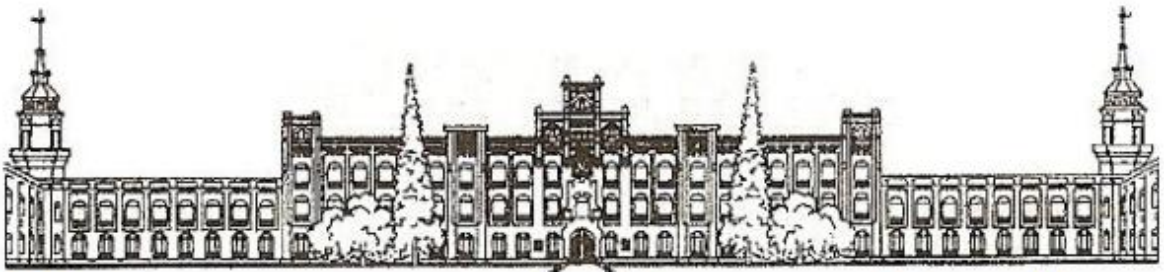


МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

А.Є.КОВАЛЕНКО

ТЕОРІЯ ІНФОРМАЦІЇ І КОДУВАННЯ: КУРС ЛЕКЦІЙ

Рекомендовано Методичною радою КПІ ім. Ігоря Сікорського як навчальний посібник для здобувачів ступеня бакалавра за спеціальністю 124 «Системний аналіз»

Київ
КПІ ім. Ігоря Сікорського
2020

Теорія інформації і кодування: курс лекцій [Електронний ресурс] : навч. посіб. для здобувачів ступеня бакалавра за спеціальністю 124 «Системний аналіз» / КПІ ім. Ігоря Сікорського ; уклад.: А.Є.Коваленко. Електронні текстові дані (1 файл: 5,758 Мбайт). Київ : КПІ ім. Ігоря Сікорського, 2020. 248 с..

Гриф надано Методичною радою КПІ ім. Ігоря Сікорського
(протокол № 10 від 18.06. 2020 р.)
за поданням Вченої ради інституту прикладного системного аналізу
(протокол №5 від 25.05. 2020 р.)

Електронне мережне навчальне видання

ТЕОРІЯ ІНФОРМАЦІЇ І КОДУВАННЯ: КУРС ЛЕКЦІЙ

Укладач: ***Коваленко Анатолій Єпіфанович*, канд. техн. наук, доц**.

Відповідальний редактор *Романенко В.Д., д-р.техн.наук, проф.*

Рецензент: *Бідюк П.І.., д-р.техн.наук, проф.*

Мета курсу лекцій – системне подання матеріалу лекційного курсу.

Курс лекцій включає теоретичний матеріал значну кількість прикладів розв'язання задач з різних розділів лекційного курсу і проведення практичних занять. Приклади розв'язання задач спрямовані на набуття практичних навичок застосування алгоритмів кодування даних і оцінки основних ентропійних характеристик інформаційних систем..

Призначено для студентів підготовки бакалаврів за спеціальністю 124 «Системний аналіз».

Посібник бути корисними для студентів старших курсів вищих навчальних закладів у процесі підготовки і самоконтролю при засвоєнні лекційного курсу.

ЗМІСТ

ПЕРЕДМОВА

Основною метою курсу «Теорія інформації і кодування» є отримання знань з алгоритмізації обробки даних і практичне застосування алгоритмів стисненні даних і побудови завадостійких кодів. Тому побудова кодів на основі типових алгоритмів кодування даних є однією з провідних задач вивчення дисципліни «Теорія інформації і кодування». Робота на практичних заняттях і самостійна робота студентів вимагає поступового системного практичного засвоєння і застосування цих алгоритмів.

Базова література курсу «Теорія інформації і кодування» [1-6] надає студентом необхідну методичну підтримку теоретичної базової частини лекційної курсу, а допоміжна література [7-10] дозволяє розширити знання з цього курсу.

ЛЕКЦІЯ 1 ОСНОВНІ ПОНЯТТЯ ТЕОРІЇ ІНФОРМАЦІЇ І КОДУВАННЯ

1.1 Теóрія інформáції і кодування

Теóрія інформáції (ТІ) — це розділи математики та кібернетики, які досліджують процеси зберігання, перетворення і передачі інформації. Теорія інформації ґрунтується на теорії ймовірностей і математичній статистиці. Базові положення ТІ пов'язані з вимірюванням і оцінкою кількості інформаціх, інформаційною ентропією, комунікаційними системами, криптографією, корекцією помилок і іншими важливими областями. Теорія інформації дозволяє отримати відповіді на два фундаментальні питання теорії передавання даних – границі стискання даних і межу швидкості передавання даних у комунікаційних ситемах із завадами.

Теорія кодування (ТК) вивчає властивості кодів та їх придатность для розв'язання специфічних задач у різних галузях науки і техніки і ґрунтується на базових положеннях і результатах ТІ. Зокрема ТК застосовують при побудові комунікаційних систем, систем захисту інформації, стиснення даних, криптографії, для знаходження і виправлення помилок, при мережевому кодуванні даних.

Дисципліна «Теóрія інформáції та кодування» (ТІК) комплексно вивчає характеристики інформаційних процесів та розробляє алгоритми стискання і передавання інформації. Результати, отримані у теорії інформації та кодуванні, застосовують у розв'язанні багатьох проблем сучасної науки і нових інформаційних технологій. Зокрема, ТІК використовують у біоінформатиці,

розподілених пошукових системах, стисканні мультимедійних даних великого об’єму тощо.

Важливими напрямами теорії кодування є стиснення даних і кодування даних інформаційних джерел (source coding) та канальне кодування (forward error correction, channel coding). Основною метою кодування джерела є стиснення даних для їх ефективнішої передачі. Канальне кодування забезпечує завадостійкість кодів.

1.2 Кодування інформації

1.2.1 Узагальнена класифікація кодів

Кодування інформації для зменшення надлишковості повідомлень називають економним, ефективним кодуванням, або стисненням даних. Мета стиснення полягає у зменшенні кількості біт, необхідних для зберігання і передавання інформації.

Завадостійке кодування інформації за рахунок надлишковості кодів забезпечує підвищення надійності передавання даних цифровими каналами. Надлишковість коду полягає у додаванні надлишкових символів, які залежать від інформаційних. Розрізняють блокові і згорткові завадостійкі коди.

Алгоритми кодування залежать від цілей кодування, форми подання кодових слів і математичних моделей оперування з кодовими словами.

1.2.2. Коди і алгоритми стиснення даних

Метою стиснення інформації є зменшення надлишковості подання кодових слів з врахуванням статистичних характеристик повідомлень.

Значного поширення набули алгоритми стиснення інформації на основі методів Шеннона-Фано, методів Хаффмана, блокування повідомлень і використання словників груп повідомлень..

1.2.3 Завадостійкі коди і алгоритми їх побудови

Завадостійкість кодів визначає ціль введення надлишковості (виявлення, виправлення помилок), моделі помилок (одинарні, кратні). Алгоритми аналізу і синтезу кодів суттєво залежать від вибраного математичного апарату опису кодів (циклічні, блокові, поліноміальні тощо).

1.3 Інформаційна ентропія, кількість інформації

1.3.1 Моделювання джерел інформації

Інформаційна ентропія в теорії інформації ґрунтується на моделюванні джерел інформації як випадкових процесів, властивості яких залежать від особливостей цих джерел.

Інформаційна ентропія є мірою невизначеності повідомлень джерела, які розглядають як випадкові величини. Повідомлення розглядають як послідовність незалежних та однаково розподілених випадкових величин.

Для джерела дискретних повідомлень інформаційне джерело (a discrete time information source) X моделюють випадковими числовими величинами X_i

на скінченному алфавіті для випадкової змінної (random variable, random quantity, aleatory variable, or stochastic variable) X. Випадкову величину називають дискретною випадковою величиною (ДВВ), $X=\{X_i\}$, якщо $|X| < \infty$, або неперервною випадковою величиною X (НВВ), якщо X відповідає значенням на множині дійсних чисел R або на її підмножині, $|X| = \infty$.

Напростішою моделлю джерела є джерело дискретних повідомлень без пам'яті (the discrete memoryless source model - DMS) за якою змінні X_i генеруються незалежно і мають єдиний розподіл ймовірностей для X. Тобто вибір наступного повідомлення джерела не залежить від попередніх виборів повідомлень.

1.3.2 Інформаційна ентропія

За Шенноном (Claude Shannon) інформаційну ентропію визначають як абсолютну межу найкращого стиснення даних без втрат.

Кількість інформації окремого дискретного повідомлення x_i оцінюють як

$$H(x_i) = -\log_2 p_i$$

де $p_i = p(x = x_i)$ - ймовірність значення x_i ДВВ X

Кількість інформації джерела дискретних повідомлень x_i, оцінюють середньою кількістю інформації, що припадає на одне елементарне повідомлення, як ентропією джерела за формулою

$$\boldsymbol{H}(\boldsymbol{X}) = -\sum_{\boldsymbol{i}=1}^{\boldsymbol{k}} \boldsymbol{p}_{\boldsymbol{i}} \log_2 \boldsymbol{p}_{\boldsymbol{i}} \quad , \qquad i=1\ldots k \ ,$$

де k - об'єм алфавіту джерела для незалежних повідомлень;

$p_i = p(x = x_i)$ - ймовірність значення x_i ДВВ X.

1.3.3 Базові аксіоми визначення функції ентропії

В моделі DMS джерело інформації розглядають як алфавіт $X=\{x_1, x_2, \ldots, x_n\}$ з розподілом ймовірностей p_i появи символів $P=\{ p_1, p_2, \ldots, p_n \}$,

$$p_i = \Pr(X=x_i), \ p_i \geq 0, \qquad (1.1)$$

$$\sum_{i=1}^{n} p_i = 1$$

Інтуїтивно ентропію джерела X за Shannon C. [1] визначають як деяку функцію $H(X)= H_n(p_1, p_2, \ldots, p_n)$, яка залежить лише від $p_1, p_2, \ldots, p_n$ і задовольняє чотирьом аксіомам [2].

Аксіома 1.1. Для двох рівно ймовірних символів ентропія дорівнює одиниці:

$$H_2(1/2 , 1/2) = 1 \text{ bit (binary unit)} \qquad (1.2)$$

Аксіома дозволяє визначити одиницю виміру значення ентропії.

Аксіома 1.2. Для символів функція H_2(p , 1-p) є неперервною функцією від p ($p \in [0,1]$)

Аксіома 1.3. Для довільної перестановки σ перестановки n ймовірностей символів виконується умова

$$H_n(\sigma(p_1, p_2, \ldots, p_n)) = H_n(p_1, p_2, \ldots, p_n) \quad (1.3)$$

Тобто значення ентропії не залежить від перестановки символів.

Аксіома 1.4.

$$H_n(p_1, p_2, \ldots, p_n) = H_{n-1}(p_1 + p_2, p_3, \ldots, p_n) + (p_1 + p_2) H_2\left(\frac{p_1}{p + p_2}, \frac{p_2}{p + p_2}\right) \quad (1.4)$$

Тобто при об'єднанні двох подій друга складова формули дає додаткову ентропію, обумовлену розглядом цих подій окремо.

Як наслідок цих аксіом маємо твердження.

Твердження 1.1 Нехай

$$A(n) = H_n(1/n, 1/n, \ldots, 1/n) \quad (1.5)$$

Тоді $A(n)$ є монотонно зростаюча функця від n.

1.3.4 Визначення функції ентропії

На основі аксіом 1.1-1.4 і твердження 1.1 можна визначити вид функції ентропії $H_n(p_1, p_2, ..., p_n)$ [2].

Теорема 1.1 Існує лише одна функція ентропії, яка задовольняє аксіомам 1.1-1.4

$$H_n(p_1, p_2, ..., p_n) = -\sum_{i=1}^{n} p_i \log_2 p_i \qquad (1.6)$$

1.3.5 Функція Хартлі

Функція Хартлі (Hartley function), введена Ralph Hartley у 1928 р., визначає ступінь невизначеності для скінченної множини елементів А

$$H_0(A) = \log_b |A| \qquad (1.7)$$

де $|A|$ - кількість елементів А;

b- основа логарифму.

Хартлі використовував значення b=10, а Шеннон b=2.

Функція Хартлі дозволяє визначити максимальну кількість інформації для ДВВ, яка подається множиною елементів А з однаковим розподілом ймовірностей елементів. Тобто

$$p_i = Pr(X=x_i)=1/k,$$
$$k=|A|$$

$$\sum_{i=1}^{n} p_i = 1$$

Максиальну інформаційну ентропію Шеннона за функцією Хартлі визначають як

$$H_0(A) = \log_2 |A|$$

Renyi було доведено, що функція Хартлі з основою 2 є єдиною функцією, яка відображає натуральні числа у дійсні числа і задовольняє вимогам адитивності

$$H(mn) = H(m) + H(n)$$

монотонності

$$H(m) \leq H(m+1)$$

і нормованості

$$H(2) = 1$$

1.4 Одиниці вимірювання кількості інформації

В загальному випадку кількість інформації окремого дискретного повідомлення x_i оцінюють як

$$H(x_i) = -\log_b p_i$$

де $p_i = p(x = x_i)$ - ймовірність значення x_i ДВВ X;

b- основа логарифму.

Вибір значення b залежить від зручності застосування відповідної рцінки кількості інформації для конкретних випадків застосування. Найбільш поширеними значеннями b є 2, 3, e, 10, 256.

1.4.1 bit, shannon

Інформаційну ентропію за Шенноном визначають як логарифм з основою b =2, тобто

$$H(x_i) = -\log_2 p_i$$

Тому одиницю виміру кількості інформації назвивають Шенноном, Shannon, Sh або bit (binary information).

Відповідно кількість інформації джерела

$$H(x_i) = -\log_2 p_i$$

де $p_i = p(x = x_i)$ - ймовірність значення x_i ДВВ X

вимірюють у бітах на символ джерела повідомлення (bit/symbol, біт/символ).

1.4.2 nat, nit, nepit

Інформаційну ентропію як натуральний логарифм з основою b =e, тобто

$$H(x_i) = -\log_e p_i$$

називають nat (від natural unit of information), nit, nepit, нат.

Відповідно для кількості інформації джерела визначають як середнє значення у нат на символ джерела.

1.4.5 ban, dit

Інформаційну ентропію як десятковий логарифм з основою b =10, тобто

$$H(x_i) = -\log_{10} p_i$$

називають Hartley, hart, ban, dit (від decimal unit of information).

Відповідно для кількості інформації джерела визначають як середнє значення у hart, ban, dit на символ джерела.

1.4.6 trit

Інформаційну ентропію як логарифм з основою b =3, тобто

$$H(x_i) = -\log_3 p_i$$

називають trit (від ternary unit of information), тріт.

Відповідно для кількості інформації джерела визначають як середнє значення у trit, тріт на символ джерела.

1.4.7 byte

Інформаційну ентропію як логарифм з основою b =256 тобто

$$H(x_i) = -\log_{256} p_i$$

називають byte, байт, оскільки байт містить 8 двійкових розрядів.

Відповідно для кількості інформації джерела визначають як середнє значення у байт на символ джерела.

1.4.8 Перетворення одиниць інформації

Практичне застосування різних одиниць інформації потребує застосування формул для взаємних перетворень при переході від одних одиниць вимірювання кількості інформації до інших.

Наприклад, для перетворення 1 nat (e^1 , $\ln_e e$) у sh отримуємо рівняння

$$2^x = e^1$$

Звідси отримуємо

$$x = \frac{1}{\ln 2} sh$$

Для перетворення 1 nat (e^1 , $\ln_e e$) у dit отримуємо рівняння

$$10^x = e^1$$

Звідси отримуємо

$$x = \frac{1}{\ln 10} hart = 0{,}434 hart$$

Аналогічно отримуємо значення

1 nat = $\log_2 e$=1.11 bit

1 bit=ln2=0,693 nat

1trit=ln3=1,099nat

1hart=ln10=2,303nat

1byte=ln256=5,545nat

1.5 Надмірність джерела інформації

Для нерівноймовірних елементарних повідомлень x_i джерела повідомлень ентропія зменшується, що оцінюють статистичною надмірністю (надлишковістю, redundancy)

$$\rho_X = 1 - \boldsymbol{H}(\boldsymbol{X}) \Big/ \boldsymbol{H}(\boldsymbol{X})_{\max} = 1 - \frac{\boldsymbol{H}(\boldsymbol{X})}{\log_2 \boldsymbol{k}},$$

де H(X) - ентропія джерела повідомлень;

$H(X)_{max}=\log_2 k$ - максимально досяжна ентропія даного джерела

Значеннняч $H(X)_{max}$ визначають за функція Хартлі для b=2. Максимальна надмірність відповідає найбільшій невизначеності джерела повідомлень. Це використовується у системах захисту інформації, зокрема для побудови і криптографічних алгоритмів та визначенні їх криптостійкості. У системах передавання даних надмірність дозволяє оцінити алгоритми стискання даних і їх передавання в умовах наявних завад каналів зв’язку.

1.6 Продуктивність джерела

Продуктивність джерела щодо конкретного повідомлення $\mathbf{x_i}$ визначають як

$$V_{si} = \frac{I(X_i)}{t_i},$$

де t_i – проміжок часу вибору повідомлення x_i;

$I(x_i)$ – кількість інформації повідомлення x_i.

Продуктивність джерела інформації характеризують середнім значенням

$$V_s = \frac{1}{t_m}\sum_{i=1}^{k} p_i I(X_i) = \frac{H(X)}{t_m},$$

де t_m – середній час вибору джерелом одного повідомлення;

p_i - ймовірність появи повідомлення x_i.

Значення t_m може бути визнавчено у різний спосіб. Зокрема, експериментально значення τ_m обчислюють як

$$t_m = \frac{\sum_{i=1}^{k} t_i}{k},$$

або задають апріорно. Можливо також обчислення середнього часу як середнього статистичного значення

$$t_m = \sum_{i=1}^{k} p_i t_i$$

1.7 Спільна ентропія двох випадкових величин

Спільна ентропія (joint entropy) для двох дискретних випадкових величин X, Y обчислюється за формулою

$$H(X,Y) = -\sum_{j}\sum_{i} p(x_i, y_j) \log_2 p(x_i, y_j),$$

де $p(x_i, y_j)$ – спільна ймовірность, що визначає закон розподілу ймовірностей ДВВ, тобто $p(x_i, y_j) = \Pr(X = x_i, Y = y_j)$.

1.8 Кодування джерела інформації

Кодування полягає у перетворенні інформації на впорядкований набір символів, елементів, знаків. При кодуванні кожному повідомленню з деякої множини, що називається ансамблем повідомлень, ставиться у відповідність кодова комбінація − набір символів (елементів, знаків). Множина повідомлень називається алфавітом повідомлень, або первинним алфавітом, а множина символів (елементів, знаків) для кодування називається алфавітом джерела, або вторинним алфавітом. Побудовану відповідно до певної схеми кодування множину кодових наборів (комбінацій) називають кодом.

Приклади кодування наведені ансамблю повідомлень у таблиці 1.1.

Таблиця 1.1 – Кодування ансамблю повідомлень

Ансамбль повідомлень	Схема кодування				
	Двійкова система	Десяткова система	Код Фібоначчі	Код Грея	Префіксний код
A	000	0	0000	000	00
B	001	1	0001	001	010
C	010	2	0011	011	011
D	011	3	0101	010	10
E	100	4	0111	110	110
F	101	5	1100	100	111

При кодуванні у десятковій системі числення застосовують вторинний алфавіт {0,1,…,9}, в для інших схем – алфавіт {0,1}. Але часто для символів 0,1,…,9 застосовують двійкові символи (у двійково-десятковому коді). Тобто кожний символ кодують за допомогою чотирьох двійкових. Наприклад, 9 має кодовий набір 1001.

Для рівномірних кодів довжина кодових наборів однакова, а для нерівномірних кодів довжина кодових наборів може бути різною. У таблиці 1.1 для префіксного коду вона становить від 2 до 3.

1.6 Приклади розв'язування задач

Приклад 1. ДВВ X1 та X2 визначаються підкиданням двох гральних кубиків. Задано ДВВ Y=X1+X2. Знайти кількість інформації I(Y,X1), I(X1,X1), I(Y,Y).

Розв'язання

Побудуємо розподіл ймовірностей ДВВ X_1 (або X_2) (табл.1.1).

Таблиця 1.1 - Розподіл ймовірностей ДВВ X

X_1	1	2	3	4	5	6
p_i	1/6	1/6	1/6	1/6	1/6	1/6

Звідси знаходимо ентропію ДВВ X_1, що у даному випадку буде максимальною, оскільки значення рівноймовірні:

$$HX_1=HX_2=\log_2 6=1+\log_2 3\approx 1+1{,}585\approx 2{,}585 \text{ (біт/сим).}$$

Оскільки X_1 та X_2 незалежні ДВВ., то їх сумісна ймовірність визначається так:

$P(X_1=i, X_2=j)=P(X_1=i)\cdot P(X_2=j)=1/36$; i=1...6, j=1...6.

Побудуємо допоміжною таблицю значень ДВВ Y $=X_1+X_2$ (табл.1.2).

Таблиця 1.2 - Значення ДВВ Y

X_2	X_1					
	1	2	3	4	5	6
1	2	3	4	5	6	7
2	3	4	5	6	7	8
3	4	5	6	7	8	9
4	5	6	7	8	9	10
5	6	7	8	9	10	11
6	7	8	9	10	11	12

Обчислимо ймовірності P(Y=j) (j=2, 3, …, 12):

$P(Y=2)=1/36$; $P(Y=3)=2\cdot 1/36=1/18$; $P(Y=4)=3\cdot 1/36=1/12$;

$P(Y=5)=4\cdot 1/36=1/9$; $P(Y=6)=5\cdot 1/36=5/36$; $P(Y=7)=6\cdot 1/36=1/6$; $P(Y=8)=5\cdot 1/36=5/36$; $P(Y=9)=4\cdot 1/36=1/9$; $P(Y=10)=3\cdot 1/36=1/12$; $P(Y=11)=2\cdot 1/36=1/18$; $P(Y=12)=1/36$;

Скориставшись допоміжною таблицею (табл. 1.2), запишемо безумовний закон розподілу ДВВ Y (табл. 1.3).

Таблиця 1.3 - Безумовний закон розподілу ДВВ Y

Y_j	2	3	4	5	6	7	8	9	10	11	12
q_j	1/36	1/18	1/12	1/9	5/36	1/6	5/36	1/9	1/12	1/18	1/36

Звідси знаходимо ентропію ДВВ Y так:

$$\boldsymbol{HY} = -\sum_{j=2}^{12} \boldsymbol{q}_j \log_2 \boldsymbol{q}_j = 2\frac{1}{36}\log_2 36 + 2\frac{1}{18}\log_2 18 + 2\frac{1}{12}\log_2 12 +$$

$$+ 2\frac{1}{9}\log_2 9 + 2\frac{5}{36}\log_2 \frac{36}{5} + 2\frac{1}{36}\log_2 36 + \frac{1}{6}\log_2 6 = \frac{23}{18} +$$

$$+ \frac{5}{3}\log_2 3 - \frac{5}{18}\log_2 5 \approx 3{,}27 \text{ (біт/сим)}$$

Побудуємо таблицю розподілу ймовірностей системи ДВВ (X_1, Y) $p_{ij}=P(X_1=i, Y=j)$ (табл. 1.4).

Таблиця 1.4 - Розподіл ймовірностей системи ДВВ (X_1,Y)

X_1	Y										
	2	3	4	5	6	7	8	9	10	11	12
1	1/36	1/36	1/36	1/36	1/36	1/36	0	0	0	0	0
2	0	1/36	1/36	1/36	1/36	1/36	1/36	0	0	0	0
3	0	0	1/36	1/36	1/36	1/36	1/36	1/36	0	0	0
4	0	0	0	1/36	1/36	1/36	1/36	1/36	1/36	0	0
5	0	0	0	0	1/36	1/36	1/36	1/36	1/36	1/36	0
6	0	0	0	0	0	1/36	1/36	1/36	1/36	1/36	1/36
	1/36	1/18	1/12	1/9	5/36	1/6	5/36	1/9	1/12	1/18	1/36

Тоді взаємна ентропія ДВВ. X_1, Y

$$\boldsymbol{H}(\boldsymbol{X}_1,\boldsymbol{Y}) = -\sum_{ij} \boldsymbol{p}_{ij} \log_2 \boldsymbol{p}_{ij} = \log_2 36 = 2\log_2 6 = 2(1+\log_2 3) \approx 2 + + 2\cdot 1{,}585 \approx 5{,}17$$

(біт/сим).

Кількість інформації, що містить. Y стосовно X_1, обчислимо за формулою $\boldsymbol{I}(\boldsymbol{Y},\boldsymbol{X}) = \sum_{ij} \boldsymbol{p}_{ij} \log_2 \frac{\boldsymbol{p}_{ij}}{\boldsymbol{p}_i \boldsymbol{q}_j}$. Отже,

$$\boldsymbol{I}(\boldsymbol{Y},\boldsymbol{X}_1) = \sum_{j=1}^{6} \sum_{1\le j-i\le 6} \boldsymbol{p}_{ij} \log_2 \frac{\boldsymbol{p}_{ij}}{\boldsymbol{p}_i \boldsymbol{q}_j} = \frac{1}{36} \sum_{j=1}^{6} \sum_{1\le j-i\le 6} \log_2 \left(\frac{1}{36 \cdot 1/6 \cdot \boldsymbol{q}_j} \right) =$$

$$= \frac{1}{36}(\sum_{j=2}^{7}\log_2\frac{1}{6\boldsymbol{q}_j} + \sum_{j=3}^{8}\log_2\frac{1}{6\boldsymbol{q}_j} + \sum_{j=4}^{9}\log_2\frac{1}{6\boldsymbol{q}_j} + \sum_{j=5}^{10}\log_2\frac{1}{6\boldsymbol{q}_j} +$$

$$+ \sum_{j=6}^{11}\log_2\frac{1}{6\boldsymbol{q}_j} + \sum_{j=7}^{12}\log_2\frac{1}{6\boldsymbol{q}_j}) = \frac{1}{36}((\log_2\frac{6}{1} + \log_2\frac{6}{2} + \log_2\frac{6}{3} +$$

$$+ \log_2\frac{6}{4} + \log_2\frac{6}{5} + \log_2\frac{6}{6}) + (\log_2\frac{6}{2} + \log_2\frac{6}{3} + \log_2\frac{6}{4} + \log_2\frac{6}{5} +$$

$$+ \log_2\frac{6}{6} + \log_2\frac{6}{5}) + (\log_2\frac{6}{3} + \log_2\frac{6}{4} + \log_2\frac{6}{5} + + \log_2\frac{6}{6} + \log_2\frac{6}{5} +$$

$$+ \log_2\frac{6}{4}) + (\log_2\frac{6}{4} + \log_2\frac{6}{5} + \log_2\frac{6}{6} + \log_2\frac{6}{5} + \log_2\frac{6}{4} + \log_2\frac{6}{3}) +$$

$$+ (\log_2\frac{6}{5} + \log_2\frac{6}{6} + \log_2\frac{6}{5} + \log_2\frac{6}{4} + \log_2\frac{6}{3} + \log_2\frac{6}{2}) + (\log_2\frac{6}{6} +$$

$$+ \log_2\frac{6}{5} + \log_2\frac{6}{4} + + \log_2\frac{6}{3} + \log_2\frac{6}{2} + \log_2\frac{6}{1})) = \frac{1}{36}(2\log_2 6 +$$

$$+ 4\log_2 3 + 6\log_2 2 + 8\log_2\frac{3}{2} + 10\log_2\frac{6}{5} + +6\log_2 1) = \frac{1}{36}(10 +$$

$$+ 24\log_2 3 - 10\log_2 5) \approx 0{,}69 \text{ (біт/сим)}.$$

Кількість інформації $I(Y,X_1)$ здебільшого зручніше знайти, скориставшись властивістю кількості інформації і ентропії:

$$\boldsymbol{I}(\boldsymbol{X},\boldsymbol{Y}) = \boldsymbol{HX} + \boldsymbol{HY} - \boldsymbol{H}(\boldsymbol{X},\boldsymbol{Y}).$$

Оскільки Y=X1+X2, де X1 та X2 – незалежні ДВВ, то

$$\boldsymbol{H}(\boldsymbol{Y},\boldsymbol{X}_1) = \boldsymbol{H}(\boldsymbol{X}_1 + \boldsymbol{X}_2, \boldsymbol{X}_1) = \boldsymbol{HX}_1 + \boldsymbol{HX}_2 = 2\boldsymbol{HX}_1.$$

$$\boldsymbol{I}(\boldsymbol{Y},\boldsymbol{X}_1) = \boldsymbol{HY} + \boldsymbol{HX}_1 - 2\boldsymbol{HX}_1 = \boldsymbol{HY} - \boldsymbol{HX}_1 = \frac{23}{18} + \frac{5}{3}\log_2 3 - \frac{5}{18}\times$$

$$\times \log_2 5 - (1 + \log_2 3) = \frac{5}{18} + \frac{2}{3}\log_2 3 - \frac{5}{18}\log_2 5 \approx 0{,}69 \text{ (біт/сим)}.$$

Відповідь: $\boldsymbol{HX}_1 = \boldsymbol{HX}_2 = 1 + \log_2 3 \approx 2{,}585$ (біт/сим);

$$\boldsymbol{I}(\boldsymbol{Y},\boldsymbol{Y}) = \boldsymbol{HY} = \frac{23}{18} + + \frac{5}{3}\log_2 3 - \frac{5}{18}\log_2 5 \approx 3{,}27 \text{ (біт/сим)};$$

$$\boldsymbol{I}(\boldsymbol{Y},\boldsymbol{X}) = \frac{5}{18} + \frac{2}{3}\log_2 3 - \frac{5}{18}\log_2 5 \approx 0{,}69 \text{ (біт/сим)}.$$

Приклад Знайти ентропії ДВВ X, Y, Z та кількість інформації, що містить ДВВ $\boldsymbol{Z}=|\boldsymbol{X}-\boldsymbol{Y}|$ стосовно X та Y. X, Y – незалежні ДВВ., які задаються розподілами ймовірностей (табл. 1.5, табл. 1.6).

Таблиця - Розподіл ймовірностей системи ДВВ X

X	1	2	3	4
p	1/8	1/8	1/4	1/2

Таблиця 1.6- Розподіл ймовірностей ДВВ Y

Y	1	2	3	4
q	1/4			

Розв'язання

Скориставшись відповідним рядом розподілу ймовірностей ДВВ X та Y, знаходимо їх ентропії.

Ентропія ДВВ X

$$\boldsymbol{HX}=2\frac{1}{8}\log_2 8+\frac{1}{4}\log_2 4+\frac{1}{2}\log_2 2=\frac{1}{4}3+\frac{1}{4}2+\frac{1}{2}=1{,}75$$ (біт/сим).

Ентропія ДВВ Y $\boldsymbol{HY}=\log_2 4=2$ (біт/сим).

Побудуємо допоміжну таблицю значень ДВВ $Z=|X\text{-}Y|$ та їх ймовірностей (табл. 1.7). Оскільки X та Y – незалежні ДВВ, то сумісна ймовірність випадання пар значень (x_i,y_j)

$$\boldsymbol{p_{ij}=P(X=i,Y=j)=p(x_i)\cdot p(y_j)=p_i\cdot q_j},\quad \boldsymbol{i,j}=\overline{1,4}.$$

Таблиця 1.7 - Сумісна ймовірність випадання пар значень (x_i,y_j)

X	Y				$\sum_j p_{ij} = p_i$
	1	2	3	4	
1	0	1	2	3	1/8
	1/32	1/32	1/32	1/32	
2	1	0	1	2	1/8
	1/32	1/32	1/32	1/32	
3	2	1	0	1	1/4
	1/16	1/16	1/16	1/16	
4	3	2	1	0	1/2
	1/8	1/8	1/8	1/8	
$\sum_i p_{ij} = q_j$	1/4	1/4	1/4	1/4	1

Знайдемо ймовірності системи ДВВ (Z=j, X=i, $j = \overline{0,3}$, $i = \overline{1,4}$):

P(Z=0, X=1)=1/32, P(Z=1, X=1)=1/32, P(Z=2, X=1)=1/32, P(Z=3, X=1)=1/32; P(Z=0, X=2)=1/32, P(Z=1, X=2)=1/32+1/32=1/16, P(Z=2, X=2)=1/32, P(Z=3, X=2)=0; P(Z=0, X=3)=1/16, P(Z=1, X=3)=1/16+1/16=1/8, P(Z=2, X=3)=1/16, P(Z=3, X=3)=0; P(Z=0, X=4)=1/8, P(Z=1, X=4)=1/8, P(Z=2, X=4)=1/8, P(Z=3, X=4)=1/8.

Побудуємо таблицю розподілу ймовірностей системи ДВВ (X, Z) (табл. 1.8).

Таблиця 1.8 – Розподіл ймовірністей системи ДВВ (X, Z)

X	Z				$\sum_j p_{ij}$
	0	1	2	3	
1	1/32	1/32	1/32	1/32	1/8
2	1/32	1/16	1/32	0	1/8
3	1/16	1/8	1/16	0	1/4
4	1/8	1/8	1/8	1/8	1/2
$\sum_i p_{ij}$	1/4	11/32	1/4	5/32	1

Тоді взаємна ентропія ДВВ Z та X

$$\boldsymbol{H(Z,X)} = -\sum_{ij} \boldsymbol{p_{ij}} \log_2 \boldsymbol{p_{ij}} = 6 \cdot \frac{1}{32} \log_2 32 + 3 \cdot \frac{1}{16} \log_2 16 +$$

$$+ 5 \cdot \frac{1}{8} \log_2 8 = \frac{3}{16} \cdot 5 + \frac{3}{16} \cdot 4 + \frac{5}{8} \cdot 3 = \frac{57}{16} \approx 3{,}563 \,(\text{біт/сим}).$$

Скориставшись табл. 1.7 або табл. 1.8, побудуємо розподіл ймовірностей ДВВ. Z (табл. 1.9).

Таблиця 1.9 – Розподіл ймовірностей ДВВ Z

	0	1	Z	3
p_i	1/4	11/32	1/4	5/32

Звідси знаходимо ентропію ДВВ Z:

$$\boldsymbol{HZ} = \frac{2}{4} \log_2 4 + \frac{11}{32} \log_2 \frac{32}{11} + \frac{5}{32} \log_2 \frac{32}{5} = \frac{1}{2} 2 + \frac{11}{32} (5 - \log_2 11) +$$

$$+ \frac{5}{32} (5 - \log_2 5) = \frac{7}{2} - \frac{5}{32} \log_2 5 - \frac{11}{32} \log_2 11 \approx 1{,}948 \,(\text{біт/сим}).$$

Кількість інформації, що містить ДВВ Z стосовно ДВВ X, знаходимо, скориставшись властивістю кількості інформації і ентропії:

$$\boldsymbol{I(Z,X)} = \boldsymbol{HZ} + \boldsymbol{HX} - \boldsymbol{H(Z,X)} = \frac{7}{2} - \frac{5}{32} \log_2 5 - \frac{11}{32} \log_2 11 +$$

$$+ \frac{7}{4} - \frac{57}{16} = \frac{27}{16} - \frac{5}{32} \log_2 5 - \frac{11}{32} \log_2 11 \approx 0{,}136 \,(\text{біт/сим}).$$

Побудуємо таблицю розподілу ймовірностей системи ДВВ (Y,Z) (табл. 1.10). Для цього, скориставшись табл. 3, обчислимо ймовірності:

P(Z=0, Y=1)=1/32, P(Z=1, Y=1)=1/32, P(Z=2, Y=1)=1/16, P(Z=3, Y=1)=1/8; P(Z=0, Y=2)=1/32, P(Z=1, Y=2)=1/32+1/16=3/32, P(Z=2, Y=2)=1/8, P(Z=3, Y=2)=0; P(Z=0, Y=3)=1/16, P(Z=1, Y=3)=1/32+1/8=5/32, P(Z=2, Y=3)=1/32, P(Z=3, Y=3)=0; P(Z=0, Y=4)=1/8, P(Z=1, Y=4)=1/16, P(Z=2, Y=4)=1/32, P(Z=3, Y=4)=1/32.

Таблиця 1.8 – Розподіл ймовірністей системи ДВВ (X, Z)

Y	Z				$\sum_j p_{ij}$
	0	1	2	3	
1	1/32	1/32	1/16	1/8	1/4
2	1/32	3/32	1/8	0	1/4
3	1/16	5/32	1/32	0	1/4
4	1/8	1/16	1/32	1/32	1/4
$\sum_i p_{ij}$	1/4	11/32	1/4	5/32	1

Тоді взаємна ентропія ДВВ Z та Y

$$\boldsymbol{H}(\boldsymbol{Z},\boldsymbol{Y}) = 6\frac{1}{32}\log_2 32 + 3\cdot\frac{1}{16}\log_2 16 + 3\frac{1}{8}\log_2 8 + \frac{3}{32}\log_2\frac{32}{3} +$$

$$+\frac{5}{32}\log_2\frac{32}{5} = \frac{3}{16}\cdot 5 + \frac{3}{16}\cdot 4 + \frac{3}{8}\cdot 3 + \frac{3}{32}(5-\log_2 3) + \frac{5}{32}(5 -$$

$$-\log_2 5) = \frac{65}{16} - \frac{3}{32}\log_2 3 - \frac{5}{32}\log_2 5 \approx 4{,}063 - \frac{3}{32}\cdot 1{,}585 - \frac{5}{32}\times$$

$$\times 2{,}322 \approx 3{,}551\,(\text{біт/сим}).$$

Отже, кількість інформації, що містить ДВВ Z стосовно ДВВ Y

$$\boldsymbol{I}(\boldsymbol{Z},\boldsymbol{Y}) = \boldsymbol{HZ} + \boldsymbol{HY} - \boldsymbol{H}(\boldsymbol{Z},\boldsymbol{Y}) = \frac{7}{2} - \frac{5}{32}\log_2 5 - \frac{11}{32}\log_2 11 + 2 -$$

$$-\left(\frac{65}{16} - \frac{3}{32}\log_2 3 - \frac{5}{32}\log_2 5\right) = \frac{23}{16} + \frac{3}{32}\log_2 3 - \frac{11}{32}\log_2 11 \approx$$

$$\approx 1{,}438 + \frac{3}{32}\cdot 1{,}585 - \frac{11}{32}\cdot 3{,}459 \approx 0{,}397\ (\text{біт/сим}).$$

Відповідь: HX = 1,75 (біт/сим); HY = 2 (біт/сим);

$\boldsymbol{HZ} = \frac{7}{2} - \frac{5}{32}\log_2 5 - \frac{11}{32}\log_2 11 \approx 1{,}948\,(\text{біт/сим})$;

$$\boldsymbol{I}(\boldsymbol{Z},\boldsymbol{X}) = \frac{27}{16} - \frac{5}{32}\log_2 5 - \frac{11}{32}\log_2 11 \approx 0{,}136\,(\text{біт/сим});$$

$$\boldsymbol{I}(\boldsymbol{Z},\boldsymbol{Y}) = \frac{23}{16} + \frac{3}{32}\log_2 3 - \frac{11}{32}\log_2 11 \approx 0{,}397\,(\text{біт/сим}).$$

ЛЕКЦІЯ 2 ХАРАКТЕРИСТИКИ ДИСКРЕТНОГО КАНАЛУ ПЕРЕДАВАННЯ ІНФОРМАЦІЇ

2.1 Ентропія джерела

2.1.1 Безумовна ентропія джерела

Джерело повідомлень **X,** $X=\{x_1, x_2, .., x_i, .., x_k\}$, алфавіту статистично незалежних повідомлень $\mathbf{x_i}$ характеризують відповідним розподілом ймовірностей $P(X)=\{P(x_1), P(x_2), \ldots . P(x_k)\}=(p_1, p_2 .. p_k)$.

Ентропію такого джерела визначають як середню кількість інформації джерела повідомлень на одне з k статистично незалежних повідомлень

$$\boldsymbol{H}(\boldsymbol{X}) = -\sum_{\boldsymbol{i}=1}^{\boldsymbol{k}} \boldsymbol{p}_{\boldsymbol{i}} \log_2 \boldsymbol{p}_{\boldsymbol{i}} \quad , \qquad \text{i=1…k ,}$$

і називають безумовною ентропією.

2.1.2 Умовна ентропія повідомлення джерела

За наявності статистичної залежності між повідомленнями $\mathbf{x_i}$ джерела $X=\{x_1, x_2, .., x_i, .., x_k\}$ факт вибору одного з повідомлень зменшує або збільшує ймовірності вибору інших повідомлень до умовних ймовірностей. Відповідно

змінюється ентропія й кількість інформації, що міститься в кожному з цих повідомлень

$$\boldsymbol{I}(\boldsymbol{X}_i / \boldsymbol{X}_{i-1}, \boldsymbol{X}_{i-2}, \ldots) = -\log_2 \boldsymbol{p}(\boldsymbol{x}_i / \boldsymbol{x}_{i-1}, \boldsymbol{x}_{i-2}, \ldots)$$

Таку ентропію називають умовною ентропією повідомлення.

Одним з найпростіших випадків є залежність повідомлення x_i лише від одного значення x_{i-1}. Для опису таких процесів часто використовують ланцюги Маркова

$$I(X_i / X_{i-1}) = -\log_2 p(x_i / x_{i-1})$$

2.2 Статистична модель канала зв'язку

2.2.1 Узагальнена модель системи передачі даних

Статистичну модель канала зв'язку від одного джерела повідомлень до іншого джерела повідомлень можна розглядати як модель системи спостереження, перетворення, збору і зберігання інформації. Ця модель складається з двох джерел X та Y, між повідомленнями яких існує статистичний взаємозв'язок. Повідомлення від джерела X надсилаються до джерела Y, яке описують множиною власний повідомлень.

Джерело X задано моделлю - ансамблем повідомлень $X = \{x_1, x_2, \ldots, x_k\}$ і рядом розподілу $P(X) = \{P(x_1), P(x_2), \ldots . P(x_k)\}$ ймовірностей. Джерело Y задано

ансамблем $Y = \{y_1, y_2, ..., y_l\}$ і розподілом $R(Y) = \{R(y_1), R(y_2), \ldots R(y_l)\}$. Ніяких обмежень на алфавіти X і Y не накладається.

Для багатьох застосувань вважають, що кількість відправлених і отриманих повідомлень співпадають, тобто l=k. Для моделей із надісланими але втраченими повідомленнями часто використовують припущення l=k+1.

2.2.2 Матриця прямих переходів повідомлень

Статистична залежність джерела Y від джерела X задається матрицею прямих переходів повідомлень x_i (i=1…k) джерела X в повідомлення y_j (j=1…k) джерела Y:

P(Y/X)=

X	**Y**					
	Y_1	**Y_2**	**…**	**y_i**	**…**	**y_k**
x_1	**$p(y_1/x_1)$**	**$p(y_2/x_1)$**	**…**	**$p(y_j/x_1)$**	**…**	**$p(y_k/x_1)$**
x_2	**$p(y_1/x_2)$**	**$p(y_2/x_2)$**	**…**	**$p(y_j/x_2)$**	**…**	**$p(y_k/x_2)$**
…	**…**	**…**	**…**	**…**	**…**	**…**
x_i	**$p(y_1/x_i)$**	**$p(y_2/x_i)$**	**…**	**$p(y_j/x_i)$**	**…**	**$p(y_k/x_i)$**
…	**…**	**…**	**…**	**…**	**…**	**…**
x_k	**$p(y_1/x_k)$**	**$p(y_2/x_k)$**	**…**	**$p(y_j/x_k)$**	**…**	**$p(y_k/x_k)$**

На головній діагоналі цієї матриці розташовані умовні ймовірності прямої відповідності типу $\mathbf{x_1 \to y_1}$, $\mathbf{x_2 \to y_2}$, …, $\mathbf{x_k \to y_k}$, які характеризують правильний вибір джерелом **Y** повідомлень.

Для кожного рядка повинна виконуватися умова нормування

$$\sum_j p(y_j / x_i) = 1, \quad i = 1..k$$

2.2.3 Матриця зворотних переходів повідомлень

Статистична залежність джерела X від джерела Y подається матрицею зворотних переходів типу $x_i \leftarrow y_j$ з умовних ймовірностей $p(x_i/y_j)$:

P(X/Y)=

X	**Y**					
	y_1	Y_2	..	y_i	…	y_k
x_1	$p(x_1/y_1)$	$p(x_1/y_2)$	..	$p(x_1/y_j)$	…	$p(x_1/y_k)$
x_2	$p(x_2/y_1)$	$p(x_2/y_2)$	..	$p(x_2/y_j)$	…	$p(x_2/y_k)$
…			..	…	…	…
x_i	$p(x_i/y_1)$	$p(x_i/y_2)$	..	$p(x_i/y_j)$	…	$p(x_i/y_k)$
…		…	..	…	…	…
x_k	$p(x_k/y_1)$	$p(x_k/y_2)$	..	$p(x_k/y_j)$	…	$p(x_k/y_k)$

Матриця складається з **k** розміщених стовпцями варіантів первинних розподілів ймовірностей ансамблю **X**, що на собі відчуває статистичний вплив повідомлень $\mathbf{y_j}$ джерела **Y**. Для кожного такого розподілу виконується умова нормування

$$\sum_i p(x_i / y_j) = 1, \quad j = 1..k$$

2.2.4 Матриця сумісних ймовірностей

Якщо задані ансамбль **X** і матриця прямих переходів, то, використовуючи безумовні імовірності **P(X)={p(x$_i$)}** можна знайти матрицю сумісних ймовірностей

$$p(x_i, y_j) = \begin{bmatrix} p(x_1, y_1)\ p(x_1, y_2) \dots p(x_1, y_k) \\ p(x_2, y_1)\ p(x_2, y_2) \dots p(x_2, y_k) \\ \dots \quad \dots \quad \dots \quad \dots \\ p(x_k, y_1)\ p(x_k, y_2) \dots p(x_k, y_k) \end{bmatrix}.$$

Виконавши згортку за **i**, дістанемо ряд розподілу безумовних ймовірностей **P(Y)={p(y$_j$)}**, **j=1…k**:

$$p(y_j) = \sum_{i=1}^{k} p(x_i, y_j), \quad j = 1 \dots k,$$

Виконавши згортку за j отримуємо розподіл P(X)={p(x$_i$)}, i=1…k:

$$p(x_i) = \sum_{j=1}^{k} p(x_i, y_j), \quad i = 1 \dots k.$$

Розподіли $P(X) = \{P(x_1), P(x_2), \dots P(x_k)\}$, $P(Y) = \{P(y_1), P(y_2), \dots P(y_l)\}$ використовують при аналізі статистичних характеристик відповідних джерел у процесі розв'язання задач з оцінки кількості інформації, яка пересилається каналом передавання даних.

2.3 Умовні ентропії передавання повідомлень

2.3.1 Умовні ймовірності джерел інформації

За умови наявності статистичної залежності між деякими повідомленнями x і y застосовують умовну ймовірність p(x/y) появи повідомлення x_i за умови, що вже вибрано повідомлення y_j або умовну ймовірність появи повідомлення y_j, якщо вже отримане повідомлення x_i p(y/x). При цьому у загальному випадку p(x/y)≠p(y/x).

Умовну ймовірність можна отримати з безумовної ймовірності p(x) чи p(y) та сумісної ймовірності системи випадкових величин p(x,y) за формулою множення ймовірностей:

$$p(x,y)=p(x)\cdot p(y/x),$$

$$p(x,y)=p(y)\cdot p(x/y),$$

Звідгси

$$p(y/x)=p(x,y)/p(x),$$

$$p(x/y)=p(x,y)/p(y).$$

Для статистично незалежних повідомлень маємо: p(y/x)=p(y), p(x/y)=p(x).

2.3.2 Поняття умовної ентропії

Для алфавітів повідомлень $X=\{x_1, x_2, .., x_i, .., x_k\}$, $Y = \{y_1, y_2, ..., y_j, ..., y_l\}$ можна визначити дві часткові умовні ентропії: $H(Y/x_i)$ та $H(X/y_j)$.

Часткова умовна ентропія H(Y/xi) є ентропія для повідомленя x_i джерела X, щодо якого визначається алфавіт Y, за умови вибору джерелом X повідомлення x_i.

Часткова умовна ентропія $H(X/y_j)$ є ентропія алфавіту X за умови вибору повідомлення y_j; алфавіту Y.

2.3.3 Обчислення часткової умовної ентропії $H(X/y_j)$

Для обчислення умовної ентропії H(X/yj) використовують умовні ймовірності $p(x_i/y_j)$ матрици зворотних переходів.

Часткова умовна ентропія - це кількість інформації, що припадає на одне повідомлення джерела X за умови встановлення факту вибору джерелом Y повідомлення y_j:

$$H(X/y_j) = -\sum_i p(x_i/y_j)\log_2 p(x_i/y_j), j = 1..l,$$

де $X=\{x_1, x_2, .., x_i, .., x_k\}$, $Y = \{y_1, y_2, ..., y_j, ..., y_l\}$ - алфавіти повідомлень;

y_j - певне повідомлення джерела Y, щодо якого визначається часткова умовна ентропія $H(X/y_j)$ алфавіту X за умови вибору повідомлення y_j;

i - номер повідомлення з алфавіту X;

j - номер повідомлення з алфавіту Y;

$p(x_i/y_j)$– умовні імовірності.

2.3.4 Обчислення часткової умовної ентропії $H(Y/x_i)$

Для обчислення умовної ентропії H(Y/xi використовують умовні ймовірності $p(y_j/x_i)$ матрици прямих переходів.

Часткова умовна ентропія - це кількість інформації, що припадає на одне повідомлення джерела Y за умови, що відомий стан джерела X:

$$\boldsymbol{H(Y/x_i)=-\sum_j p(y_j/x_i)\log_2 p(y_j/x_i), i=1..k},$$

де $\boldsymbol{X=\{x_1, x_2, .., x_i, .., x_k\}}$, $\boldsymbol{Y=\{y_1, y_2, ..., y_j, ..., y_l\}}$ - алфавіти повідомлень;

x_i - певне повідомлення джерела **X**, щодо якого визначається часткова умовна ентропія $H(Y/x_i)$ алфавіту **Y** за умови вибору джерелом **X** повідомлення x_i ;

i - номер повідомлення з алфавіту X;

j - номер повідомлення з алфавіту Y;

$p(y_j/x_i)$ – умовні імовірності.

2.3.5 Загальна умовна ентропія H(X/Y)

Загальна або повна умовна ентропія визначається так:

$$\boldsymbol{H}(\boldsymbol{X}/\boldsymbol{Y}) = \sum_j \boldsymbol{p}(\boldsymbol{y}_j)\boldsymbol{H}(\boldsymbol{X}/\boldsymbol{y}_j),$$

тобто середньостатистична кількість інформації (математичне сподівання), що припадає на будь-яке повідомлення джерела X, якщо відомий його статистичний взаємозв'язок з джерелом Y.

Ці умовні ентропії визначають як

$$\boldsymbol{H}(\boldsymbol{X}/\boldsymbol{Y}) = -\sum_j \boldsymbol{p}(\boldsymbol{y}_j)\sum_i \boldsymbol{p}(\boldsymbol{x}_i/\boldsymbol{y}_j)\log_2 \boldsymbol{p}(\boldsymbol{x}_i/\boldsymbol{y}_j) =$$
$$= -\sum_j \sum_i \boldsymbol{p}(\boldsymbol{x}_i, \boldsymbol{y}_j)\log_2 \boldsymbol{p}(\boldsymbol{x}_i/\boldsymbol{y}_j),$$

Статистична надмірності, обумовленої наявністю статистичної залежності між елементами повідомлення, визначають як

$$\rho_{X/Y} = 1 - \boldsymbol{H}(\boldsymbol{X}/\boldsymbol{Y})/\boldsymbol{H}(\boldsymbol{X}),$$

де H(X/Y) - загальна умовна ентропія джерела **X** стосовно джерела **Y**;

H(X) - безумовна ентропія джерела **X**.

2.3.6 Загальна умовна ентропія H(Y/X)

$$\boldsymbol{H(Y/X)=\sum_i p(x_i)H(Y/x_i)}.$$

тобто середня кількість інформації, яка міститься в повідомленнях джерела **Y** за наявності статистичного взаємозв'язку з джерелом **X**.

Ці умовні ентропії визначають як

$$\boldsymbol{H(Y/X)=-\sum_i p(x_i)\sum_j p(y_j/x_i)\log_2 p(y_j/x_i)=}$$
$$\boldsymbol{=-\sum_i\sum_j p(x_i,y_j)\log_2 p(y_j/x_i),}$$,

де $p(x_i, y_j)$ - сумісна імовірність появи повідомлень x_i, y_j;

$p(y_j/x_i)$ – їх умовні імовірності.

2.3.7 Загальна статистична надлишковість алфавіту джерела інформації

Загальна статистична надлишковість алфавіту джерела інформації визначається так:

$$\boldsymbol{\rho_{X,Y}=\rho_X+\rho_{X/Y}-\rho_X\rho_{X/Y}}.$$

У разі малих значень ρ_X, $\rho_{X/Y}$ статистична надлишковість визначається виразом

$$\rho_{X,Y} = \rho_X + \rho_{X/Y}.$$

2.3.8 Ентропію H(X,Y) об'єднання двох джерел інформації X і Y

Ентропію **H(X,Y)** об'єднання двох джерел інформації **X** і **Y** знаходять через імовірності $\mathbf{p(x_i,y_j)}$ системи випадкових повідомлень $\mathbf{x_i}$, $\mathbf{y_j}$ для всіх **i=1...k**, і **j=1...l**. Для цього складається матриця ймовірностей системи двох статистично залежних джерел

$$p(x_i,y_j)=\begin{bmatrix} p(x_1,y_1)\ p(x_1,y_2)\ldots p(x_1,y_l) \\ p(x_2,y_1)\ p(x_2,y_2)\ldots p(x_2,y_l) \\ \ldots \quad \ldots \quad \ldots \quad \ldots \\ p(x_k,y_1)\ p(x_k,y_2)\ \ldots p(x_k,y_l) \end{bmatrix}.$$

Ентропія об'єднання двох джерел H(X,Y) (взаємна ентропія) - це середня кількість інформації, що припадає на два будь-які повідомлення джерел **X** і **Y**:

$$H(X,Y)=-\sum_{ij} p(x_i,y_j)\log_2 p(x_i,y_j).$$

З рівності $p(x_i,y_j)= p(y_j,x_i)$ випливає, що

$$\mathbf{H(X,Y)=H(Y,X)}.$$

$$\mathbf{H(X, Y)=H(Y)+H(X/Y)},$$

Звідси

$$\mathbf{H(Y/X)=H(X,Y)-H(X)},$$

$$\mathbf{H(X/Y)=H(X,Y)-H(Y)}.$$

2.4 Кількість інформації каналу зв’язку

Кількість інформації, що припадає на одне повідомлення, передане по каналу зв'язку джерелом **X** спостерігачу **Y** за наявності завад і статистичного взаємозв'язку ансамблів **X** і **Y** знаходиться за формулою

$$\mathbf{I(X,Y)=H(Y)+H(X)-H(X,Y)=H(X)-H(X/Y)=H(Y)-H(Y/X)}.$$

2.5 Швидкість передачі інформації каналом зв'язку

Повідомлення x_i передається по каналу зв'язку спостерігачеві Y, роль якого відіграє приймальний пристрій. Вибір повідомлень $y_j \in Y$ джерелом Y характеризує процес передачі інформації по каналу зв'язку від джерела X на вихід джерела Y. При цьому взаємна кількість інформації I(X,Y) - це середня

кількість інформації про стан джерела X, що міститься в одному повідомленні джерела Y.

Оскільки на вибір кожного повідомлення y_j джерелом Y витрачається час τ, то швидкість передачі інформації по каналу зв'язку знаходиться за формулою

$$V = \frac{I(X,Y)}{\tau}.$$

2.6 Кількість інформації каналу зв’язку за відсутності завад

При повній відсутності інформаційних втрат (завад в каналі)

$$\mathbf{I(X,Y)=H(X)=H(Y)=H(X,Y)}.$$

2.7 Кількість інформації каналу зв’язку за наявності завад

а при високому рівні завад спостерігається повна статистична незалежність джерел X і Y

$$\mathbf{H(X/Y)=H(X)},$$
$$\mathbf{H(Y/X)=H(Y)},$$
$$\mathbf{H(X,Y)= H(X)+H(Y)}.$$
$$\mathbf{I(X, Y)= 0}.$$

Оцінка втрат інформації у каналі за наявності завад грнтується на використанні повних умовних ентропій каналу зв’язку.

У проміжному випадку завади деякою мірою спотворюють передані повідомлення і умовна ентропія змінюється в межах

$$\mathbf{0 \le H(X/Y) \le H(X)},$$

$$\mathbf{0 \le H(Y/X) \le H(Y)}.$$

Тому кількість переданої по каналу зв'язку інформації як кількість нового відсутнього знання визначається різницею **HX** і **H(X/Y)**:

$$\mathbf{I(X, Y)=H(X)-H(X/Y)}.$$

2.8 Пропускна здатність передавання інформаціїї

2.8.1 Пропускна здатність каналу зв’язку

Максимально можлива швидкість передавання інформації по каналу називається пропускною здатністю, або ємністю каналу зв'язку С

$$\boldsymbol{C} = \frac{1}{\tau}\left[\boldsymbol{I}(\boldsymbol{X},\boldsymbol{Y})\right]_{\max} = \frac{1}{\tau}\left[\boldsymbol{H}(\boldsymbol{X}) - \boldsymbol{H}(\boldsymbol{X}/\boldsymbol{Y})\right]_{\max}.$$

2.8.2 Пропускна здатність каналу за відсутності завад

У разі рівноімовірних повідомлень пропускна здатність за відсутності завад

$$\boldsymbol{C} = \frac{1}{\boldsymbol{\tau}} \boldsymbol{H}(\boldsymbol{X})_{\max} = \frac{1}{\boldsymbol{\tau}} \log_2 \boldsymbol{k} \ .$$

2.8.3 Пропускна здатність каналу за наявності завад

За наявних завад умовна ентропія **H(X/Y)** знаходиться в діапазоні

$$\mathbf{0 \le H(X/Y) \le H(X),}$$

а пропускна здатність каналу дорівнює

$$\boldsymbol{C} = \frac{1}{\boldsymbol{\tau}} \left[\log_2 \boldsymbol{k} - \boldsymbol{H}(\boldsymbol{X}/\boldsymbol{Y}) \right] \ .$$

При зменшенні рівня завад пропускна здатність каналу **C** прямує до максимального значення, а при збільшенні рівня завад – до нуля.

2.9 Приклади розв'язування задач

Приклад 1 Матриця умовних ймовірностей каналу зв'язку між джерелом **X** і спостерігачем **Y** має вигляд

$$\boldsymbol{p}(\boldsymbol{y}_j / \boldsymbol{x}_i) = \begin{bmatrix} 0{,}97 & 0{,}02 & 0{,}01 \\ 0{,}1 & 0{,}86 & 0{,}04 \\ 0{,}03 & 0{,}08 & 0{,}89 \end{bmatrix}.$$

Знайти часткову та загальну умовні ентропії повідомлень у цьому каналі, якщо задано розподіл ймовірностей джерела $\mathbf{P_x}=\{0{,}65;\ 0{,}3;\ 0{,}05\}$.

Розв'язання

Умовні ймовірності $p(y_j/x_i)$ при **i=j** характеризують вплив завад у каналі зв'язку.

Часткова умовна ентропія спостерігача **Y** щодо джерела **X** визначається за формулою

$$\boldsymbol{H}(\boldsymbol{Y} / \boldsymbol{x}_i) = -\sum_j \boldsymbol{p}(\boldsymbol{y}_j / \boldsymbol{x}_i) \log_2 \boldsymbol{p}(\boldsymbol{y}_j / \boldsymbol{x}_i)\,.$$

Знайдемо часткові умовні ентропії для всіх $\mathbf{x_i}$, **i**=1, …,3

$$\boldsymbol{H}(\boldsymbol{Y} / \boldsymbol{x}_1) = -\left(0{,}97 \log_2 0{,}97 + 0{,}02 \log_2 0{,}02 + 0{,}01 \log_2 0{,}01\right) = 1{,}98 + 2 \times$$
$$\times \log_2 5 - 0{,}97 \log_2 97 \approx 1{,}98 + 2 \cdot 2{,}322 - 0{,}97 \cdot 6{,}6 \approx 0{,}222\ (\text{біт/сим});$$

$$\boldsymbol{H}(\boldsymbol{Y} / \boldsymbol{x}_2) = -\left(0{,}1 \log_2 0{,}1 + 0{,}86 \log_2 0{,}86 + 0{,}04 \log_2 0{,}04\right) = 0{,}96 + 1{,}9 \times$$
$$\times \log_2 5 - 0{,}86 \log_2 43 \approx 0{,}96 + 1{,}9 \cdot 2{,}322 - 0{,}86 \cdot 5{,}426 \approx 0{,}705\ (\text{біт/сим});$$

$$\boldsymbol{H}(\boldsymbol{Y} / \boldsymbol{x}_3) = -\left(0{,}03 \log_2 0{,}03 + 0{,}08 \log_2 0{,}08 + 0{,}89 \log_2 0{,}89\right) =$$
$$= 1{,}76 + 2 \log_2 5 - 0{,}03 \log_2 3 - 0{,}89 \log_2 89 \approx 1{,}76 + 2 \cdot 2{,}322 - 0{,}03 \times$$

$\times,585 - 0,89 \cdot 6,476 \approx 0,593$ (біт/сим).

Знаходимо загальну умовну ентропію ДВВ **Y** стосовно ДВВ **X**:

$$\boldsymbol{H}(\boldsymbol{Y}/\boldsymbol{X}) = \sum_i \boldsymbol{p}(\boldsymbol{x}_i)\boldsymbol{H}(\boldsymbol{Y}/\boldsymbol{x}_i) = 0,65 \cdot 0,222 + 0,3 \cdot 0,705 + 0,05 \times$$

$$\times 0,593 \approx 0,385 \text{ (біт/сим)}.$$

Скориставшись формулою множення ймовірностей

$$p(x_i, y_j)=p(x_i)\cdot p(y_j/x_i),$$

побудуємо матрицю ймовірностей системи ДВВ **X**, **Y**:

$$\boldsymbol{p}(\boldsymbol{x}, \boldsymbol{y}) = \begin{bmatrix} 0,97 \cdot 0,65 & 0,02 \cdot 0,65 & 0,01 \cdot 0,65 \\ 0,1 \cdot 0,3 & 0,86 \cdot 0,3 & 0,04 \cdot 0,3 \\ 0,03 \cdot 0,05 & 0,08 \cdot 0,05 & 0,89 \cdot 0,05 \end{bmatrix} = \begin{bmatrix} 0,6305 & 0,013 & 0,0065 \\ 0,03 & 0,258 & 0,012 \\ 0,0015 & 0,004 & 0,0445 \end{bmatrix}.$$

Перевіряємо умову нормування: $\sum_{ij} \boldsymbol{p}_{ij}$ = 0,6305+0,013+0,0065+

+0,03+0,258+0,012+0,0015+0,004+0,0445=1.

З матриці сумісних ймовірностей **p(x, y)** знаходимо взаємну ентропію ДВВ **X**, **Y**:

$$\boldsymbol{H}(\boldsymbol{X}, \boldsymbol{Y}) = -\big(0,6306\log_2 0,6305 + 0,013\log_2 0,013 + 0,0065\log_2 0,0065 +$$
$$+ 0,03\log_2 0,03 + 0,258\log_2 0,258 + 0,012\log_2 0,012 + 0,0015 \times$$

$$\times \log_2 0,0015 + 0,004\log_2 0,004 + 0,0445\log_2 0,0445\big) = 3,363 + 2,97 \times$$

$$\times \log_2 5 - 0,3015\log_2 3 - 0,65\log_2 13 - 0,258\log_2 43 - 0,0445\log_2 89 -$$
$$- 0,6305\log_2 97 \approx 3,363 + 2,97 \cdot 2,322 - 0,3015 \cdot 1,585 - 0,65 \cdot 3,7 -$$

$$- 0,258 \cdot 5,426 - -0,0445 \cdot 6,476 - 0,6305 \cdot 6,6 \approx 1,527 \text{ (біт/сим)}.$$

Скориставшись заданим рядом розподілу ймовірностей д. в. в. **X**, знайдемо ентропію **X**:

$$HX = -(0{,}65\log_2 0{,}65 + 0{,}3\log_2 0{,}3 + 0{,}05\log_2 0{,}05) = 1{,}7 + \log_2 5 - 0{,}3\log_2 3 -$$
$$-0{,}65\log_2 13 \approx 1{,}7 + 2{,}322 - 0{,}3 \cdot 1{,}585 - 0{,}65 \cdot 3{,}7 \approx 1{,}142 \text{(біт/сим)}.$$

Перевірка: H(X, Y)=HX+H(Y/X)=1,142+0,385≈1,527 (біт/сим).

Виконавши в матриці сумісних ймовірностей $\mathbf{p_{ij}}$= **p(x, y)** згортку за **i**, отримаємо приблизний безумовний розподіл ДВВ **Y**:

$$p(y_1) = \sum_i p(x_i, y_1) = 0{,}6305 + 0{,}03 + 0{,}0015 = 0{,}662;$$

$$p(y_2) = \sum_i p(x_i, y_2) = 0{,}013 + 0{,}258 + 0{,}004 = 0{,}275;$$

$$p(y_3) = \sum_i p(x_i, y_3) = 0{,}0065 + 0{,}012 + 0{,}0445 = 0{,}063.$$

Перевіряємо умову нормування

$$\mathbf{p(y_1)+ p(y_2)+ p(y_3)}=0{,}662+0{,}275+0{,}063=1.$$

Знаючи приблизний безумовний закон розподілу $\mathbf{P_y}$={0,662; 0,275; 0,063}, знайдемо матрицю умовних ймовірностей **p(x/ y)**, скориставшись формулою

$$p(x/y) = \frac{p(x,y)}{p(y)}.$$

$$p(x_i / y_j) = \begin{bmatrix} \frac{0,6305}{0,662} & \frac{0,013}{0,275} & \frac{0,0065}{0,063} \\ \frac{0,03}{0,662} & \frac{0,258}{0,275} & \frac{0,012}{0,063} \\ \frac{0,0015}{0,662} & \frac{0,004}{0,275} & \frac{0,0445}{0,063} \end{bmatrix} = \begin{bmatrix} 0,9524 & 0,0473 & 0,1032 \\ 0,0453 & 0,9382 & 0,1905 \\ 0,0023 & 0,0145 & 0,7063 \end{bmatrix}.$$

$$y_1 \qquad y_2 \qquad y_3$$

Перевіряємо умови нормування:

$$\sum_i p(x_i / y_1) = 0,9524 + 0,0453 + 0,0023 = 1;$$

$$\sum_i p(x_i / y_2) = 0,0473 + 0,9382 + 0,0145 = 1;$$

$$\sum_i p(x_i / y_3) = 0,1032 + 0,1905 + 0,7063 = 1.$$

Виходячи з розподілу безумовних ймовірностей ДВВ **Y**, знайдемо ентропію Y:

$$HY = -(0,662\log_2 0,662 + 0,275\log_2 0,275 + 0,063\log_2 0,063) \approx$$

$$\approx 1,157 \text{ (біт/сим)}.$$

З матриці умовних ймовірностей p(x/y) знайдемо часткові умовні ентропії ДВВ X стосовно ДВВ Y:

$$H(X / y_1) = -(0,9524\log_2 0,9524 + 0,0453\log_2 0,0453 + 0,0023 \times$$

$$\times \log_2 0,0023) \approx 0,289 \text{ (біт/сим)};$$

$$H(X / y_2) = -(0,0473\log_2 0,0473 + 0,9382\log_2 0,9382 + 0,0145 \times$$

$$\times \log_2 0,0145) \approx 0,383 \text{ (біт/сим)};$$

$$\boldsymbol{H}(\boldsymbol{X}/\boldsymbol{y}_3) = -(0{,}1032\log_2 0{,}1032 + 0{,}1905\log_2 0{,}1905 + 0{,}7063 \times$$

$$\times \log_2 0{,}7063) \approx 1{,}148 \text{ (біт/сим)}.$$

Тоді загальна умовна ентропія X стосовно Y

$$\boldsymbol{H}(\boldsymbol{X}/\boldsymbol{Y}) = \sum_j \boldsymbol{p}(\boldsymbol{y}_j)\boldsymbol{H}(\boldsymbol{X}/\boldsymbol{y}_j) = 0{,}662\,0{,}289 + 0{,}275\,0{,}383 +$$

$$+ 0{,}063 \cdot 1{,}148 \approx 0{,}369 \text{ (біт/сим)}.$$

Перевірка:

H(X, Y)=HY+H(X/Y)=1,157+0,369=1,527 (біт/сим).

Відповідь:

$H(Y/x_1)\approx 0{,}222$ (біт/сим);

$H(Y/x_2)\approx 0{,}705$ (біт/сим);

$H(Y/x_3)\approx 0{,}593$ (біт/сим);

H(Y/X)≈0,385 (біт/сим);

$H(X/y_1)\approx 0{,}289$ (біт/сим);

$H(X/y_2)\approx 0{,}383$ (біт/сим);

$H(X/y_3)\approx 1{,}148$ (біт/сим);

H(X/Y)≈0,369 (біт/сим).

Приклад 2 Матриця сумісних ймовірностей каналу зв'язку має вигляд

$$\boldsymbol{p}(\boldsymbol{x}_i, \boldsymbol{y}_j) = \begin{bmatrix} 0{,}15 & 0{,}15 & 0 \\ 0 & 0{,}25 & 0{,}1 \\ 0 & 0{,}2 & 0{,}15 \end{bmatrix}.$$

Знайти інформаційні втрати, пропускну здатність і швидкість передачі інформації по дискретному каналу зв'язку, якщо час передачі одного повідомлення $\tau=10^{-3}$ с.

Розв'язання

Інформаційні втрати в каналі зв'язку визначаються умовною ентропією **H**(**X**/**Y**) одного джерела щодо іншого.

Для того щоб обчислити повну умовну ентропію **H**(**X**/**Y**), потрібно знайти розподіли безумовних ймовірностей $\mathbf{p}(\mathbf{x_i})$, $\mathbf{p}(\mathbf{y_j})$ і побудувати матрицю умовних ймовірностей $\mathbf{p}(\mathbf{x_i}/\mathbf{y_j})$.

Безумовний закон розподілу $\mathbf{p}(\mathbf{x_i})$ знаходимо, виконавши в матриці сумісних ймовірностей $\mathbf{p}(\mathbf{x_i}, \mathbf{y_j})$ згортку за **j**:

$$p(x_i) = \sum_j p(x_i, y_j)$$

$\mathbf{p}(\mathbf{x_1})$= 0,15+0,15+0=0,3, **i**=1;

$\mathbf{p}(\mathbf{x_2})$= 0+0,25+0,1=0,35, **i**=2;

$\mathbf{p}(\mathbf{x_3})$= 0+0,2+0,15=0,35, **i**=3.

Перевіряємо умову нормування

$$\mathbf{p(x_1)+ p(x_2)+ p(x_3)}=0,3+0,35+0,35=1.$$

Виходячи з розподілу безумовних ймовірностей ДВВ **X**, обчислимо її ентропію:

$$\boldsymbol{HX} = -(0,3\log_2 0,3+0,35\log_2 0,35+0,35\log_2 0,35) \approx 1,581 \text{ (біт/сим)}.$$

Безумовний закон розподілу $\mathbf{p(y_j)}$ знаходимо, виконавши в матриці сумісних ймовірностей $\mathbf{p(x_i, y_j)}$ згортку за **i**:

$$p(y_j) = \sum_i p(x_i, y_j)$$

$\mathbf{p(y_1)}$= 0,15+0+0=0,15, **j**=1;
$\mathbf{p(y_2)}$= 0,15+0,25+0,2=0,6, **j**=2;
$\mathbf{p(y_3)}$= 0+0,1+0,15=0,25, **j**=3.

Перевіряємо умову нормування:

$$\mathbf{p(y_1)+ p(y_2)+ p(y_3)}=0,15+0,6+0,25=1.$$

Матрицю умовних ймовірностей знаходимо, скориставшись формулою множення ймовірностей

$$\mathbf{p(x_i, y_j)=p(y_j)\cdot p(x_i/y_j)}.$$

Звідси випливає, що

$$p(x_i / y_j) = \frac{p(x_i, y_j)}{p(y_j)}.$$

Отже, матриця умовних ймовірностей $\mathbf{p(x_i/y_j)}$ знаходиться так:

$$p(x_i / y_j) = \begin{matrix} \\ x_1 \\ x_2 \\ x_3 \end{matrix} \begin{matrix} \begin{bmatrix} \frac{0,15}{0,15} & \frac{0,15}{0,6} & \frac{0}{0,25} \\ \frac{0}{0,15} & \frac{0,25}{0,6} & \frac{0,1}{0,25} \\ \frac{0}{0,15} & \frac{0,2}{0,6} & \frac{0,15}{0,25} \end{bmatrix} \\ y_1 \quad y_2 \quad y_3 \end{matrix} = \begin{bmatrix} 1 & 0,25 & 0 \\ 0 & 0,4167 & 0,4 \\ 0 & 0,3333 & 0,6 \end{bmatrix}.$$

Для матриці умовних ймовірностей $\mathbf{p(x_i/y_j)}$ повинна виконуватися умова нормування

$$\sum_i p(x_i / y_j) = 1.$$

Перевіряємо цю умову:

$$\sum_i p(x_i / y_1) = 1 + 0 + 0 = 1,$$

$$\sum_i p(x_i / y_2) = 0,25 + 0,42 + 0,33 = 1,$$

$$\sum_i p(x_i / y_3) = 0 + 0,4 + 0,6 = 1.$$

Скориставшись матрицею умовний ймовірностей $\mathbf{p(x_i/y_j)}$, обчислимо часткові умовні ентропії **X** стосовно **Y**:

$$H(X/y_1) = -\sum_i p(x_i/y_1)\log_2 p(x_i/y_1) = -\log_2 1 = 0 \text{ (біт/сим);}$$

$$H(X/y_2) = -\sum_i p(x_i/y_2)\log_2 p(x_i/y_2) = -(0{,}25\log_2 0{,}25 +$$

$$+0{,}42\log_2 0{,}42 + 0{,}33\log_2 0{,}33) \approx 1{,}555 \text{ (біт/сим);}$$

$$H(X/y_3) = -\sum_i p(x_i/y_3)\log_2 p(x_i/y_3) = -(0 + 0{,}4\log_2 0{,}4 +$$

$$+0{,}6\log_2 0{,}6) \approx 0{,}971 \text{ (біт/сим).}$$

Виходячи з безумовного закону розподілу ДВВ **Y** та знайдених часткових умовних ентропій $\mathbf{H(X/y_j)}$, відшукуємо їх математичне сподівання – загальну умовну ентропію

$$H(X/Y) = \sum_j p(y_j)\cdot H(X/y_j) = 0{,}15\cdot 0 + 0{,}6\cdot 1{,}555 + 0{,}25\cdot 0{,}971 \approx$$

$$\approx 1{,}176 \text{ (біт/сим).}$$

Отже, інформаційні втрати в каналі зв'язку $H(X/Y) \approx 1{,}18$ (біт/сим). Пропускна здатність каналу із шумом обчислюється за формулою

$$C = \frac{1}{\tau}\left[\log_2 k - H(X/Y)\right],$$

де **k** - об'єм алфавіту джерела;

τ - час вибору повідомлення джерелом.

Отже, отримаємо

$$C = \frac{1}{10^{-3}}\left[\log_2 3 - 1{,}176\right] \approx 0{,}409\cdot 10^3 = 409 \text{ (бод).}$$

Кількість переданої по каналу інформації, що припадає на одне повідомлення джерела, знаходиться, виходячи із середньої кількості інформації, що виробляється джерелом – його ентропії і інформаційних втрат в каналі:

$$\mathbf{I(X,Y)=HX-H(X/Y)}=1,581-1,176\approx 0,406 \text{ (біт/сим)}.$$

Швидкість передачі інформації знаходиться так:

$$\boldsymbol{v}=\frac{\boldsymbol{I(X,Y)}}{\boldsymbol{\tau}}=0,406\cdot 10^{3}=406 \text{ (бод)}.$$

Відповідь:

$$\mathbf{H(X/Y)}\approx 1,18 \text{ (біт/сим)};\ \mathbf{C}\approx 409 \text{ (бод)};\ \mathbf{v}=406 \text{ (бод)}.$$

Джерело повідомлень X задано ансамблем $\{x_1, x_2, x_3\}$ з ймовірностями $p(x_1)=0,65;\ p(x_2)=0,25;\ p(x_3)=0,1.$ Матриця умовних ймовірностей каналу має вигляд

$$p(y_j/x_i)=\begin{pmatrix} 0,99 & 0,005 & 0,005 \\ 0,13 & 0,750 & 0,120 \\ 0,15 & 0,350 & 0,500 \end{pmatrix}.$$

Знайти кількість інформації, передану в одному та 100 повідоленнях джерела, інформаційні втрати в каналі при передачі 100 повідомлень з алфавіту X?

Умовні ймовірності $p(y_j/x_i)$ при i=j характеризують вплив завад у каналі зв'язку.

Часткова умовна ентропія спостерігача Y щодо джерела X визначається за формулою

$$H(Y/x_i)=-\sum_i p(y_j/x_i)*\log_2 p(y_j/x_i)$$

Знайдемо часткові умовні ентропії для i=1,...,3

$H(Y/x_1)$ = -(0,99*log0,99 + 0,005*log0,005 + 0,005*log0,005)= -(-0,01435-0,382-0,382)=-(-0,0908)= 0,0908 (біт/сим).

$$H(Y/x_2)=-(0,13\log_2 0,13+0,75\log_2 0,75+0,12\log_2 0,12)=-(-0,38264-0,31129-0,36707)=1,061$$ (біт/сим).

$$H(Y/x_3)=-(0,15\log_2 0,15+0,35\log_2 0,35+0,5\log_2 0,5)=-(-0,410545-0,5301-0,5)=1,44065$$ (біт/сим)

Інформаційні втрати визначаються умовною ентропією. Знаходимо загальну умовну ентропію ДВВ Y стосовно ДВВ X(для одного повідомлення):

$$H(Y/X)=\sum_i p(x_i)*H(Y/x_i)=0,65*0,0908+0,25*1,061+0,1*1,44065=0,46834$$ (біт/сим).

$$H_{100}(Y/X)=100*0,4683=43,83$$ (біт/сим)

Скориставшись формулою множення ймовірностей

$$p(x_i, y_j)=p(x_i)*p(y_j/x_i)$$,

побудуємо матрицю ймовірностей системи ДВВ X, Y:

$$p(x,y)=\begin{pmatrix} 0,99*0,65 & 0,005*0,65 & 0,005*0,65 \\ 0,13*0,25 & 0,75*0,25 & 0,12*0,25 \\ 0,15*0,10 & 0,35*0,10 & 0,50*0,10 \end{pmatrix}=\begin{pmatrix} 0,6435 & 0,00325 & 0,00325 \\ 0,0325 & 0,18750 & 0,03000 \\ 0,0150 & 0,03500 & 0,05000 \end{pmatrix}$$

Знайдемо H(Y). Для цього знайдемо $p(y_1), p(y_2), p(y_{3.})$, виконавши в матрицы сумісних ймовірностей згортку за i.

$$p(y_1)=0,6435+0,0325+0,015=0,691$$
$$p(y_2)=0,00325+0,1875+0,035=0,22575$$
$$p(y_3)=0,00325+0,03+0,05=0,08325$$

Перевіряємо умову нормування:

$$p(y_1)+p(y_2)+p(y_3)=0,691+0,22575+0,0825=1$$

Ентропія Y:

$$H(Y)=-(0,691*\log_2 0,691+0,22575*\log_2 0,22575+0,0825*\log_2 0,0825)=-0,3685$$
$$-0,484731-0,296956=1,1502$$

Перевіримо умову нормування

$$\sum_{ij} p_{ij}=0,6435+0,00325+0,00325+0,0325+0,1875+0,03+0,015+0,035+0,05=1$$

Отже кількість інформації, передана в 1 повідомленні дорівнює

$$I(X,Y)=H(Y)-H(Y/X);$$

$$I(X,Y)=1,1502-0,46834=0,68186 \text{ (біт/сим);}$$

Для 100 повідомлень

$$I_{100}(X,Y)=100*0,68186=68,186 \text{ (біт/сим)}$$

Відповідь

1)Інформаційні втрати, передані в 1 повідомленні

$$H(Y/X)=\sum_i p(x_i)*H(Y/x_i)=0,46834 \text{ (біт/сим)}$$

2)Інформаційні втрати,передані в 100 повідомленнях

$$H_{100}(Y/X)=100*0,4683=43,83 \text{ (біт/сим)}$$

3)Кількість інформації, передана в 1 повідомленні

$$I(X,Y)=1,1502-0,46834=0,68186 \text{ (біт/сим)}$$

4)Кількість інформації, передена в 100 повідомленнях

$$I_{100}(X,Y)=100*0,68186=68,186 \text{ (біт/сим)}$$

ЛЕКЦІЯ 3 ОПТИМАЛЬНІ СТАТИСТИЧНІ МЕТОДИ СТИСНЕННЯ ІНФОРМАЦІЇ

3.1 Основні поняття

3.1.1 Стиснення інформації

Кодування інформації для зменшення надмірності (надлишковості) повідомлень називають економним або ефективним кодуванням. Застосування економних кодів призводить до стиснення інформації. Мета стиснення - зменшення кількості біт, необхідних для зберігання і передачі інформації. Оптимальні коди стискання інформації забезпечують алгоритми або тоди оптимального кодування.

Поширеними способами побудови економних кодів є кодові таблиці, кодові дерева, метод Шеннона-Фано, метод Хаффмена.

3.1.2 Кодові таблиці

Подання кодів кодовими таблицями полягає у зіставленні символам повідомлень певних кодових комбінацій. Приклад кодової таблиці рівномірних кодів подається в табл. 3.1.

Таблиця 3.1 – Кодова таблиця

Символ x_i	Число λ_i	Код з основою 10	Код з основою 4	Код з основою 2
А	0	0	00	000
Б	1	1	01	001
В	2	2	02	010
Г	3	3	03	011
Д	4	4	10	100
Е	5	5	11	101
Ж	6	6	12	110
З	7	7	13	111

3.1.3 Кодові дерева

Наочним і зручним способом подання кодів є кодові дерева (рис.3.1).

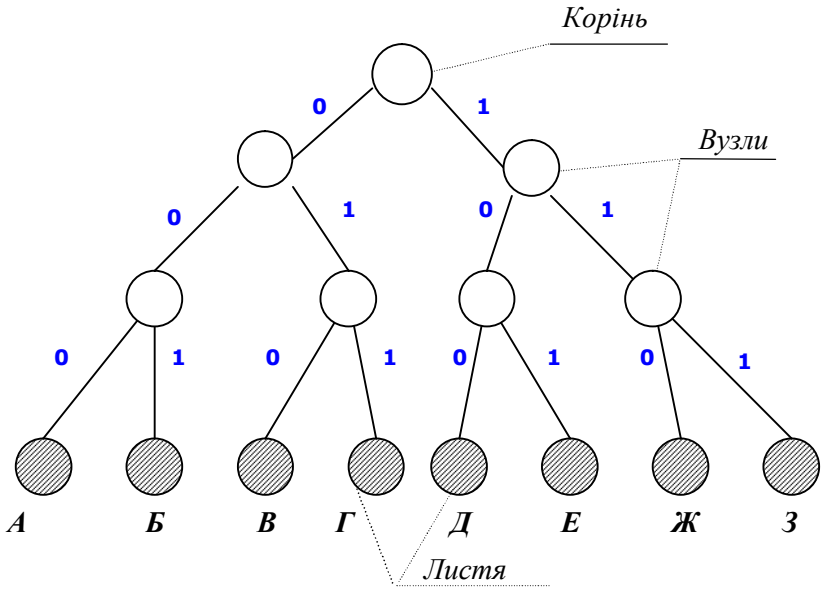


Рисунок 3. 1 – Подання символів рівномірними кодами

Для цього, починаючи з деякої точки - кореня кодового дерева, проводяться гілки, що позначаються 0 або 1. На листях кодового дерева знаходяться символи алфавіту джерела, причому кожному символу відповідає свій лист і свій шлях від кореня до відповідного листа, який утворює певну кодову комбінацію.

3.1.4 Метод Шеннона – Фано

Метод Шеннона-Фано. Значення ДВВ символів розміщуються у порядку спадання ймовірностей. Потім уся сукупність розділяється на дві приблизно рівні за сумою ймовірностей частини. До коду першої частини додається 0, а до коду другої - 1. Кожна з частин за тим самим принципом знову розділяється (якщо це можливо) на дві частини і т.і.

Побудуємо таблицю кодів за методом Шеннона-Фано для повідомлень, заданих розподілом ймовірностей (табл. 3.2).

Приклад. Побудувати код Шеннона –Фано джерела алфавіту для заданого віиразу.

Вираз	Код
навчай інших — і сам навчишся.	Код Хаффмена першого порядку

Визначити середню довжину коду. Обчислити довжину закодованого виразу і довжину для ASCII-коду виразу. Визначити коефіцієнт стискання.

Розв'язок задачі

Побудуємо таблицю кодування код Шеннона –Фано.

Для цього спочатку визначаємо частотність появи окремих символів a_i і задаємо їх як ймовірність p_i.

Далі упорядковуємо (сортуємо) символи за ймовірністю p_i у порядку не зростання або зменшення.

Починаємо формувати перший розряд кодових наборів. Для цього розділяємо множину символів на дві групи символів з найменшою різницею сум їх ймовірностей. Оскільки знаменник у нас число 30, то це приблизно 15/30 на кожну групу.

Тому першу групу можуть утворити символи: _(проміжок), а, н, с з ймовірністю 14/30, а іншу групу символи и, ш, і,…, я з ймовірністю 16/30. Різниця між ними становить 2/30.

символ, a_i	Ймовірність, p_i	Код	довжина коду l_i	$p_i l_i$
_	$\frac{5}{30}$	000	3	$\frac{15}{30}$
А	$\frac{4}{30}$	001	3	$\frac{12}{30}$
Н	$\frac{3}{30}$	010	3	$\frac{9}{30}$
С	$\frac{2}{30}$	011	3	$\frac{6}{10}$
И	$\frac{2}{30}$	1000	4	$\frac{8}{30}$
Ш	$\frac{2}{30}$	1001	4	$\frac{8}{30}$
І	$\frac{2}{30}$	1010	4	$\frac{8}{30}$
Ч	$\frac{2}{30}$	1011	4	$\frac{8}{30}$
В	$\frac{2}{30}$	1100	4	$\frac{8}{30}$
.	$\frac{1}{30}$	11010	5	$\frac{5}{30}$
-	$\frac{1}{30}$	11011	5	$\frac{5}{30}$
й	$\frac{1}{30}$	11100	5	$\frac{5}{30}$
м	$\frac{1}{30}$	11101	5	$\frac{5}{30}$
х	$\frac{1}{30}$	11110	5	$\frac{5}{28}$
Я	$\frac{1}{30}$	11111	5	$\frac{5}{28}$

Іншим варіантом є: перша група символів _(проміжок), а, н, с, и з ймовірністю 16/30, а іншу групу символи ш, і,…, я з ймовірністю 14/30. Різниця між ними становить 2/30.

Тому, за методом Шеннона –Фано, варіанти є рівноцінними і можна вибрати довільний варіант.

Вибираємо перший варіант. Тобто символи: _(проміжок), а, н, с отримають перший розряд кодового набору 0, а символи и, ш, і,…, я перший розряд кодового набору 1 (перший стовпчик таблиці кодування).

Далі ці групи розглядаємо окремо для отримання другого розряду кодових наборів символів.

Для символів _, а, н, с робимо розбивку на дві підгрупи: _, а та н, с або _, та а, н, с. Вони є рівноцінними с точки зору методу Шеннона-Фано вони є рівноцінними за різницею ймовірностей. Але перший варіант надалі дозволяє отримати символи з меншими кодовими наборами.

Тому для першого варіанту підгрупам з одного елементу _ та а надають значень відповідно 0 і 1 для третього розряду кодових наборів.

Аналогічну розбивку проводимо для групи и, ш, і,…, я на підгрупи.

Заповнюємо відповідну графу кодових наборів і обчислюємо добуток $p_i l_i$.

Обчислюємо середню довжину коду.

$$\overline{L} = \sum_{i=1}^{15} p_i * l_i = 6*5/30 + 5*8/30 + 6/30 + 9/30 + 12/30 + 15/30 = 112/30 = 3.733 біт./сим$$

Закодуємо наш вираз:

При кодуванні підсталяємо значення кодових наборів.

Наприклад, «навчай» отримує кодову послідовність

010 001 1100 1011 001 11100

Довжина даного коду для виразу становить 112 біт.

Довжина даного повідомлення в коді ASCII: 8*30=240 біт.

Коефіцієнт стискання k=240/112=2.14

3.1.5 Метод Хаффмена

За методом Хаффмана код будують за допомогою бінарного дерева. Спочатку ймовірності ДВВ вихідного ансамблю повідомлень розміщують у спадному порядку і приписують листкам кодового дерева, які відповідають окремим повідомленням.

Метод Хаффмана полягає у побудові дерева Хаффмана, яке складається з вузлів. Вузол може бути кінцевим, тобто листком, або проміжним (батьківським) Проміжний вузол утворюють з двох іншиз вузлів. Величину, що приписують вузлу дерева, називають його вагою.

Два листи або вузли з найменшими значеннями ваги утворюють батьківський вузол. Вага батьківського вузла дорівнює сумарній вазі вузлів, що його утворюють, а самі дочірні вузли виключають з подальшого розгляду.

Надалі батьківський вузол враховують нарівні з вузлами, що лишилися. Надалі листя або вузли, що утворили батьківський вузол, більше не розглядаються.

Після отримання кінцевого вузла - кореня дерева, визначають кодовий набір кожного повідомлення. Для цього кожну гілку, що виходить з батьківського вузла, позначають 0 (зазвичай, це ліва гілка) або 1 (права гілка). Кодовий набір значень ДВВ – це послідовності 0 і 1 на шляху від кореня

кодового дерева до листа із заданою імовірністю ДВВ повідомлення.

Приклад. Побудувати код Хаффмена джерела алфавіту для заданого віиразу. Визначити середню довжину коду. Закодувати вираз отриманим кодом. Обчислити довжину закодованого виразу і довжину для ASCII-коду виразу. Визначити коефіцієнт стискання.

Вираз	Код
навчай інших — і сам навчишся.	Код Хаффмена першого порядку

Побудуємо таблицю коду Хаффмена першого порядку. Для цього спочатку визначаємо частотність появи окремих символів a_i і задаємо їх як ймовірність p_i.

символ, a_i	Ймовірність, p_i	Код	l_i – довжина коду	$p_i l_i$
.	1/30	00000	5	5/30
-	1/30	00001	5	5/30
й	1/30	00010	5	5/30
м	1/30	01100	5	5/30
х	1/30	01101	5	5/30
Я	1/30	00011	5	5/30
С	2/30	1010	4	8/30
И	2/30	1001	4	8/30
Ш	2/30	0111	4	8/30
І	2/30	1000	4	8/30
Ч	2/30	110	3	6/30
В	2/30	1011	4	8/30
Н	3/30	111	3	9/30
А	4/30	001	3	12/30
_	5/30	010	3	15/30

Побудуємо кодове дерево Хаффмена (рис. 3.2).

Визначаємо кодові набори для окремих символів і записуємо їх у таблицю.

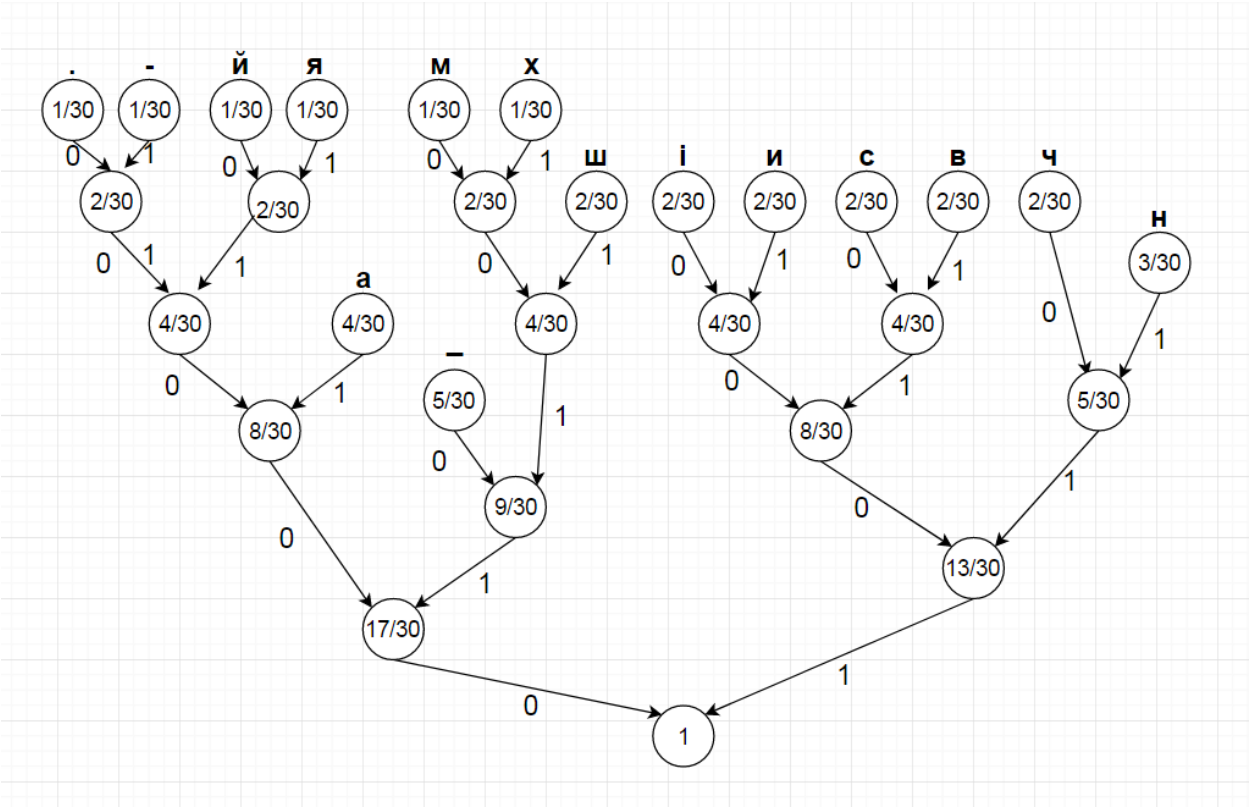


Рисунок 3.2 – Дерево Хаффмена

Обчислюємо середню довжину коду.

$$\overline{L} = \sum_{i=1}^{15} p_i * l_i = 6*5/30 + 5*8/30 + 6/30 + 9/30 + 12/30 + 15/30 = 112/30 = 3.733 біт./сим$$

Закодуємо наш вираз:

111 001 1011 110 001 00010 010 1000 111 0111 1001 01101 010 00001 010 1000 010 1010 001 01100 010 111 001 1011 110 1001 0111 1010 00011 00000

Довжина даного коду: 112 біт.

Довжина даного повідомлення в коді ASCII: 8*30=240 біт.

Коефіцієнт стискання k=240/112=2.14

3.2 Приклади розв'язування задач

3.2.1 Алгоритм Шеннона -Фано

Приклад Обчислити середні довжини кодів за алгоритмами Шеннона-Фано для дискретної випадкової величини X, заданої таким розподілом ймовірностей:

X	**1**	**2**	**5**	**6**	**7**
P	*0,2*	*0,1*	*0,3*	*0,25*	*0,15*

.

Розв'язання

Побудуємо таблицю кодів для дискретної випадкової величини (ДВВ) X за алгоритмом Шеннона-Фано.

Код Шеннона-Фано

Значення x_i	**Імовірність** $P(x_i)$	**Код** $Code(x_i)$	**Довжина коду** l_i	$p_i l_i$
5	***0,3***	**00**	2	0,6
6	***0,25***	**01**	2	0,5
1	***0,2***	**10**	2	0,4
7	***0,15***	**110**	3	0,45
2	***0,1***	**111**	3	0,3
				$\sum l_i \cdot p_i = 2,25$

Середня довжина отриманого коду $\overline{L(X)} = 2,25$ (біт/сим).

Приклад Для ансамблю повідомлень, заданого таким розподілом ймовірностей: 0,18; 0,17; 0,16; 0,15; 0,1; 0,08; 0,05; 0,05; 0,04; 0,02, побудувати двійкові коди Шеннона-Фано. Визначити основні характеристики кодів: ентропію, середню довжину, ефективність коду. надлишковість коду.

Розв'язання

Верхня границя компактності кодування повідомлень ДВВ визначається її ентропією

$$\overline{L_{\min}} = HX = -\sum_i p_i \log_2 p_i \approx 3{,}07 \text{ (біт/сим).}$$

Обчислимо середню довжину оптимальних кодів Шеннона-Фано для кодування даної ДВВ.

Побудуємо таблицю кодів Шеннона-Фано (таблиця 3.4).

Код Шеннона-Фано

Символ	**Імовірність p_i**	**Код Code(x_i)**	**Довжина коду l_i**	**$p_i l_i$**
a	0,18	**00**	2	0,36
b	0,17	**010**	3	0,51
c	0,16	**011**	3	0,48
d	0,15	**100**	3	0,45
e	0,1	**101**	3	0,3
f	0,08	**1100**	4	0,32
g	0,05	**1101**	4	0,2
h	0,05	**1110**	4	0,2
l	0,04	**11110**	5	0,2
m	0,02	**11111**	5	0,1
				$\Sigma p_i l_i$=3,12

Середня довжина даного коду: $\overline{L(X)} = \sum_i l_i p_i = 3{,}12$ (біт/сим).

Отже, ефективність коду $\chi = \frac{\overline{L_{\min}}}{\overline{L}} = \frac{3{,}07}{3{,}12} \approx 0{,}98$,

надлишковість коду $\rho_к = 1 - \chi \approx 1 - 0{,}98 = 0{,}02$.

3.2.2 Алгоритм Хаффмена

Приклад 2 Для ансамблю повідомлень, заданого таким розподілом ймовірностей: 0,18; 0,17; 0,16; 0,15; 0,1; 0,08; 0,05; 0,05; 0,04; 0,02, побудувати двійковий код Хаффмена. Визначити основні характеристики кодів: ентропію, середню довжину, ефективність коду. надлишковість коду.

Побудуємо кодове дерево за алгоритмом Хаффмена, використовуючи ймовірності заданої ДВВ (рис.3.3), і відповідно до кодового дерева побудуємо таблицю кодів

Код Хаффмена

	Імовірність p_i	**Код Code(x_i)**	**Довжина коду l_i**	**$p_i l_i$**
a	0,18	**11**	2	0,36
b	0,17	**000**	3	0,51
c	0,16	**001**	3	0,48
d	0,15	**010**	3	0,45
e	0,1	**100**	3	0,3
f	0,08	**0110**	4	0,32
g	0,05	**1010**	4	0,2
h	0,05	**1011**	4	0,2
l	0,04	**01110**	5	0,2
m	0,02	**01111**	5	0,1
				$\Sigma p_i l_i = 3,12$

Середня довжина коду $\overline{L(X)} = 3,12$ (біт/сим).

Отже, ефективність коду χ=0,98, надлишковість $\rho_к \approx 0,02$.

Для порівняння: при кодуванні даної ДВВ рівномірним двійковим кодом середня довжина коду $\overline{L(X)} = 2$ (біт/сим).

Ефективність коду

$$\chi = \frac{3,07}{4} \approx 0,77 .$$

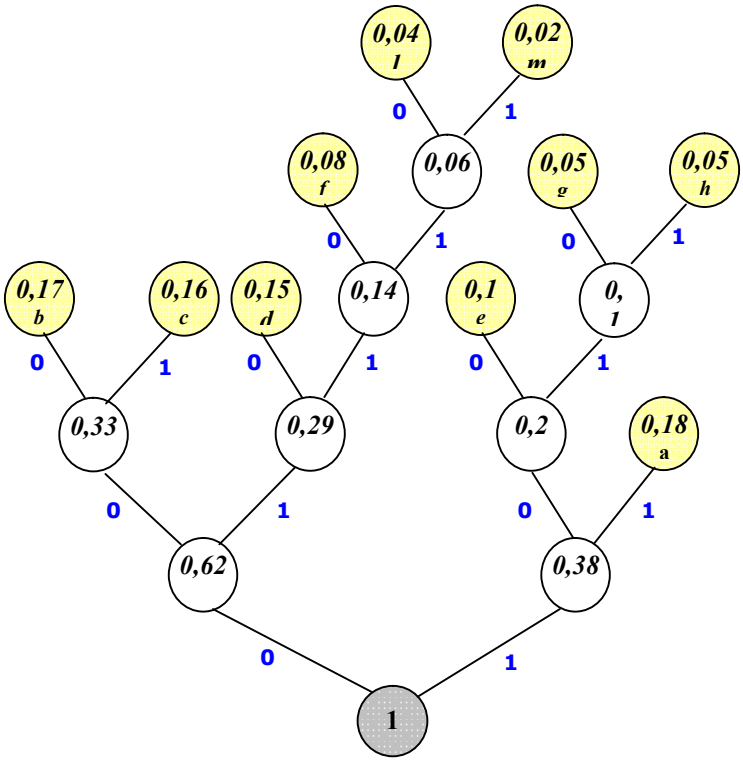


Рисунок 3.3 – Дерево Хаффмена

Надлишковість коду $\rho_к=1-\chi \approx 0,23$.

Приклад Обчислити середні довжини кодів за алгоритмами Шеннона-Фано, Хаффмена, арифметичним для дискретної випадкової величини X, заданої таким розподілом ймовірностей:

X	**1**	**2**	**5**	**6**	**7**
P	*0,2*	*0,1*	*0,3*	*0,25*	*0,15*

.

Розв'язання

Для ДВВ X, побудуємо кодове дерево (рис. 3.4) і відповідну таблицю кодів (таблиця 4.5) за алгоритмом Хаффмена.

Код Хаффмена

Значення x_i	**Імовірність** $P(x_i)$	**Код** $Code(x_i)$	**Довжина коду** l_i	$p_i \cdot l_i$
5	***0,3***	**00**	2	0,6
6	***0,25***	**01**	2	0,5
1	***0,2***	**11**	2	0,4
7	***0,15***	**100**	3	0,45
2	***0,1***	**101**	3	0,3
				$\sum l_i \cdot p_i$**=2,25**

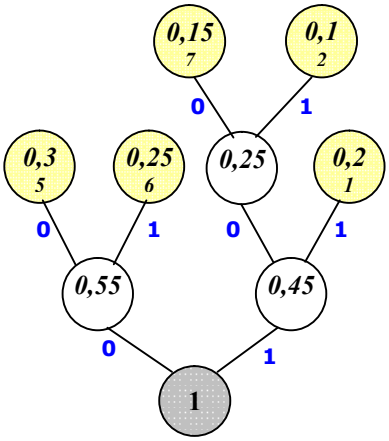


Рисунок 3.4 –Дерево Хаффмена

Середня довжина отриманого коду $\overline{L(X)} = 2{,}25$ (біт/сим).

ЛЕКЦІЯ 4 АРИФМЕТИЧНЕ КОДУВАННЯ

4.1 Основні поняття

4.1.1 Загальна характеристика арифметичного алгоритму

За розподілом ймовірностей ДВВ складається таблиця з пересічних в граничних точках відрізків для кожного із значень ДВВ Об'єднання цих відрізків утворює інтервал [0;1], а їхні довжини пропорційні ймовірностям значень ДВВ

4.1.2 Арифметичний алгоритм кодування

Алгоритм кодування полягає в побудові інтервалу, що однозначно визначає конкретну послідовність значень ДВВ Інтервали повідомлення будуються так. Якщо є відрізок повідомлення завдовжки n-1 символів, то для побудови відрізка повідомлення завдовжки n попередній інтервал розбивається на стільки частин, скільки можливих значень має ДВВ Для знаходження початку і кінця нового інтервалу повідомлення до початку попереднього інтервалу необхідно додати значення добутків його ширини на відповідні границі відрізка поточного нового символу з таблиці символів і їхніх інтервалів (таблиці кодера). З отриманих інтервалів вибирається той, що відповідає конкретному повідомленню завдовжки n символів.

Для побудованого таким чином інтервалу повідомлення знаходиться число, що належить цьому відрізку, як правило, це ціле число, розділене на мінімальний степінь 2. Це дійсне число і буде кодом даного повідомлення.

У міру надходження символів повідомлення його інтервал звужується, відповідно кількість розрядів, необхідна для подання інтервалу збільшується. Більш імовірні символи меншою мірою звужують інтервал, ніж менш імовірні, і, отже, додають менше розрядів до результату.

Так для слова «МАТЕМАТИКА» частоти символів і відрізки, довжини яких визначають імовірністі появи символів, відповідають даним таблиці 4.1.

Таблиця 4.1 – Імовірності появи і інтервали символів

Символ	***Імовірність***	***Інтервал***
М	0,2	***[0; 0,2)***
А	0,3	***[0,2; 0,5)***
Т	0,2	***[0,5; 0,7)***
Е	0,1	***[0,7; 0,8)***
И	0,1	***[0,8; 0,9)***
К	0,1	***[0,9; 1,0)***

Для першого символу «М» кодер звужує початковий інтервал [0; 1) до нового [0; 0,2). Наступний символ «А» кодується підінтервалом [0,04;0,1): lowi=0+0,2*0,2=0,04; highi=0+0,2*0,5=0,1).

Букві «Т» відповідає інтервал [0,5; 0,7), а новий інтервал буде [0,07; 0,082) (lowi=0,04+0,06*0,5=0,07; highi=0,04+0,06*0,7=0,082).

Послідовність інтервалів кодування повідомлення «МАТЕМАТИКА» наведена в таблиці 4.2.

Таблиця 4.2 – Кодування повідомлення

Символ – інтервал	***Інтервал повідомлення***	***Ширина інтервалу***
М - ***[0; 0,2)***	***[0; 0,2)***	**0,2**
А - ***[0,2; 0,5)***	***[0,04; 0,1)***	**0,06**
Т - ***[0,5; 0,7)***	***[0,07; 0,082)***	**0,012**
Е - ***[0,7; 0,8)***	***[0,0784; 0,0796)***	**0,0012**
М - ***[0; 0,2)***	***[0,0784; 0,07864)***	**0,00024**
А - ***[0,2; 0,5)***	***[0,078448; 0,07852)***	$\mathbf{0{,}72 \cdot 10^{-4}}$
Т - ***[0,5; 0,7)***	***[0,078484; 0,0784984)***	$\mathbf{0{,}144 \cdot 10^{-4}}$
И - ***[0,8; 0,9)***	***[0,07849552; 0,07849696)***	$\mathbf{0{,}144 \cdot 10^{-5}}$
К - ***[0,9; 1,0)***	***[0,078496816; 0,07849696)***	$\mathbf{0{,}144 \cdot 10^{-6}}$
А - ***[0,2; 0,5)***	***[0,0784968448; 0,078496888)***	$\mathbf{0{,}432 \cdot 10^{-7}}$

З результату кодування повідомлення «МАТЕМАТИКАотримаємо інтервал [0,0784968448; 0,078496888). Для цього числа знаходять мінімальний степінь 2 - це $0{,}07849687 = 1316959/2^{24}$, яка визначає 24 - розрядний код повідомлення.

Двійковий 24-розрядний код числа 131625910 = =000101000001100001011111 2 і є арифметичним кодом даного повідомлення: Code(МАТЕМАТИКА)=000101000001100001011111.

Довжина коду L(X)=24 біт.

Середня довжина коду $\overline{L(X)} = 24/10 = 2{,}4$ (біт/сим).

Арифметичний код можна будувати для повідомлень алфавіту. Нехай ДВВ може набувати значень 0 і 1 з ймовірностями відповідно 2/3 і 1/3. Побудуємо арифметичний код для повідомлень завдовжки 3 символи.

Коди за арифметичним алгоритмом для всіх можливих повідомлень завдовжки 3 символи для заданої ДВВ подаються таблицею 4.3.

Таблиця 4.3 – Арифметичне кодування послідовності символів

Повідомлення та їх інтервали						**Код**	p_i
1	**[2/3; 1)**	**11**	**[8/9; 1)**	**111**	**[26/27; 1] э 31/32 →**	**11111**	*1/27*
				110	**[8/9; 26/27] э 15/16 →**	**1111**	*2/27*
		10	**[2/3; 8/9)**	**101**	**[22/27; 8/9] э 7/8 →**	**111**	*2/27*
				100	**[2/3; 22/27] э 3/4 →**	**11**	*4/27*
0	**[0; 2/3)**	**01**	**[4/9; 2/3)**	**101**	**[16/27; 2/3] э 5/8 →**	**101**	*2/27*
				100	**[4/9; 16/27] э 1/2 →**	**1**	*4/27*
		00	**[0; 4/9]**	**001**	**[8/27; 4/9] э 3/8 →**	**011**	*4/27*
				000	**[0; 8/27] э 1/4 →**	**01**	*8/27*

Середня довжина коду

$$\overline{L(X)} = \frac{1}{n}\sum_{i=1}^{k} L(\vec{X}_i)\cdot p_i = \frac{1}{81}(5\cdot 1+4\cdot 2+3\cdot 2+2\cdot 4+3\cdot 2+1\cdot 4+3\cdot 4+2\cdot 8) = \frac{65}{81} \approx 0{,}8025$$

(біт/сим).

4.1.3 Арифметичний алгоритм декодування

Декодеру, як і кодеру, відома таблиця розподілу інтервалів символів алфавіту. Декодування арифметичного коду повідомлення здійснюється за таким алгоритмом:

Крок 1 За таблицею інтервалів символів алфавіту визначається відрізок, що містить значення поточного коду, і за цим інтервалом з тієї самої таблиці однозначно визначається символ повідомлення. Якщо це маркер кінця повідомлення, то кінець, інакше - перехід до кроку 2.

Крок 2 Від поточного коду віднімається нижня границя його інтервалу. Різниця ділиться на довжину цього інтервалу. Отримане значення вважається новим значенням поточного коду. Перехід до кроку 1.

Наприклад, арифметичний код повідомлення з 10 символів є $000101000001100001011111_2 = 1316259_{10}$, а символи і інтервали повідомлення наведені у таблиці 4.1. Розкодуємо це повідомлення.

Дійсне число, яке належить інтервалу, що однозначно визначає закодоване повідомлення $1316959 / 2^{24} =$ 0,07849687. Це число є значенням поточного коду. З таблиці 4.1 визначають відрізок, якому належить це число: [0; 0,2). Тому перший закодований символ є «М».

Віднімемо від поточного коду нижню границю інтервалу розкодованого символу «М» і розділимо отриманий результат на ширину цього інтервалу: $\frac{0,07849687 - 0}{0,2} = 0,39248435$. Це число належить відрізку [0,2;0,5), що відповідає символу «А».

Виключимо з отриманого інтервалу [0,2; 0,5) вплив букви «А» відніманням нижньої границі цього інтервалу і розділимо на його ширину: $\frac{0,39248435 - 0,2}{0,3} = 0,6416145$.

Результат належить відрізку [0,5;0,7) символу «Т» тощо. Тому на виході декодера отримаємо таке повідомлення .

Декодоване число	Символ на виході	Інтервал	Ширина інтервалу
0,07849687	*М*	[0; 0,2)	***0,2***
0,39248435	*А*	[0,2; 0,5)	***0,3***
0,6416145	*Т*	[0,5; 0,7)	***0,2***
0,7080725	*Е*	[0,7; 0,8)	***0,1***
0,080725	*М*	[0; 0,2)	***0,2***
0,403625	*А*	[0,2; 0,5)	***0,3***
0,67875	*Т*	[0,5; 0,7)	***0,2***
0,89375	*И*	[0,8; 0,9)	***0,1***
0.9375	*К*	[0,9; 1,0)	***0,1***
0.375	*А*	[0,2; 0,5)	***0,3***

4.2 Приклади розв'язування задач

Арифметичний метод

Приклад Обчислити середні довжини кодів за арифметичним алгоритмом для дискретної випадкової величини X, заданої таким розподілом ймовірностей:

X	**1**	**2**	**5**	**6**	**7**
P	*0,2*	*0,1*	*0,3*	*0,25*	*0,15*

.

Побудуємо інтервалів і кодів для даної ДВВ X.

Таблиця арифметичного коду

Значення x_i	**Імовірність p_i**	**Інтервал**	**Число інтервалу**	**Код**	**Довжина коду l_i**	$l_i p_i$
1	***0,2***	**[0,8; 1)**	**7/8∈[0,8;1)**	**111**	3	0,6
2	***0,1***	**[0,7; 0,8)**	**3/4∈[0,7;0,8)**	**11**	2	0,2
5	***0,3***	**[0,4; 0,7)**	**1/2∈[0,4;0,7)**	**1**	1	0,3
6	***0,25***	**[0,15; 0,4)**	**1/4∈[0,15;0,4)**	**01**	2	0,5
7	***0,15***	**[0; 0,15)**	**1/8∈[0;0,15)**	**001**	3	0,45
					$\sum l_i p_i$=2,05	

Обчислимо середню довжину арифметичного коду

$$\overline{L(X)} = 2{,}05 \text{ (біт/сим).}$$

Приклад Закодувати за арифметичним алгоритмом повідомлення BAABCB, отримане від дискретної випадкової величини X, заданої таким розподілом ймовірностей: P(X=A)=1/3; P(X=B)=7/15; P(X=C)=1/5.

Розв'язання

Побудуємо таблицю 4.7 символів і відповідних їм інтервалів.

Таблицю Символи і їх інтервали

Символ	Імовірність	Інтервал
A	*1/3*	**[2/3; 1)**
B	*7/15*	**[1/5; 2/3)**
C	*1/5*	**[0; 1/5)**

Ця таблиця зберігається разом із кодом стисненого повідомлення і призначена для кодування й декодування за арифметичним алгоритмом.

Процес кодування повідомлення BAABCB зручно подати у вигляді такої таблиці 4.8.

Таблиця 4.8 – Кодування повідомлення

Повідомлення	Інтервал	Ширина інтервалу
	$[0;1)$	1
B	$\left[\frac{1}{5};\frac{2}{3}\right)$	$\frac{7}{15}$
A	$\left[\frac{23}{45};\frac{2}{3}\right)$	$\frac{7}{45}$
A	$\left[\frac{83}{135};\frac{2}{3}\right)$	$\frac{7}{135}$
B	$\left[\frac{422}{675};\frac{263}{405}\right)$	$\frac{49}{2025}$
C	$\left[\frac{422}{675};\frac{6379}{10125}\right)$	$\frac{49}{10125}$
B	$\left[\frac{31699}{50625};\frac{19088}{30375}\right)$	$\frac{343}{151875}$

Повідомлення BAABCB однозначно визначає інтервал $\left[\frac{31699}{50625};\frac{19088}{30375}\right)\approx[0{,}626;\,0{,}628)$. Знайдемо дійсне число, що належить цьому інтервалу і є часткою від ділення цілого додатного числа на мінімальний степінь 2.

Таке число

$$\frac{321}{2^9}\in\left[\frac{31699}{50625};\frac{19088}{30375}\right).$$

Двійкове подання чисельника буде арифметичним кодом повідомлення. Розрядність коду визначається степенем 2.

Отже, знайдемо двійковий 9-розрядний код числа 321: 32110=1010000012. Таким чином, арифметичний код заданого повідомлення: Code(BAABCB)=101000001. Довжина коду L(X)=9 біт.

Приклад 3 Декодувати повідомлення довжиною 5 символів за арифметичним алгоритмом. Код повідомлення 010001011.

Таблиця символів і відповідних їм інтервалів така:

Таблицю 4.9 - Символи і їх інтервали

Символ	Імовірність	Інтервал
C	*1/4*	**[3/4; 1)**
B	*1/2*	**[1/4; 3/4)**
A	*1/4*	**[0; 1/4)**

Розв'язання

Заданий арифметичний код повідомлення 010001011 – це 9-розрядна двійкова комбінація, що визначає число, яке належить відрізку закодованого повідомлення.

Знайдемо це число: 0100010112 = 13910.

Число $\frac{139}{2^9} \approx 0,272$ є поточним кодом повідомлення. Воно належить інтервалу, який визначає перший символ повідомлення 0,272 $\in$ [1/4; 3/4) - це інтервал символу «B».

Для отримання наступного значення поточного коду від попереднього його значення віднімають нижню границю інтервалу розкодованого символу і ділять. Після визначення символу міняємо інтервал і т.і. Процес декодування повідомлення показано у таблиці 4.10.

Таблиця 4.10 – Декодування повідомлення

Поточний код повідомлення	**Символ**	**Інтервал**	**Ширина інтервалу**
0,272	***B***	**[1/4; 3/4)**	*1/2*
$\frac{0,272-1/4}{1/2}$ **=0,043**	***A***	**[0; 1/4)**	*1/4*
$\frac{0,043}{1/4}$ **=0,172**	***A***	**[0; 1/4)**	*1/4*
$\frac{0,172}{1/4}$ **=0,688**	***B***	**[1/4; 3/4)**	*1/2*
$\frac{0,688-1/4}{1/2}$ **=0,876**	***C***	**[3/4; 1)**	*1/4*

ЛЕКЦІЯ 5 АДАПТИВНИЙ АЛГОРИТМ ХАФФМЕНА

5.1 Основні поняття

5.1.1 Загальна характеристика

В адаптивному (динамічному) алгоритмі Хаффмена таблиця кодів не передається. У процесі кодування залежно від значення поточного символу, що надходять на вхід алгоритму, кодове дерево коригується відповідно до зміни статистики вхідного потоку. При декодуванні відбувається той самий процес. Для однозначності декодування використовується упорядкована структура кодового дерева.

Упорядкованим деревом Хаффмана називається бінарне дерево, вузли якого можуть бути перелічені у порядку неубування їх ваги зліва-направо на кожному рівні і знизу-вверх за рівнями.

При зміні ваги існуючого вузла в дереві достатньо поміняти місцями два вузли: вузол, що порушив упорядкованість, і останній з наступних за ним, вузлів меншої ваги. Після обміну вузлів місцями необхідно перерахувати вагу всіх вузлів.

5.1.2 Принципи побудова адаптивного дерева Хаффмена

На початку роботи алгоритму дерево містить тільки один спеціальний символ <ESC>, що завжди має частоту 0. Він необхідний для занесення в кодове дерево нового символу, що передається безпосередньо після <ESC>. При появі нового символу праворуч від вузла <ESC> додається лист і потім,

якщо необхідно, дерево упорядковується. Ліві гілки кодового дерева позначаються 0, а праві - 1.

Так при додаванні у дерево Хаффмена рис. 5.1а двох букв необхідно поміняти місцями вузли «A» і «D» (рис. 5.1б). При додаванні двох нових букв «A» потрібно поміняти місцями спочатку вузол «A» та вузол - батько «D» і «B», а потім вузол «E» і вузол - брат «E» (рис. 5.1 в-г).

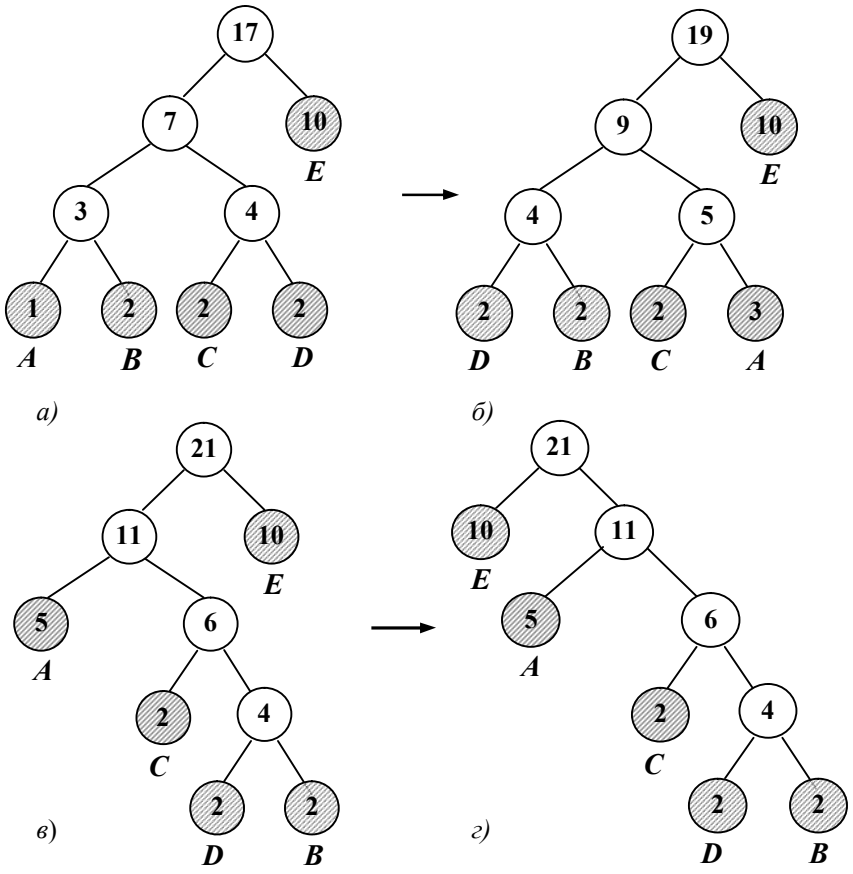


Рисунок 5.1 – Коригування дерева Хаффмена

5.2 Приклади розв'язування задач

5.2.1 Кодування за адаптивним алгоритмом Хаффмена

Приклад

Для заданого виразу spes vana побудувати таблицю кодування (закодувати) повідомлення виразу. Обчислити ентропію джерела. Виконати декодування закодованого виразу.

Побудувати дерево кодування виразу індивідуального завдання адаптивним кодом Хаффмена. Закодувати заданий вираз отриманим кодом Хаффмена.

Обчислити довжини заданого за індивідуальним завданням виразу в ASCII-кодах, закодованого виразу за заданим кодомі кодом Хаффмена.

Порівняти довжини виразу в ASCII-кодах і закодованого виразу . кодом Хаффмена.

Spes vana	Марна надія.	Адаптивний алгоритм Хаффмена

Розв'язання

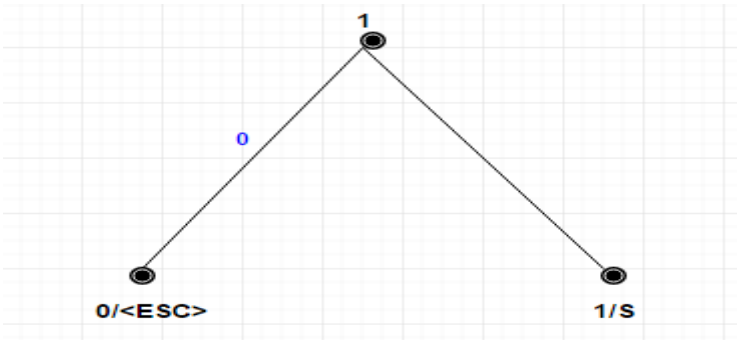
Процес кодування повідомлення і відповідні зміни кодового дерева подаються в таблиці 5.1 і на рис. 5.2.

Таблиця 5.1 – Опис потоку кодування вхідних символів

Вхідні дані	Код	Довжина коду	Номер дерева
S	‘S’	8	1
P	0’P’	9	2
E	00’E’	10	3
S	0	1	4
_	100’_’	11	5
V	1100’V’	12	6
A	1000’A’	12	7
N	0100’N’	12	8
A	001	3	9

Побудуємо дерева для кожного з послідовності символів і визначимо їх довжини.

1.



2.

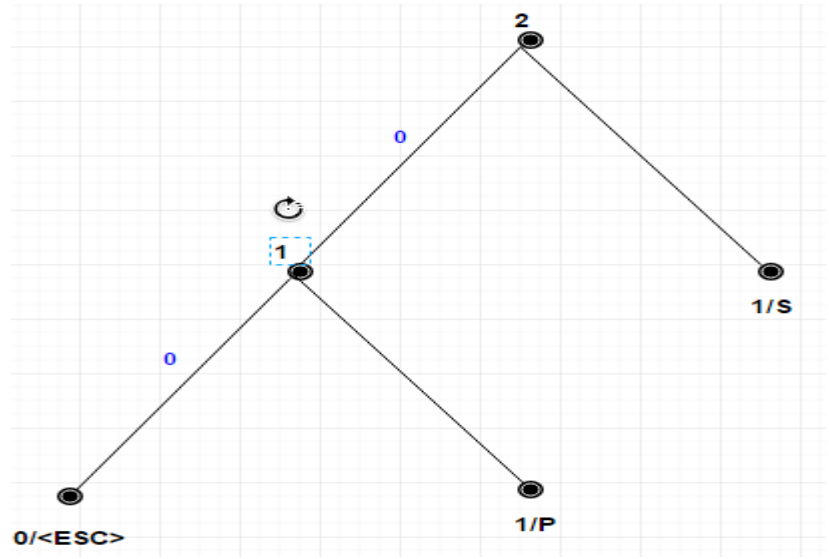
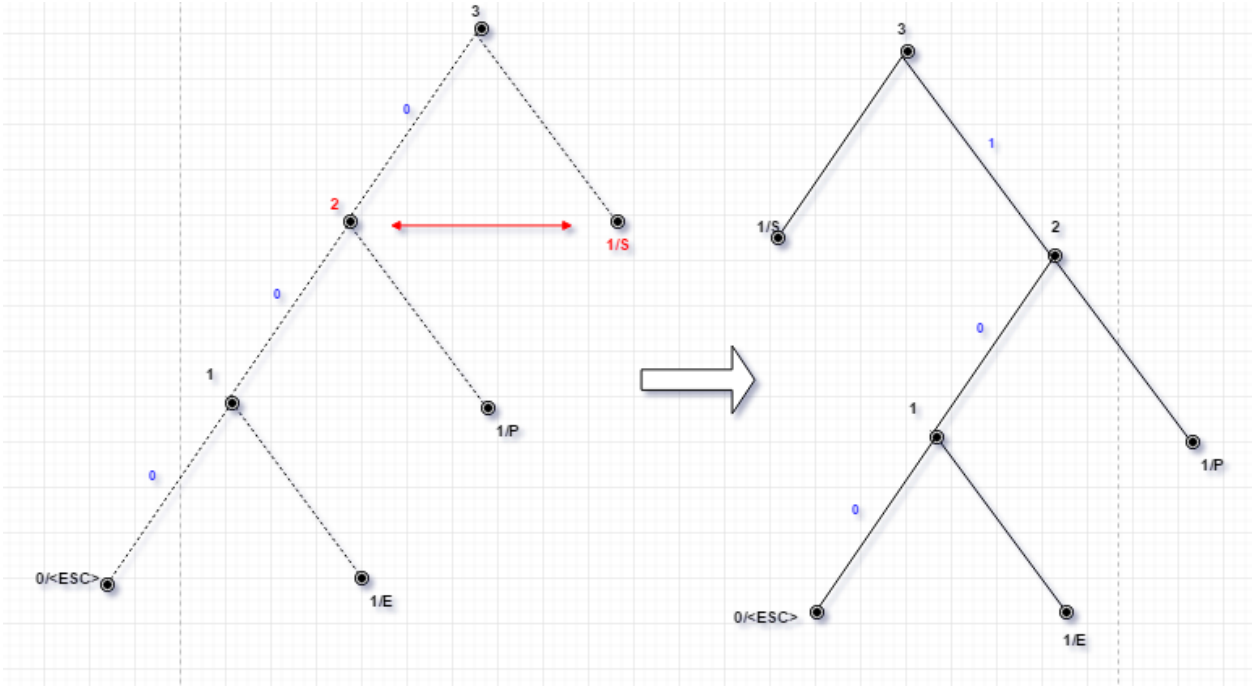
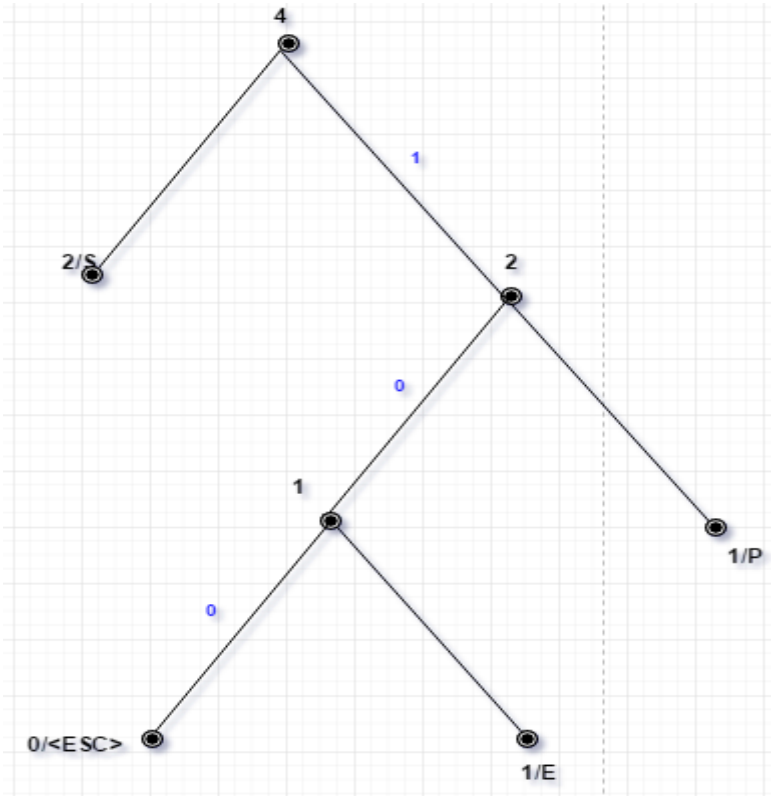


Рисунок 5.2 – Дерева кодування Хаффмена

3.

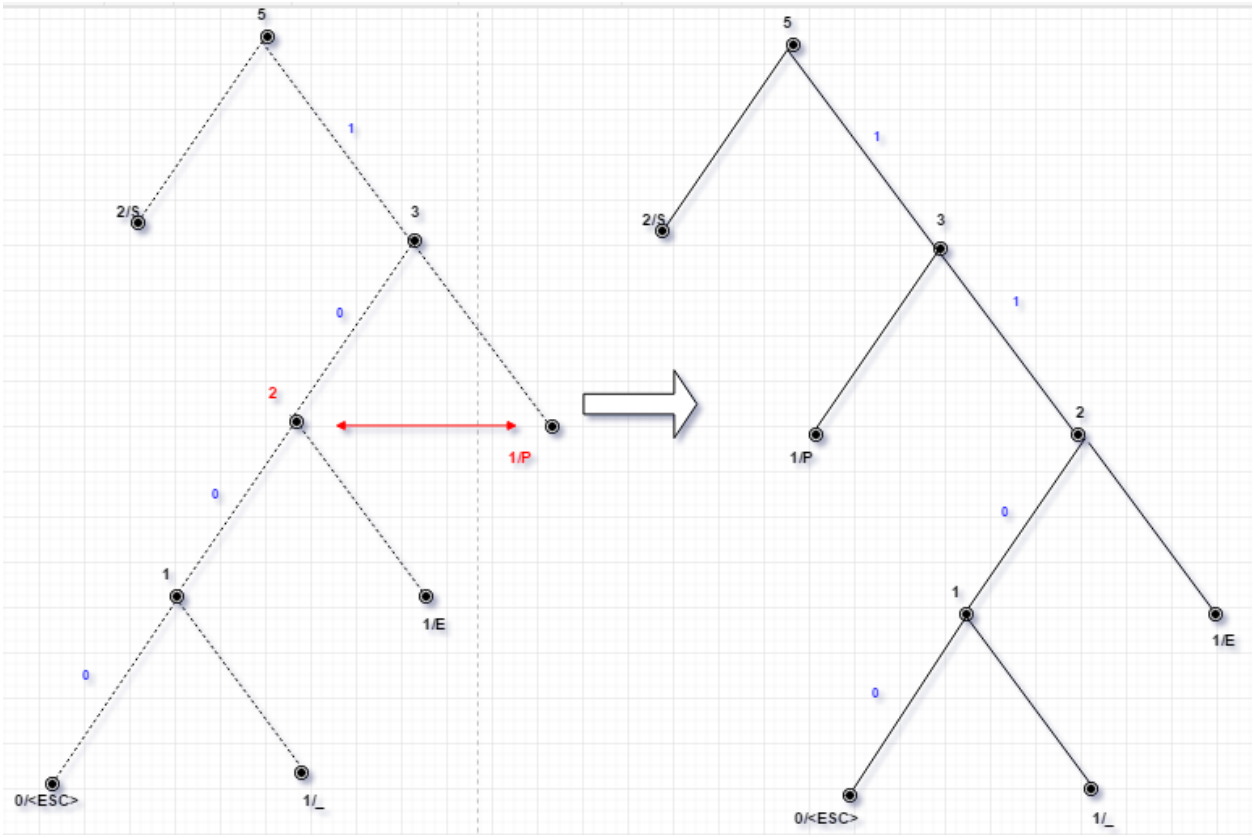


4.

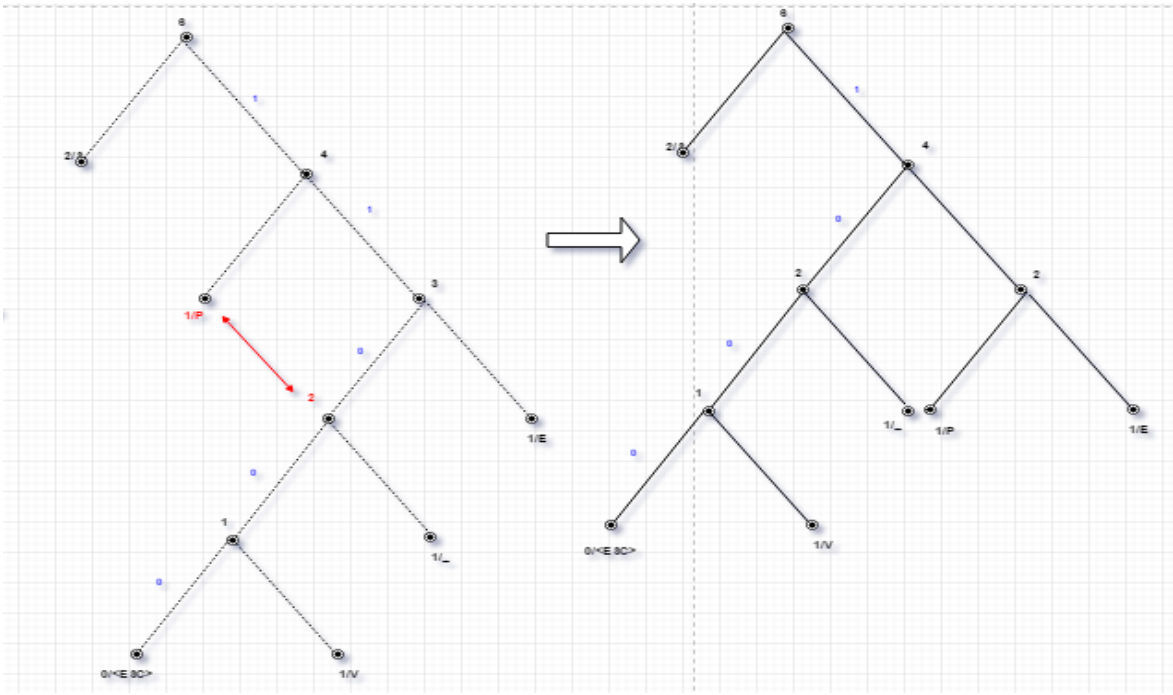


Продовження 1 рисунку 5.2

5.

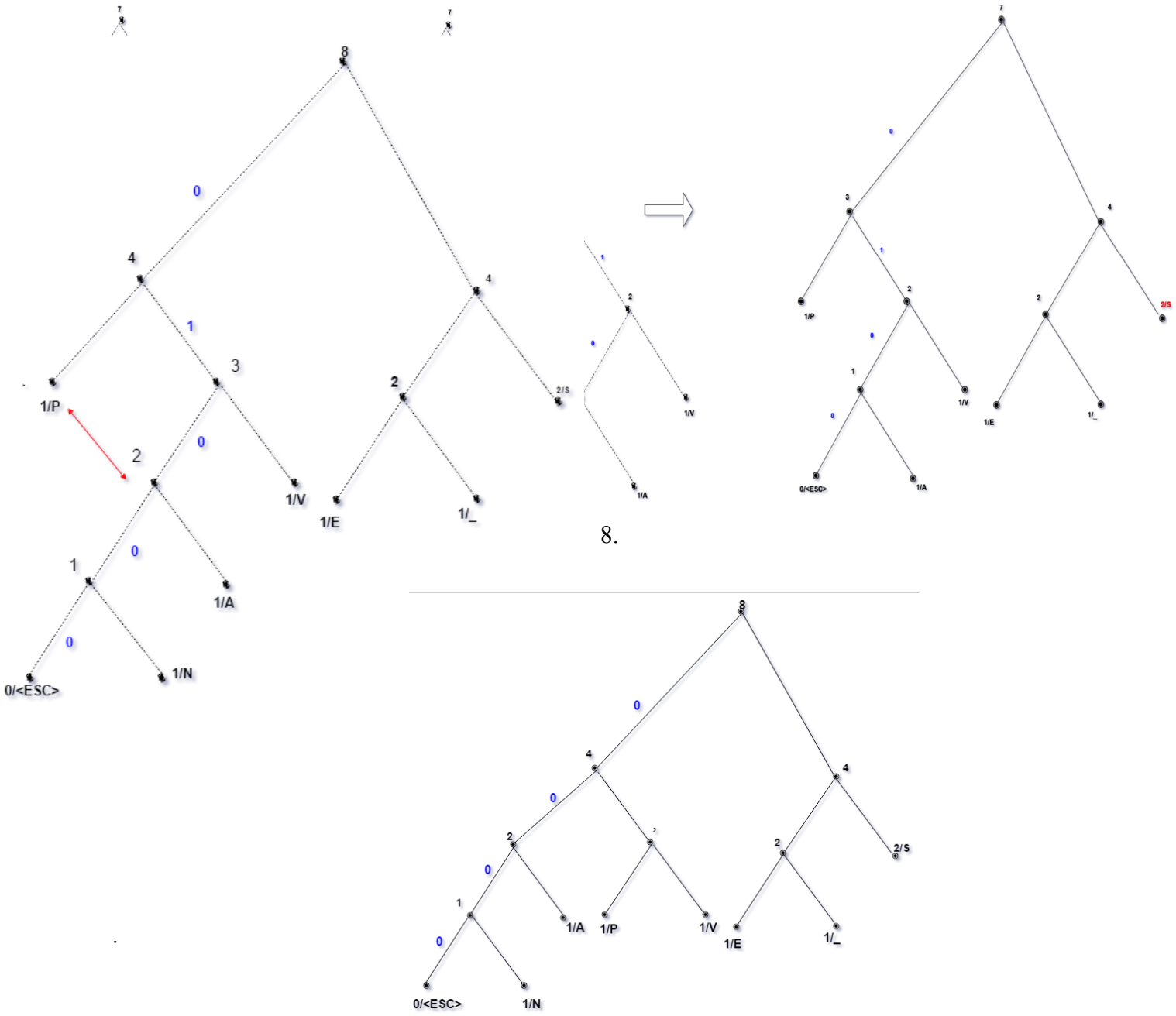


6.



Продовження 2 рисунку 5.2

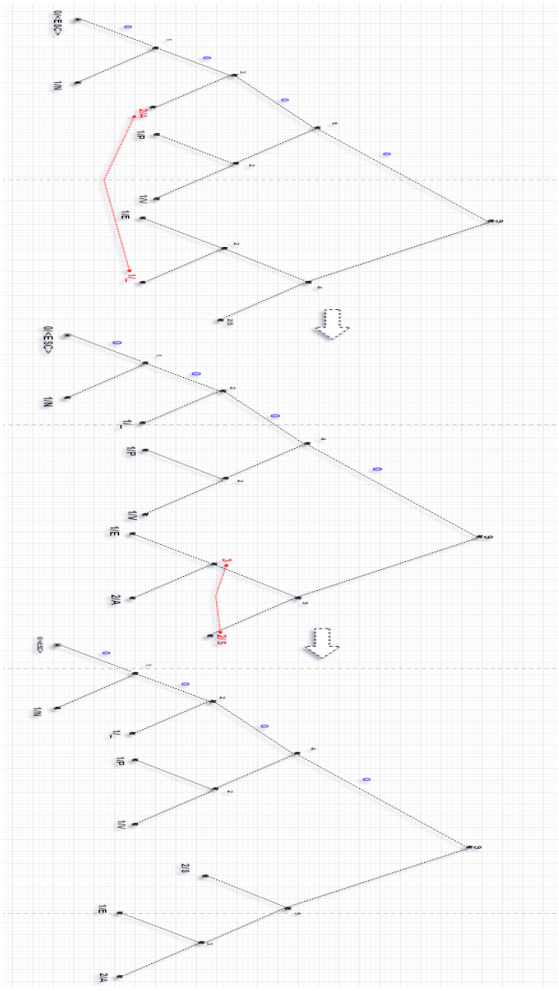
7.



8.

Продовження 3 рисунку 5.2

9)



Продовження 4 рисунку 5.2

Повідомлення кодується безпосередньо послідовністю кодів для кожного символу.

Закодоване повідомлення :

‘S’0’P’00’E’0100’_’1100’V’1000’A’0100”n”001

5.2.2 Декодування за адаптивним алгоритмом Хаффмена

Приклад Декодувати повідомлення

‘S’0’P’00’E’0100’_’1100’V’1000’A’0100”n”001,

закодоване за адаптивним алгоритмом Хаффмана.

Декодування відбувається безпосередньо за заданим кодом за допомогою побудови дерев за правилами:

1.Елемети вхідного повідомлення зчитуються побітно

2.Кожний раз при зчитуванні полсідовностеей 0 та 1 відбувається переміщення від кореня вниз по відповідній гілці бінарного дерева Хаффмана, до того часу, поки не буде досягнутий будь який, так званий листок дерева.

3.Якщ досягнутий листок, відповідний символ записується у вихідне повідомлення. Вага листка збільшується на один, вага вузлів-предків корегується ,дерево за необхідністю впорядковується.

4.Якщо ж досягнутий escape-символ, то у дерево записується новий символ, вага вузлів-предків корегуються, потім за необхідністю відбувається його упорядкування.

На початку декодування дерево Хаффмана містить тільки esc символ з частотою 0.3 розкодуванням кожного нового символу дерево перебудовується, а на виході декодера отримуємо кодову послідовність.

Процес кодування повідомлення і відповідні зміни кодового дерева подаються в таблиці 5.2 і на рис. 5.3.

Побудуємо дерева для кожного з послідовності символів і визначимо їх довжини.

1.

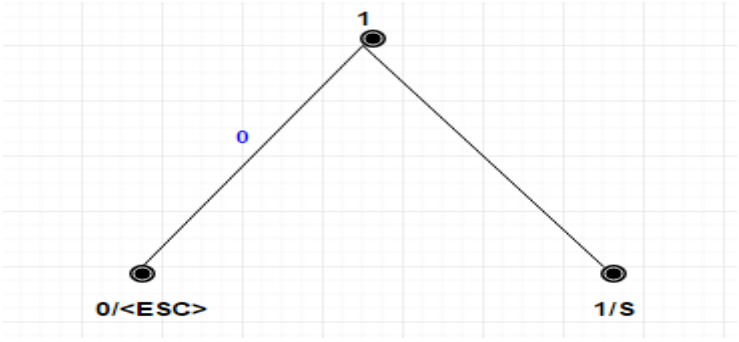
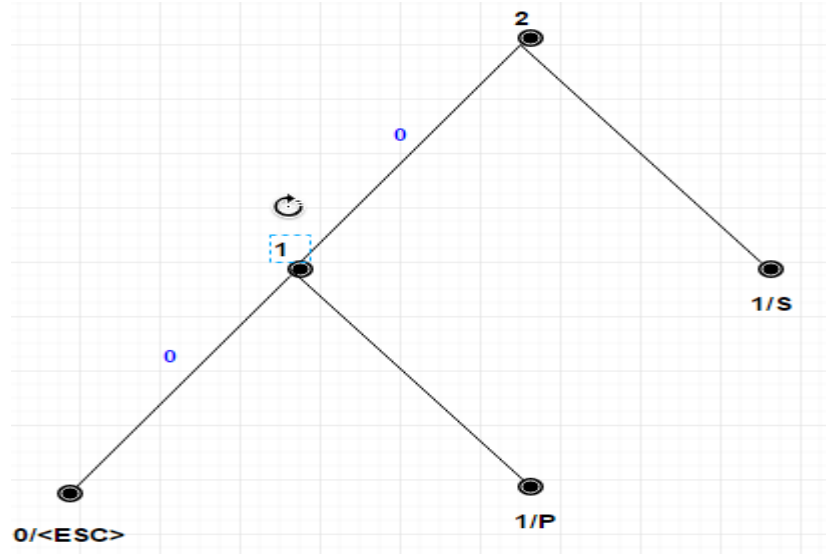
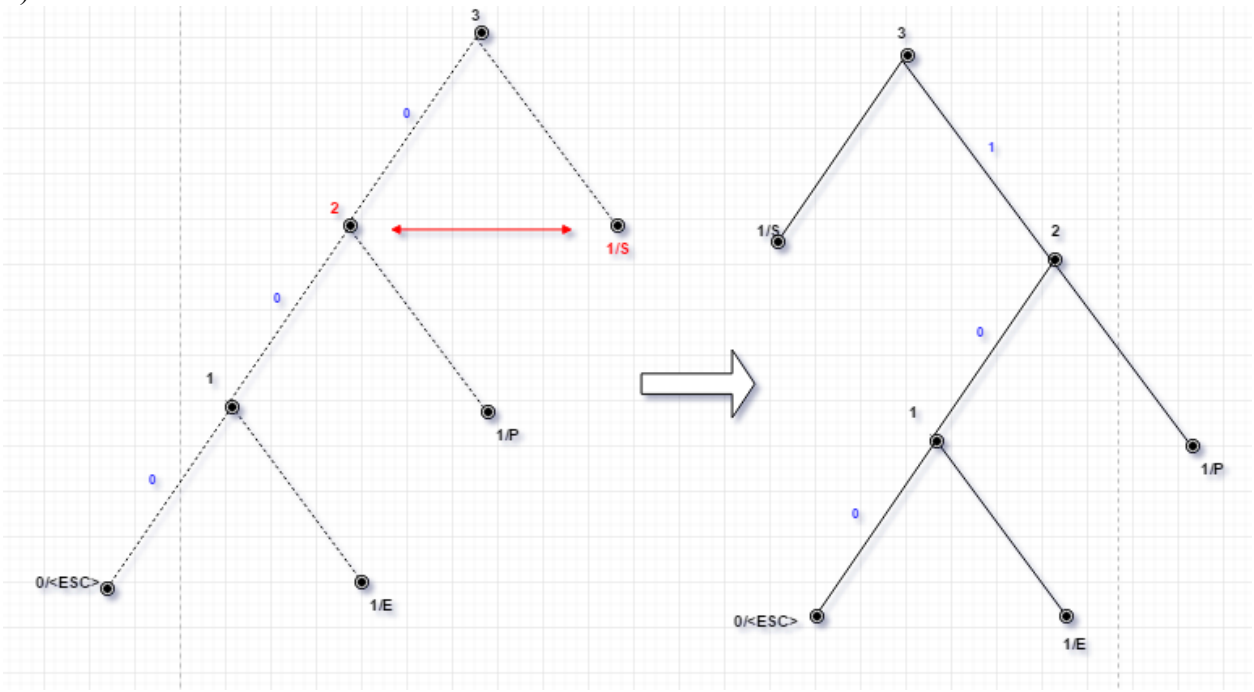


Рисунок 5.3 – Дерева декодування Хаффмена

2.

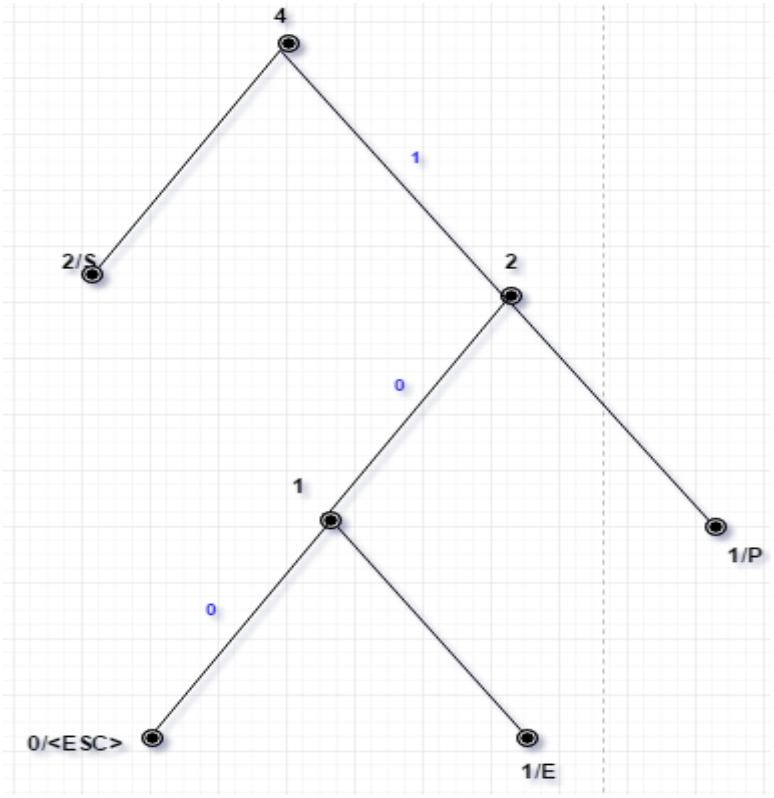


3).



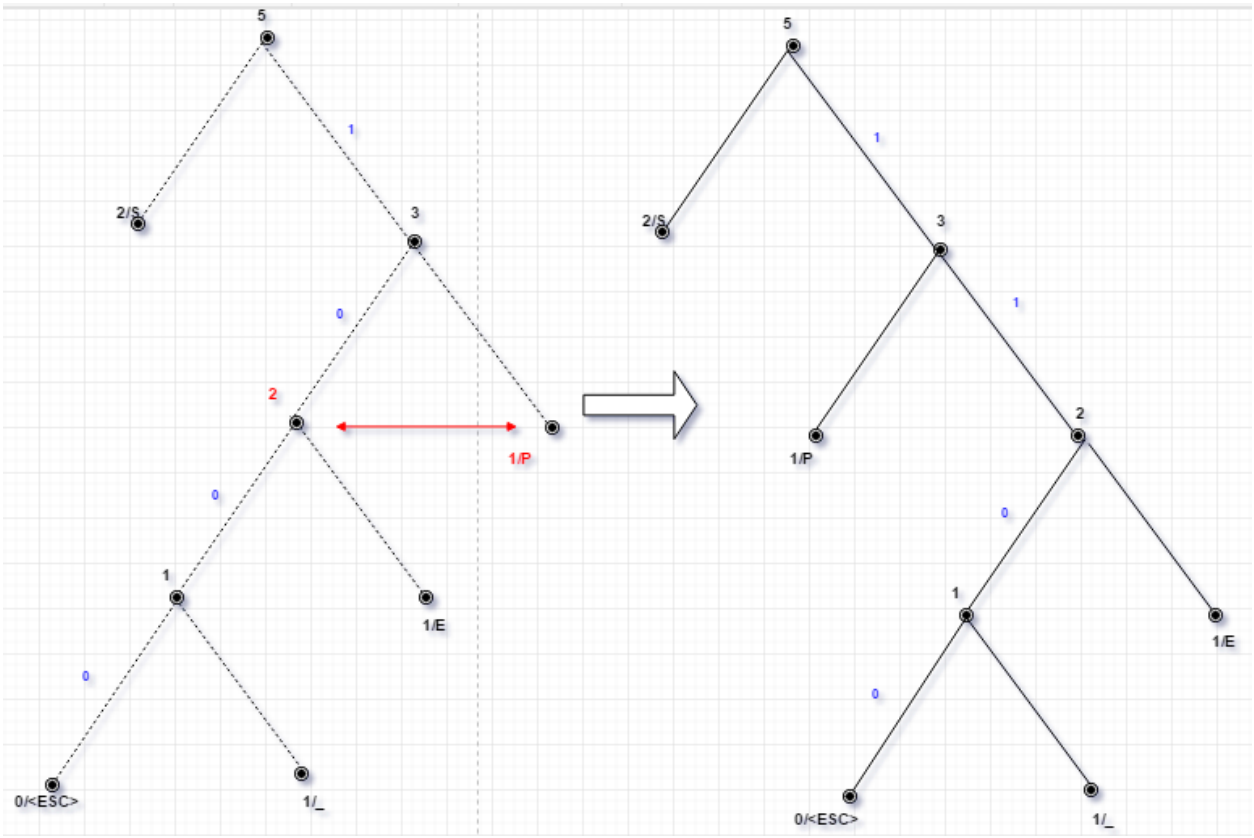
Продовження 1 рисунку 5.3

4.

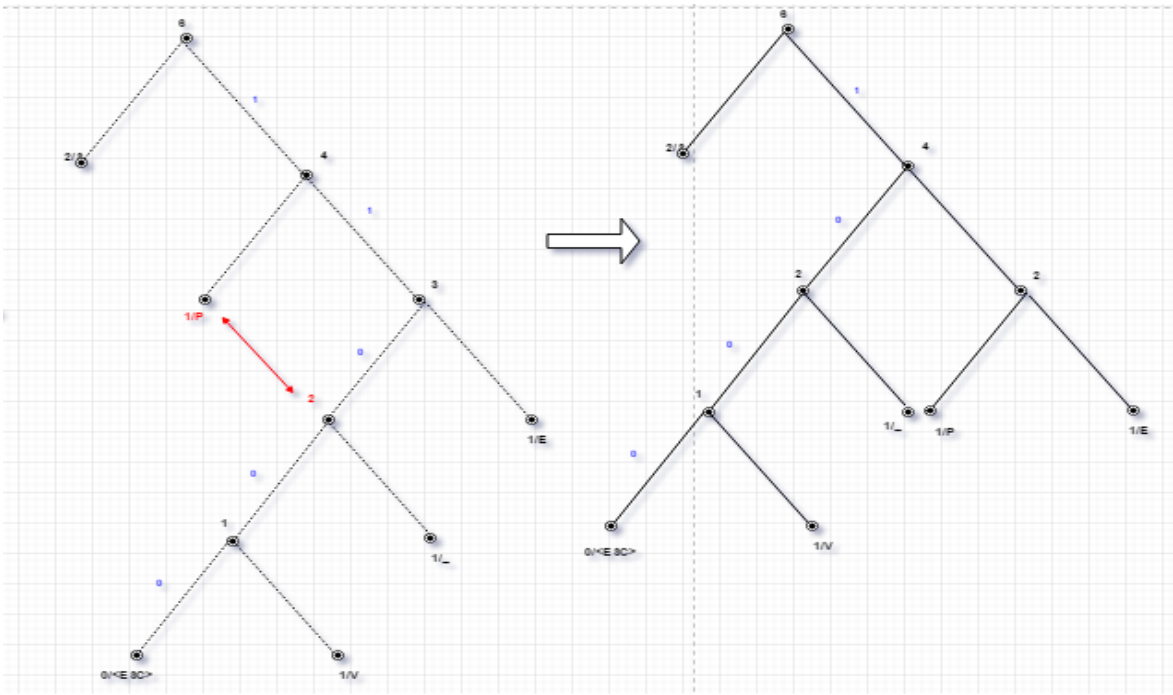


Продовження 2 рисунку 5.3

5.

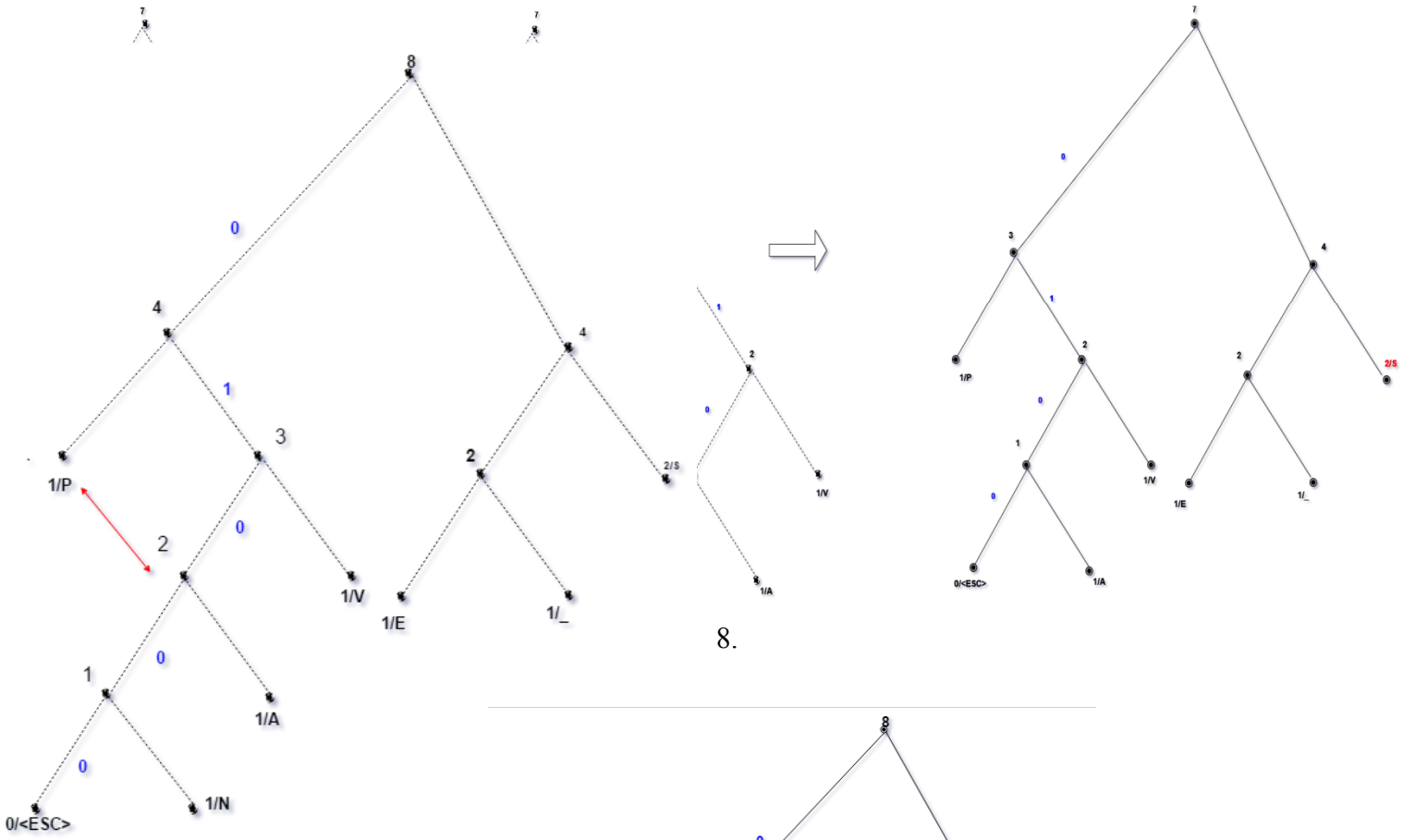


6.

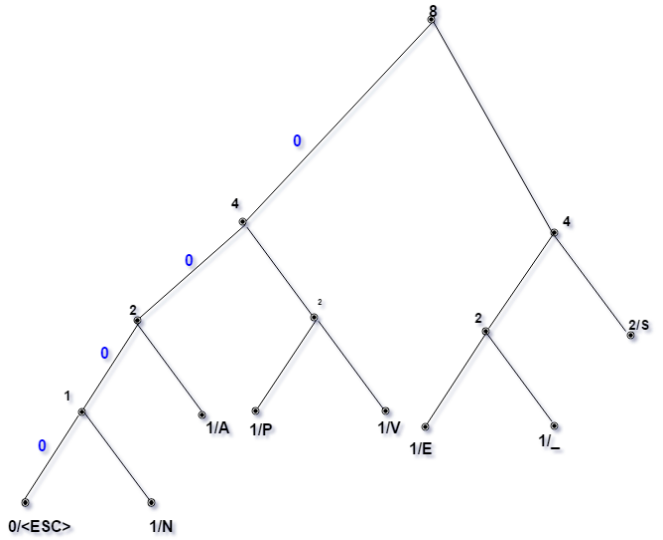


Продовження 3 рисунку 5.3

7.

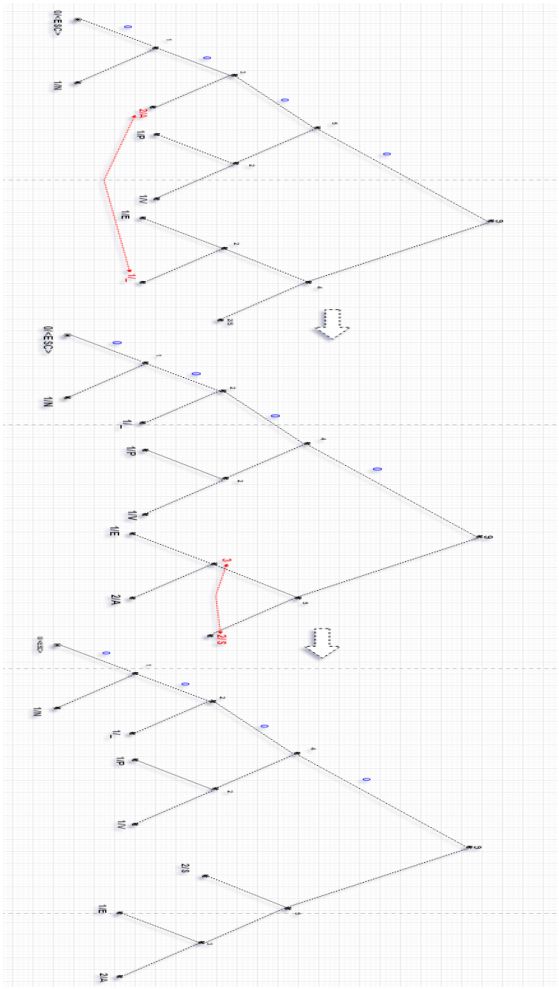


8.



Продовження 4 рисунку 5.3

9)



Продовження 5 рисунку 5.3

Таблиця 5.2 – Опис потоку декодування вхідних символів

Вхідні дані	Код	Довжина коду	Номер дерева
‘S’	S	8	1
0’P’	P	9	2
00’E’	E	10	3
0	S	1	4
100’_’	_	11	5
1100’V’	V	12	6
1000’A’	A	12	7
0100’N’	N	12	8
001	A	3	9

ЛЕКЦІЯ 6 БЛОКОВІ СТАТИСТИЧНІ АЛГОРИТМИ СТИСНЕННЯ ІНФОРМАЦІЇ

6.1 Основні поняття

6.1.1 Границя стиснення

За Шенноном верхня границя стиснення інформації є ентропію джерела H(X). Середнє значення обчислюється як математичне сподівання:

$$\overline{L(X)} = \sum_{i=1}^{k} p(x_i) L(X_i) .$$

$$L(X)=len(code(X_i)),$$

де code(X_i) значенню X_i ставить у відповідність деякий бітовий код;

len() - повертає довжину цього коду.

Наслідком теореми Шеннона про кодування джерела у відсутності завад є

$$\overline{L(X)} \geq HX$$

для будь-якої ДВВ X і будь-якого її коду.

Для вектору даних завдовжки n $\vec{X} = (x_1, x_2, \ldots, x_n)$ з . вектором частот символів у $\vec{X}$ $\vec{F} = (F_1, F_2, \ldots, F_k)$ середня кількість біт коду на одиницю повідомлення $\vec{X}$ обчислюється як

$$\overline{L(X)} = \frac{L(\vec{X})}{n} = \frac{1}{n} \sum_{i=1}^{k} F_i L_i ,$$

де $L(\vec{X})$ - довжина коду повідомлення $\vec{X}$;

$L(\vec{X}) = len(code(\vec{X}))$ $(L_1, L_2, \ldots, L_k)$ - вектор Крафта для $\vec{X}$

.

Ентропія повідомлення $\vec{X}$ обчислюється за формулою

$$HX \cong \frac{1}{n}\sum_{i=1}^{k} F_i \log_2 \frac{n}{F_i}.$$

Якщо вектор Крафта $(L_1^*, L_2^*, \ldots, L_k^*)$ пов'язан з частотами символів за

$$L_I^* \cong -\log_2\left(\frac{F_i}{n}\right).$$

то $\overline{L(X)} = HX$

Границі стиснення інформації при оптимальному статистичному кодуванні визначаються так:

$$HX \le \overline{L(X)} \le HX + 1.$$

6.1.2 Блоковий код k-го порядку

Блоковий код розділяє вектор даних на блоки певної довжини, і потім кожний блок замінює кодовим словом з префіксної множини кодових слів. Отриману послідовність кодових слів об'єднують в остаточну двійкову послідовність на виході кодера.

Блоковий код називається блоковим кодом k-го порядку, якщо всі його блоки мають довжину k символів. За заданим ε>0 можемо знайти таку довжину блоку k (для повідомлення довжиною n буде n/k блоків), що за умови використання оптимального статистичного кодування блоків можна досягти середньої довжини коду більше ентропії менш ніж на ε.

6.1.3 Границя стиснення блокових кодів

Для незалежних ДВВ X1, X2, …, Xn з однаковим розподілом ймовірностей ентропія n-вимірної ДВВ $\vec{X} = (X_1, X_2, \dots, X_n)$

$$H\vec{X} = nHX_1.$$

Нехай $\vec{Y} = (\vec{Y}_1, \vec{Y}_2, \dots, \vec{Y}_{n/k})$ - повідомлення джерела, де $\vec{Y}_1 = (X_1, X_2, \dots, X_k)$, $\vec{Y}_2 = (X_{k+1}, X_{k+2}, \dots, X_{2k})$, $\vec{Y}_i = (X_{k(i-1)+1}, X_{k(i-1)+2}, \dots, X_{ki})$ - блоки повідомлення. Тоді

$$H\vec{Y}_1 = kHX_1.$$

При оптимальному кодуванні k-послідовностей векторної ДВВ. $\vec{Y}$, що розглядаються як одиниці повідомлення, справедлива нерівність

$$\overline{L(\vec{Y}_1)} \le H\vec{Y}_1 + 1.$$

Середня кількість біт на одиницю повідомлення X

$$\overline{L(X)} = \frac{\overline{L(\vec{Y}_i)}}{k}.$$

$$k\overline{L(X)} \le kHX + 1$$

і

$$\overline{L(X)} \le HX + \frac{1}{k},$$

тобто достатньо вибрати $k = 1/\varepsilon$.

6.2 Приклади розв'язування задач

Приклад ДВВ X задана таким розподілом ймовірностей: P(X=A)=1/3; P(X=B)=7/15; P(X=C)=1/5. Побудувати таблицю кодів для блокового коду Хаффмена 2-го порядку. Визначити середню довжину коду.

Розв'язання

Побудуємо розподіл ймовірностей векторної ДВВ $\vec{X} = (X_1, X_2)$, що являє собою блок повідомлення довжиною в два символи (таблиця 6.1).

Виходячи з отриманого ряду ймовірностей, побудуємо кодове дерево за алгоритмом Хаффмена для значень ДВВ $\vec{X}$ (рис.6.1) і відповідну таблицю 6.1 кодових слів.

Таблиця 6.1 – Таблиця кодових слів блокового коду

$\vec{X}$	*BB*	*BA*	*AB*	*AA*	*BC*	*CB*	*AC*	*CA*	*CC*
$P(\vec{X})$	*49/225*	*7/45*	*7/45*	*1/9*	*7/75*	*7/75*	*1/15*	*1/15*	*1/25*
$Code(\vec{X})$	**10**	**001**	**010**	**011**	**111**	**0000**	**0001**	**1100**	**1101**
$L(\vec{X})$	2	3	3	3	3	4	4	4	4
$P_i \cdot L_i$	98/225	7/15	7/15	1/3	7/25	28/75	4/15	4/15	4/25

Середня довжина коду для блокового коду Хаффмена 2-го порядку

$$\overline{L(X)} = \frac{\overline{L(\vec{X})}}{n} = \frac{\sum_i P_i \cdot L_i}{n} = \frac{686}{225} \Big/ 2 = \frac{686}{450} \approx 1{,}524$$ (біт/сим).

Для порівняння: наведемо кодове дерево (рис.6.2) і відповідну таблицю кодів (таблиця 6.2) для одновимірної ДВВ

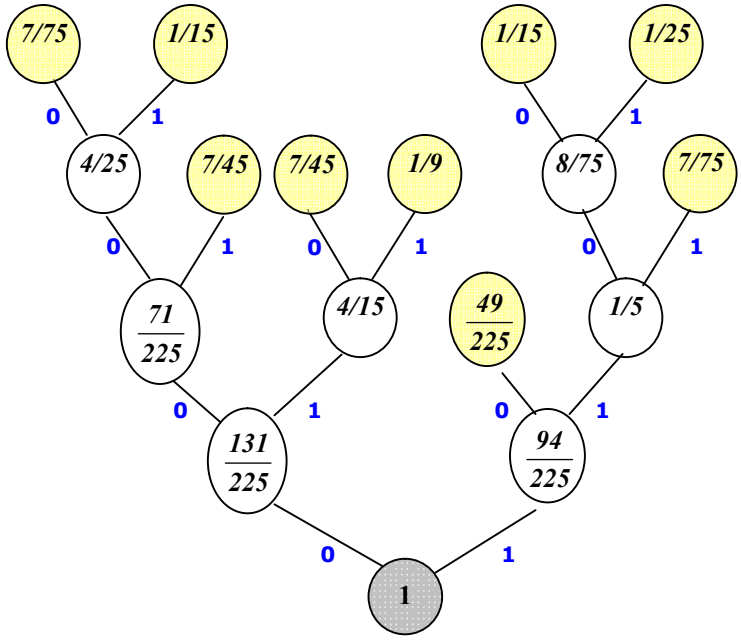


Рисунок 6.1 – Кодове дерево

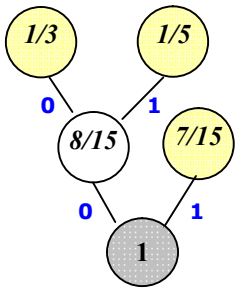


Рисунок 6.2 – Кодове дерево

Таблиця 6.2 – Таблиця кодування символів

x_i	B	A	C	Σ
p_i	7/15	1/3	1/5	1
$Code(x_i)$	1	00	01	
l_i	1	2	2	
$p_i l_i$	7/15	2/3	2/5	23/15

Середня довжина коду $\overline{L(X)} = \frac{23}{15} \approx 1{,}533$ (біт/сим).

Мінімальна середня довжина коду для кодування даної ДВВ визначається її ентропією:

$$\overline{L}_{\min} = HX = \frac{7}{15}\log_2\frac{15}{7} + \frac{1}{3}\log_2 3 + \frac{1}{5}\log_2 5 = \frac{2}{3}\log_2 5 +$$

$$+\frac{4}{5}\log_2 3 - \frac{7}{15}\log_2 7 \approx 1{,}506 \text{ (біт/сим).}$$

Отже, надлишковість блокового коду

$$\rho_к = 1 - \frac{1{,}506}{1{,}524} \approx 0{,}012,$$

а надлишковість неблокового коду

$$\rho_к = 1 - \frac{1{,}506}{1{,}533} \approx 0{,}017$$.

Для рівномірного коду надлишковість

$$\rho_к = 1 - \frac{1{,}506}{2} \approx 0{,}25$$

істотно більше.

Приклад Закодувати повідомлення ABAAABBA за алгоритмом Хаффмена і блоковим алгоритмом Хаффмена 2-го порядку, обчислити довжини отриманих кодів. Приблизний закон розподілу ймовірностей визначити з аналізу повідомлення.

Розв'язання

За частотами символів у повідомленні побудуємо приблизний закон розподілу їх ймовірностей:

Символ x_i	***A***	***B***
Імовірність p_i	***5/8***	***3/8***

Побудуємо таблицю кодів за алгоритмом Хаффмена:

Символ x_i	***A***	***B***
Імовірність p_i	*5/8*	*3/8*
Код	**0**	**1**

Скориставшись даною таблицею кодів, закодуємо повідомлення

Code(ABAAABBA)=01000110.

Довжина коду L(X)=8 біт.

Повідомлення ABAAABBA можна розбити на 4 блоки довжиною 2 символи.

Побудуємо приблизний статистичний закон розподілу ймовірностей блоків повідомлення, розглядаючи їх як одиниці повідомлення.

Блок повідомлення $\vec{X}$	*AB*	*AA*	*BA*
Імовірність p_i	*1/2*	*1/4*	*1/4*

Побудуємо таблицю кодів за алгоритмом Хаффмена для блоків повідомлення .

Блок повідомлення $\vec{X}$	*AB*	*AA*	*BA*
Імовірність p_i	*1/2*	*1/4*	*1/4*
Код	**0**	**10**	**11**

Блоковий код повідомлення

Code(ABAAABBA)=010011.

Довжина коду L(X)=6 біт.

Приклад

1.Для заданого коду і виразу за індивідуальним завданням скласти таблицю розподілу ймовірностей букв(блоків) алфавіту джерела.

2. Обчислити ентропію джерела.
3. Обчислити надлишковість джерела
4. Побудувати код джерела алфавіту.
5. Обчислити середню довжину коду
6. Обчислити надлишковість отриманого коду..
7. Закодувати заданий вираз отриманим кодом

8. Порівняти довжини заданого за індивідуальним завданням виразу в ASCII-кодах і закодованого виразу.

Вираз	Код
життя біжить — як музика дзвенить.	Код Шеннона-Фано другого порядку

Розв’язання задачі

Складаємо таблицю розподілу ймовірностей блоків:

ЖИТТЯ_БІЖИТЬ_—_ЯК_МУЗИКА_ДЗВЕНИТЬ

i	a_i	$p(a_i)$	i	a_i	$p(a_i)$
1	“ЖИ”	2/17	9	“МУ”	1/17
2	“ТЬ”	1/17	10	“ЗИ”	1/17
3	“ТТ”	1/17	11	“КА”	1/17
4	“Я_”	1/17	12	“_Д”	1/17
5	“БІ”	1/17	13	“ЗВ”	1/17
6	“_–”	1/17	14	“ЕН”	1/17
7	“_Я”	1/17	15	“ИТ”	1/17
8	“К_”	1/17	16	“Ь”	1/17

$$P(a_i)=n/N$$

де N – кількість блоків всього,

n – кількість повторів блоку a_i

Обчислюємо ентропію джерела

$$H(X) = -\sum_{i=1}^{16} p_i \log_2 p_i = -\left(\frac{2}{17}\log_2\frac{2}{17} + \frac{15}{17}\log_2\frac{1}{17}\right) = 0.1176 * 3.088 +$$

$$+0.8824 * 4.088 = 0.3631 + 3.6073 = \mathbf{3.9704}\ ^{\text{біт}}/_{\text{символ}}$$

Обчислюємо надлишковість джерела .

$$\rho_{дж} = 1 - \frac{H(X)}{H(X)_{MAX}} = 1 - \frac{3.9704}{\log_2 16} = 1 - \frac{3.9704}{4} = 1 - 0.9926 = \mathbf{0.0074}$$

Будуємо код джерела за алгоритмом Шенона-Фано

i	a_i	$p(a_i)$	Код					l_i	$l_i p_i$
1	“ЖИ”	$^2/_{17}$	0	0	0			3	$^6/_{17}$
2	“ТЬ”	$^1/_{17}$	0	0	1	0		4	$^4/_{17}$
3	“ТТ”	$^1/_{17}$	0	0	1	1		4	$^4/_{17}$
4	“Я_”	$^1/_{17}$	0	1	0	0		4	$^4/_{17}$
5	“БІ”	$^1/_{17}$	0	1	0	1		4	$^4/_{17}$
6	“_–”	$^1/_{17}$	0	1	1	0		4	$^4/_{17}$
7	“_Я”	$^1/_{17}$	0	1	1	1		4	$^4/_{17}$
8	“К_”	$^1/_{17}$	1	0	0	0		4	$^4/_{17}$
9	“МУ”	$^1/_{17}$	1	0	0	1		4	$^4/_{17}$
10	“ЗИ”	$^1/_{17}$	1	0	1	0		4	$^4/_{17}$
11	“КА”	$^1/_{17}$	1	0	1	1		4	$^4/_{17}$
12	“_Д”	$^1/_{17}$	1	1	0	0		4	$^4/_{17}$
13	“ЗВ”	$^1/_{17}$	1	1	0	1		4	$^4/_{17}$
14	“ЕН”	$^1/_{17}$	1	1	1	0		4	$^4/_{17}$
15	“ИТ”	$^1/_{17}$	1	1	1	1	0	5	$^5/_{17}$
16	“Ь”	$^1/_{17}$	1	1	1	1	1	5	$^5/_{17}$
$\sum$		1							$^{68}/_{17}$

Обчислюємо середню довжину коду:

$$\overline{L} = \sum_{i=1}^{n} l_i p_i = \frac{6}{17} + 13 * \frac{4}{17} + 2 * \frac{5}{17} = \frac{68}{17} = 4\ \text{біти}/_{\text{символ}}$$

Обчислюємо надлишковість отриманого коду:

$$\rho_{\text{коду}} = 1 - \frac{3.9704}{\overline{L}} = \frac{3.9704}{4} = \mathbf{0.0074}$$

Кодуємо даний вираз знайденим кодом:

ЖИ	**ТТ**	**Я_**	**БІ**	**ЖИ**	**ТЬ**	**_Я**	**К_**
000	*0011*	*0100*	*0101*	*000*	*0010*	*0111*	*1000*
МУ	**ЗИ**	**КА**	**_Д**	**ЗВ**	**ЕН**	**ИТ**	**Ь**
1001	*1010*	*1011*	*1100*	*1101*	*1110*	*11110*	*11111*

Порівнюємо довжини заданого за індивідуальним завданням виразу в ASCII-кодах і закодованого виразу:

Довжина даного виразу в ASCII коді: $33 * 8 = \mathbf{264}$ біт

Довжина даного виразу в коді Шенона-Фано другого порядку: **64 біт**

$\frac{264}{64} * 100\% = 412.5\%$

Отже, код Шенона-Фано в даній ситуації на 412.5 % коротший за ASCII-код.

ЛЕКЦІЯ 7 СЛОВНИКОВІ МЕТОДИ СТИСНЕННЯ ДАНИХ

7.1 Основні поняття

71.1 Принципи словникових методів стискання даних

У словникових методах у процесі кодування і декодування створюють словники на основі вихідних даних з повідомлень. Ефект стискання даних досягають за рахунок того, що повторювані підрядки у повідомленні замінюють покажчиками (вказівниками) на їх місці появи у словнику.

Декодування стиснутого повідомлення здійснюється заміною покажчика готовою фразою із словника, на яку цей покажчик вказує.

Значне місце серед словникових методів посідають LZ-методи (за прізвищами вчених Ziv та Lempel). Перевагами LZ-методів є високий рівень стиснення даних і висока швидкість кодування.

7.1.2 Класифікація словникових алгоритмів

Словникові методи поділяють на методи з використанням . «ковзного» вікна і методи із застосуванням словника фраз.

Ковзне вікно містить дві частини: більший за розміром словник, який будується у процесі кодування або декодування; буфера, який застосовують для кодування даних. Буфер є значно меншим за словник. Витрати пам’яті на кодування пов’язані з розміром вказівника, який залежить від об’єму словника, і розміру буфера. Із збільшенням розмірів словника і буфера зростає ймовірність появи співпадаючих послідовностей з буфера зі змістом словника.

Методи із застосуванням словника фраз дозволяють будувати словник з одно, двох, трьох тощо фраз, яким призначають відповідний індекс або покажчик у словнику.

7.1.3 Алгоритми з використанням ковзного вікна

Для алгоритмів з використанням «ковзного» за повідомленням вікна виділяють дві нерівні за об'ємом частини: першу, більшу за розміром, яка включає фрагмент повідомлення, що вже проглянуто; другу, набагато меншу частину вікна, що виступає у якості буфера, що містить ще незакодовані символи вхідного потоку.

Першу частину використовують як словник. Зазвичай об'єм ковзного вікна складає кілька кілобайтів, а розмір буфера - не більше 100 байтів. Алгоритми цієї групи відшукують у словнику (більшій частині вікна) ланцюжки символів, що збігаються із вмістом буфера, і замінюють ці ланцюжки покажчиками на їхнє попереднє входження у повідомлення, тобто на вміст словника.

Словник в неявному вигляді міститься у закодованих даних, а зберігаються покажчики на повторювані ланцюжки символів (підрядки), що зустрічаються у повідомленні.

Характерними представниками алгоритмів першої групи є LZ77 та модифікований алгоритм LZSS.

7.1.4 Алгоритми з використанням словника фраз

Алгоритми другої групи доповнюють початковий словник джерела словником фраз, що є повторюваними у повідомленні комбінаціями символів початкового словника. При цьому розмір словника збільшується, і для його кодування потрібне більше число біт, але значна частина словника подає не окремі букви, а сполучення букв або цілі слова. Якщо кодер знаходить фразу, що раніше зустрічалася, він замінює її індексом цієї фрази у словнику. Довжина коду індексу є набагато меншою довжини коду незакодованого підрядка.

Базовою реалізацією є алгоритм LZ78, а удосконаленим LZW.

7.2 Алгоритм LZ77

7.2.1 Кодування за алгоритмом LZ77

За алгоритмом LZ77 друге і подальші входження деякого підрядка символів у повідомленні замінюються покажчиками на його перше або попереднє входження. Алгоритм використовує частину повідомлення, що вже проглянуто, як словник. Для стиснення алгоритм робить спробу замінити наступну фразу повідомлення покажчиком на вміст словника.

Процедури кодування та декодування за алгоритмом LZ77 такі.

Кодер:

```
While (lookAheadBuffer not empty)
  get a pointer(position,  match) to the longest match
  in the window for the lookahead buffer;
 if (lehgth>Minimum_Match_Length)
    output a(position, length) pair;
    shift the window length characters along;
 else
    output the first character in the lookaheadbuffer;
    shift the window 1 character along.
```

7.2.2 Декодування за алгоритмом LZ77

Декодер:

```
Whenever a(position, length) pair is encountered,
  go  to  that  (position)  in  the  window  and  copy
(length) bytes to the output.
```

З N символів розміра «ковзного» вікна та F - розміру буфера, перші N-F символів є вже закодовані символи, що містить словник, а останні F символів – вміст випереджуючого буфера.

При кодуванні вмісту буфера серед попередніх N-F символів, тобто у словнику, шукається найдовший підрядок, що збігається з початком буфера. Знайдений найбільший збіг кодується тріадою <i, j, a>, де і - зсув у словнику підрядка, що збігається із початком буфера; j - довжина підрядка, що збігається; а - перший символ, що йде за підрядком, що збігається. Далі алгоритм виконує зсув усього вмісту вікна на j+1 символів і водночас зчитує стільки ж символів вхідного потоку у буфер.

Об'єм пам'яті, що потребує алгоритм-кодер або декодер, визначається розміром вікна N. Довжина коду обчислюється так: довжина підрядка, що співпав із вмістом словника, не може бути більше розміру буфера F, а зсув

цього підрядка у словнику не може бути більше розміру словника мінус 1. Отже, довжина двійкового коду зсуву і буде округлений до більшого цілого $\lceil \log_2(N-F) \rceil$, а довжина коду довжини підрядка j буде округлений у більшу сторону $\lceil \log_2(F+1) \rceil$, а символ a кодується 8 бітами за таблицею ASCII+.

При декодуванні виконується той же самий порядок роботи з вікном, що й при кодуванні, але на відміну від пошуку підрядків, що збігаються, вони, навпаки, копіюються декодером з вікна згідно з черговою тріадою коду <i, j, a>.

7.2.3 Приклад кодування за алгоритмом LZ77

Кодування виразу алгоритмом LZ77: NATURA RERUM

Словник															Буфер					код
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	1	2	3	4	5	
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	N	A	T	U	R	<0,0,N>
-	-	-	-	-	-	-	-	-	-	-	-	-	-	N	A	T	U	R	A	<0,0,A>
-	-	-	-	-	-	-	-	-	-	-	-	-	N	A	T	U	R	A	" "	<0,0,T>
-	-	-	-	-	-	-	-	-	-	-	-	N	A	T	U	R	A	" "	R	<0,0,U>
-	-	-	-	-	-	-	-	-	-	-	N	A	T	U	R	A	" "	R	E	<0,0,R>
-	-	-	-	-	-	-	-	-	-	N	A	T	U	R	A	" "	R	E	R	<11,1," ">
-	-	-	-	-	-	-	-	N	A	T	U	R	A	" "	R	E	R	U	M	<12,1,E>
-	-	-	-	-	-	N	A	T	U	R	A	" "	R	E	R	U	M	-	-	<10,1,U>
-	-	-	-	N	A	T	U	R	A	" "	R	E	R	U	M	-	-	-	-	<0,0,M>

Код повідомлення: <0,0,N> <0,0,A> <0,0,T> <0,0,U> <0,0,R> <11,1,” ”> <12,1,E> <10,1,U> <0,0,M>

Порівняємо довжини виразів:

Довжина коду стисненого повідомлення

L_{CODE}=9*([$\log_2 15$] + [$\log_2(5+1)$]+8)=9*(8+3+4) =135 біт

Довжина нестисненого повідомлення

$L_{ASCII+} = 8 * 12 = 96$ біт

Середня довжина символу стиснутого повідомлення

$t_{CODE} = \frac{135}{12} = 11{,}25$ біт/сим.

Це більше, ніж довжина символу 8 біт у ASCII-коді.

7.2.4 Приклад кодування за алгоритмом LZ77

Декодування коду повідомлення:

<0,0,N> <0,0,A> <0,0,T> <0,0,U> <0,0,R> <11,1,” ”> <12,1,E> <10,1,U> <0,0,M>:

Вхідний код	Вихід	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
<0,0,N>	“N”	-	-	-	-	-	-	-	-	-	-	-	-	-	-	N
<0,0,A>	“A”	-	-	-	-	-	-	-	-	-	-	-	-	-	N	A
<0,0,T>	“T”	-	-	-	-	-	-	-	-	-	-	-	-	N	A	T
<0,0,U>	“U”	-	-	-	-	-	-	-	-	-	-	-	N	A	T	U
<0,0,R>	“R”	-	-	-	-	-	-	-	-	-	-	N	A	T	U	R
<11,1,” “>	“A ”	-	-	-	-	-	-	-	-	N	A	T	U	R	A	“ “
<12,1,E>	“RE”	-	-	-	-	-	-	N	A	T	U	R	A	“ “	R	E
<10,1,U>	“RU”	-	-	-	-	N	A	T	U	R	A	“ ”	R	E	R	U
<0,0,M>	“M”	-	-	-	N	A	T	U	R	A	“ “	R	E	R	U	M

7.3 Алгоритм LZSS

7.3.1 Кодування за алгоритмом LZSS

Алгоритм LZSS є модифікацією алгоритму LZ77. Код алгоритму починається однобітовим префіксом, що відділяє код підрядка, що збігається, від незакодованого символу.

Кодовий набір складається з пари <i, j> - зсуву i у словнику підрядка, що збігається з початком буфера, і довжини j цього підрядка, як і для LZ77.

Вікно зсувається рівно на довжину знайденого підрядка або на 1, якщо входження підрядка буфера у словнику не знайдено.

Довжина підрядка у алгоритмі LZSS завжди більше 0 і не може перевищувати розмір буфера F, тому довжина двійкового коду довжини підрядка, що збігається, j - це округлений до більшого цілого $\lceil \log_2(F) \rceil$, а довжина коду зсуву i – округлений до більшого цілого $\lceil \log_2(N-F) \rceil$.

Приклад кодування за алгоритмом LZSS рядка «POST_FACTUM» з розміром словника 13 байтів і буфера 6 байтів.

Словник													Буфер						Код	Довжина
0	1	2	3	4	5	6	7	8	9	10	11	12	1	2	3	4	5	6		
													P	O	S	T	_	F	0'P'	9
												P	O	S	T	_	F	A	0'O'	9
											P	O	S	T	_	F	A	C	0'S'	9
										P	O	S	T	_	F	A	C	T	0'T'	9

									P	O	S	T	_	F	A	C	T	U	0'_'	9
								P	O	S	T	_	F	A	C	T	U	M	0'F'	9
							P	O	S	T	_	F	A	C	T	U	M		0'A'	9
						P	O	S	T	_	F	A	C	T	U	M			0'C'	9
					P	O	S	T	_	F	A	C	T	U	M				1<8,1>	8
				P	O	S	T	_	F	A	C	T	U	M					0'U'	9
			P	O	S	T	_	F	A	C	T	U	M						0'M'	9
		P	O	S	T	_	F	A	C	T	U	M								

Код повідомлення:

0'P' 0'O' 0'S' 0'T' 0'_' 0'F' 0'A' 0'C' 1<8,1> 0'U' 0'M'

Довжина коду стиснутого повідомлення: $L\,code = 10 \times 9 + 8 = 98\,біт$

Довжина нестиснутого повідомлення: $L\,ASCII = 11 \times 8 = 88\,біт$

Вхідний код	Вихід	Словник												
		0	1	2	3	4	5	6	7	8	9	10	11	12
0’P’	P													P
0’O’	O												P	O
0’S’	S											P	O	S
0’T’	T										P	O	S	T
0’_’	_									P	O	S	T	_
0’F’	F								P	O	S	T	_	F
0’A’	A							P	O	S	T	_	F	A
0’C’	C						P	O	S	T	_	F	A	C
1<11,1>	T					P	O	S	T	_	F	A	C	T
0’U’	U				P	O	S	T	_	F	A	C	T	U
0’M’	M			P	O	S	T	_	F	A	C	T	U	M

7.3.2 Декодування за алгоритмом LZSS

Приклад декодування (розпаковування) повідомлення

0’P’ 0’O’ 0’S’ 0’T’ 0’_’ 0‘F’ 0’A’ 0’C’ 1<8,1> 0’U’ 0’M’

Вхідний код	Вихід	Словник												
		0	1	2	3	4	5	6	7	8	9	10	11	12
0'P'	P													P
0'O'	O												P	O
0'S'	S											P	O	S
0'T'	T										P	O	S	T
0'_'	_									P	O	S	T	_
0'F'	F								P	O	S	T	_	F
0'A'	A							P	O	S	T	_	F	A
0'C'	C						P	O	S	T	_	F	A	C
1<11,1>	T					P	O	S	T	_	F	A	C	T
0'U'	U				P	O	S	T	_	F	A	C	T	U
0'M'	M			P	O	S	T	_	F	A	C	T	U	M

Недоліками алгоритмів LZ77, LZSS є:

- неможливість кодування повторюваних підрядків, що знаходяться на відстані, більшій за довжину словника;

- довжина підрядка, який можна закодувати, обмежується розміром буфера;

- збільшення розміру словника і буфера призводить до суттєвого зростання часу кодування.

7.4 Приклади розв'язування задач

Приклад Закодувати вираз: QUI VIVRA, VERRA, використовуючи словниковий алгоритм LZ77. Розмір словника – 15 байтів, буфера – 5 байтів. Обчислити довжини кодів.

Розв'язання. Кодуємо повідомлення за алгоритмом LZ77.

Кодування повідомлення

Словник															Буфер					Код
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	1	2	3	4	5	<i, j, `a`>
															Q	U	I		V	<0,0,`Q`>
														Q	U	I		V	I	<0,0,`U`>
													Q	U	I		V	I	V	<0,0,`I`>
												Q	U	I		V	I	V	R	<0,0,` `>
											Q	U	I		V	I	V	R	A	<0,0,`V`>
										Q	U	I		V	I	V	R	A	,	<12,1,`V`>
								Q	U	I		V	I	V	R	A	,		V	<0,0,`R`>
							Q	U	I		V	I	V	R	A	,		V	E	<0,0,`A`>
						Q	U	I		V	I	V	R	A	,		V	E	R	<0,0,`,`>
					Q	U	I		V	I	V	R	A	,		V	E	R	R	<8,1,`V`>
			Q	U	I		V	I	V	R	A	,		V	E	R	R	A		<0,0,`E`>
		Q	U	I		V	I	V	R	A	,		V	E	R	R	A			<9,1,`R`>
Q	U	I		V	I	V	R	A	,		V	E	R	R	A					<0,0,`A`>

Довжина коду:

$$L_{code} = 13*(8+(\lceil \log_2 15 \rceil + 1)+(\lceil \log_2 5 \rceil + 1)) = 13*15 = 195(біт)$$

$$L_{ASCII} = 16*8 = 128(біт)$$

Довжина символу при кодуванні становить для LZ77становить 195/16 біт/сим. Це значно більше у порівняні з 8 іт біт /сим в ASCII –кодах.

Тому цікавим є порівння методу LZ77 зі статистичними методами кодування, зокрема з методом Хаффмена.

При кодуванні виразу за алгоритмом Хаффмена для заданого виразу розподіл ймовірностей появи символів джерела повідомлень становить

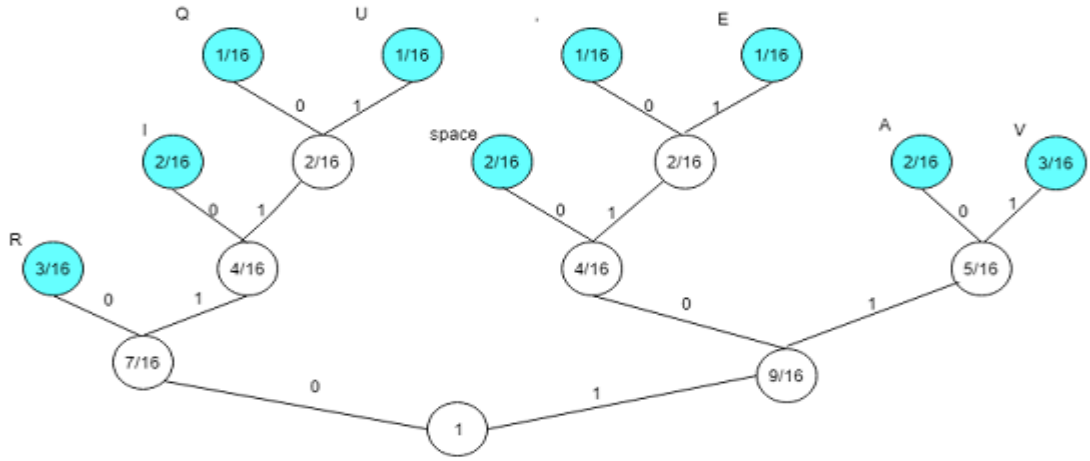
Символ, x_i	Ймовірність, p_i
R	3/16
V	3/16
space	2/16

I	2/16
A	2/16
Q	1/16
U	1/16
,	1/16
E	1/16

Обчислимо ентропію джерела:

$$H(X) = -\sum_i p_i * \log_2 p_i = -(4*1/16*\log_2 1/16 + 3*2/16*\log_2 2/16 + 2*3/16*\log_2 3/16 \approx 2.65562(біт/символ)$$

Дерево Хаффмена має такий вигляд



Отримуємо таблицю розподілу символів

Символ, x_i	Ймовірність, p_i	Код	Довжина, l_i	$l_i\ p_i$
R	3/16	00	2	6/16
V	3/16	111	3	9/16
space	2/16	100	3	6/16
I	2/16	010	3	6/16
A	2/16	110	3	6/16
Q	1/16	0110	4	4/16
U	1/16	0111	4	4/16
,	1/16	1010	4	4/16
E	1/16	1011	4	4/16

Закодуємо заданий вираз отриманим кодом:

Q	U	I		V	I	V	R	A	,
0110	0111	010	100	111	010	111	00	110	1010

	V	E	R	R	A
100	111	1011	00	00	110

Тобто довжина закодованого повідомлення становить

L_{Huff} =49 біт.

Середня довжина коду:

$$L(\acute{x}) = \frac{L(\vec{x})}{n} = \frac{\sum_l P_l L_l}{n} = \frac{49}{16} = 3{,}0625(\text{біт/сим})$$

Виконаємо декодування закодованого виразу для LZ77.

Вхідний код	Вихід	Словник														
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
<0,0,'Q'>	Q															Q
<0,0,'U'>	U														Q	U
<0,0,'I'>	I													Q	U	I
<0,0,' '>													Q	U	I	
<0,0,'V'>	V											Q	U	I		V
<12,1,'V'>	IV									Q	U	I		V	I	V
<0,0,'R'>	R								Q	U	I		V	I	V	R
<0,0,'A'>	A							Q	U	I		V	I	V	R	A
<0,0,','>	,						Q	U	I		V	I	V	R	A	,
<8,1,'V'>	V				Q	U	I		V	I	V	R	A	,		V
<0,0,'E'>	E			Q	U	I		V	I	V	R	A	,		V	E
<9,1,'R'>	RR	Q	U	I		V	I	V	R	A	,		V	E	R	R
<0,0,'A'>	A	U	I		V	I	V	R	A	,		V	E	R	R	A

Приклад Закодувати вираз: MEA CULPA, використовуючи словниковий алгоритм LZSS. Розмір словника – 13 байтів, буфера – 6 байтів. Обчислити довжини кодів.

Розв'язання. Кодуємо повідомлення за алгоритмом LZSS.

Кодування повідомлення

Словник(13 байт)													Буфер(6 байт)						код
0	1	2	3	4	5	6	7	8	9	10	11	12	1	2	3	4	5	6	
													M	E	A		C	U	0 'M'
												M	E	A		C	U	L	0 'E'
											M	E	A		C	U	L	P	0 'A'
										M	E	A		C	U	L	P	A	0 '_'
									M	E	A		C	U	L	P	A		0 'C'
								M	E	A		C	U	L	P	A			0 'U'
							M	E	A		C	U	L	P	A				0 'L'
						M	E	A		C	U	L	P	A					0 'P'
					M	E	A		C	U	L	P	A						1 <7,1>

Довжина закодованого виразу

L_{LZSS} = 9*(1+3+4)=72 біт.

Виконаємо декодування закодованого виразу

вихідний код	вихід	словник												
		0	1	2	3	4	5	6	7	8	9	10	11	12
0 'M'	L													M
0 'E'	A												M	E
0 'A'	P											M	E	A
0 '_'	S										M	E	A	
0 'C'	U									M	E	A		C
0 'U'	S								M	E	A		C	U
0 ' L'	" "							M	E	A		C	U	L
0 'P'	C						M	E	A		C	U	L	P
1 <7,1>	A					M	E	A		C	U	L	P	A

ЛЕКЦІЯ 8 МЕТОДИ СТИСНЕННЯ ІЗ ЗАСТОСУВАННЯМ СЛОВНИКА ФРАЗ

8.4 Алгоритм LZ78

8.4.1 Кодування за алгоритмом LZ78

Алгоритм LZ78 не використовує вікно, а зберігає словник із фраз повідомлення, що вже проглянуто. Спочатку словник містить лише один порожній рядок. Алгоритм зчитує символи повідомлення доти, доки накопичуваний підрядок повністю збігається з однією із фраз словника. Як тільки зчитаний підрядок не відповідає фразі словника, алгоритм генерує код з індексу фрази у словнику, що до останнього зчитаного символу містила вхідний підрядок, і символу, що порушив збіг. Потім зчитаний новий підрядок заноситься у словник, і пошук збігу наступного підрядка з однією із фраз словника починається знову. Якщо словник вже заповнений, то з нього видаляють фразу, що найменше використовується у порівняннях.

Для визначення розміру кодів ключовим є розмір словника, оскільки кожний код за алгоритмом LZ78 містить номер фрази у словнику. Звідси випливає, що коди алгоритму мають постійну довжину, яка дорівнює округленому до більшого цілого $\lceil \log_2([\textit{розмір словника}]) \rceil + 8$, де 8 - кількість біт для кодування наступного символу, що йде за підрядком, що збігається, за таблицею ACSII+.

В узагальненому вигляді алгоритм LZ78 має такий вигляд

```
w := NIL;
While (there is input){
K  := next symbol from input;
If (wK exists in the dictionary) {
```

```
w := wK;
}
Else {
Output (index(w), K);
Add wK to the dictionary;
w := NIL;
}
}
```

Розглянемо кодування за алгоритмом LZ78 виразу **NATURA RERUM**

Побудуємо таблицю кодування для словника з кількістю фраз 16.

Словник	Код	Індекс фрази
“”	-	0
“N”	<0,”N”>	1
“A”	<0,”A”>	2
“T”	<0,”T”>	3
“U”	<0,”U”>	4
“R”	<0,”R”>	5
“A_”	<2.”_”>	6
“RE”	<5,”E”>	7
“RU”	<5,”U”>	8
“M”	<0,”M”>	9

Отримано закодоване плвідомлення

<0,”N”><0,”A”><0,”T”><0,”U”><0,”R”><2,”_”><5,”E”><5,”U”><0,”M”>

Довжина закодованого повідомлення становить

L_{LZ78}= 9*($\log_2$16 +8)=108 біт.

Незакодоване повідомлення для виразу становить

L_{ASCII}=12*8=96 біт.

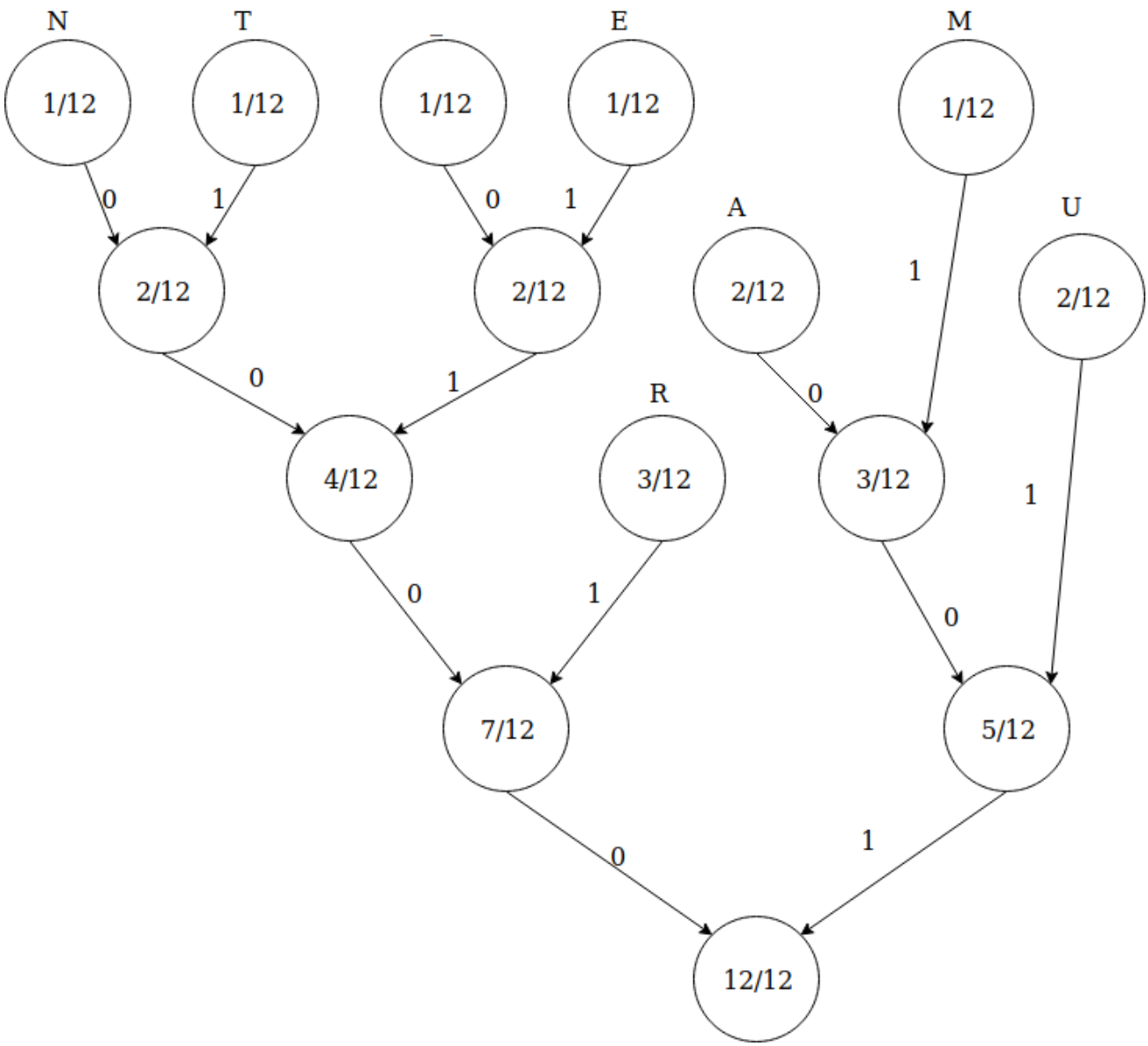
Проведемо порівняння даного алгоритму з алгоритмом Huffman.

Символ	P_i
N	1/12
A	2/12
T	1/12
U	2/12
R	3/12
_	1/12
E	1/12
M	1/12

Побудуємо дерево кодування виразу кодом Хаффмена з використанням таблиці.

Символ	P_i	Код Хаффмана
N	1/12	0000
A	2/12	100
T	1/12	0001
U	2/12	11
R	3/12	01
_	1/12	0010
E	1/12	0011
M	1/12	101

Можна підрахувати середнюдовжину кода



Кодуємо вираз отриманим кодом Хаффмана:

0000 100 0001 11 01 100 0010 01 0011 01 11 101

Обчислимо довжини заданого за індивідуальним завданням виразу в ASCII-кодах, закодованого виразу за заданим алгоритмом і кодом Хаффмана.

L_{ASCII}=12*8=96(біт)

$L_{HUFFMAN}$=35(біт)

Обчислимо ентропію джерела. Ентропія джерела становить

$$H(X) = \sum(-p_i)\,log_2(p_i) = 2.855\,\text{біт/символ}$$

Тому мінімальна границя кодування становить

L_{min} = 2,855*12= 34 біт.

8.4.2 Декодування за алгоритмом LZ78

Приклад Виконуємо декодування закодованого повідомлення виразу

<0,”N”><0,”A”><0,”T”><0,”U”><0,”R”><2,”_”><5,”E”><5,”U”><0,”M”>

Код	Індекс фрази	Словник
-	0	“”
<0,”N”>	1	“N”
<0,”A”>	2	“A”
<0,”T”>	3	“T”
<0,”U”>	4	“U”
<0,”R”>	5	“R”
<2,” ”>	6	“A ”
<5,”E”>	7	“RE”
<5,”U”>	8	“RU”
<0,”M”>	9	“M”

Тобто на виході декодера маємо вираз: NATURA RERUM

8.5 Алгоритм LZW

8.5.1 Кодування за алгоритмом LZW

Алгоритм LZW є модифікацією LZ78. Алгоритм починає роботу зі словника розміром 4К, що містить за адресами від 0 до 255 посилання на окремі символи (таблиця ASCII+), а від 256 до 4095 - посилання на підрядки завдовжки більше одного символу.

У процесі кодування текст розбивається на підрядки, де кожний новий підрядок є найдовшою фразою, що збігається, що вже проглянуто, плюс один символ. Новий підрядок кодується індексом його префікса (фрази словника) і символом, що порушив збіг, після чого нова фраза заноситься у словник (таблицю рядків).

При переповнюванні словника з нього видаляється або найменш уживана фраза, або усі підрядки завдовжки більше одного символу.

Алгоритму кодування LZW має такий вигляд.

```
Initialize table with single character strings
P = first input character
WHILE not end of input stream
C = next input character
 IF P + C is in the string table
 P = P + C
ELSE
output the code for P
 add P + C to the string table
P = C
END WHILE
```

LZW-коди мають постійну довжину, що визначається округленим до більшого цілого $\lceil \log_2([\textit{розмір словника}]) \rceil$.

Розглянемо кодування за алгоритмом LZW-виразу **NATURA RERUM**

Побудуємо таблицю кодування для словника з кількістю фраз 512.

Словник “”	Код -	Індекс фрази 0
“NA”	0”N”	256
“AT”	0”A”	257
“TU”	0”T”	258
“UR”	0”U”	259
“RA” “A_”	0”R” <257>	260 261
“_R”	0”_”	262
“RE”	0”R”	263
“ER”	0”E”	264
“RU”	0”R”	265
“UM”	0”U”	266
“M”	0”M”	

Довжина закодованого повідомлення становить

L_{LZW}= 11*(1+8) +$\log_2$512 =108 біт.

8.5.2 Декодування за алгоритмом LZW

LZW-декодер також, як і кодер, заносить нові фрази у словник кожного разу, якщо знаходить у вхідному потоці новий код. Наприкінці декодування декодер має такий самий словник фраз, що був накопичений кодером при кодуванні.

При декодуванні за алгоритмом LZW потрібно дотримуватися такого правила: словник поповнюється тільки після зчитування першого символу фрази, що розкодовується із наступного за поточним коду.

Виконуємо декодування закодованого повідомлення виразу **NATURA RERUM**.

Код	Вихід декодера	Індекс фрази	Словник
-	-	0	“”
0”N”	N	256	“NA”
0”A”	A	257	“AT”
0”T”	T	258	“TU”
0”U”	U	259	“UR”
0”R”	R	260	“RA”
<257>	A	261	“A_”
0”_”	_	262	“_R”
0”R”	R	263	“RE”
0”E”	E	264	“ER”
0”R”	R	265	“RU”
0”U”	U	266	“UM”
0”M”	M		“M”

Тобто на виході декоднера маємо вираз: NATURA RERUM

8.6 Приклади розв'язування задач

Приклад Декодувати повідомлення, закодовані за алгоритмом LZW.

0‘A’ 0‘F’ 0‘X’ <256> <257> <257> 0‘A’ <258> 0‘F’ 0‘F’ 0‘A’

Обчислити довжину їх кодів.

Повідомлення стиснуте за алгоритмом LZW (словник містить таблицю ASCII+ і 512 фраз).

Код стиснутого повідомлення такий: 0‘A’ 0‘F’ 0‘X’ <256> <257> <257> 0‘A’ <258> 0‘F’ 0‘F’ 0‘A’.

Вхідний код	**Вихідна фраза**	**Словник**	**Індкс словника**
		ASCII+	**0 – 255**
0‘*A*’	***“A”***	***“AF”***	**256**
0‘*F*’	***“F”***	***“FX”***	**257**
0‘*X*’	***“X”***	***“XA”***	**258**
<256>	***“AF”***	***“AFF”***	**259**
<257>	***“FX”***	***“FXF”***	**260**
<257>	***“FX”***	***“FXA”***	**261**
0‘*A*’	***“A”***	***“AX”***	**262**
<258>	***“XA”***	***“XAF”***	**263**
0‘*F*’	***“F”***	***“FF”***	**264**
0‘*F*’	***“F”***	***“FA”***	**265**
0‘*A*’	***“A”***		

Закодоване повідомлення: AFXAFFXFXAXAFFA.

Довжина стиснутого повідомлення Lcode=11*9=99 біт.

ЛЕКЦІЯ 9 ПРИНЦИПИ ЗАВАДОСТІЙКОГО КОДУВАННЯ

9.1 Лінійні блокові коди

9.1.1 Функції кодерів блокових і згортокових кодів

Кодер для блокових кодів перетворює інформаційні блоки завдовжки k символів у повідомлення з n символів, з яких r=n-k є надлишковими (перевірними, контрольними). Перевірні символи забезпечують виявлення й виправлення помилок, спричинених завадами у каналі зв'язку. Кожний блок з n символів залежить лише від відповідного k-символьного інформаційного блоку і не залежить від інших блоків.

Кодер для згорткових кодів перетворює послідовність з k інформаційних символів у блок з n кодових символів, де n>k. Кодовий n-символьний блок залежить не тільки від k-символьного інформаційного блоку, наявного на вході у поточний момент часу, але і від попередніх m блоків повідомлення. Згорткові коди часто використовують для побітового передавання даних каналами.

9.1.2 Систематичні коди

Блоковий код завдовжки n символів, що складається з 2^k кодових слів, називають лінійним (n, k)-кодом за умови, що всі 2^k кодових слів утворюють k-вимірний підпростір векторного простору n-послідовностей двійкового поля GF(2), тобто порозрядна сума за модулем 2 (mod 2) двох кодових слів також є кодовим словом даного коду.

Систематичний код має інформаційну частину з k символів і надлишкову (перевірну) частину з n-k символів постійної довжини. Найпростішим лінійним блоковим кодом є (n-1,n) - код з контролем парності з використанням перетворення виду $E(m_1, ... , m_k)=(m_1, ..., m_k, m_{k+1})$, де m_{k+1} є символом парності або непарності інформаційних розрядів $m_1, ... , m_k$.

9.1.3 Оцінка впливу помилок симетричних каналів

Помилки у бітах для двійкових симетричних каналів вважають рівноймовірними й незалежними і оцінюють як: p - ймовірність безпомилкової передачі біта і q = 1-p - ймовірність помилкової передачі повідомлення. Тому ймовірність передачі n бітів з k помилками визначається за формулою Бернуллі

$$P_n(k) = C_n^k q^k p^{n-k},$$

де $C_n^k = \frac{n!}{k!(n-k)!}$ - кількість сполук з n елементів по k

9.2 Ітеративний код

9.2.1 Поняття ітеративного коду

Ітеративний код є найпростішим кодом з виправленням помилок. Повідомлення m розміщують у вигляді матриці з додаванням до кожного

рядка і стовпця контрольних символів перевірки на парність. За умови виникнення однієї помилки перевірка на парність у відповідному рядку і стовпці не виконуватиметься, а координати помилки однозначно визначають номерам стовпця і рядка, в яких не виконується перевірка на парність.

9.2.2 Виявлення і виправлення помилок

Розглянемо приклад застосування ітеративного коду.

Приклад. Для кодування послідовності з k=12 інформаційних елементів застосовується ітеративний метод. Записати твірну матрицю еквівалентного лінійного блокового коду. Закодувати повідомлення (110111000110). Виправити помилку в прийнятій послідовності (110111100011100011111). Визначити надлишковість коду.

Для розв'язання задачі інформаційну послідовність розмістимо у вигляді матриці розмірністю 3×4. Вихідна матриця кодування для інформаційної послідовності (m_1, m_2, … , m_{12}) має вигляд

$$\begin{pmatrix} m_1 & m_2 & m_3 & m_4 \\ m_5 & m_6 & m_7 & m_8 \\ m_9 & m_{10} & m_{11} & m_{12} \end{pmatrix} \Rightarrow \begin{pmatrix} m_1 & m_2 & m_3 & m_4 & r_1 \\ m_5 & m_6 & m_7 & m_8 & r_2 \\ m_9 & m_{10} & m_{11} & m_{12} & r_3 \\ r_4 & r_5 & r_6 & r_7 & r_8 \end{pmatrix},$$

де r_1, r_2, …, r_8 – перевірні елементи.

Тоді послідовність, що зберігається або передається по каналу зв’язку, є:

u= (m_1, m_2, m_3, m_4, r_1, m_5, m_6, m_7, m_8, r_2, m_9, m_{10}, m_{11}, m_{12}, r_3, r_4, r_5, r_6, r_8).

З вихідної матриці отримуємо систему перевірних рівнянь, що визначає правила знаходження перевірних елементів:

$$\begin{cases} r_1 = m_1 + m_2 + m_3 + m_4, \\ r_2 = m_5 + m_6 + m_7 + m_8, \\ r_3 = m_9 + m_{10} + m_{11} + m_{12}, \\ r_4 = m_1 + m_5 + m_9, \\ r_5 = m_2 + m_6 + m_{10}, \\ r_6 = m_3 + m_7 + m_{11}, \\ r_7 = m_4 + m_8 + m_{12}, \\ r_8 = m_1 + m_2 + \ldots + m_{12}. \end{cases}$$

Для системи перевірних рівнянь знаходимо твірну матрицю $G_{12\times20}$ еквівалентного лінійного блокового коду, яку можна легко привести до канонічного вигляду, розмістивши стовпці, що відповідають перевірним елементам у правій частини твірної матриці, а одиничну підматрицю, що визначає інформаційну частину кодового слова – у лівій:

$$\boldsymbol{G}_{12\times20} = \begin{pmatrix} 1\,0\,0\,0 & 1 & 0\,0\,0\,0 & 0 & 0\,0\,0 & 0 & 0 & 0\,1\,0\,0\,0\,1 \\ 0\,1\,0\,0 & 1 & 0\,0\,0\,0 & 0 & 0\,0\,0 & 0 & 0 & 0\,0\,1\,0\,0\,1 \\ 0\,0\,1\,0 & 1 & 0\,0\,0\,0 & 0 & 0\,0\,0 & 0 & 0 & 0\,0\,0\,1\,0\,1 \\ 0\,0\,0\,1 & 1 & 0\,0\,0\,0 & 0 & 0\,0\,0 & 0 & 0 & 0\,0\,0\,0\,1\,1 \\ 0\,0\,0\,0 & 0 & 1\,0\,0\,0 & 1 & 0\,0\,0 & 0 & 0 & 0\,1\,0\,0\,0\,1 \\ 0\,0\,0\,0 & 0 & 0\,1\,0\,0 & 1 & 0\,0\,0 & 0 & 0 & 0\,0\,1\,0\,0\,1 \\ 0\,0\,0\,0 & 0 & 0\,0\,1\,0 & 1 & 0\,0\,0 & 0 & 0 & 0\,0\,0\,1\,0\,1 \\ 0\,0\,0\,0 & 0 & 0\,0\,0\,1 & 1 & 0\,0\,0 & 0 & 0 & 0\,0\,0\,0\,1\,1 \\ 0\,0\,0\,0 & 0 & 0\,0\,0\,0 & 0 & 1\,0\,0 & 0 & 0 & 1\,1\,0\,0\,0\,1 \\ 0\,0\,0\,0 & 0 & 0\,0\,0\,0 & 0 & 0\,1\,0 & 0 & 0 & 1\,0\,1\,0\,0\,1 \\ 0\,0\,0\,0 & 0 & 0\,0\,0\,0 & 0 & 0\,0\,0 & 1 & 0 & 1\,0\,0\,1\,0\,1 \\ 0\,0\,0\,0 & 0 & 0\,0\,0\,0 & 0 & 0\,0\,0 & 0 & 1 & 1\,0\,0\,0\,1\,1 \end{pmatrix}.$$

$$\boldsymbol{m_1 m_2 m_3 m_4} \;\; \boldsymbol{r_1} \;\; \boldsymbol{m_5 m_6 m_7 m_8} \;\; \boldsymbol{r_2} \;\; \boldsymbol{m_9 m_{10} m_{11} m_{12}} \;\; \boldsymbol{r_3} \;\; \boldsymbol{r_4} \;\; \boldsymbol{r_5} \;\; \boldsymbol{r_6} \;\; \boldsymbol{r_7} \;\; \boldsymbol{r_8}$$

Закодуємо задану інформаційну послідовність m=(110111000110).

Розмістимо її у вигляді матриці 3×4 і допишемо перевірні елементи до кожного рядка і стовпця:

$$\begin{vmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{vmatrix} \Rightarrow \left|\begin{array}{cccc|c} 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ \hline 0 & 1 & 1 & 1 & 1 \end{array}\right| .$$

Тобто кодовим слово на виході кодера буде послідовність

(11011110000110001111).

Нехай за умовою задачі на приймайльній стороні прийнято послідовність (11011110001110001111). Декодуємо її за ітеративним методом.

Запишемо прийняте слово у вигляді матриці (у даному випадку розмірністю 4×5) і виконаємо перевірку на парність за її кожним рядком і стовпчиком. У разі відсутності помилки контрольні елементи парності за рядками і стовпчиками матриці мають значення 0. Невиконання контролю парності у стовпчику і рядку однозначно визначить координати помилкового елемента – це дозволить його виправити.

$$\begin{vmatrix} 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 \end{vmatrix} \Rightarrow \left|\begin{array}{ccccc|c} 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ \hline 1 & 0 & 0 & 0 & 0 & 1 \end{array}\right| .$$

Помилковим є елемент m[3,1], який виправляємо.

$$\begin{vmatrix} 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 \end{vmatrix} \Rightarrow \begin{vmatrix} 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 \end{vmatrix} .$$

Отже, виправлена комбінація така:

y=(11011110000110001111).

Декодуємо її, прибравши перевірні елементи:

(11011110000110001111) $\Rightarrow$ (110111000110).

Надлишковість коду $\rho_к = 1 - \frac{12}{20} = 0{,}4$.

9.3 Способи подання лінійних кодів

9.3.1 Основні способи подання лінійних кодів

Основними способами подання лінійних кодів є табличний, за допомогою систем перевірних рівнянь, за допомогою твірної матриці, перевірної матриці. За табличним способом кожній інформаційній послідовності ставиться у відповідність кодове слово з таблиці кодів.

9.3.2 Система перевірних рівнянь

Система перевірних рівнянь визначає правила знаходження перевірних символів залежно від інформаційних, наприклад

$$\begin{cases} r_1 = m_1, \\ r_2 = m_2, \\ r_3 = m_3, \\ r_4 = m_1 + m_2 + m_3; \end{cases}$$

де r_1, r_2, r_3, r_4 - перевірні (контрольні) символи.

9.3.3 Подання коду твірною матрицею

Твірна матриця G розміром k×n

$$G_{k\times n} = \left\langle \begin{array}{c|c} 1\,0\,0\,\ldots\,0 & p_{11}\ p_{12}\ \cdots\ p_{1,n-k} \\ 0\,1\,0\,\ldots\,0 & p_{21}\ p_{22}\ \cdots\ p_{2,n-k} \\ 0\,0\,1\,\ldots\,0 & \ldots\ldots\ \ldots\ldots\ \ldots\ldots \\ \ldots\ldots\ \ldots & \ldots\ldots\ \ldots\ldots\ \ldots\ldots \\ 0\,0\,0\,\ldots\,1 & p_{k1}\ p_{k2}\ \cdots\ p_{k,n-k} \end{array} \right\rangle.$$

одинична підматриця I_{kxk} *перевірна підматриця* $P_{kx(n-k)}$

лінійного блокового систематичного (n, k)- коду визначає кожне кодове слово як лінійну комбінацією рядків матриці G, а кожну лінійну комбінація рядків G - кодовим словом.

Для блока повідомлення m=(m_1, m_2, ... , m_k) кодове слово є послідовність

$$u=m\times G,$$

де для i=1, 2, ..., k $u_i = m_i$;

для i=k+1, ..., n $u_i = m_1p_{1i} + m_2p_{2i} + ... + + m_kp_{ki}$;

i=1, 2, ..., n-k - номер стовпчика перевірної частини $P_{k\times(n-k)}$ твірної матриці $G_{k\times n}$.

Наприклад, для твірної матриці

$$\boldsymbol{G}_{4\times 7} = \begin{pmatrix} 1\,0\,0\,0 & 1\,1\,0 \\ 0\,1\,0\,0 & 0\,1\,1 \\ 0\,0\,1\,0 & 1\,1\,1 \\ 0\,0\,0\,1 & 1\,0\,1 \end{pmatrix}.$$

кодові слова визначають як інформаційні і перевірні розряди

$$\boldsymbol{u}=\boldsymbol{m}\times\boldsymbol{G}=(\boldsymbol{m}_1, \boldsymbol{m}_2, \boldsymbol{m}_3, \boldsymbol{m}_4)\times \begin{pmatrix} 1\,0\,0\,0 & 1\,1\,0 \\ 0\,1\,0\,0 & 0\,1\,1 \\ 0\,0\,1\,0 & 1\,1\,1 \\ 0\,0\,0\,1 & 1\,0\,1 \end{pmatrix} =(\boldsymbol{m}_1, \boldsymbol{m}_2, \boldsymbol{m}_3, \boldsymbol{m}_4, \boldsymbol{m}_1+\boldsymbol{m}_3+\boldsymbol{m}_4, \boldsymbol{m}_1+\boldsymbol{m}_2+\boldsymbol{m}_3, \boldsymbol{m}_2+\boldsymbol{m}_3+\boldsymbol{m}_4).$$

9.4 Правила побудови твірної матриці коду

Основними правилами побудови твірної матриці лінійного блокового (n, k)- коду є:

1) кількість початкових кодових комбінацій (число рядків) твірної матриці дорівнює k, тобто кількості інформаційних елементів;

2) усі кодові комбінації твірної матриці повинні відрізнятися, але нульова комбінація до їх складу не входить;

3) усі кодові комбінації твірної матриці лінійно незалежні;

4) кількість одиниць в кожній кодовій комбінації твірної матриці повинна бути не меншою за d_{min};

5) кодова відстань між будь-якою парою кодових слів повинна бути не меншою за d_{min}.

ЛЕКЦІЯ 10 ВИЯВЛЕННЯ І ВИПРАВЛЕННЯ ПОМИЛОК ЛІНІЙНИМ БЛОКОВИМ КОДОМ

10.1 Будова перевірної матриці

Перевірна матриця $H_{(n-k)\times n}$ розмірності (n-k)×n має властивість: для кодового слова u $u\times H^T = 0$, тобто ортогональна будь-якій кодовій послідовності даного коду і має структуру

$$\underline{\boldsymbol{H}}_{(n-k)\times n} = \left\langle \begin{array}{cccc|c} p_{11} & p_{21} & \cdots & p_{k1} & 1\,0\,0\ldots 0 \\ p_{12} & p_{22} & \cdots & p_{k2} & 0\,1\,0\ldots 0 \\ \cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots \\ \cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots \\ p_{1n-k} & p_{2n-k} & \cdots & p_{k,n-k} & 0\,0\,0\ldots 1 \end{array} \right\rangle, \qquad)$$

$$\underbrace{\boldsymbol{P}}_{(n-k)\times k}\overbrace{\underbrace{\boldsymbol{I}}_{(n-k)\times(n-k)}}^{T}$$

де $\boldsymbol{P}_{(n-k)\times k}^{T}$ - транспонована перевірна підматриця твірної матриці $G_{k\times n}$; $\boldsymbol{I}_{(n-k)\times(n-k)}$ - одинична підматриця.

10.2 Визначення належності кодової послідовності коду

За допомогою перевірної матриці можна визначити, чи є прийнята послідовність кодовим словом даного коду. Наприклад, для матриці кода (7, 4)

$$\underline{\boldsymbol{H}}_{3\times 7} = \left(\begin{array}{cccc|ccc} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{array}\right).$$

і послідовності y=(1011001)отримаємо

$$\boldsymbol{y}\times\underline{\boldsymbol{H}}^{\mathrm{T}}=(\mathbf{1011001})\times\left(\begin{array}{c|c}1011 & 100\\ 1110 & 010\\ 0111 & 001\end{array}\right)^{\mathrm{T}}=(\mathbf{1\ 0\ 1})\neq\mathbf{0}.$$

Тобто y=(1011001) не є кодовим словом даного коду.

10.3 Визначення вектора помилок

Перевірні матриці використовують для перевірки наявності помилок $e=(e_1, e_2, \ldots, e_n)$ у повідомленнях $u=(u_1, u_2, \ldots, u_n)$, що надходять до каналу зв'язку. На виході канала зв'язку з шумами отримують послідовність y = u+e, де u - передане кодове слово; e – вектор помилок у каналі. Вектор помилок $e=(e_1, e_2, \ldots, e_n)$, є двійковою послідовністю завдовжки n з одиницями у тих позиціях, де виникли помилки.

Наприклад, вектор помилок e=(0001000) означає однократну помилку у четвертому біті, e=(1100000) - двократну помилку у першому і другому бітах. Для u=(0001000), e=(0001000), y=(0000000).

10.4 Визначення кодового синдрома помилок

Для перевірки наявності помилок у прийнятій послідовності у, декодер обчислює кодовий синдром S як (n-k)- послідовність

$$\boldsymbol{S}=(\boldsymbol{S_1}, \boldsymbol{S_2}, \ldots, \boldsymbol{S_{n-k}})=\boldsymbol{y}\times \boldsymbol{H}^{\mathrm{T}},$$

де у - прийнята кодована послідовність;

$\boldsymbol{H}^{\mathrm{T}}$ - транспонована перевірна матриця коду.

Послідовність у є кодовим словом при S=(0 0 … 0), і не є кодовим словом за умови S≠0.

Наприклад, для у=(y_1, y_2, …, y_7) синдром

$$\boldsymbol{S}=\boldsymbol{y}\times \boldsymbol{H}^{\mathrm{T}}_{3\times 7}=(\boldsymbol{y_1}, \boldsymbol{y_2}, \ldots, \boldsymbol{y_7})\times \begin{pmatrix} 1\,0\,1\,1\,1\,0\,0 \\ 1\,1\,1\,0\,0\,1\,0 \\ 0\,1\,1\,1\,0\,0\,1 \end{pmatrix}^{\mathrm{T}} =((\boldsymbol{y_1}+\boldsymbol{y_3}+\boldsymbol{y_4}+\boldsymbol{y_5}), (\boldsymbol{y_1}+\boldsymbol{y_2}+\boldsymbol{y_3}+\boldsymbol{y_6}), (\boldsymbol{y_2}+\boldsymbol{y_3}+\boldsymbol{y_4}+\boldsymbol{y_7})).$$

Декодер не виявляє помилки, для яких синдром S=у$\times \boldsymbol{H}^{\mathrm{T}}$=0.

Кодовий синдром залежить лише від вектора помилок і не залежить від переданого слова, оскільки

$$\boldsymbol{S}=\boldsymbol{y}\times \boldsymbol{H}^{\mathrm{T}}=(\boldsymbol{u}+\boldsymbol{e})\times \boldsymbol{H}^{\mathrm{T}}=\boldsymbol{u}\times \boldsymbol{H}^{\mathrm{T}}+\boldsymbol{e}\times \boldsymbol{H}^{\mathrm{T}}=\boldsymbol{0}+\boldsymbol{e}\times \boldsymbol{H}^{\mathrm{T}}=\boldsymbol{e}\times \boldsymbol{H}^{\mathrm{T}},$$

10.5 Виправлення помилок на основі синдрома

За наявним вектором e можна відновити кодове слово: u*=y+e. Наприклад для e_4=(**0001000**)

$$e_4 \times H^{\mathrm{T}} = (0001000) \times \begin{pmatrix} 1011100 \\ 1110010 \\ 0111001 \end{pmatrix}^{\mathrm{T}} = (101) \text{ - помилка у 4-му біті}$$

Помилку можна вилучити за u*=y+e.

Існує однозначна відповідність між координатами поодиноких помилок і їх синдромами, тобто, знаючи синдром, можна визначити позицію коду, в якій виникла помилка.

Наприклад, з перевірної матриці лінійного блокового (7,4)- коду знайдемо систему перевірних рівнянь:

$$u \times H_{3\times 7}^{\mathrm{T}} = (u_1, u_2, \ldots, u_7) \times H_{3\times 7}^{\mathrm{T}} = (m_1, m_2, m_3, m_4, r_1, r_2, r_3) \times$$

$$\times \underbrace{\begin{pmatrix} 1011100 \\ 1110010 \\ 0111001 \end{pmatrix}^{\mathrm{T}}}_{m_1 m_2 m_3 m_4 r_1 r_2 r_3} = (r_1 + m_1 + m_3 + m_4, r_2 + m_1 + m_2 + m_3,$$

$$r_3 + m_2 + m_3 + m_4) = (0\,0\,0).$$

Звідси

$$\begin{cases} \boldsymbol{u}_5 = \boldsymbol{r}_1 = \boldsymbol{m}_1 + \boldsymbol{m}_3 + \boldsymbol{m}_4; \\ \boldsymbol{u}_6 = \boldsymbol{r}_2 = \boldsymbol{m}_1 + \boldsymbol{m}_2 + \boldsymbol{m}_3; \\ \boldsymbol{u}_7 = \boldsymbol{r}_3 = \boldsymbol{m}_2 + \boldsymbol{m}_3 + \boldsymbol{m}_4; \end{cases}$$

де (m_1, m_2, m_3, m_4) – інформаційні символи;

r_1, r_2, r_3 - перевірні символи;

$(u_1, u_2, \ldots, u_7)=(m_1, m_2, m_3, m_4, r_1, r_2, r_3)$ - кодове слово.

Виникнення помилки у кодовій комбінації призведе до невиконання тих рівнянь у системі перевірних рівнянь коду, в які входить значення помилкового розряду.

Наприклад, якщо помилка у четвертому розряді кодової комбінації

$$u=(u_1, u_2, u_3, u_4, u_5, u_6, u_7)=(m_1, m_2, m_3, m_4, r_1, r_2, r_3),$$

то не виконаються перше і третє рівняння, в які входить помилковий символ $u_4=m_4$. Вектором синдрому буде послідовність S=(101), що збігається з четвертим стовпцем перевірної матриці коду H.

У такий спосіб номер помилкового розряду кодової комбінації є номером стовпчика перевірної матриці H, що збігається з вектором синдрому. Це дозволяє визначити місце виникнення помилки і таким чином її виправити.

Кодовий синдром виправляє тільки однократні помилки в коді. Наприклад, e=(0100010) і

$$\boldsymbol{S}=(0100010)\times\begin{pmatrix} 1011100 \\ 1110010 \\ 0111001 \end{pmatrix}^{\mathrm{T}}=(001).$$

Цей синдром збігається з сьомим стовпцем перевірної матриці H, що означає наявність однієї помилки у сьомому біті. Отже, декодер не тільки не виправить помилкові біти, але і внесе помилку у позицію, де її не було. Тобто лінійний блоковий (7,4) - код не виправляє двократні помилки і помилки більшої кратності.

ЛЕКЦІЯ 11 КОДИ ХЕММІНГА

11.1 Основні поняття

Коди Хеммінга є узагальненням (7,4) лінійних кодів з виправленням помилок Хеммінга, запропонованих у 1950 р. Коди спроможні виявляти до двох і виправляти до однієї помилки у бітах повідомлення. Коди часто застососовують у пристроях пам'яті. Для кодів Хеммінга належать коди з мінімальною кодовою відстанню $d_{min}=3$, здатні виправляти поодинокі помилки.

11.2 Умова виявлення однократних помилок

Для повідомлення довжини n за однократних помилок можливим є n ситуацій виникнення помилки. Тому для визначення місця помилки необхідними є наявність 2^r комбінацій перевірних елементів, які враховують всі можливі помилкові ситуації в коді плюс ситуація, коли помилки немає

$$2^r \geq n+1 .$$

Тобто мінімальна кількість перевірних елементів r задовольняє співвідношенню

$$2^r - 1 = n .,$$

а кількість інформаційних елементів k=n-r.

11.3 Будова перевірної матриці Хеммінга

Хеммінгом запропоновано розміщувати стовпці перевірної матриці $\boldsymbol{H}_{(n-k)\times n}$ так, щоб і-й стовпець матриці, що відповідає номеру і помилкового розряду кодової комбінації, був двійковим поданням цього номера. Тоді синдром виправлення однократних помилок кодом Хеммінга є двійковим поданням номера розряду, в якому виникла помилка. Для цього перевірні розряди, що відповідають стовпцям одиничної підматриці перевірної матриці коду, повинні знаходитися не в правій частині кодового слова, а у позиціях з номерами степеня двійки, тобто 2^0, 2^1, 2^2,…, 2^{r-1}.

Наприклад, для r=3 перевірна матриця Хеммінга має вигляд

$$\boldsymbol{H}_{3\times 7} = \begin{pmatrix} 0\,0\,0\,1\,1\,1\,1 \\ 0\,1\,1\,0\,0\,1\,1 \\ 1\,0\,1\,0\,1\,0\,1 \end{pmatrix}.$$

Для даного лінійного блокового (7, 4)- коду перший, другий, четвертий розряди (u_1, u_2, u_4) будуть перевірними, а третій, п'ятий, шостий і сьомий розряди (u_3,u_5,u_6,u_7) - символами інформаційного повідомлення у звичайному порядку, тобто (m_1,m_2 ,m_3,m_4) відповідно.

Отже, для (n, k)- коду Хеммінга в кожному кодовому слові u=(u_1, u_2, u_3, u_4, …, u_8, … u_n), r=n-k бітів з номерами степеня двійки є перевірними, а інші – бітами повідомлення, тобто кодування здійснюється перетворенням

$E(m_1, m_2, \ldots, m_k) = (u_1, u_2, \ldots, u_n) = (r_1, r_2, m_1, r_3, m_2, m_3, m_4, r_4, m_5, m_6, \ldots, m_k)$.

11.4 Типові перевірні матриці (n, k)- коду Хеммінга

Перевірна матриця (n, k)- коду Хеммінга складається із r рядків і $\boldsymbol{n} \le 2^r - 1$ стовпців, що є двійковими комбінаціями числа і, де і - номер стовпця. Наприклад, для r=2, 3, 4 перевірні матриці коду Хеммінга мають вигляд

$$\boldsymbol{H}_{2\times 3} = \begin{pmatrix} 0\,1\,1 \\ 1\,0\,1 \end{pmatrix}; \qquad \boldsymbol{H}_{3\times 7} = \begin{pmatrix} 0\,0\,0\,1\,1\,1\,1 \\ 0\,1\,1\,0\,0\,1\,1 \\ 1\,0\,1\,0\,1\,0\,1 \end{pmatrix};$$

$$\boldsymbol{H}_{4\times 15} = \begin{pmatrix} 0\,0\,0\,0\,0\,0\,0\,1\,1\,1\,1\,1\,1\,1\,1 \\ 0\,0\,0\,1\,1\,1\,1\,0\,0\,0\,0\,1\,1\,1\,1 \\ 0\,1\,1\,0\,0\,1\,1\,0\,0\,1\,1\,0\,0\,1\,1 \\ 1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1 \end{pmatrix}.$$

Синдром, що визначає систему перевірних рівнянь коду, знаходиться із рівняння

$$u \times \boldsymbol{H}^{\mathrm{T}} = 0.$$

Наприклад, для r = 3 система перевірних рівнянь буде отримуємо

$$\begin{cases} \underline{\boldsymbol{u}_4} + \boldsymbol{u}_5 + \boldsymbol{u}_6 + \boldsymbol{u}_7 = 0, \\ \underline{\boldsymbol{u}_2} + \boldsymbol{u}_3 + \boldsymbol{u}_6 + \boldsymbol{u}_7 = 0, \\ \underline{\boldsymbol{u}_1} + \boldsymbol{u}_3 + \boldsymbol{u}_5 + \boldsymbol{u}_7 = 0. \end{cases}$$

Звідси визначають перевірні розряди (контрольні суми)

$$\begin{cases} \boldsymbol{r}_1 = \boldsymbol{u}_1 = \boldsymbol{u}_3 + \boldsymbol{u}_5 + \boldsymbol{u}_7 = \boldsymbol{m}_1 + \boldsymbol{m}_2 + \boldsymbol{m}_4, \\ \boldsymbol{r}_2 = \boldsymbol{u}_2 = \boldsymbol{u}_3 + \boldsymbol{u}_6 + \boldsymbol{u}_7 = \boldsymbol{m}_1 + \boldsymbol{m}_3 + \boldsymbol{m}_4, \\ \boldsymbol{r}_3 = \boldsymbol{u}_4 = \boldsymbol{u}_5 + \boldsymbol{u}_6 + \boldsymbol{u}_7 = \boldsymbol{m}_2 + \boldsymbol{m}_3 + \boldsymbol{m}_4. \end{cases}$$

Інформаційні розряди повідомлення кодують ($m \to u$) так, що значення розрядів u_i є значеннями послідовності m_k і мають індекси і, які не є степенню двійки, а перевірні розряди з індексами 2^j знаходять з системи перевірних рівнянь. У кожне рівняння системи входить тільки одна контрольна сума.

Наприклад, повідомлення m=(0111) (7, 4)-кодом Хеммінга кодують кодовим словом $u=(u_1\ u_2\ 0\ u_4\ 1\ 1\ 1)$, де u_1, u_2, u_4 - контрольні суми r_1, r_2, r_3 відповідно. Контрольні суми отримують з системи перевірних рівнянь

$$r_1=u_1=u_3+u_5+u_7=m_1+m_2+m_4=0+1+1=0,$$

$$r_2=u_2=u_3+u_6+u_7=m_1+m_3+m_4=0+1+1=0,\ r_3=u_4=u_5+u_6+u_7=m_2+m_3+m_4=1+1+1=1.$$

Отже, кодовим словом буде послідовність u=(0001111).

11.5 Декодування коду Хеммінга

Декодування коду Хеммінга здійснюється у такий спосіб. Обчислюють синдром прийнятої послідовності y

$$S=y\times \boldsymbol{H}^{T},$$

де $\boldsymbol{H}^{T}$ - перевірна матриця коду.

Якщо синдром є нульовим вектором, то вважають, що отримано слово y без помилок. За ненульового синдрома значення синдрому відповідає двійковому поданню номера помилкового розряду.

Тобто для виправлення помилки необхідно змінити значення помилкового інформаційного біта в отриманому повідомленні, визначивши номер біта за місцем розташування перевірного біта.

Наприклад, прийнято послідовність y=(0011111) пловідомлення, закодованоного (7, 4)- кодом Хеммінга. Для декодування послідовності y=(0011111) обчислюємо синдром прийнятого вектора

$$y\times \boldsymbol{H}^{T} = (0011111)\times \begin{pmatrix} 0\,0\,0\,1\,1\,1\,1 \\ 0\,1\,1\,0\,0\,1\,1 \\ 1\,0\,1\,0\,1\,0\,1 \end{pmatrix}^{T} = (011)_2=3_{10}$$

Тобто помилка виникла у третомі біті. Тому виправляємо помилку, змінюючи значення у помилковому біті: (0011111)→ (0001111). З виправленого повідомлення виділяємо інформаційні розряди

$$D(u_1, u_2, u_3, u_4, u_5, u_6, u_7)=D(0001111)=(0111).$$

11.6 Твірна матриця (n, k)-коду Хеммінга

Твірна матриця (n, k)-коду Хеммінга - це матриця розмірності (k×n), в якій стовпці з номерами, що не є степенями двійки, утворюють одиничну підматрицю, а решту стовпців визначають з перевірних рівнянь коду. Така матриця при кодуванні копіюватиме біти повідомлення у позиції з номерами не степеня 2 і заповнюватиме інші позиції коду згідно з перевірним рівнянням знаходження відповідного контрольного розряду.

Наприклад, для системи перевірних рівнянь (7, 4)- коду Хеммінга

$$r_1 = u_1 = u_3+u_5+u_7 = m_1+m_2+m_4,$$
$$r_2 = u_2 = u_3+u_6+u_7 = m_1+m_3+m_4,$$
$$r_3 = u_4 = u_5+u_6+u_7 = m_2+m_3+m_4.$$

відповідною твірною матрицею коду є матриця

$$\boldsymbol{G}_{4\times 7} = \begin{pmatrix} 1\,1\,1\,0\,0\,0\,0 \\ 1\,0\,0\,1\,1\,0\,0 \\ 0\,1\,0\,1\,0\,1\,0 \\ 1\,1\,0\,1\,0\,0\,1 \end{pmatrix}.$$

11.7 Приклади розв'язування задач

Приклад Закодувати повідомлення 11001010110 двійковим кодом Хеммінга. Побудувати твірну матрицю коду. Навести приклад виправлення помилки і декодування повідомлення. Визначити надлишковість коду.

Розв'язання

При використанні завадостійкого (n, k)-коду Хеммінга контрольні суми, що додаються до інформаційного повідомлення, необхідно розмістити не в правій частині кодового слова, а в позиціях цілого степеня двійки.

Тоді стовпці перевірної матриці коду $H_{T(n-k)\times n}$ будуть двійковим поданням номера розряду кодової комбінації, в якому виникла помилка.

У даному випадку кількість інформаційних елементів k=11.

Необхідна кількість перевірних розрядів r=4.

Отже, потрібно побудувати (15, 11) -код Хеммінга для заданого повідомлення.

Перевірна матриця (15, 11) -коду Хеммінга має вигляд

$$\boldsymbol{H}_{4\times 15} = \begin{pmatrix} 0\,0\,0\,0\,0\,0\,0\,1\,1\,1\,1\,1\,1\,1\,1 \\ 0\,0\,0\,1\,1\,1\,1\,0\,0\,0\,0\,1\,1\,1\,1 \\ 0\,1\,1\,0\,0\,1\,1\,0\,0\,1\,1\,0\,0\,1\,1 \\ 1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1 \end{pmatrix}.$$

Систему перевірних рівнянь, що визначає правила знаходження перевірних розрядів (контрольних сум), дістанемо з рівняння кодового синдрому

$$S=u\times H^T=0,$$

Де u – кодове слово лінійного блокового коду, заданого перевірною матрицею H.

$$S=(u_1, u_2, \ldots, u_{15})\times \begin{pmatrix} 0\,0\,0\,0\,0\,0\,0\,1\,1\,1\,1\,1\,1\,1\,1 \\ 0\,0\,0\,1\,1\,1\,1\,0\,0\,0\,0\,1\,1\,1\,1 \\ 0\,1\,1\,0\,0\,1\,1\,0\,0\,1\,1\,0\,0\,1\,1 \\ 1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1 \end{pmatrix}^{T} =$$

$$(u_8+u_9+u_{10}+u_{11}+ +u_{12}+u_{13}+u_{14}+u_{15},, u_4+u_5+u_6+u_7+u_{12}+u_{13}+u_{14}+u_{15},$$

$$u_2+u_3+u_6+u_7+ +u_{10}+u_{11}+u14+ u_{15},, u_1+ u_3+u_5+u_7+u_9+u_{11}+ u_{13}+u_{15}) = 0.$$

Звідси випливає система рівнянь

$$\begin{cases} u_1 + u_3 + u_5 + u_7 + u_9 + u_{11} + u_{13} + u_{15} = 0, \\ u_2 + u_3 + u_6 + u_7 + u_{10} + u_{11} + u_{14} + u_{15} = 0, \\ u_4 + u_5 + u_6 + u_7 + u_{12} + u_{13} + u_{14} + u_{15} = 0, \\ u_8 + u_9 + u_{10} + u_{11} + u_{12} + u_{13} + u_{14} + u_{15} = 0. \end{cases}$$

Перевірні розряди - це біти з номерами цілого степеня 2 – u_1, u_2, u_4, u_8, інші розряди - символи інформаційного повідомлення, тобто кодове слово будується так:

$$\underset{u_1}{(r_1}, \underset{u_2}{r_2}, \underset{u_3}{m_1}, \underset{u_4}{r_3}, \underset{u_5}{m_2}, \underset{u_6}{m_3}, \underset{u_7}{m_4}, \underset{u_8}{r_4}, \underset{u_9}{m_5}, \underset{u_{10}}{m_6}, \underset{u_{11}}{m_7}, \underset{u_{12}}{m_8}, \underset{u_{13}}{m_9}, \underset{u_{14}}{m_{10}}, \underset{u_{15}}{m_{11}})\text{,}$$

де $m=(m_1, m_2, \ldots, m_{11})$ – інформаційне повідомлення;

r1, r2, r3, r4 – контрольні суми.

Система перевірних рівнянь коду має вигляд

$$\begin{cases} r_1 = m_1 + m_2 + m_4 + m_5 + m_7 + m_9 + m_{11}, \\ r_2 = m_1 + m_3 + m_4 + m_6 + m_7 + m_{10} + m_{11}, \\ r_3 = m_2 + m_3 + m_4 + m_8 + m_9 + m_{10} + m_{11}, \\ r_4 = m_5 + m_6 + m_7 + m_8 + m_9 + m_{10} + m_{11}. \end{cases}$$

Закодуємо задане повідомлення:

$$\underset{m_1\,m_2\,m_3\,m_4\,m_5\,m_6\,m_7\,m_8\,m_9\,m_{10}\,m_{11}}{(1\,1\,0\,0\,1\,0\,1\,0\,1\,1\,0)} \rightarrow \underset{u_1\,u_2\,u_3\,u_4\,u_5\,u_6\,u_7\,u_8\,u_9\,u_{10}\,u_{11}\,u_{12}\,u_{13}\,u_{14}\,u_{15}}{(r_1\,r_2\,1\,r_3\,1\,0\,0\,r_4\,1\,0\,1\,0\,1\,1\,0)}.$$

Отримаємо контрольні суми з системи перевірних рівнянь:

r_1=1+1+0+1+1+1+0=1;

r_2=1+0+0+0+1+1+0=1;

r_3=1+0+0+0+1+1+0=1;

r_4=1+0+1+0+1+1+0=0.

Тобто закодована послідовність буде

(11001010110)→ (111110001010110).

Нехай у цьому кодовому слові при передачі виникла помилка, наприклад, прийнятий вектор y=(111110101010110).

На приймальній стороні обчислюємо синдром

$S = y \times H^T$ = (111110101010110)×

$$\times \begin{pmatrix} 0000000011111111 \\ 0001111000001111 \\ 0110011001100 11 \\ 1010101010101 01 \end{pmatrix}^{T} = (0111).$$

Ненульове значення кодового синдрому означає наявність помилки у сьомому розряді, оскільки випадку $(0111)_2=7_{10}$. Виправляємо значення помилкового біта і декодуємо кодову послідовність, вилучивши з неї контрольні суми:

(111110101010110)→(111110001010110)→(11 001010110).

У твірній матриці стовпчики з номерами, що не є 2^j j=0, 1, 2, … відповідали інформаційним елементам повідомлення, тобто утворювали одиничну підматрицю, а стовпці з номерами 2^j (перевірна частина матриці) визначалися рівняннями знаходження відповідних контрольних сум.

У такий спосіб, твірна матриця (15, 11) -коду Хеммінга має вигляд

$$\boldsymbol{G}_{11\times15} = \begin{pmatrix} 111000000000000 \\ 100110000000000 \\ 010101000000000 \\ 110100100000000 \\ 100000011000000 \\ 010000010100000 \\ 110000010010000 \\ 000100010001000 \\ 100100010000100 \\ 010100010000010 \\ 110100010000001 \end{pmatrix}.$$

Кодові слова лінійного блокового коду знаходяться множенням інформаційного повідомлення на твірну матрицю

$$u=(m_1, m_2, ..., m_{11})\times \begin{pmatrix} 1\,1\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0 \\ 1\,0\,0\,1\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0 \\ 0\,1\,0\,1\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0 \\ 1\,1\,0\,1\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0 \\ 1\,0\,0\,0\,0\,0\,0\,1\,1\,0\,0\,0\,0\,0\,0 \\ 0\,1\,0\,0\,0\,0\,0\,1\,0\,1\,0\,0\,0\,0\,0 \\ 1\,1\,0\,0\,0\,0\,0\,1\,0\,0\,1\,0\,0\,0\,0 \\ 0\,0\,0\,1\,0\,0\,0\,1\,0\,0\,0\,1\,0\,0\,0 \\ 1\,0\,0\,1\,0\,0\,0\,1\,0\,0\,0\,0\,1\,0\,0 \\ 0\,1\,0\,1\,0\,0\,0\,1\,0\,0\,0\,0\,0\,1\,0 \\ 1\,1\,0\,1\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,1 \end{pmatrix} =$$

$$=(\boldsymbol{m}_1+\boldsymbol{m}_2+\boldsymbol{m}_4+\underset{r_1}{\overset{u_1}{\boldsymbol{m}_5}}+\boldsymbol{m}_7+\boldsymbol{m}_9+\boldsymbol{m}_{11}, \boldsymbol{m}_1+\boldsymbol{m}_3+\boldsymbol{m}_4+\underset{r_2}{\overset{u_2}{\boldsymbol{m}_6}}+\boldsymbol{m}_7+\boldsymbol{m}_{10}+\boldsymbol{m}_{11},$$

$$\overset{u_3}{\boldsymbol{m}_1}, \boldsymbol{m}_2+\boldsymbol{m}_3+\boldsymbol{m}_4+\underset{r_3}{\overset{u_4}{\boldsymbol{m}_8}}+\boldsymbol{m}_9+\boldsymbol{m}_{10}+\boldsymbol{m}_{11}, \overset{u_5}{\boldsymbol{m}_2}, \overset{u_6}{\boldsymbol{m}_3}, \overset{u_7}{\boldsymbol{m}_4},$$

$$\boldsymbol{m}_5+\boldsymbol{m}_6+\boldsymbol{m}_7+\underset{r_4}{\overset{u_8}{\boldsymbol{m}_8}}+\boldsymbol{m}_9+\boldsymbol{m}_{10}+\boldsymbol{m}_{11}, \overset{u_9}{\boldsymbol{m}_5}, \overset{u_{10}}{\boldsymbol{m}_6}, \overset{u_{11}}{\boldsymbol{m}_7}, \overset{u_{12}}{\boldsymbol{m}_8}, \overset{u_{13}}{\boldsymbol{m}_9}, \overset{u_{14}}{\boldsymbol{m}_{10}}, \overset{u_{15}}{\boldsymbol{m}_{11}}).$$

Звідси випливає система перевірних рівнянь коду.

Надлишковість коду

$$\rho_к = 1-\frac{k}{n}=1-\frac{11}{15}=\frac{4}{15}\approx 0{,}267 .$$

ЛЕКЦІЯ 12 ПОЛІНОМІАЛЬНЕ КОДУВАННЯ ІНФОРМАЦІЇ

12.1 Поліноміальні коди

12.1.1 Поліноміальне подання лінійного блокового (n, k)-коду

Лінійний блоковий (n, k)-код можна подати коефіцієнтами полінома

$$u(x) = u_0 + u_1 x + u_2 x^2 + \ldots + u_{n-1} x^{n-1},$$

В полі двійкових символів GF(2) суму двох поліномівf(x) і g(x) визначають як поліном з GF(2)

$$f(x) + g(x) = \sum_{i=0}^{n-1} (f_i + g_i) x^i,$$

тобто додавання виконують як операцію додавання за модулем два коефіцієнтів при однакових степенях x.

Добуток двох поліномів f(x) і g(x) в полі двійкових символів GF(2) є поліномом з GF(2)

$$f(x) \cdot g(x) = \sum_{i=0}^{n-1} (\sum_{j=0}^{i} f_i g_{i-j}) x^i,$$

тобто знаходять як добуток поліномів за правилом множення степеневих функцій, а коефіцієнти однієї степені додаються за модулем два.

Наприклад:

$$x^3 + x^2 + 0 \cdot x + 1$$

$$\begin{array}{r} \times\ x^2 + x \\ \hline x^4 + x^3 + 0\cdot x^2 + x \\ +\ x^5 + x^4 + 0\cdot x^3 + x^2 \\ \hline x^5 + 0\cdot x^4 + x^3 + x^2 + x = x^5 + x^3 + x^2 + x. \end{array}$$

Ділення поліномів виконують за правилами ділення степеневих функцій, а операція віднімання замінюють додаванням за модулем два.

Для кожної пари поліномів c(x) і d(x) (d(x)≠0) існує єдина пара поліномів q(x), ρ(x), для яких

$$c(x)= q(x)\cdot d(x) + \rho(x),$$

де q(x) – частка,

ρ(x) - залишок від ділення

Наприклад:

$$\begin{array}{l} x^4 + 0\cdot x^3 + x^2 + x + 1 \quad | \quad x + 1 \\ +\ x^4 + x^3 \quad\quad\quad\quad\quad\quad\ \ | \quad x^3 + x^2 + 1 \\ \hline x^3 + x^2 + x + 1 \\ +\ x^3 + x^2 \\ \hline x + 1 \\ +\ x + 1 \\ \hline 0 \end{array}$$

0 – залишок ρ(x).

12.1.2 Поліноміальний (n, k)-код

Поліноміальним кодом називають множину всіх многочленів степені не більше n-1, що мають спільний множник – деякий фіксований многочлен g(x) степеня r=n-k. Цей многочлен g(x) називаютья твірним многочленом коду.

Поліноміальний код з твірним многочленом g(x) кодує повідомлення m(x) поліномом

$$u(x)=m(x)\cdot g(x)=u_0 + u_1x + u_2x^2 + \ldots + u_{n-1}x^{n-1},$$

або кодовим словом з коефіцієнтів цього многочлена $u= (u_0,u_1,..,u_{n-1})$.

12.1.3 Будова і використання твірної матриці поліноміального коду

Матриця $\boldsymbol{G}_{k\times n}$ поліноміального коду з твірним многочленом g(x) степеня r=n-k має вигляд

$$\boldsymbol{G}_{k\times n}=\begin{pmatrix} \boldsymbol{g}_0\ \boldsymbol{g}_1\ \boldsymbol{g}_2 \cdots \boldsymbol{g}_r \ldots\ldots\ldots\ldots 0 \\ 0\ \ \boldsymbol{g}_0\ \boldsymbol{g}_1 \cdots \boldsymbol{g}_{r-1}\ \boldsymbol{g}_r \ \ \ldots\ \ \ 0 \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ 0\ \ 0\ \ 0\ \ldots\ldots\ldots\ldots\ldots\ldots\ \boldsymbol{g}_r \end{pmatrix},$$

Ненульові елементи в і-му рядку є послідовністю коефіцієнтів твірного многочлена, розташованих з j-го по (j+r)-й стовпець.

Приклад Поліноміальний (5, 8)- код задано твірним многочленом $g(x)=1 + x^2 + x^3$. Закодуємо за його допомогою повідомлення (01011).

Повідомлення (01011) можна подати многочленом

$$m(x)= 0\cdot x^0+1\cdot x+0\cdot x^2+1\cdot x^3+ 1\cdot x^4 = x + x^3 + x^4$$

Тоді кодовим словом буде поліном

$$u(x)= m(x)\cdot g(x) = (x + x^3 + + x^4)\cdot(1 + x^2 + x^3) = x + x^3 + x^4 + x^3 +$$
$$+x^5+ x^6 +x^4+x^6+x^7=x+(1+1)\cdot x^3+(1+1)\cdot x^3+x^5+(1+1)\cdot x^6+x^7= x + x^5 + x^7.$$

Цьому поліному відповідає кодова послідовність u = (01000101).

Поліноміальнийо (5, 8)- код можна також подати за допомогою твірної матриці для твірного полінома $g(x)=1+x^2+x^3$

$$\boldsymbol{G}_{5\times 8} = \begin{pmatrix} 1\,0\,1\,1\,0\,0\,0\,0 \\ 0\,1\,0\,1\,1\,0\,0\,0 \\ 0\,0\,1\,0\,1\,1\,0\,0 \\ 0\,0\,0\,1\,0\,1\,1\,0 \\ 0\,0\,0\,0\,1\,0\,1\,1 \end{pmatrix}$$

або відображення:

$$00000 \rightarrow 00000000,\ 00001 \rightarrow 00001011,\ 00010 \rightarrow 00010110,\ \ 01011 \rightarrow 01000101.$$

12.1.4 Визначення і виявлення помилок поліноміальним кодом

Вектор помилок $e=e_0, \ldots, e_{n-1}$ є не визначеним у тому і лише у тому випадку, коли його многочлен $e(x)=e_0+e_1x+\ldots+e_{n-1}x^{n-1}$ ділиться на твірний поліном коду g(x) без остачі. Тобто прийнята послідовність $c(x)=m(x)g(x)+e(x)$ ділиться на g(x) без остачі тоді і тільки тоді, коли e(x) ділиться на g(x) без остачі.

Тому помилка, многочлен якої не ділиться на g(x) можна визначити, а

помилка, многочлен якої ділиться на g(x), не можна визначити. Як наслідок, виявлення помилки поліноміальним кодом з твірним поліномом g(x) може бути здійснене за допомогою ділення многочленів. Якщо залишок від ділення многочлена прийнятої послідовності на твірний поліном g(x) ненульовий, то при передачі відбулося спотворення даних.

Якщо твірний поліном g(x) не є дільником жодного з многочленів вигляду x^{j+1}, де $j<n$, то мінімальна відстань між кодовими словами відповідного поліноміального коду $d_{min} \geq 3$.

12.2 Циклічні коди

12.2.1 Означення циклічного коду

До простих і поширених поліноміальних кодів відносять циклічні коди. Лінійний блоковий (n, k) - код називають циклічним, якщо в результаті циклічного зсуву кодового слова $u=(u_0, u_1, \ldots, u_{n-1})$ отримують нове кодове слово $v=(u_{n-1}, u_0, u_1, \ldots, u_{n-2})$, яке також належить даному коду.

Особливості застосування циклічних поліноміальних кодів визначають такі їх властивості: кожний ненульовий поліном повинен мати степінь в межах від (n-k) до n-1; існує лише один кодовий поліном $g(x)=1 + g_1x + g_2x^2 + \ldots + g_{n-k-1}\cdot x^{n-k-1} + x^{n-k}$ степені (n-k), що є дільником кожного кодового полінома $u(x)=m(x)\cdot g(x)$ і який називають твірним поліномом коду.

Кодове слово циклічного коду складається з перевірної частини з (n-k) перевірних символів і інформаційної частини m завдовжки k символів. Перевірні символи є коефіцієнтами полінома ρ(x) – остачі від ділення кодового слова $u(x)=m(x)\cdot x^{n-k}$ на твірний поліном g(x).

12.2.2 Алгоритм побудови циклічного (n, k)- коду

Алгоритм побудови циклічного (n, k)- коду для послідовності $m=(m_0, m_1, m_2, \dots, m_{k-1})$ такий:

1) многочлен інформаційної послідовності m(x) домножають на x^{n-k}, що забезпечує зсув праворуч на n-k розрядів;

2) отриманий поліном ділять на твірний поліном коду g(x);

3) остача від ділення $x^{n-k}\cdot m(x)$ на g(x) додається до $x^{n-k}\cdot m(x)$, тобто записується у молодших n-k розрядах.

Важлива властивість циклічного (n, k)- коду є те, що твірний поліном g(x) ділить без остачі двочлен x^n+1, тобто комбінації лінійного коду мають властивість циклічності за виконання умови

$$x^n+1=g(x)\cdot h(x).$$

Двочлен x^n+1 може бути розкладений на кілька незвідних поліномів, тобто поліномів, які не можуть бути подані як добуток многочленів меншого степеня і діляться або самі на себе, або на 1.

У якості твірних поліномів циклічних кодів використовують незвідні поліноми і їх добутки, оскільки вони є дільниками двочлена x^n+1. Поширеними є такі твірні поліноми (таблиця 12.1).

Поліном h(x) степеня k, що є часткою від ділення двочлена x^n+1 на твірний поліном коду g(x), називають перевірним поліномом. Оскільки h(x) однозначно зв'язаний з g(x), то він також визначає код. Це дозволяє дати інше визначення циклічного лінійного кода і сндрома.

Циклічним називають лінійний (n, k)- код, усі 2^k кодові комбінації якого подані поліномами степеня n-1 і менше, які діляться без остачі на деякий поліном g(x) степеня r=n-k, що є дільником двочлена x^n+1.

Для канала зв'язку позначимо через u(x) та y(x) поліноми, що відповідають переданому кодовому слову u і прийнятій послідовності y.

Розділивши поліном прийнятого повідомлення y(x) на твірний поліном коду g(x), отримаємо поліноми частки від ділення q(x) і остачі s(x), тобто

$$y(x)=q(x)g(x)+s(x).$$

Таблиця 12.1-Твірні поліноми

Твірний поліном g(x)	*Двійковий запис полінома*
$1+x$	11
$1+x+x^2$	111
$1+x+x^3$	1101
$1+x^2+x^3$	1011
$1+x+x^4$	11001
$1+x^3+x^4$	10011
$1+x+x^2+x^4$	11101
$1+x^2+x^3+x^4$	10111
$1+x+x^2+x^3+x^4$	11111
$1+x^2+x^5$	101001
$1+x^3+x^5$	100101
$1+x+x^2+x^3+x^5$	111101
$1+x+x^2+x^4+x^5$	111011
$1+x^4+x^5+x^6$	1000111
$1+x+x^2+x^5+x^6+x^7+x^8$	111001111
$1+x^3+x^5+x^9$	1001010001
$1+x+x^5+x^6+x^{10}+x^{11}$	110100100011
$1+x+x^2+x^3+x^5+x^7+x^8+x^{11}$	111101011001
$1+x^3+x^4+x^6+x^8+x^9+x^{10}+x^{11}$	100110101111
$1+x+x^2+x^3+x^4+x^{12}+x^{13}+x^{14}+x^{15}$	1111100000001111
$1+x^5+x^{12}+x^{16}$	10000100000010001

Якщо y(x) є кодовим поліномом, то він ділиться на g(x) без остачі, тобто s(x)=0. Ненульова остача s(x)≠0 є ознакою наявності помилки у прийнятій послідовності, тобто s(x) є кодовим синдромом.

Поліноми синдрому s(x) і вектора помилок в каналі e(x) мають вигляд

$$s(x)=s_0+s_1x+\ldots+s_{n-k-1}x^{n-k-1}.$$
$$e(x)=e_0+e_1x+\ldots+e_{n-1}x^{n-1}$$

Многочлен синдрому однозначно зв'язаний з многочленом помилок e(x), дозволяє не тільки виявляти, але й виправляти помилки у прийнятій послідовності, поліном y(x) якої визначають як

$y(x)=u(x)+e(x)$. І $u(x)+y(x)=e(x)$.

Оскільки

$y(x)=q(x)\cdot g(x) + s(x)$, $u(x)=m(x)\cdot g(x)$,

то

$e(x)=[m(x)+q(x)]\cdot g(x)+s(x)=f(x)\cdot g(x)+s(x)$,

тобто поліном синдрому s(x) – це остача від ділення многочлена помилки e(x) на твірний поліном коду g(x). Тому за поліномом синдрому можна однозначно визначити вектор помилок і виправити помилку.

ЛЕКЦІЯ 13 СТИСНЕННЯ СТАТИЧНИХ ЗОБРАЖЕНЬ

Існує кілька основних алгоритмів стиснення статичних зображень. Найбільш вживаними є алгоритми JPEG, JPEG2000, LS JPEG. В основі цих алгоритмів покладено перетворення вихідної інформації зображень з подальшим стисненням отриманих даних за відомими методами Хаффмена (за визначенням), алгоритмами LZ.

Розрізняють алгоритми стиснення зображень без втрат (наприклад, LS JPEG) і алгоритми із втратами (JPEG, JPEG2000). Повний аналіз алгоритмів стиснення зображень і можливих апаратних і програмних реалізацій у пристроях потребує більш ґрунтовної роботи з виявлення особливостей виконання окремих перетворень.

У даному розділі розглянуто основні принципи, які лежать в основі цих алгоритмів. Детальний їх аналіз і програмні реалізації алгоритмів стиснення можуть бути предметом самостійної наукової роботи студентів, яку враховують у системі рейтингових оцінок успішності навчання студента.

13.1 Алгоритм JPEG

13.1.1 Загальна характеристика

Алгоритм описує стискання повнокольорових графічних зображень з втратами. Алгоритм JPEG (Joint Photographic Expert Group) розроблено у рамках ISO як стандарт. При роботі алгоритму оперують з областями 8X8 пікселів з плавною зміною яскравості і кольору. JPEG використовує 24 бітове

подання на піксель (для кольорового або монохромного зображення), що набагато перевищує інші формати. Наприклад, у форматі GIF використовують лише 8 бітів на пік сель, тобто 256 кольорів.

Основою алгоритму є дискретні косиносоідальні перетворення (Discrete Cosine Transform - DCT) до матриці зображень для отримання нової матриці коефіцієнтів. Для отримання повного зображення зі стиснутого застосують зворотне перетворення.

Зображення розкладають за амплітудами певних частот. Частина з коефіцієнтів такого розкладу наближається до нуля. Виходячи з особливостей людського зору можна знехтувати цими складовими, що дозволяє виконати стиснення даних. Крім цього, частину коефіцієнтів при косинусоідальному розкладі можна замінити їх наближеннями. Процес наближення досягають квантуванням коефіцієнтів (quantization). У найпростішому випадку це досягається зсувом двійкового запису вправо.

Алгоритм JPEG є основою багатьох алгоритмів стискання рухомих зображень, зокрема MPEG (Moving or Motion-JPEG).

Розрізняють алгоритми JPEG з втратами стискання (lossy image compression) і без втрат. На рис. 13.1 подано модель прямого кодування зображення (Forward Discrete Cosine Transform- FDCT) JPEG з втратами стискання.

Кожний блок коефіцієнтів DCT складається з одного коефіцієнта Direct Current (DC) середньої інтенсивності у блоку і решткі 63 коефіцієнтів (Alternating Current - AC).

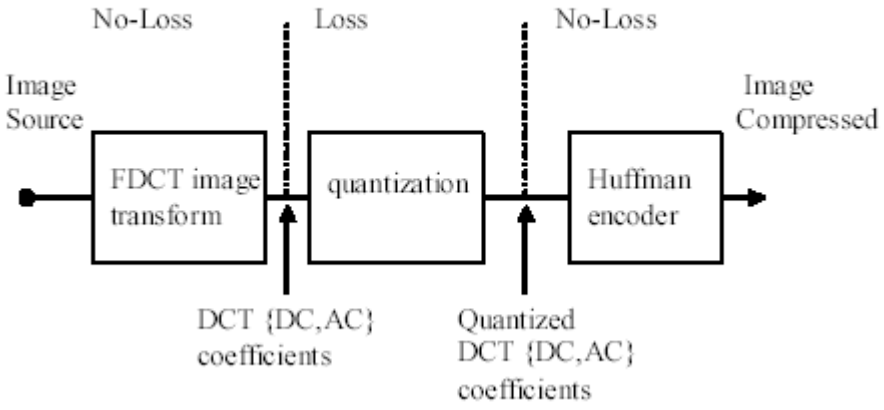


Рисунок 13.1 - Модель JPEG з втратами стискання

13.1.2 Основні етапи виконання алгоритму JPEG

Розглянемо порядок виконання алгоритму JPEG у 24 бітовому поданні пікселів трьох кольорів зображень, тобто 16 міліонів кольорів(рис. 13.2) .

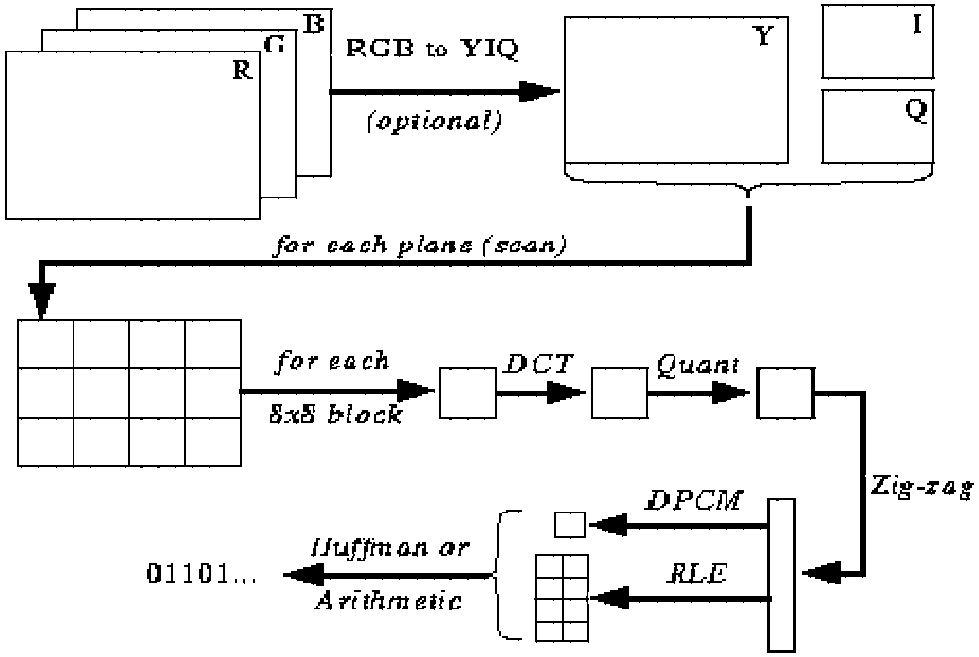


Рисунок 13.2 - Модель кодування JPEG

Крок 1.

Перетворюємо кольоровий простір RGB (червоний Red – R, зелений Green – G, та синій Blue – B) кольорової крапки у кольоровий простір YCrCb (відомий іноді як YUV). У просторі YCrCb величина Y є складовою яскравості, а складові Cr, Cb відповідають за колір (хроматично червоний і хроматично синій).

Зір людини є більш чутливим до яскравості, ніж до кольору. Тому масиви для Cr, Cb можна стискати (архівувати) з великими втратами, тобто з великими коефіцієнтами стискання. Зокрема, у телебаченні для цього використовують більш вузьку полосу частот.

Пряме перетворення з RGB у YCrCb відповідає матриці переходів

$$\begin{vmatrix} Y \\ Cb \\ Cr \end{vmatrix} = \begin{vmatrix} 0.299 & 0.587 & 0.114 \\ 0.5 & -0.4187 & -0.0813 \\ 0.1687 & -0.3313 & 0.5 \end{vmatrix} * \begin{vmatrix} R \\ G \\ B \end{vmatrix} + \begin{vmatrix} 0 \\ 128 \\ 128 \end{vmatrix}$$

А зворотне перетворення з YCrCb у RGB виконують за зворотною матрицею переходів

$$\begin{vmatrix} R \\ G \\ B \end{vmatrix} = \begin{vmatrix} 1 & 0 & 1.402 \\ 1 & -0.34414 & -0.71414 \\ 1 & 1.772 & 0 \end{vmatrix} * \left(\begin{vmatrix} Y \\ Cb \\ Cr \end{vmatrix} - \begin{vmatrix} 0 \\ 128 \\ 128 \end{vmatrix} \right)$$

Крок 2.

Вихідне зображення розділяють на матриці розміром 8x8. Для кожної матриці формуємо три робочі матриці DCT по 8 бітів окремо по кожній компоненті.

Для великих коефіцієнтів стискання зображення можна застосовувати більш складну процедуру. Наприклад, компоненти Y формують як матриці для кожного значення, а для компонентів Cr і Cb матриці набирають через рядок і через стовпчик.

Наприклад, для вихідної матриці 16x16 отримаємо одну робочу матрицю DCT. При цьому втрачається ¾ корисної інформації про кольорові складові зображення і отримуємо стискання у два рази у просторі YCrCb. На результуюче зображення RGB це суттєво не впливає.

Крок 3.

Застосовуємо DCT до кожної робочої матриці. В результаті отримаємо матрицю, яка у лівому верхньому куті має низькочастотні складові y[i,j], а у правому нижньому куті – високочастотні складові зображення.

Спрощено це можна подати як

$$Y[u,v] = \frac{1}{4}\sum_{i=0}^{n-1}\sum_{j=0}^{n-1} C(i,u)\times C(j,v)\times \quad y[i,j]$$

де

$$C(i,u) = A(u)\times \quad \cos\left(\frac{(2\times i+1)\times u\times \pi}{2n}\right)$$

$$A(u) = \begin{cases} \frac{1}{\sqrt{2}}, & \text{for} \quad u \equiv 0 \\ 1, & \text{for} \quad u \neq 0 \end{cases}$$

Крок 4.

Виконуємо квантування. Для цього ділимо робочу матрицю по елементам на матрицю квантування. Для кожного компоненту (Y, U и V) задаємо свою матрицю квантування q[u,v] (умовно МК) за виразом

$$Yq[u,v] = IntegerRound\left(\frac{Y[u,v]}{q[u,v]}\right)$$

На цьому кроці регулюємо коефіцієнти стискання, визначаючи ступінь стискання. Для МК з великими коефіцієнтами стискання отримуємо велику кількість нулів і відповідно більше стискання.

У стандарті JPEG пропонують множення на експериментально визначені значення числа gamma. При великих значеннях gamma на низьких частотах можуть бути великі втрати. Як результат є розбивка зображення на квадрати 8x8. А на високих частотах це призводить до утворення контуру навколо зображень з різким переходом у кольорі (ефект Гіббса).

Крок 5.

Переводимо матрицю 8x8 у 64 – елементний вектор з використанням вибору елементів зігзагом, як показано на рис. 13.1

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$	$a_{0,4}$	$a_{0,5}$	$a_{0,6}$	$a_{0,7}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	$a_{1,4}$	$a_{1,5}$	$a_{1,6}$	$a_{1,7}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$	$a_{3,0}$			
$a_{3,0}$	$a_{3,0}$	$a_{3,0}$	$a_{3,0}$				
$a_{4,0}$	$a_{4,1}$	$a_{4,2}$					
$a_{5,0}$	$a_{5,1}$						
$a_{6,0}$	$a_{6,1}$						
$a_{7,0}$	$a_{7,1}$						

Рисунок 13.1-Вибір елементів зигзагом

На початку вектора отримуємо коефіцієнти для малих частот, а в кінці – для великих частот.

Крок 6.

Згортаємо вектор за допомогою алгоритму групового кодування. Як результат отримуємо пари чисел (пропуск, число). Пропуск є кількість нулів,

які пропускають, число – значення, яке записуємо у наступну комірку. Наприклад, вектор 42 3 0 0 0 -2 0 0 0 0 1 … згортається у пари (0,42) (0,3) (3,-2) (4,1) ... .

Крок 7.

Згортаємо отримані пари кодуванням за Хаффменом з фіксованою таблицею.

Відновлення зображення виконують симетрично. Алгоритм дозволяє стискати зображення у 10-15 разів без суттєвих втрат.

13.1.3 Переваги і недоліки JPEG

Перевагами алгоритму є:

- можливість визначення ступені стискання;
- подання вихідного кольорового зображення у вигляді 24 біт на точку.

Недоліками алгоритму є:

- при підвищенному коефіцієнті стискання отримуємо розбивку зображення на квадрати 8x8, що пов’язано з великими втратами на низьких частотах;
- виявляються результати Гіббса як ореоли по границях різкого переходу кольорів;
- при друку можливо поява горизонтальних та вертикальних полос, які не проявляються на екрані дисплею.

Існують також проблеми перетворень з одного формату в інший, наприклад з 8 бітового GIF у 24-бітовий JPEG, і навпаки, оскільки втрати

відбуваються в обох випадках. Але виграш в стисканні в 3-20 разів за малих втратах якості часто переважає такі втрати.

Існує кілька варіантів реалізації алгоритму, зокрема за значеннями gamma. Зокрема існують модифікації алгоритму JPEG, зокрема, у форматах Quick Time, PostScript Level 2, Tiff 6.0.

13.2 Алгоритм RLE

Кодування довжин серій (Run-length encoding - RLE) або Кодування повторів— простий алгоритм стиснення даних, який оперує серіями даних, тобто послідовностями, в яких один і той же символ зустрічається кілька разів підряд. При кодуванні рядок однакових символів, що становлять серію, замінюється рядком, який містить сам повторюваний символ і кількість його повторів

Алгоритм відноситься до методів групового кодування. Алгоритм його роботи полягає в наступному: програма, що створює файл, зчитує значення окремих пікселів і запам'ятовує їх. Якщо програма знаходить підряд пікселі з однаковим значенням, вона не записує колірне значення пікселя ще раз, а просто запам'ятовує, скільки пікселів з цим значенням повторюються послідовно.

Бітове уявлення = 0,0,0,1,1,1,1,1,1,1,0,0,0,0,0,1,1,1,0,0,1,0,0,0,0,1,1,1,0,0,0,1

Результат стиснення: (3,0),(7,1),(5,0),(3,1),(2,0),1,(4,0),(3,1),(3,0),1. Пари у дужках визначають кількість повторень та значення байта.

У цьому методі можливі варіанти. Програма при створенні стиснутого файлу може спочатку записувати кількість пікселів у рядку і потім їхній колір, але може і навпаки. Це породжує деякі проблеми, якщо програма, що зчитує файл, очікує появи даних в іншому порядку, чим програма, що записує цей файл. Метод стиску RLE легко доступний для розуміння і порівняно просто реалізується, але він не є найефективнішим для більшості растрових файлів.

Метод RLE найкраще працює з зображеннями, у яких обмежена кількість кольорів і великі області однотонного фарбування, і гірше - з сканованими чи "фотореалістичними" малюнками, оскільки у них немає довгих рядків однакових пікселів, які можна стиснути. Коли кожен піксель зображення відрізняється від сусіднього, метод RLE не є ефективним, оскільки алгоритм стиску застосуються до кожного окремого пікселю, що не дозволяє одержати помітної зміни розміру файлу зображення і навіть розмір масиву з закодованим зображенням може бути більше вихідного.

Розглянемо зображення, що містить простий чорний текст на суцільному білому фоні. Тут буде багато серій білих пікселів в порожніх місцях, і багато коротких серій чорних пікселів в тексті. Нехай як приклад наведено якийсь довільний рядок зображення в чорно-білому варіанті. Тут B (англ. Black) представляє чорний піксель, а W (англ. White) позначає білий:

WWWWWWWWWWWWBWWWWWWWWWWWWBBBWWWWWW
WWWWWWWWWWWWWWWWWWBWWWWWWWWWWWWW

Якщо ми застосуємо просте кодування довжин серій до цього рядка, то отримаємо наступне:

12W1B12W3B24W1B14W

Останній запис інтерпретується як «дванадцять W», «одна B», «дванадцять W», «три B» і т. д. Таким чином, код представляє вихідні 67 символів у вигляді всього лише 18.

Однак, у випадку, якщо рядок складається з великої кількості неповторюваних символів, її обсяг може зрости.

ABCABCABCDDDFFFFFF
1A1B1C1A1B1C1A1B1C3D6F

Проблема вирішується досить просто. Алфавіт, в якому записані довжини серій, розділяється на дві (зазвичай рівні) частини. Алфавіт цілих чисел можна, наприклад, розділити на дві частини: позитивні і негативні. Позитивні використовують для запису кількості повторюваних однакових символів, а негативні — для запису кількості неоднакових.

-9ABCABCABC3D6F

Так як чисельні типи даних на комп'ютері завжди мають певну межу, виникає ще одна проблема. Припустимо, ми використовуємо тип signed char для запису довжин серій. Тоді ми не можемо записати серію довше 127 символів однією парою «довжина-символ». Якщо поспіль записано 256 символів A, їх розділяють на мінімальну кількість груп:

127A127A2A

Реалізація на конкретній мові програмування алгоритму RLE з урахуванням цих обмежень може бути нетривіальною.

Звичайно, кодування, яке використовується для зберігання зображень, оперує з двійковими даними, а не з символами ASCII, як у розглянутому прикладі, однак принцип залишається той же.

Таке кодування ефективно для даних, що містять велику кількість серій, наприклад, для простих графічних зображень, таких як іконки та графічні малюнки. Однак це кодування погано підходить для зображень з плавним переходом тонів, таких як фотографії.

Поширені формати для упаковки даних за допомогою RLE включають в себе PackBits, PCX і ILBM.

Методом кодування довжин серій можуть бути стислі довільні файли з двійковими даними, оскільки специфікації на формати файлів часто включають в себе повторювані байти в області вирівнювання даних. Проте, сучасні системи стиснення (наприклад, DEFLATE) частіше використовують алгоритми на основі LZ77, які є узагальненням методу кодування довжин серій і оперують з послідовностями символів виду «BWWBWWBWWBWW».

Звукові дані, які мають довгі послідовні серії байт (такі як низькоякісні звукові семпли) можуть бути стислі за допомогою RLE після того, як до них буде застосовано Дельта-кодування.

Нижче наведені програмні реалізації RLE.

Реалізація алгоритму мовою C/C++

```
#include <stdio.h>
#include <string.h>

int main()
{
    int cnt;
    char smb;
    char *code = new char [80];
    char *encode = new char [80];
    char *str = new char [80];
```

```
    scanf("%s", code);

    strcpy(encode, "");
    smb = code[0];
    cnt = 0;

    for (int i = 0; i <= strlen(code); i++) {
        if (code[i]==smb) {
            cnt++;
        }
        else {
            sprintf(str, "%d", cnt);
            strcat(encode, str);
            sprintf(str, "%c", smb);
            strcat(encode, str);
            smb = code[i];
            cnt = 1;
        }
    }

    printf("%s\n", encode);

    return 0;
}
```

Реалізація алгоритму мовою PHP

```
<?php
$code = 'fafaaaaaaaaaaaaa';
$encode = '';

for ($i = 0; $i < strlen($code);$i++){
      $smb = $code[$i] ;
      $count = 1 ;
      for ($b = $i; $b < strlen($code);$b++){
            if ($code[$b + 1] != $smb) break ;
            $count++ ;

            $i++ ;
      }
      $encode .= $count . $smb ;
}
```

```
      print 'Строку: ' . $code . ' вдалося зжати до ' . $encode .
'.<br> і ми заощадили ' . (strlen($code) - strlen($encode)) . '
байт.'
      ?>
```

Реалізації алгоритму на Visual Basic 6

```
      Public Function Encode(ByVal SrcString As String) As String
          Dim I As Long, N As Long, sResult As String
          N = 1

          For I = 1 To Len(SrcString)
              If Not Mid(SrcString, I, 1) = Mid(SrcString, I + 1,
1) Then
                  sResult = sResult & IIf(I - N + 1 > 1, CStr(I -
N + 1), CStr("")) & Mid(SrcString, I, 1)
                  N = I + 1
              End If
          Next

          Encode = sResult
      End Function
```

ЛЕКЦІЯ 14 СТИСКАННЯ МУЛЬТИМЕДІЙНИХ ДАНИХ

Мультимедійні дані передбачають одночасне подання кількох різновидів даних, зокрема відеоінформації, звуку, тексту. Несумісність цих даних за швидкостями утворення і передавання, різноманіття методів стиснення зумовлюють різні підходи до стиснення даних. Найбільш відомими алгоритмами подання і оперування з цими даними є стандарти MPEG. На їх основі створюють специфікації для виробників кодерів і декодерів цих даних.

14.1 Загальна характеристика MPEG

Стандарти MPEG (від абревіатури Moving Picture Experts Group) від ISO і IEC спрямовані на розробку єдиних норм кодування аудіо-і відеосигналів. Стандарти використовують у технологіях CD-i і CD-Video, DVD, у цифровому радіомовлення, у кабельному і супутниковому TV, Інтернет-радіо, мультимедійних комп'ютерних продуктах, в комунікаціях по каналах ISDN, багатьох інших електронних інформаційних системах.

14.2 Структура побудови зображень MPEG

У MPEG-стисненні використовують такі основні ідеї:

1) усунення тимчасової надмірності відео, що враховує той факт, що в межах коротких інтервалів часу більшість фрагментів сцени виявляються нерухомими або незначно зміщуються по полю;

2) усунення просторової надмірності зображень, зокрема зменшенням впливу дрібних деталей сцени, несуттєвих для візуального сприйняття людиною;

3) використання більш низького колірного дозволу при yuv-поданні зображень (y –яскравість, u і v - кольорорознісні сигнали);

4) підвищення інформаційної щільності результуючого цифрового потоку шляхом вибору оптимального математичного коду для його опису (наприклад, використання більш коротких кодових слів для найбільш часто повторюваних значень);

5) поділ зображень на послідовності зображень різних т ипів (I -intra, що грають роль опорних при відновленні інших зображень за їх разностям; P divdicted , що містять різниця поточного зображення з попереднім I або P з урахуванням зсувів окремих фрагментів; B - bidirectionally divdicted, що містять різниця поточного зображення з попереднім і наступним зображеннями типів I або P з урахуванням зсувів окремих фрагментів).

Зображення об'єднують у групи (GOP - Group Of Pictures), що являють собою мінімальний повторюваний набір послідовних зображень, які можуть бути декодовані незалежно від інших зображень у послідовності. Типовою є група виду (I0 B1 B2 P3 B4 B5 P6 B7 B8 P9 B10 B11) (I12 B13 B14 P15 B16 B17 P18 ...), в якій I тип повторюється кожні півсекунди. У зображенні P3 основну частину фрагментів сцени формують на основі відповідних зміщених фрагментів зображення I0. Власне кодуванню піддаються тільки різниці цих пар фрагментів. Аналогічно P6 будують на базі P3, а P9 - на основі P6 ощо.

Більшість фрагментів B1 і B2 обчислюють як полусумми зміщених фрагментів з I0 і P3, B4 і B5 - з P3 і P6, B7 і B8 - з P6 і P9 тощо. B-зображення не використовують для передбачення інших зображень.

Внаслідок залежності зображень у процесі їх кодування змінюється порядок їх проходження. Для наведеної послідовності він буде наступним: I0 P3 B1 B2 P6 B4 B5 P9 B7 B8 I12 B10 B11 P15 B13 B14 P18 B16 B17

Точність кодування має бути максимальною для I, нижче - для P, мінімальної - для B. Встановлено, що для типових сцен хороші результати досягаються при відведенні числа біт для I в 3 рази більше, ніж для P, і для P в 2-5 разів більше, ніж для B. Ці відносини зменшуються для динамічних сцен і збільшуються для статичних.

Окремі зображення складаються з макроблоків. Макроблок - це основна структурна одиниця фрагментації зображення. Він відповідає ділянці зображення розміром 16 * 16 пікселів. Саме для них визначаються вектора зміщення відносно I-або P-зображень. Загальне число макроблоків в зображенні - 396.

Для підвищення стійкості процесу відновлення зображень до можливих помилок передачі даних послідовні макроблоки об'єднують в незалежні один від одного розділи слайди (slices). У граничному випадку «чистої» передачі на зображення доводиться всього один розділ із 396 макроблоків.

Кожен макроблок складається з шести блоків, чотири з яких несуть інформацію про яскравість Y, а по одному визначають колірні U-і V-компоненти. Кожен блок являє собою матрицю 8 * 8 елементів. Блоки є базовими структурними одиницями, над якими здійснюються основні операції кодування, в тому числі виконується дискретне косинусне перетворення (DCT - Discrete Cosine Transform) і квантування отриманих коефіцієнтів.

14.3 Структура побудови зображень MJPEG

Існує стиснення MJPEG (Motion JPEG), яке грунтується на незалежному кодуванні кожного кадру та об'єднанні отриманої послідовності у файл. Стиснення відео здійснюють за JPEG-алгоритмом: кожне зображення розбивають на квадрати 8x8 точок і подають у векторній формі шляхом дискретного перетворення і високочастотної фільтрації отриманого спектра. Тобто стисле відео є послідовність незалежних JPEG-зображень.

Ступінь стиснення MJPEG відео складає до 15 і дозволяє зберігати відеоінформацію практично без втрати якості, від 15 до 25. Для великих коефіцієнтів стиснення (1:30 і вище) стиснення відео за алгоритмом MJPEG супроводжується характерними для формату JPEG спотвореннями: на кордонах сітки розбиття (квадрати 8x8 точок), порушується гладкість зображення (мозаїчний ефект). не дуже велика ефективність стиснення, неможливість створення відеофрагментів розміром більше 2 Гб,

В даний час застосовуються програмні методи "склеювання" MJPEG-файлів, що дозволяють перемикатися між ними практично непомітно. Кілька років тому компресія MJPEG стала стандартом в області мультимедіа, що спонукало розробників апаратного і програмного забезпечення до створення власних MJPEG-кодеків. Формат використовує просту обробку кодованого аналогового відеосигналу з роздільною здатністю 768x576 пікселів. Зараз цей формат практично не використовують, оскільки стислі ролики займають досить багато місця. У деяких моделях пристроїв (наприклад, фотокамерах з функцією відео) зустрічається спрощений варіант M-JPEG з роздільною здатністю 320x240 пікселів.

14.4 Характеристики мультимедійних відеоданих

Під якістю зображення зазвичай розуміють кількість відтворених вертикальних ліній. Якість відеоданих виміряють за допомогою формальних метрик, зокрема, PSNR, або з використанням суб'єктивного порівняння із залученням експертів.

Метрика PSNR (peak signal to noise ratio - пікове відношення сигналу до шуму, яке вимірюють у дБ) аналогічна середньоквадратичному відхиленню але за логарифмічним масштабом шкали.

Кількість (частота) кадрів на секунду - це кількість нерухомих зображень, що змінюють одне одне впродовж однієї секунди відеоматеріалу. Системи телебачення PAL і SECAM використовують 25 кадрів у секунду (25 fps), а система NTSC використовує 29,97 кадри на секунду.

Комп'ютерні оцифровані відеоматеріали високої якості, зазвичайя використовують частоту 30 кадрів на секунду. Верхня гранична частота мерехтіння, що сприймається людським мозком, в середньому становить 39 - 42 Герца й індивідуальна для кожної людини. Деякі сучасні професійні камери можуть знімати з частотою до 120 кадрів в секунду. А спеціальні камери для надшвидкої зйомки з частотою до 1000 кадрів в секунду дозволяють, наприклад, вивчати траєкторії польоту кулі або структури вибуху.

Розгортка відеоматеріалу може бути прогресивною або чересстрочной. При прогресивній розгортці всі горизонтальні лінії зображення (рядки) відображаються одночасно. А за через строковій розгортці показують почергово парні й непарні рядки (поля кадру). Черезстрокову розгортку часто називають інтерлейсом (від слова interlace) або інтерлейсінгом.

Системи PAL, SECAM і NTSC є системами з черезстроковою розгорткою. Нові цифрові стандарти телебачення, наприклад, HDTV

передбачають прогресивну розгортку. Але існують технології, що дозволяють імітувати прогресивне розгорнення при показі матеріалу з інтерлейсом.

Будь-який відеосигнал також має дозвіл (resolution) по горизонталі і вертикалі, який вимірюють у пікселях. Новий стандарт високочіткого відображення (high-definition) цифрового телебачення HDTV передбачає роздільну здатність до 1920x1080 пікселів при частоті кадрів 60 Герц з прогресивною розгорткою, тобто 1920 пікселів на лінію, 1080 ліній.

Роздільну здатність у випадку трьовимірному відео вимірюють у вокселях - елементах зображення, що являють собою крапки (кубики) у трьохвимірному просторі. Наприклад, для простого трьохвимірного відео зараз використовують в основному роздільну здатність 512x512x512.

Кількість кольорів і кольорову розрядність описують колірними моделями кольорів. Для стандарту PAL застосовують колірну модель YUV, для SECAM модель YDbDr, для NTSC модель YIQ, у комп'ютерній техніці застосовують переважно модель RGB (і αRGB), іноді модель HSV, а для друку - модель CMYK.

Кількість кольорів, що може відобразити монітор або проектор залежить від якості монітора або проектора. Людське око може сприйняти, по різних підрахунках, від 5 до 10 мільйонів відтінків кольорів. Кількість кольорів у відеоданих визначається числом біт, відведеним для кодування кольору кожного пікселя (bits per pixel, bpp). Зазвичай кодують від одного біта (звичайно чорний і білий), до 8 бітів -256 кольорів ($2^8 = 256$), до 16 бітів - 65 536 кольорів (2^{16}), або до 24 бітів - 16 777 216 кольорів (2^{24}). Але існує також стандарт 32 біти на піксель (αRGB), але додатковий α-байт (8 біт) використовують для кодування коефіцієнта прозорості пікселя (α), а не для передачі кольору (RGB). При обробці пікселя відеоадаптером, RGB-значення буде змінено залежно від значення α-байта і кольору належного пікселя (який

стане «видимим» через «прозорий» піксель), а потім α-байт буде відкинутий, і на монітор піде тільки колірний сигнал RGB.

Важливою характеристикою відеопотоку (для цифрового відео) є ширина (або швидкіскість) - бітрейт (від слів bit rate). Бітрейт - це кількість оброблюваних біт відеоінформації на секунду (позначають як біт / с, bit / s - біт в секунду, або частіше Мбіт / с, Mbit / s - мегабіт на секунду). Чим вища ширина відеопотоку, тим краща якість відео. Наприклад, для формату VideoCD ширина відеопотоку складає приблизно 1 Мбіт / с, а для DVD складає приблизно 5 мбіт / с. Формат же цифрового телебачення HDTV використовує ширину відеопотоку біля 10 Мбіт / с.

За допомогою швидкості відеопотоку дуже зручно оцінювати якість відео даних при передачі через Інтернет.

Чим більше bitrate, тим більше місця на диску займає дана інформація з поліпшенням якості зображення. Bitrate в 8000 kbit / s відповідає 1 мегабайту в секунду. Наприклад, для 60 секунд відео, закодованого з bitrate 8000 kbit / s, потрібно 60 мегабайт.

Розрізняють два види керування шириною потоку в відеокодеку - постійний бітрейт (англ. constant bit rate, CBR) і змінний бітрейт (англ. variable bit rate, VBR). Концепція VBR, нині дуже популярна, оскільки покликана максимально зберегти якість відео, зменшуючи при цьому сумарний обсяг переданого відеопотоку.

У VBR на швидких сценах руху ширина відеопотоку зростає, а на повільних сценах, де картинка змінюється повільно, ширина потоку спадає. Це дуже зручно для буферизованих відеотрансляцій і передачі збереженого відеоматеріалу по комп'ютерних мережах. Але для безбуферний систем реального часу і для прямого ефіру (наприклад, для телеконференцій) VBR не підходить.

Для відображення даних використовують параметр Aspect Ratio - відношення ширини до висоти екрана. Співвідношення 4:3 означає, що розміри по горизонталі на третину більше розмірів по вертикалі. Стандартний телевізійний розмір 4:3 (або 1.33:1). Широкоформатні (widescreen) DVD і HTDV мають співвідношення 16:9 (або 1.78:1).

Кодеком (Codec) називають алгоритм або спеціальну комп'ютерну програму що дозволяє обробити (застосувати компресію) і зменшити розміри великого файлу.

ЛЕКЦІЯ 15 СТАНДАРТИ MPEG

15.1 Сфери застосування стандартів MPEG

Стандарти MPEG охоплюють регламентацію шикого кола питань, пов’язаних зі створенням і застосуванням мультимедійних систем. Стандарти складають основу уніфікації вимог до апаратних і програмних рішень виробників мультимедіа. Значна частина стандартів охоплюють питання побудови кодів, спрямованих на зменшення трафіку у телекомунікаційних системах та об’ємів даних для збереження мультимедійної інформації. Стандарти враховують поточний розвиток мікроелектронних пристроїв, технології обробки і подання даних.

До основних стандартів, пов’язаних з кодуванням різнородних даних, можна віднести такі стандарти

MPEG 1

MPEG 2

MPEG 3

MPEG 4

MPEG 7

MPEG 21

15.2 Стандарт MPEG 1

Стандарт MPEG-1 визначає дуже популярний формат у всьому світі, з основою, взятої від кодека JPG. Стиснення в ньому виконують серіями по три

кадри. Якість зображення невисока, наближена до звичного аналогового формату VHS. Картинку подають як 352x288 пікселів.

Кадр (frame) можна розбити на дві складові частини - задній план, який зберігається один раз, а потім підставляється під час відтворення всіх кадрів, і область, що рухається і яку доведеться записувати окремо для кожного кадру.

У форматі MPEG-1 всі кадри відеоролика поділяються на три типи: I-, P-і B-кадри. До першого типу (I-кадри, Intra Frames) відносяться опорні кадри. Їх зображення зберігаються у повному обсязі в форматі JPEG. Для P-кадрів (Predicted Frames) записуються лише відзнаки від попереднього і-кадру, що вимагає набагато менше дискового простору. Для B-кадрів (Bi-DirectiOnally Interpolated Frames) зберігаються відмінності від попереднього і наступного I- або P-кадру. У підсумку розмір стисненого файлу складає приблизно 1 / 35 від початкового. Це означає, що півторагодинний фільм з якістю, еквівалентним аналогового запису на касеті VHS, у форматі MPEG-1 поміститься на два компакт-диска. Для передачі через Internet або в мережах супутникового мовлення цей стандарт зазвичай не підходить.

15.3 Стандарт MPEG 2

Стандарт MPEG-2 є подальшим розширенням MPEG-1. У ньому збільшено рекомендований розмір кадру до 1920 x 1080 пікселів, додана підтримка шестиканального звуку. Слід зазначити, що велася робота над створенням стандарту MPEG-3 для систем цифрового телебачення високої чіткості HDTV. Але робота над ним була перервана, оскільки потрібні для HDTV вимоги вдалося реалізувати у вигляді розширень до MPEG-2.

Домінуючий формат MPEG-2 має роздільну здатність 720x576 пікселів. Всі DVD-video диски працюють у форматі MPEG-2. Трансляції з супутників в кілька каналів на одній частоті, ефірна трансляція, в тому числі TV високої чіткості, різноманітні плеєри DVD, microMV-відеокамери використовують цей формат стиснення.

MPEG-2 підходить для запису півторагодинного фільму відмінної якості на стандартний диск DVD (7, 4 Гб). Крім того, в цьому форматі можна записувати на подвійні DVD (9 Гб) фільми підвищеної якості з використанням декількох різних доріжок звуку (дубляж), різних форматів багатоканального звучання, субтитрів, різних кутів огляду відеоматеріалу (кілька синхронних доріжок відео) і інших цифрових нововведень. Серед них, наприклад, присутній довільний миттєвий доступ до будь-якої частини відеоматеріалу на диску і відсутність перемотування при досягненні кінця відеоматеріалу, що раніше було досить великою проблемою.

Ієрархію потоку відеоі даних показано на рис. 15.1, а групу GOP на рис. 15.2.

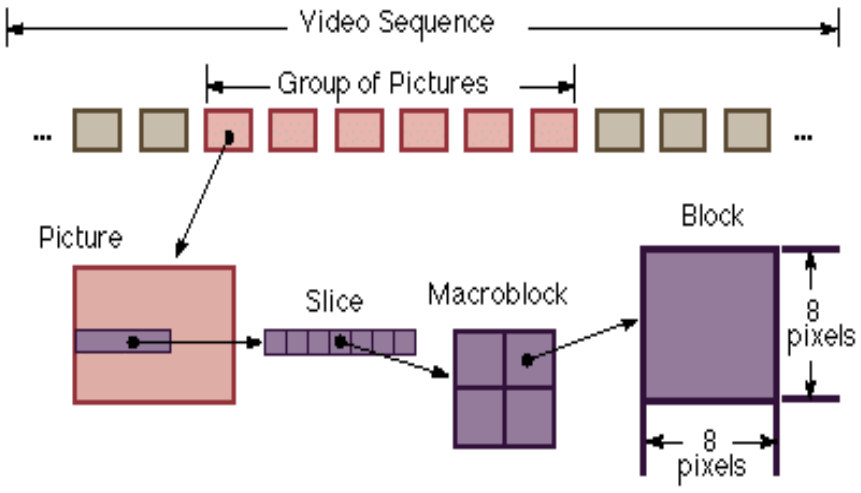


Рисунок 15.1- Ієрархію потоку відео даних

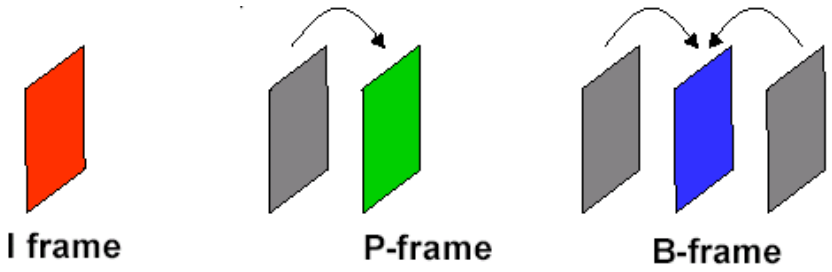


Рисунок 15.2- Група кадрів GOP

Основні характеристики формування потоків даних звуку і зображення наведено у таблиці 15.1

Таблиця 15.1- Формування потоків даних

Потоки даних	Характеристики потоків
Elementary Stream (ES)	Digital Control Data Digital Audio Digital Video Digital Data
Packetised Elementary Stream (PES)	Each ES combined into stream of PES packets. A PES packet may be fixed (or variable) sized block. Each block has up to 65536 bytes per block and a 6 byte protocol header.
MPEG Program Stream	Tightly coupled PES packets Used for video playback and network application
MPEG Transport Stream	Each PES packet is broken into fixed-sized transport packets

Особливості цього формату широко використовує компанія Sony у своєму розширеному стандарті microMV, хоча потік інформації там підвищений до 12 Мбіт / с (в порівнянні з максимальним стандартом DVD 9,8 Мбіт / с), а розмір касети зменшено (в порівнянні з DV). І все ж стандарт DV

відрізняється більшою стійкістю і великим поширенням по всьому світу. Нещодавно з'явилися камери, які пишуть відразу на miniDVD диски у форматі MPEG-2. Вони мають кілька важливих переваг - перезапис дисків до 1000 разів без втрати якості, доступність матеріалу і деякі інші переваги.

15.4 Стандарт MPEG 4

Формати MPEG-1 і MPEG-2 не забезпечували реальної можливості трансляції відео данних мережами Internet і створення інтерактивного телебачення на їх основі, оскільки вимагали дуже великих розмірів файлів.

MPEG-4 орієнтовано переважно не на стискання відео, а на створення так званого "мультимедійного контенту", тобто злиття інтерактивного телебачення, 3D-графіки, тексту тощо. Формат MPEG-4 поєднує відмінний звук і максимальне ущільнення відеосигналу (до 30-40% краще ніж у попередника).

Різниця полягає в тому, що кодується послідовність більш ніж з трьох кадрів (зазвичай до 250 кадрів). Тим самим досягається більше стиснення і можливість дивитися в режимі реального часу якісне потокове відео в інтернет. Динамічне стиснення також ефективно використовує ресурси. Тому на звичайний компакт-диск вміщується 1,5 години відео хорошлї якості. Проте, в більшості випадків, уважний глядач може побачити на хорошому екрані різницю між зображенням, закодованому в MPEG2 і MPEG4.

Стандарт MPEG-4 включає наступні частини.

Part 1, Systems – synchronizing and multiplexing audio and video

Part 2, Visual – coding visual data

Part 3, Audio – coding audio data, enhancements to Advanced Audio Coding and new techniques

Part 4, Conformance testing

Part 5, Reference software

Part 6, DMIF (Delivery Multimedia Integration Framework)

Part 7, optimized reference software for coding audio-video objects

Part 8, carry MPEG-4 content on IP networks

Part 9, reference hardware implementation

Part 10, Advanced Video Coding (AVC)

Part 11, Scene description and application engine; BIFS (Binary Format for Scene) and XMT (Extensible MPEG-4 Textual format)

Part 12, ISO base media file format

Part 13, IPMP extensions

Part 14, MP4 file format, version 2

Part 15, AVC (advanced Video Coding) file format

Part 16, Animation Framework eXtension (AFX)

Part 17, timed text subtitle format

Part 18, font compression and streaming

Part 19, synthesized texture stream

Part 20, Lightweight Application Scene Representation (LASeR) and Simple Aggregation Format (SAF)

Part 21, MPEG-J Graphics Framework eXtension (GFX)

Part 22, Open Font Format

Part 23, Symbolic Music Representation

Part 24, audio and systems interaction

Part 25, 3D Graphics Compression Model

Part 26, audio conformance

Part 27, 3D graphics conformance

Цікавою особливістю формату є те, що для типових об'єктів навіть розроблені окремі алгоритми передбачення і опису їх рухів. Це стосується, зокрема, ходи людей, найбільш поширених жестів, міміки. Тепер такі зміни в кадрах немає потреби записувати взагалі - їх можна розрахувати програмно. У MPEG-4 підтримується відображення тексту різними шрифтами поверх відеозображення. Більше того, цей текст може бути озвучений за допомогою синтезатора мови з можливістю імітації чоловічих і жіночих голосів. При необхідності голос синхронізується з рухами обличчя диктора відповідно до вимовними фонемами. Також може синтезуватися звучання деяких музичних інструментів (рис. 15.3). Стиснення оцифрованих звукозаписів здійснюється більш ефективно за допомогою спеціально розробленого кодека AAC (Advanced Audio Codec).

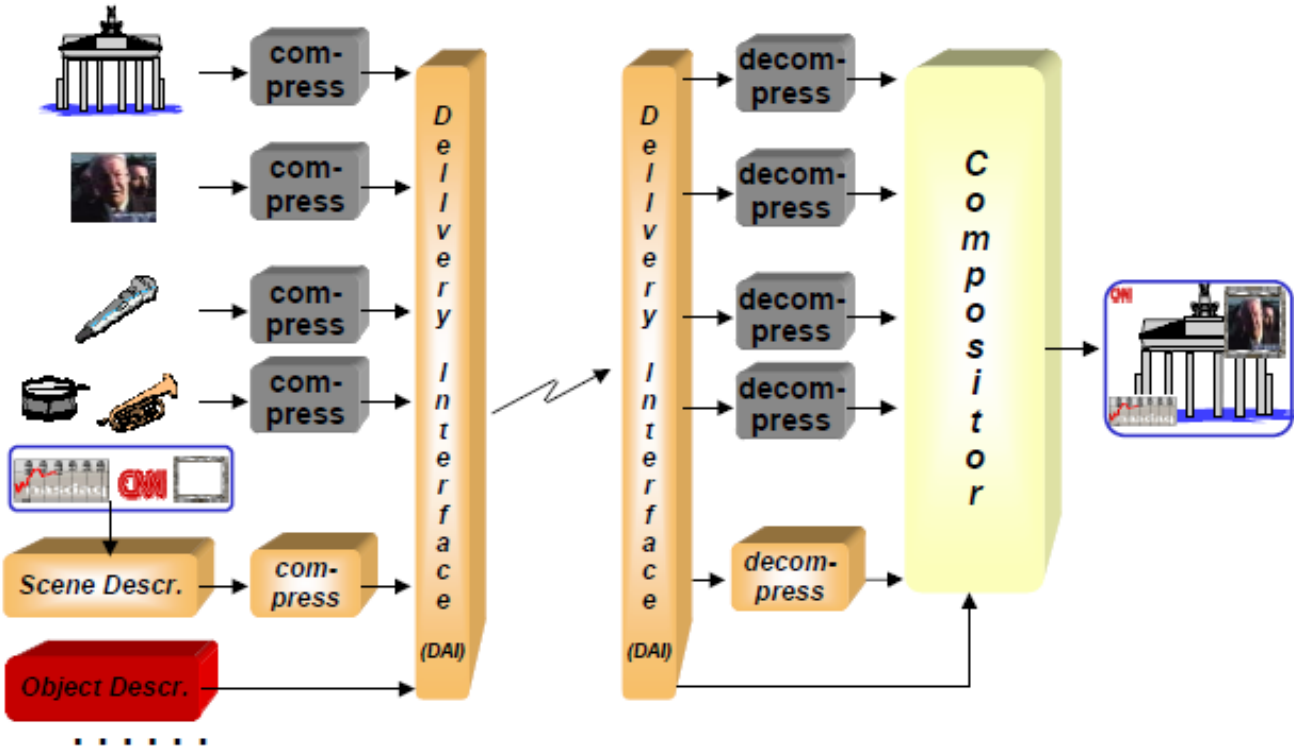


Рисунок 15.3- Кодування і декодування даних у MPEG-4

Деякі відеокамери дозволяють записувати у форматі MPEG-4 відео на власну карту пам'яті або працювати як web-камера, передаючи по USB кабелю відео зі звуком у форматі MPEG-4.

Крім того, сучасні технології дозволяють навіть відтворювати цифрове телебачення (стисле в форматі MPEG-4 або MPEG-2) за допомогою мобільних телефонів, використовуючи GPRS.

Міжнародні стандарти MPEG-1, MPEG-2, MPEG-4, H.261 і H.263 використовують комбіновану технологію кодування і компенсацію руху. Деякі сучасні алгоритми використовують технологію DWT (Discrete Wavelet Transform). Інші технології включають фрактальне стискання зображень (Fractal Image Comdivssion).

На рис. 15.4 показано формування пакетів даних у MPEG-4 мультиплксуванням (FlexMux) потоків різних т ипів.

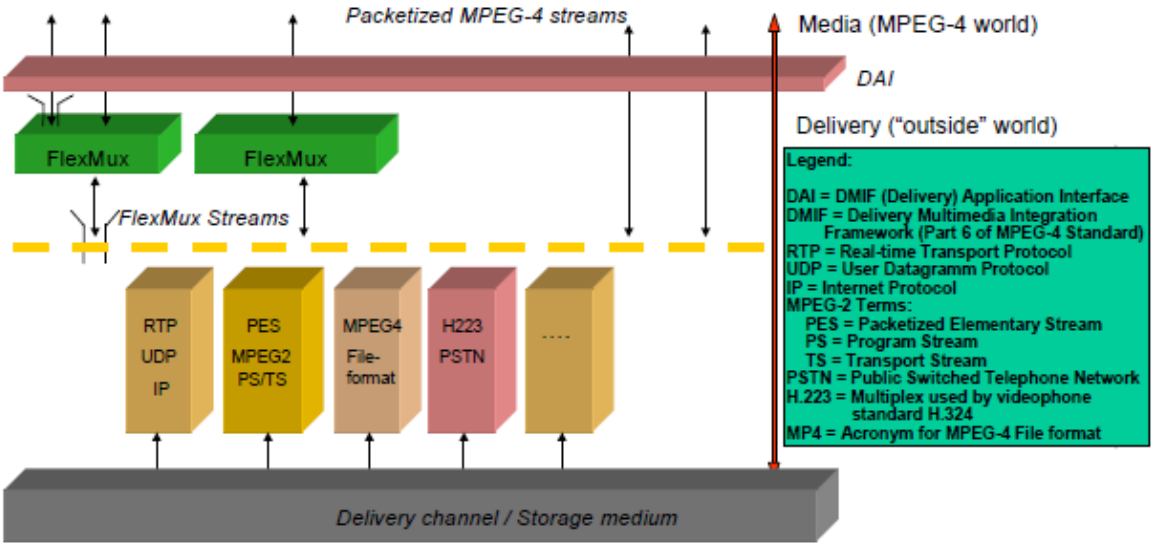


Рисунок 15.4- Формування пакетів даних у MPEG-4

Значного стискання досягають за рахунок пошуку подібних макроблоків (рис. 15.5, рис. 15.6) для попередніх і наступних кадрів (фреймів).

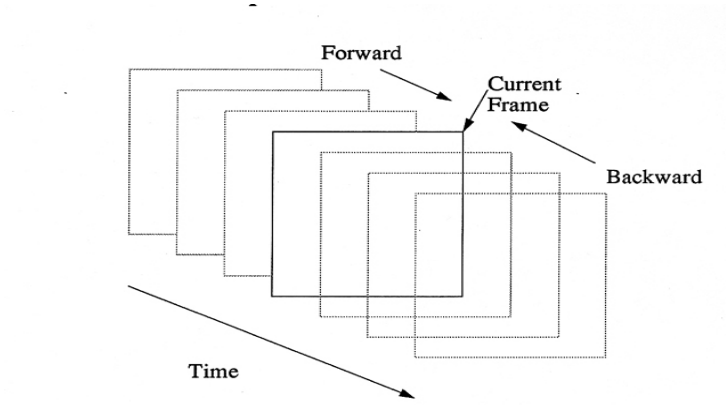


Рисунок 15.5- Формування макроблоків

Особливості окремих типів фреймів подано у таблиці 15.2.

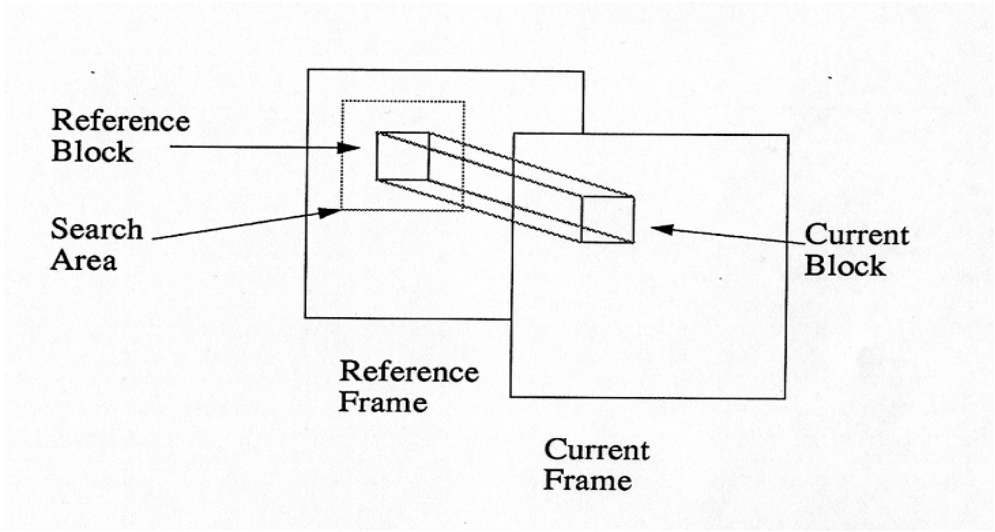


Рисунок 15.6- Пошук макроблоків фреймів

Таблиця 15.2- Характеристика фреймів даних

Фрейм	Характеристики фреймів	Особливості створення
I Frames	Spatial macro blocks only serves as "anchor" frame	Coded without reference to other frames
P Frames	Spatial, forward motion, and skip macro blocks can refer backwards to I and P frames can serve as "anchor" frame	Coded with reference to a previous reference frame (either I or P) Size is usually about 1/3rd of an I frame
B Frames	Spatial, forward/backward /interpolated motion, and skip macro blocks can refer backwards or forwards to I and P frames never used as "anchor" frame depends on a "future" frame which must be received and decoded first	frame (bi-directional predictive-coded) Coded with reference to both previous and future reference frames (either I or P) Size is usually about 1/6th of an I frame

15.5 Технологія VQ

Векторне квантування VQ (Vector Quantization,) застосовують у компресорах Indeo 3.2 і Cinepak з колірною схемою YUV (а не RGB). Процес кодування дуже трудомісткий і практично неможливий без спеціального додаткового обладнання. Процес декодування дуже швидкий. Спостерігають блокові спотворення при високих коефіцієнтах стиснення.

Основна ідея векторної квантизації полягає в розбитті зображення на блоки (розміром 4x4 піксела в колірній схемі YUV для компресорів Indeo і

Cinepak). Зазвичай, деякі блоки виявляються схожими один на одний. У цьому випадку компресор ідентифікує клас схожих блоків і замінює їх одним спільним блоком.

Крім цього, генерується двійкова таблиця (карта) таких загальних блоків з найкоротших кодових слів. Надалі VQ-декодер, використовуючи таблицю, збирає зображення поблочно із загальних блоків. Даний спосіб кодування є з втратами якості, оскільк схожість блоків дуже відносна. Тут допускається апроксимація реальних блоків зображення до загального, що їх об'єднує.

Процес кодування тривалий і трудомісткий, оскільки кодеру необхідно виявляти належність кожного блоку зображення до якого-небудь загальним блоку. Проте завдання декодування в цьому випадку зводиться до задачі побудови зображення по заданій карті із загальних блоків і не займає багато апаратних та часових ресурсів.

Таблицю або карту також називають кодовою книгою, а двійкові коди, що входять до неї - кодовими словами. Найбільше стиснення з використанням алгоритму VQ досягається шляхом зменшення числа класів загальних блоків, тобто припущенням про схожість щодо більшого числа блоків зображення. Це призводить до зменшення кодової книги. У міру зменшення розмірів кодової книги якість відтворення погіршується. Як результат на зображенні з'являється штучна «блочность».

15.6 Стандарт MPEG 7

Основна мета створення стандарту MPEG-7 є доступ за змістом до цифрових бібліотек мультимедійних даних. Стандарт MPEG-7 має назву Multimedia Content Description Interface. Він підримує мультимедійні додатки у

формі малюнків, графіки, моделей 3D models, аудіо, мовлення, відео і комбіновану інформацію. Дані MPEG-7 подають у текстовому форматі або у двійковому форматі.

Стандарт MPEG-7 складається з трьох частин:

засобів опису (Description tools), аналогічних до tool box у Matlab, які включають описи (Descriptors – D) і схеми описів (Description Schemes –DS);

мови визначення опису (Description Definition Language -DDL);

системних засобів (System tools).

Склад Description tools показано на рис. 15.7

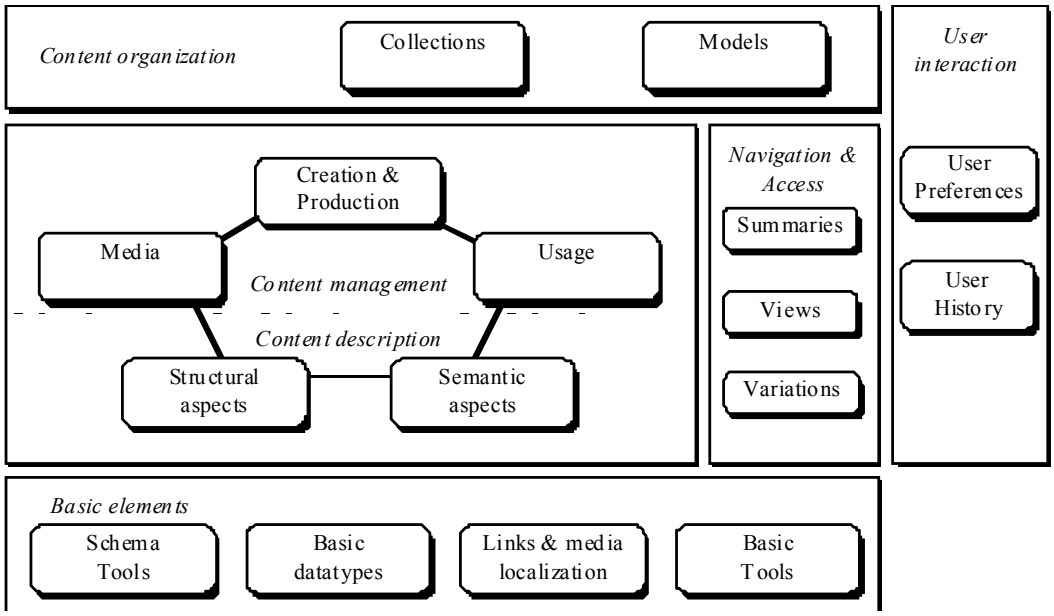


Рисунок 15.7- Структура Description tools

Приклад схеми змістовного опису DS подано на рис. 15.8.

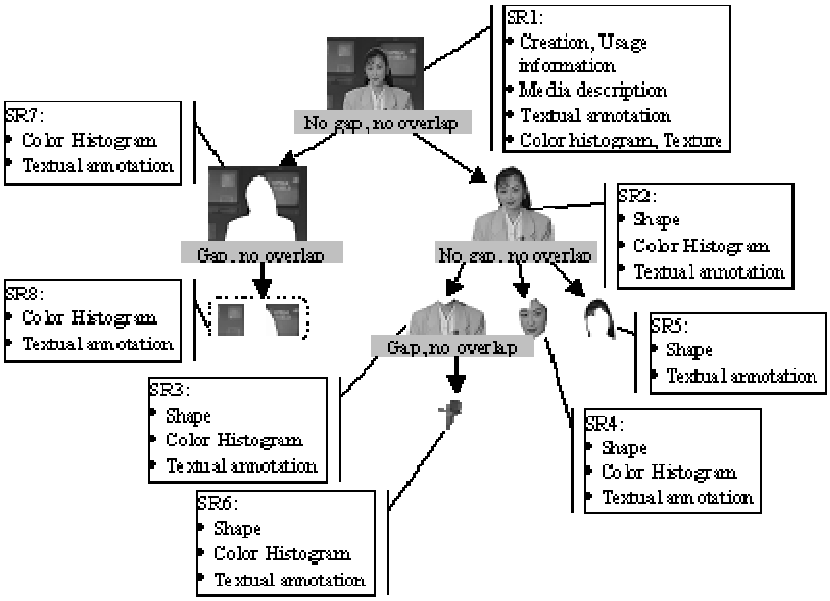


Рисунок 15.8- Приклад Description tools

Мова DDL ґрунтується на мові XML. Вона описує синтаксис опису D's і DS's та їх комбінації. Системні засоби System tools підтримують текстове і бінарне подання даних.

15.7 Стандарт MPEG 21

Стандарт MPEG 21 (ISO/IEC TR 21000) призначено для керування цифровими потоками даних і включає такі розділи:

Part 1: Vision, Technologies and Strategy

Part 2: DID – Digital Item Declaration

Part 3: DII – Digital Item Identification

Part 4: IPMP – Intellectual Property Management and Protection

Part 5: REL – Rights Expression

Part 6: RDD – Rights Data Dictionary

Part 7: DIA – Digital Item Adaptation

Part 8: MPEG-21 Reference Software

Part 9: MPEG-21 File Format

Part 10: DIP – Digital Item Processing

Part 11: Persistent Association Technology

Part 12: Test Bed for MPEG-21 Resource Delivery

Part 13: Empty

Part 14: Conformance

Part 15: Event Reporting

Part 16: BF – Binary Format

Part 17: Fragment Identification of MPEG Resources

Part 18: Digital Item Streaming

У MPEG 21 визначають базові поняття цифрового компонента (Digital Item) і користувача User. Компонент Digital Item подають як структурований об'єкт ("a structured digital object with a standard representation, identification and metadata within the MPEG-21 framework"), а користувача User як деяку данність, яка взаємодіє або використовує Digital Item ("any entity that interacts with or makes use of the Digital Items). На рис. 15.9 показано визначення Digital Item (Digital Item Declaration) і мову Digital Item (Digital Item Declaration Language).

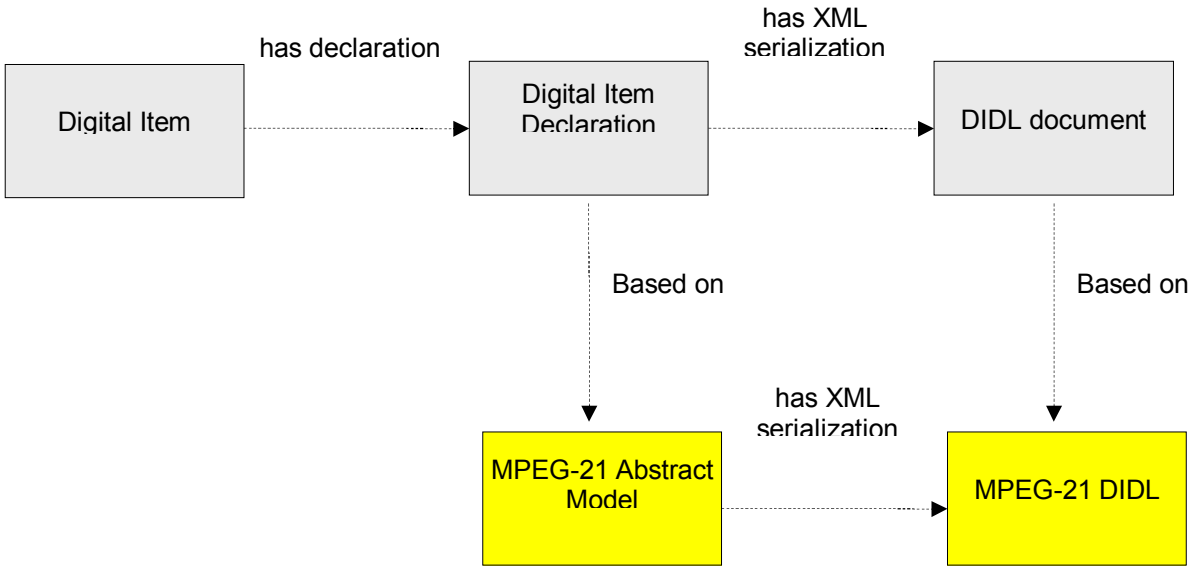


Рисунок 15.9- Опис Digital Item

Отримані документи DIDL ідентифікують для подальшого використання, як показано на рис. 15.10.

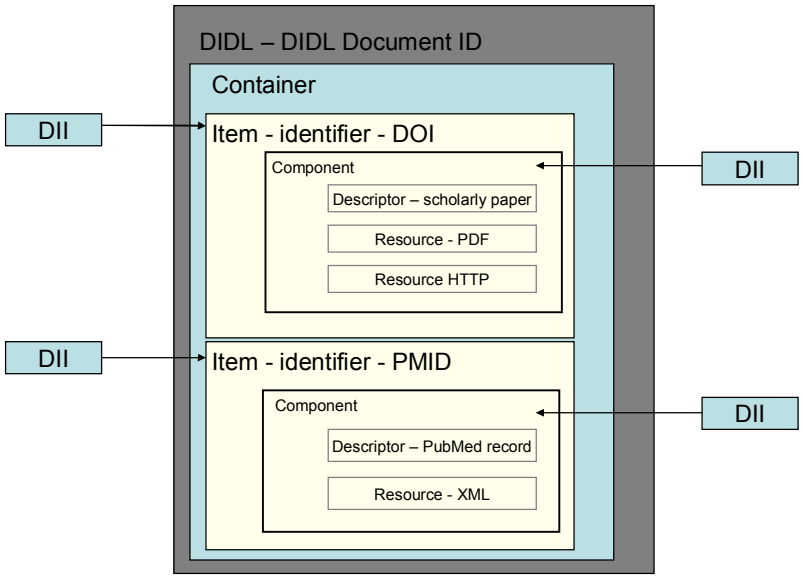


Рисунок 15.10- Опис Digital Item

Основні складові опису подано у розділах MPEG 21:

Identification information – MPEG-21 Part 3 : DII

Rights information – MPEG-21 Part 5 : REL

Processing information – MPEG-21 Part 10 : DIP

User environment information – MPEG-21 Part 7 : DIA

15.8 Алгоритми Wavelet

Відносно новий алгоритм стиснення відео при якому, на відміну від JPEG, зображення обробляється без розбивки на квадрати. Після того, як фірма Analogue Devices випустила спеціалізовану мікросхему апаратного wavelet-стиснення відео, даний формат став базисом багатоканальних цифрових систем відео спостереження і цифрових відеореєстраторів.

Як і у випадку формату JPEG, в Wavelet стиснення здійснюється з незворотними втратами інформації, але зображення не має "мозаїчних" дефектів навіть при дуже великих ступенях компресії.

Характерним є відсутність видимих дефектів навіть при великому коефіцієнті стиснення відео. При цьому знижується різкість і зображення просто стає менш чітким.

З математичної точки зору основною особливістю wavelet-перетворення є можливість розкласти зображення на два компоненти - низькочастотну частину, яка містить основну інформацію, і високочастотну частину, яка містить лише малу частку інформації. Низькочастотну частину можна знову розкласти на дві частини тощо. Частина зображення, що залишилася, містить лише малі високочастотні компоненти. У результаті послідовного застосування wavelet-перетворень виходить зображення, що займає невеликий обсяг місця на диску.

ЛЕКЦІЯ 16 КОДИ БОУЗА-ЧОУДХУРИ-ХОКВИНГЕМА

16.1 Призначення кодів BCH

Коди Боуза-Чоудхури-Хоквингема BCH (Bose-Chaudhuri-Hocquenghem Codes) або BCH дозволяють виявляти кратні помилки у пакетах даних. Основою для їх створення є алгебраїчні структури групи і поля. Основні властивості цих структур дозволяють визначити вимоги до кодових наборів кодів із заданими властивостями. Значне місце при цьому посідають поля Галуа, абелевими групи та поліноміальне подання кодів.

16.2 Властивості груп і полів

Групою G називається множина елементів з визначеною для кожної пари елементів операцією, що позначається *, що володіють наступними властивостями:

1. замкнутість: для кожної пари a і b з множини елементів c = a*b належить множині;

2. асоціативність: для будь-яких a, b і c з множини a*(b*c) = (a*b)*c;

3. існування одиниці: у множині існує елемент e, називаний одиничним, такий, що a*e = e*a = a для будь-якого елемента a множини;

4. існування зворотних елементів: для будь-якого a з множини існує деякий елемент b з множини, називаний зворотним a і такий, що a*b = b*a = e.

Якщо група G містить кінцеве число елементів, то вона називається кінцевою рупою, а число елементів G називається порядком G.

Групи, що володіють властивістю комутативності

$$a*b = b*a,$$

називаються комутативними або абелевими групами.

Полем називається множина із двома визначеними на ній операціями – додаванням і множенням, причому мають місце наступні аксіоми:

1) множина утворює абелеву групу за додаванням;

2) поле замкнуте щодо множення, і множина ненульових елементів утворює абелеву групу за множенням;

3) виконується закон дистрибутивності:

$(a + b)c = ac + bc$ для будь-яких a,b і c з поля.

Одиничний елемент щодо додавання прийнято позначати через 0 і називати нулем; адитивний зворотний елементові a елемент - через –a.

Одиничний елемент щодо множення позначають через 1 і називають одиницею;

мультиплікативний зворотний до елемента a елемент - через a^{-1}.

Під вирахуванням (a - b) розуміється a + (- b); під діленням (a / b) розуміється $(b^{-1})*a$.

Широко відомими є поля:

1) R: множина речовинних чисел;

2) C: множина комплексних чисел;

3) Q: множина раціональних чисел;

4) B: множина двійкович чисел.

Перші три поля містять нескінченну множину елементів.

В теорії кодування зазвичай використовують поля з кінцевим числом елементів.

Поле з q елементами, якщо воно існує, називають кінцевим полем або полем Галуа і позначають через GF(q).

Мінімальне поле GF(2):

+	0	1
0	0	1
1	1	0

*	0	1
0	0	0
1	0	1

Нехай F - деяке поле. Підмножина в F називається підполем, якщо вона саме є полем щодо наслідуваних з F операцій додавання і множення. У цьому випадку вихідне поле F називається розширенням поля.

16.3 Властивості поліномів над полями

Поліномом (багаточленом) над полем GF(q) називається математичний вираз

$$f(x) = f_n*x^n + f_{n-1}*x^{n-1} + \ldots + f_1*x + f_0$$

де x - невизначена (фіктивна) змінна;

$f_n,\ldots,f_0$ - коефіцієнти, які належать полю GF(q),

а індекси і показники степенів є цілими числами.

Нульовим поліномом називається поліном f(x) = 0.

Зведеним поліномом називається поліном, старший коефіцієнт f_n якого дорівнює 1.

Степенем ненульового полінома f(x) називається індекс старшого коефіцієнта fn; степінь полінома f(x) позначається через deg f(x).

Незвідними поліномам називаються поліноми, що не можуть бути представлені у виді добутку поліномів нижчих степенів з коефіцієнтами з того ж поля.

Зведені незвідні поліноми називаються простими.

Приклад. Розглянемо поле GF(4) як розширення поля GF(2) над незвідним поліномом $p(x)=x^2 + x + 1$:

$$\{0, 1, x, x+1\}.$$

Елементи поля можуть бути подані у вигляді різних позначень:

Поліноміальне	Двійкове	Десяткове	Степеневе
0	00	0	0
1	01	1	x^0
x	10	2	x^1
x +1	11	3	x^2

Використовуючи поліноміальне подання, визначимо операції додавання і множення над елементами (операції виконуються за модулем два і за модулем p(x)):

+	0	1	x	x + 1	*	0	1	x	x + 1
0	0	1	x	x+1	0	0	0	0	0
1	1	0	x+1	x	1	0	1	x	x+1
x	x	x+1	0	1	x	0	x	x+1	1
x+1	x+1	x	1	0	x+1	0	x+1	1	x

Елемент β називається коренем полінома p(x) або коренем рівняння p(x)=0, якщо

$$p(\beta) = 0.$$

Примітивним елементом поля GF(q) називається такий елемент α, що всі елементи поля, за винятком нуля, можуть бути представлені у вигляді степеня елемента α.

Приклад. У полі GF(5) {0, 1, 2, 3, 4} примітивним є елемент 2, тому що всі елементи даного поля, за винятком нуля, можуть бути представлені у виді степеня 2:

$2^0=1$,

$2^1=2$,

$2^2=4$,

$2^3= 8 \bmod 5 = 3$.

Примітивним поліномом p(x) над полем GF(q) називають простий поліном над GF(q), такий, що в розширенні поля, побудованому за модулем p(x), елемент, що відповідає поліномові x, є примітивним.

Прикладом примітивного полінома є поліном $x^2 + x + 1$.

Нехай GF(q) - деяке поле, GF(Q) – розширення поля GF(q), α - елемент GF(Q).

Простий поліном f(x) найменшого степеня над GF(q), для якого f(α)=0, називають мінімальним поліномом елемента α над GF(q).

Примітивні і мінімальні поліноми визначають за допомогою таблиць.

Приклад. У таблиці 16.1 задано подання поля GF(16) як розширення поля GF(2), побудоване за примітивним поліномом $p(z) = z^4 + z + 1$.

Можна помітити, що елементи поля $GF(2^4)$ у двійковому виді являють собою не що інше, як залишки від ділення 1 з і нулями на поліном p(z) ,

причому і відповідає показникові степені (a^i). Таким чином, ці елементи поля можна одержати в регістрі з лінійними зворотними зв'язками (РЗЛЗЗ) з породжувальний поліномом p(z).

Таблиця 16.1 - Поле GF(16) як розширення поля GF(2)

У виді степеня	У виді полінома	У двійковому виді	Мінімальний поліном
0	0	0000	-
α^0	1	0001	x+1
α^1	z	0010	x^4+x+1
α^2	z^2	0100	x^4+x+1
α^3	z^3	1000	$x^4+x^3+x^2+x+1$
α^4	z +1	0011	x^4+x+1
α^5	$z^2 + z$	0110	x^2+x+1
α^6	$z^3 + z^2$	1100	$x^4+x^3+x^2+x+1$
α^7	$z^3 + z +1$	1011	x^4+x^3+1
α^8	$z^2 + 1$	0101	x^4+x+1
α^9	$z^3 + z$	1010	$x^4+x^3+x^2+x+1$
α^{10}	$z^2 + z +1$	0111	x^2+x+1
α^{11}	$z^3 + z^2 + z$	1110	x^4+x^3+1
α^{12}	$z^3 + z^2 + z +1$	1111	$x^4+x^3+x^2+x+1$
α^{13}	$z^3 + z^2 + 1$	1101	x^4+x^3+1
α^{14}	$Z^3 + 1$	1001	x^4+x^3+1

Якщо для елемента α відомий мінімальний поліном, то цей же поліном буде мінімальним для елемента α^2, а зворотний до цього полінома поліном буде мінімальним для елемента α^{-1}.

У таблиці незвідних поліномів для її скорочення приводять тільки ті поліноми, які не можна одержати з інших відомих поліномів у відповідності з властивостями мінімальних поліномів. Наприклад, для GF(16) побудованого за примітивним поліномом $p(z) = z^4 + z + 1$, таблиця незвідних поліномів містить тільки три мінімальних поліномів: для 1, 3, 5. Всі інші визначаються

по цим поліномам, використовуючи розглянуті властивості мінімальних поліномів.

Особливе місце посідає мінімальний поліном для полінома α^0. Для розширень полів над поліномами будь-яких степеней він дорівнює x + 1.

Позначимо мінімальний поліном для елемента α^i як $M_i(x)$. Відповідно до визначення мінімального полінома, елемент α^i є коренем Mi(x). Це значить, наприклад, якщо: замість псевдозмінної x у поліномі M1(x) підставити z (z=α, оскільки $p(z) = z^4 + z + 1$ – примітивний поліном), отриманий поліном за модулем p(x) дорівнює нулю:

$$(z^4 + z + 1) \bmod p(z) = 0;$$

замість псевдозмінної x у поліномі M3(x) підставити z^3 (z=α, оскільки $p(z) = z^4 + z + 1$ – примітивний поліном), отриманий поліном за модулем p(x) дорівнює нулю: $z^{12} + z^9 + z^6 + z^3 + 1 \bmod p(z) = 0$; замість псевдозмінної x у поліномі M5(x) підставити z5 (або $z^2 + z$) (z=α, оскільки $p(z) = z^4 + z + 1$ – примітивний поліном), отриманий поліном за модулем p(x) дорівнює нулю: $(z^{10} + z^5 + 1) \bmod p(z) = 0$ тощо.

16.4 Кільця поліномів

Поліном степені n з коефіцієнтами, визначеними у кільці B, є елементом у B^{n+1} .

Можна також розглядати поліноми у кільці B як поліноми над B. Оскільки B є полем, то множина поліномів степеня n на B є лінійним простором B^{n+1} .

Наприклад, для n+1=8 кодовий набір 11000111 має подання як поліном $1+X+X^5+X^6+X^7$.

Для n=1 кодовий набір 00 має подання як поліном 0, 10 як поліном 1, 01 як поліном X, а 11 як поліном X+1.

Для n=2 кодовий набір 010 має подання як поліном X, 011 як поліном $X+X^2$ тощо.

Для n=9 кодовий набір 101010101 має подання як поліном $1+X^3+X^5+X^7+X^9$.

Добуток поліномів у B дозволяє отримати поліном добутку. Наприклад,

$$(1+X+X^2)\times(1+X) = 1+X+X+X^2+X^2+X^3 = 1+X^3$$

$$(1+X^2)\times(X+X^3) = X+X^3+X^3+X^5 = X+X^5$$

$$(X+X^3)\times(X^5+X^9) = X^6+X^8+X^{10}+X^{12}.$$

Дані поліноми відповідають таким перетворенням кордових наборів

$$111\times 11 = 1001$$

$$101\times 0101 = 010001$$

$$0101\times 0000010001 = 000000101010 1.$$

Кільце поліномів з коефіцієнтами з B та операціями додавання і множення позначатимемо B[X].

Операція ділення поліномів визначають через процедуру синтетичнго ділення (synthetic division) шляхом послідовного віднімання дільника, починаючи зі старшого розряду вихідного поліному.

Наприклад, для ділення вихідного поліному X^3+X^2+X+1 на поліном $X+1$ отримуємо таку послідовність перетворень

$$
\begin{array}{r|l}
X+1 & X^2 + 1 \\
\hline
X^3+X^2+X+1 & \\
\underline{X^3+X^2} & \\
X+1 & \\
\underline{X+1} & \\
0 &
\end{array}
$$

Залишок становить 0.

При діленні X^5+1 на X^2+X отримуємо таку послідовність

$$
\begin{array}{r|l}
X^2+X & X^3+X^2+X+1 \\
\hline
X^5+1 & \\
\underline{X^5+X^4} & \\
X^4+1 & \\
\underline{X^4+X^3} & \\
X^3+1 & \\
\underline{X^3+X^2} & \\
X^2+1 & \\
\underline{X^2+X} & \\
X+1 &
\end{array}
$$

Залишок становить X+1.

Для поліномів з коефіцієнтами в B можна визначити операцію обчислення за модулем.

Нехай p, q є поліномами з коефіцієнтами, визначеними у кільці B. Тоді модуль за поліномом r добутку поліномів p та q є залишком від ділення добутку поліномів на поліном r.

Наприклад, для поліномів $p(X)=X^4+X^2, q(X)=X^4+X$ та $r(X)=X^5+X$ добуток p*q становитиме

$$p(X)q(X) = (X^4 + X^2)(X^4 + X) = X^8 + X^6 + X^5 + X^3.$$

Результат ділення на r дає

$$X^8 + X^6 + X^5 + X^3 = (X^3 + X)(X^5 + X) + (X^4 + X^3 + X^2 + X)$$

Тобто

$$(X^4+X^2)\,\text{and}\,(X^4+X)\,\text{modulo}\,(X^5+X) = (X^4+X^3+X^2+X)$$

Множення поліномів за модулем $X^n + 1$ дає поліном степені, не більше за n-1. Множина поліномів зі степенями, меншими за n утворює кільце відносно операцій множення і додавання за модулем $X^n + 1$. Позначимо це кільце через $\mathbb{B}_n[X]/(X^n + 1)$. Ці поліноми можна визначити у $\mathbb{B}^n$ для характеристики структури кільця.

На основі теореми про залишки, залишок ділення полінома з $\mathbb{B}[X]$ на $X^n + 1$ можна отримати, згортаючи степінь X^n до $X^0 = 1$.

Наприклад, залишок від ділення X^3+X^2+X на X^3+1 становить $1+X^2+X = X^2+X+1$.

Залишок від ділення $X^4+X^3+X^2$ на X^3+1 становить $1X+1+X^2 = X^2+X+1$.

Залишок від ділення $X^7 + X^6 + X^5$ на $X^4 + 1$ становить $1X^3 + 1X^2 + 1X = X^3 + X^2 + X$.

Добуток $(X^3+X^2)(X^2+X) = X^5+X^3$ за модулем X^4+1 становить $1X+X^3 = X^3+X$.

Це дозволяє побудувати таблицю множення для кільця $\mathbb{B}_n[X]/(X^n+1)$.

Зокрема для $\mathbb{B}_2[X]/(X^2+1)$

The elements of $\mathbb{B}_2[X]/(X^2+1)$ are the polynomials 0, 1, X and $1+X$. Multiplication by 0 and 1 is trivial. The other products are

$(X)(X)$ modulo $(X^2+1) = X^2$ modulo $(X^2+1) = 1$,

$(X)(1+X)$ modulo $(X^2+1) = X+X^2$ modulo $(X^2+1) = 1+X$,

$(1+X)(1+X)$ modulo $(X^2+1) = 1+X^2$ modulo $(X^2+1) = 0$.

Тобто таблиця множення для залишків має такий вигляд

	0	1	X	$1+X$
0	0	0	0	0
1	0	1	X	$1+X$
X	0	X	1	$1+X$
$1+X$	0	$1+X$	$1+X$	0

Оскільки є відповідність поліномів з кодовими словами

0 with 00

1 with 10

X with 01

$1+X$ with 11

Та таблиця множення залишків набуває вигляду

	00 10 01 11
00	00 00 00 00
10	00 10 01 11
01	00 01 10 11
11	00 11 11 00

Аналогічно може бути побудована таблиця для елементів з $\mathbb{B}_3[X]/(X^3+1)$ Поліномами залишків будуть $0, 1, X, 1+X, X^2, 1+X^2, X+X^2, 1+X+X^2$.

А відповідна таблиця множення набуває вигляду

	0	1	X	$1+X$
0	0	0	0	0
1	0	1	X	$1+X$
X	0	X	X^2	$X+X^2$
$1+X$	0	$1+X$	$X+X^2$	$1+X^2$
X^2	0	X^2	1	$1+X^2$
$1+X^2$	0	$1+X^2$	$1+X$	$X+X^2$
$X+X^2$	0	$X+X^2$	$1+X^2$	$1+X$
$1+X+X^2$	0	$1+X+X^2$	$1+X+X^2$	0

	X^2	$1+X^2$	$X+X^2$	$1+X+X^2$
0	0	0	0	0
1	X^2	$1+X^2$	$X+X^2$	$1+X+X^2$
X	1	$1+X$	$1+X^2$	$1+X+X^2$
$1+X$	$1+X$	$X+X^2$	$1+X$	0
X^2	X	$X+X^2$	$1+X$	$1+X+X^2$
$1+X^2$	$1+X^2$	$1+X$	$1+X^2$	0
$X+X^2$	$1+X$	$1+X^2$	$X+X^2$	0
$1+X+X^2$	$1+X+X^2$	0	0	$1+X+X^2$

Оскільки є відповідність між поліномами і кодовими словами, то таблиця кодових наборів результатів добутку кодових слів має вигляд

	000	100	010	110	001	101	011	111
000	000	000	000	000	000	000	000	000
100	000	100	010	110	001	101	011	111
010	000	010	001	011	100	010	101	111
110	000	110	011	101	110	011	110	000
001	000	001	100	101	010	011	110	111
101	000	101	110	011	101	110	101	000
011	000	011	101	110	110	101	011	000
111	000	111	111	000	111	000	000	111

Множення на X у $\mathbb{B}_n[X]/(X^n+1)$ еквівалентно зсуву компонента з $\mathbb{B}^n$ вправо циклічно на одну позицію.

16.5 Побудова твірного полінома коду BCH

Твірний поліном коду BCH, що виправляє s помилок, повинен містити 2s коренів. Множина цих коренів:

$$\{\alpha^{j0}, \alpha^{j0+1}, \alpha^{j0+2},\ldots,\alpha^{j0+2s-1}\},$$

де j0 – конструктивний параметр.

Зазвичай вибирають j0 = 1. Тоді множиною цих коренів є:

$$\{\alpha, \alpha 2, \alpha 3 \ldots \alpha 2s\}$$

Приклад. Нехай задане поле GF(16), побудовано як розширення поля GF(2) над поліномом $p(z) = z^4 + z + 1$. Позначимо мінімальні поліноми в такий спосіб.

Елемент	Мінімальний поліном
α	$M_1(x) = x^4 + x + 1$
α^3	$M_3(x) = x^4 + x^3 + x^2 + x + 1$
α^5	$M_5(x) = x^2 + x + 1$
α^7	$M_7(x) = x^4 + x^3 + 1$

1.Якщо взяти s=1, то для двійкового поля код збігається з кодом Хеммінга.

2. Нехай s=2. Твірний поліном коду BCH, що виправляє подвійні помилки:

$$B(x) = НЗК\ \{M1(x), M2(x), M3(x), M4(x)\} = НЗК\ \{M1(x), M3(x)\} =$$
$$= (x^4 + x + 1)*(x^4 + x^3 + x^2 + x + 1) = x^8 + x^7 + x^6 + x^4 + 1$$

де НЗК – найменше загальне кратне),

Тут M1(x), M2(x), M4(x) мають один і той самий поліном,.

Оскільки поле GF(16) містить 15 ненульових елементів, то довжина коду n = 15: кількість перевірочних символів

$$p = \deg B(x) = 8,$$

то кількість інформаційних символів $k = n\text{-}p = 7$. Тобто одержимо код BCH (15, 7), що виправляє подвійні помилки.

3. Нехай s=3. Твірний поліном коду BCH, що виправляє потрійні помилки:

$$B(x) = НЗК\ \{M1(x), M3(x), M5(x)\} =$$
$$=(x^4 + x + 1)*(x^4 + x^3 + x^2 + x + 1)*(x^2 + x + 1) =$$
$$= x^{10} + x^8 + x^5 + x^4 + x^2 + x + 1.$$

Довжина коду n = 15, оскільки розглянуте поле GF(16) містить 15 ненульових елементів. Кількість перевірочних символів складає

$$p = \deg B(x) = 10,$$

а кількість інформаційних символів k = n-p = 5.

Тобто одержали код BCH (15,5), що виправляє потрійні помилки.

4. s=4 (5, 6, 7).

$$B(x) = НЗК\{M1(x), M3(x), M5(x), M7(x)\} =$$
$$=(x^4 + x + 1)*(x^4 + x^3 + x^2 + x + 1)*(x^2 + x + +1)*(x^4 + x^3 + 1) =$$
$$=x^{14} + x^{13} + x^{12} + x^{11} + x^{10} + x^9 + x^8 + x^7 + x^6 + x^5 + x^4 + x^3 + x^2 + x+1.$$

Довжина коду n=15, кількість перевірних символів p=14, кількість інформаційних символів k=1. Отримуємо твірний поліном (15,1) коду BCH. Це простий код з повторенням, що виправляє сім помилок (при s = 5,6,7 маємо однаковий поліном).

ЛЕКЦІЯ 17 КОДИ РІДА-СОЛОМОНА

17.1 Характеристика пакетних помилок

Для пакетів помилок послідовність даних поділяють на пакети сталої довжини з визначеними характеристиками для кожного пакету.

Циклічним пакетом довжини b називають вектор, усі ненульові компоненти якого розташовані серед b послідовних (по циклу) компонентів, перший й останній з яких відмінні від нуля (рис. 17.1).

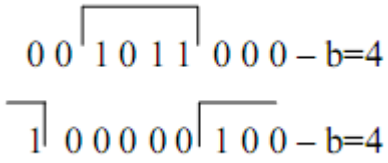


Рисунок 17.1- Циклічні пакети

Блоковий код, що виправляє пакети помилок довжини b, повинен містити, щонайменше, 2b перевірних символів (границя Рейгера).

Для кодів, що виправляють пакети помилок, визначають параметр міри неефективності

$$z=p*2b$$

де p – кількість перевірних символів;

b – довжина пакета помилок, що виправляється.

z - параметр міри неефективності.

17.2 Циклічні коди з виправленням пакетів помилок

Для виправленням пакетів помилок застосвують циклічні коди. Ці коди можна побудувати за допомогою комп'ютера шляхом перебору твірних поліномів кодових наборів або за допомогою аналітичних методів.

Приклади твірних (породжувальних) поліномів, побудувані за допомогою комп'ютера, наведені у таблиці 17.1.

Таблиця 17.1 – Твірні поліноми

Породжувальний поліном	Параметри (n, k)	Довжина пакета b
$x^4+x^3+x^2+1$	(7,3)	2
$x^5+x^4+x^2+1$	(15,10)	2
$x^6+x^5+x^4+x^3+1$	(15,9)	3
$x^6+x^5+x^4+1$	(31,25)	2
$x^7+x^6+x^5+x^3+x^2+1$	(63,56)	2
$x^8+x^7+x^6+x^3+1$	(63,55)	3
$x^{12}+x^8+x^5+x^3+1$	(511,499)	4
$x^{13}+x^{10}+x^7+x^6+x^5+x^4+x^2+1$	(1023,1010)	4

Серед найбільш поширених аналітичних методів можна виділити такі

- коди Файра;
- коди Бартона;
- коди Ріда – Соломона з виправленням поодиноких помилок;
- коди Ріда – Соломона з виправленням кратних помилок;
- коди, отримані з перерахованих вище методом укорочування і/або символьного або блочного чергування.

Методом символьного чергування степеня j можна одержати більш довгі коди, що виправляють пакети помилок довжини (j*b). При цьому,якщо вихідний код має параметри (n, k) і твірний поліном g(x), то новими параметрами будуть (jn, jk) і g(xj) відповідно.

При символьному чергуванні параметр міри неефективності z збільшується в j разів. Якщо z дорівнює нулю для вихідного коду, то для нового коду z також буде дорівнювати нулю.

Приклад. Нехай вихідний код має твірний поліном $g(x) = x^4+x^3 +x^2+1$ і параметри (n,k)=(7, 3), b=2, z = 0.

Тоді при символьному чергуванні отримуємо коди з параметрами:

для j=2: $g(x) = x^8+x^6 + x^4+1$, (n,k)= (14, 6), b=4, z=0;
для j=3: $g(x) = x^{12}+x^9 + x^6+1$, (n,k)= (21, 9), b=6, z=0.

Зокрема, для символьного чергуванні з параметром j=2 можна взяти дві копії кодера, декодера, що формують і обробляють символи кодового слова по черзі. Наприклад, якщо код (n,k)=(7, 3) виправляє пакет помилок довжини b=2, то взявши копію a1’a2’a3’b1’b2’b3’b4’ кодового слова вихідного коду a1a2a3b1b2b3b4, одержимо кодове слово нового коду a1a1’a2a2’a3a3’b1b1’b2 b2’b3b3’b4b4’ (рис. 17.2).

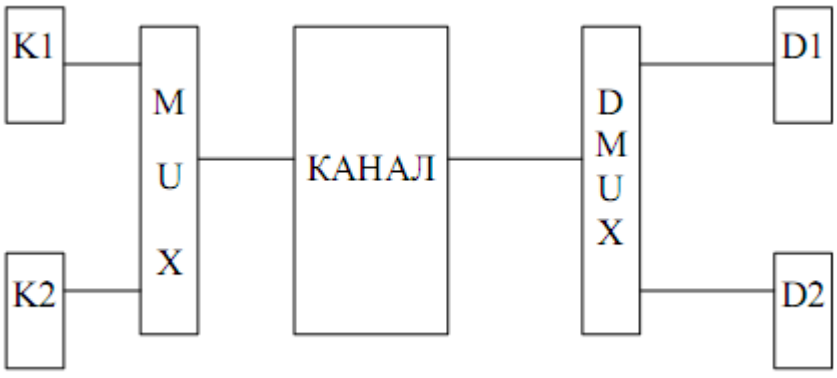


Рисунок 17.2- Формування кодового слова для j=2

Кожна пара кодер-декодер обробляє своє кодове слово довжини 7 і може виправити пакет довжини 2. Якщо розглянути будь-який пакет помилок довжини 4 у кодовому слові a1a1’a2a2’a3a3’b1b1’b2 b2’b3b3’b4b4’, то два помилкових символи, розташованих поруч, будуть відноситися до різних пар кодер-декодер..Тому такий пакет буде виправлений.

17.3 Коди Файра

Твірний поліном визначають як

$$F(x) = (x^{2b-1} + 1)* P(x),$$

де P(x) – примітивний поліном, deg P(x) = b;

b – довжина пакета помилок, що виправляється.

Довжина коду n = (2 b - 1)*(2b 1). Кількість перевірних символів p=3b-1. Кількість інформаційних символів k = n – p. Параметр міри неефективності коду z = b - 1.

Якщо b дуже велике, то використовують укорочені коди Файра, оскільки в цьому випадку довжина коду дуже велика. Методи укорочування циклічних кодів, що виправляють пакети помилок, аналогічні методам укорочування циклічних кодів Хеммінга.

17.4 Коди Бартона

Твірний поліном визначають як

$$B(x) = (x^b + 1)* P(x),$$

де P(x) – примітивний поліном, deg P(x) = b;

b – довжина пакета помилок, що виправляється.

Довжина коду $n = (2^b - 1)*b$. Кількість перевірних символів p = 2b. Кількість інформаційних символів k = n – p. Параметр міри неефективності коду z =0.

Для кодів Бартона існують обмеження на характер розташування пакета помилок. Приклад такого обмеження подано на рис. 17.3. Кодове слово довжини n можна розбити на 2b – 1 ділянок по b символів.

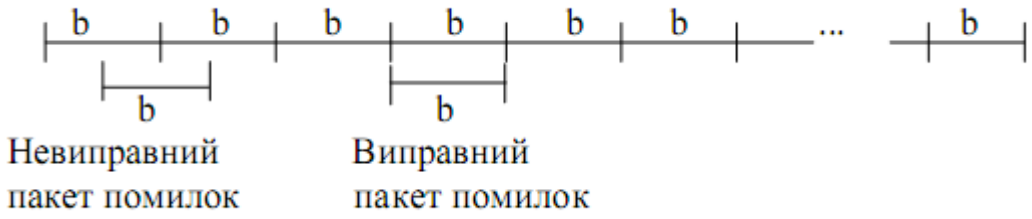


Рисунок 17.3- Обмеження на розташування пакета помилок

Якщо пакет помилок випадає у виділені ділянки, то він буде виправлений. Якщо ж пакет захоплює дві виділених ділянки, то такий пакет не може бути виправлений.

Щоб зняти це обмеження, застосовують поблокове чергування кодів Бартона. Суть його аналогічна символьному чергуванню. Але замість

чергування символів перемежовують блоки довжини b. За допомогою додаткових копій кодів Бартона з довжиною пакета b одержують перемежовуваний код з довжиною пакета b’.

$$b' = j*b - (b - 1).$$

При j=2 b’ = b+1. При j = 3 b’ = 2b + 1. Ефективність поблочного чергування зростає зі збільшенням значення j.

17.5 Коди Ріда - Соломона

17.5.1 Загальна характеристика

Коди Ріда – Соломона RS (Reed-Solomon) є окремим випадком кодів BCH. Головна відмінність кодів RS полягає у тому, що символом виступає елемент поля Галуа (декілька біт).

Сфера застосування кодів RS надзвичайно широкау цифрових системах зв'язку і збереження інформації. Зокрема, коди RS застосовують

-для космічного зв'язку NASA;

- для CD ROM, DVD використовують укорочені коди RS над полем Галуа GF(28);

- для цифрового телебачення високої чіткості HDTV використовують розширений код (128, 122, 7);

- для кабельних модемів використовують коди RS над полем Галуа GF(27).

17.5.2 Твірні поліноми кодів RS

Твірний поліном кодів RS для виправлення s помилок містить 2s коренів:

$$\{\alpha^{j0}, \alpha^{j0+1}, \alpha^{j0+2},\ldots,\alpha^{j0+2s-1}\},$$

де j0 – конструктивний параметр.

Зазвичай, j0 вибирають рівним 1. Тоді множиною коренів полінома є

$$\{\alpha, \alpha^2, \alpha^3\ldots\alpha^{2s}\}...$$

Для коду RS з виправленням s помилок твірним поліномом є

$$RS(x) = (x - \alpha)(x - \alpha^2)(x - \alpha^{3)}\ldots(x - \alpha^{2s})$$

17.5.3 Коди RS для поодиноких помилок

Для коду RS з виправленням s=1 помилок твірним поліномом є

$$RS(x) = (x - \alpha)(x - \alpha^2).$$

Можливими є кілька форм запису цього полінома. Зазвичай, для побудови кодів RS використовують розширення поля GF(2) над примітивним

поліномом p(z). У цьому випадку, відповідно до визначення примітивного полінома, елемент поля z є примітивним. Тому замість позначення примітивного елемента α можна використовувати z:

$$RS(x) = x^2 + (\alpha + \alpha^{2)}*x + \alpha^3 = x^2 + (z + z^2)*x + z^3.$$

Інша форма залежить від поля, яке застосовують для створення кодів RS.

Приклад. Побудуємо поле Галуа GF(8) як розширення поля GF(2) над примітивним поліномом

$$p(z) = z^3 + z + 1.$$

Елементи поля можуть бути подані у різному позначенні (таблиця 17.2).

Таблиця 17.2 – Подання елементів поля

У виді степеня	У виді полінома	У двійковому виді
0	0	000
α^0	1	001
α^1	Z	010
α^2	z^2	100
α^3	z + 1	011
α^4	z^2 +z	110
α^5	z^2 + z + 1	111
α^6	z^2 + 1	101

Оскільки

$$RS(x) = x^2 + (z + z^2)*x + z^3, \text{ а } (z + z^2)$$

для поля GF(8) у степеневому позначенні

$$\alpha^4, z^3, \alpha^3,$$

то твірний поліном можна подати як

$$RS(x) = x^2 + \alpha^4 x + \alpha^3.$$

При зміні j0 змінюється твірний поліном. Наприклад, при j0=2

$$RS(x) = (x - \alpha^2)(x - \alpha^3) = x^2 + (z^3 + z^2)x + z^5.$$

Для GF(8) над $p(z) = z^3 + z + 1$

$$RS(x) = x^2 + (z^2 + z + 1)x + z^2 + z + 1 = x^2 + \alpha^5 x + \alpha^5.$$

Елементи поля в загальному вигляді для поля GF(8) можна подати як

$$a_2 z^2 + a_1 z + a_0,$$

де a_2, a_1, a_0 – коефіцієнти, що приймають різні значення.

Ці елементи поля (у даному випадку тріади) є символами коду RS.

Перелік форм твірних поліномів для поодиноких помилок наведено у таблиці 17.3.

Таблиця 17.3 – Форми твірних поліномів

№	Твірний поліном	Умови застосування
1	$RS(x) = (x - \alpha)(x - \alpha^2)$	при j0 = 1 – для побудови схем у такому виді не використовується;
2	$RS(x) = (x - \alpha^2)(x - \alpha^3)$	при j0 = 2 - для побудови схем у такому виді не використовується;
3	$RS(x) = x^2 + (z + z^2)*x + z^3$	при j0 = 1 – не залежить від конкретного поля Галуа і може бути використана для побудови схем кодера і декодера;
4	RS(x) = x2 + (z3 + z2)x + z5	при j0 = 2 – не залежить від конкретного поля Галуа і може бути використана для побудови схем кодера і декодера;
5	$RS(x) = x^2 + (z + z^2)*x + z + 1$	при j0 = 1 – залежить від конкретного поля Галуа і може бути використана для побудови схем кодера і декодера;
6	$RS(x) = x^2 + (z^2 + z + 1)x + z^2 + z + 1$	при j0 = 2 – залежить від конкретного поля Галуа і може бути використана для побудови схем кодера і декодера;
7	$RS(x) = x^2 + \alpha^4 x + \alpha^3$	при j0 = 1 – залежить від конкретного поля Галуа і може бути використана для побудови схем кодера і декодера;
8	$RS(x) = x^2 + \alpha^5 x + \alpha^5$	при j0 = 2 – залежить від конкретного поля Галуа і може бути використана для побудови схем кодера і декодера..

Непарні і парні варіанти поліномів визначають різні коди. Вони мають відмінні реалізації. Але основні параметри (коригувальні можливості, довжина коду n, кількість інформаційних k і перевірних n-k=p символів) є однаковими.

Зазвичай, для функціональних схем кодуючих і декодуючих пристроїв використовують варіанти 7, 8 поліномів. Схемна реалізація є найбільш компактною. Але побудова таких поліномів потребує побудови поля Галуа.

Для розробки принципових схем краще використовувати варіанти 3-6 поліномів, оскільки вони не вимагають побудови поля Галуа. При цьому варіанти 5, 6 є кращими.

17.5.4 Коди RS для кратних помилок

Під помилкою розуміють не один помилковий двійковий символ, а елемент поля Галуа, тобто кілька двійкових символів.

Твірний поліном для s=2 містить чотири співмножники

$$RS(x) = (x-\alpha)(x-\alpha^2)(x-\alpha^3)(x-\alpha^4) =$$
$$= x^4 + x^3(z^4+z^2+z^3+z) + x^2(z^7+z^6+z^4+z^3) + x(z^9+z^8+z^7+z^6) + z^{10}$$

$$(x-\alpha)(x-\alpha^2) = x^2 + x(z^2+z) + z^3$$

$$(x-\alpha^3)(x-\alpha^4) = x^2 + x(z^4+z^3) + z^7$$

$$\begin{aligned}
RS(x) = \quad & x^2 + x(z^2+z) + z^3 \\
* \quad & \underline{x^2 + x(z^4+z^3) + z^7} \\
& z^7x^2 + x(z^9+z^8) + z^{10} \\
& + x^3(z^4+z^3) + x^2(z^6+z^4) + x^4 + x(z^7+z^6) \\
& \underline{+ x^4 + x^3(z^2+z) + x^2z^3} \\
& x^2z^7 + x\,z^9 + x\,z^8 + z^{10} \\
& + x^3z^4 + x^3z^3 + x^2z^6 + x^2z^4 + x\,z^7 + x\,z^6 \\
& \underline{+ x^4 + x^3z^2 + x^3z + x^2z^3} \\
& x^4 + x^3(z^4+z^2+z^3+z) + x^2(z^7+z^6+z^4+z^3) + x(z^9+z^8+z^7+z^6) + z^{10}
\end{aligned}$$

Цей твірний поліном є справедливим для кожного поля Галуа.

Для обчислення твірного полінома для конкретного поля необхідно обчислити відповідні коефіцієнти при псевдозмінних. Для цього кожен відповідний коефіцієнт у загальній формі необхідно розділити на примітивний поліном p(z). Залишок від ділення і буде являти собою шуканий коефіцієнт.

Приклад: Код RS виправляє подвійну помилку і будується для поля GF(16) як розширення поля GF(2) над примітивним поліномом $p(z)=z^4 + z + 1$.

Тоді довжина коду становить n=15 двійкових тетрод. Кількість перевірних символів p=4 тетради. Кількість інформаційних символів k=11 тетрад.

Побудова коду RS (15, 11) починаємо зі спрощення твірного полінома, отриманого для загального випадку. Спростити цей поліном можна, розділивши кожний з його коефіцієнтів на твірний поліном поля Галуа.

1. $z^4+z^2+z^3+z \;\big|\; z^4 + z + 1$
$- z^4 + z + 1 \;\big|\; 1$
$z^3 + z^2 + 1$

2. $z^7+z^6+z^4+z^3 \;\big|\; z^4 + z + 1$
$- z^7+z^4+z^3 \;\big|\; z^3+z^2$
z^6
$- z^6+z^3+z^2$
z^3+z^2

3. $z^9+z^8+z^7+z^6 \;\big|\; z^4 + z + 1$
$- z^9+z^6+z^5 \;\big|\; z^5+z^4+z^3$
$z^8+z^7+z^5$
$- z^8+z^5+z^4$
z^7+z^4
$- z^7+z^4+z^3$
z^3

$$
\begin{array}{rl}
4. \quad z^{10} & \left| \begin{array}{l} z^4 + z + 1 \\ \hline z^6 + z^3 + z^2 + 1 \end{array} \right. \\
\underline{- \; z^{10}+z^7+z^6} & \\
z^7+z^6 & \\
\underline{- \; z^7+z^4+z^3} & \\
z^6+z^4+z^3 & \\
\underline{- \; z^6+z^3+z^2} & \\
z^4+z^2 & \\
\underline{- \; z^4+z+1} & \\
z^2+z+1 &
\end{array}
$$

Таким чином твірним поліномом коду RS (15, 11), що виправляє подвійні помилки, для поля GF(16) є

$$RS(x) = x^4 + x^3(z^3+z^2+1) + x^2(z^3+z^2) + x\,z^3 + z^2 + z + 1$$

ЛЕКЦІЯ 18 ЗАСТОСУВАННЯ МЕТОДІВ ТЕОРІЇ ІНФОРМАЦІЇ І КОДУВАННЯ

18.1 Призначення кодів CRC

Циклічні надлишкові коди CRC (cyclic redundancy check) є алгоритмами перевірки цілісності даних на основі обчислення контрольних сум із використанням властивостей циклічних кодів. Коди CRC призначені для виявлення помилок передавання даних каналами зв’язку.

Контрольні суми підраховують для кількох блоків повідомлення. Зазвичай для отриманої контрольної суми виділяють окремий блок. Цей блок часто розміщують у кінці даних для передавання або зчитування по завершенню інформативної частини повідомлення.

Декодування інформативного повідомлення супроводжується підрахунком контрольної суми на приймальній стороні. Отриману суму порівнюють з контрольною сумою переданого повідомлення. При співпадінні сум робиться висновок про відсутність спотворення. передавання або зчитаних даних даних без спотворень. А у разі відмінностей контрольних сум ініціюють повторне виконання операцій передавання або зчитування повідомлення.

Коди CRC широко застосовують для контролю пересилання даних у комп’ютерних мережах, пристроях зберігання даних (зокрема в пристроях HDD), обчисленні хеш-функцій, для реалізації криптографічних перетворень тощо.

18.2 Загальна характеристика алгоритмів CRC

Алгоритми перетворень у кодах CRC ґрунтуються на властивостях ділення з остачею інформаційних блоків, які подають як поліноми над скінченим полем Галуа GF(2). Дільник подають як твірний поліном.

Обчислення залишку R(x) виконують за формулою

$$R(x)=P(x)*x^n \bmod G(x)$$

Де P(x) – вихідний поліном інформаційного повідомлення;

G(x) – твірний поліном степені n;

n – степень твірного полінома.

Наприклад, інформаційне повідомлення може складатись з послідовності слів: бітів (коди CRC-1) або з байтів (коди CRC-8).

За наявності 1 у старшому розряді слова, слово повідомлення зсувається вліво на один розряд і виконується операція XOR між бітами слів і твірним поліномом. За наявності 0 у старшому розряді слова, слово зсувається вліво на один розряд. Операції продовжуються до завершення послідовності слів інформаційного повідомлення. Результатом виконання цих операцій є залишок результату ділення. Він і становить значення контрольної суми інформаційного повідомлення (рис. 18.1).

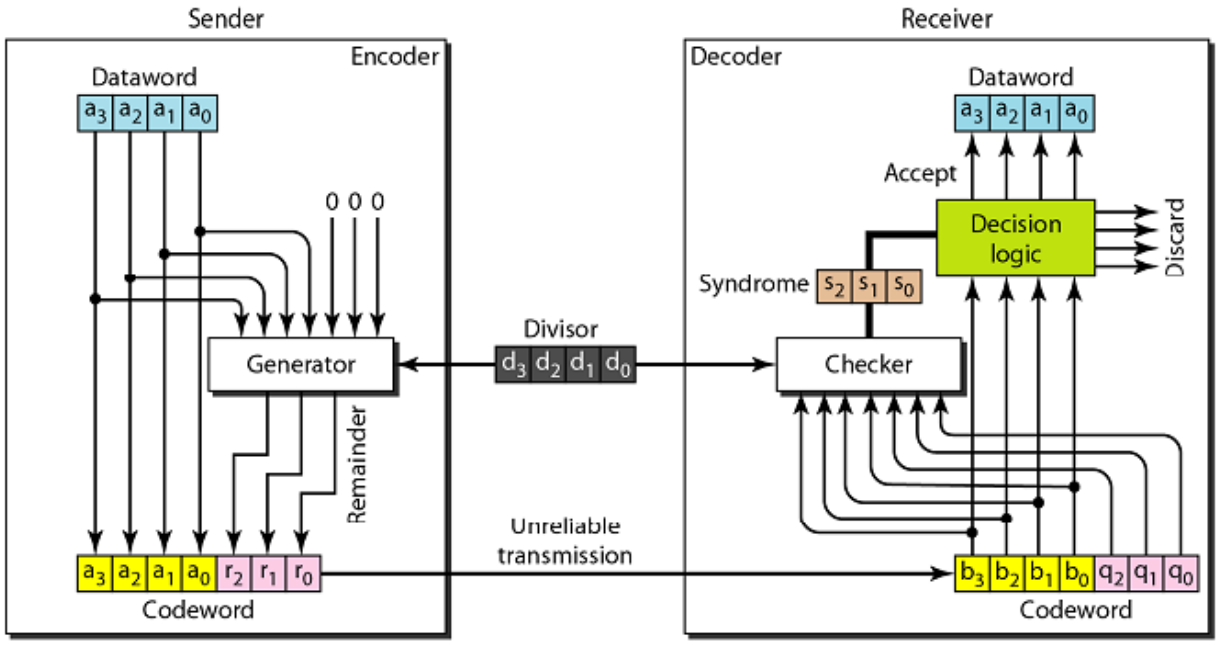


Рисунок 18.1 – Кодування і декодування за CRC

Приклад формування залишку показано на рис. 18.2

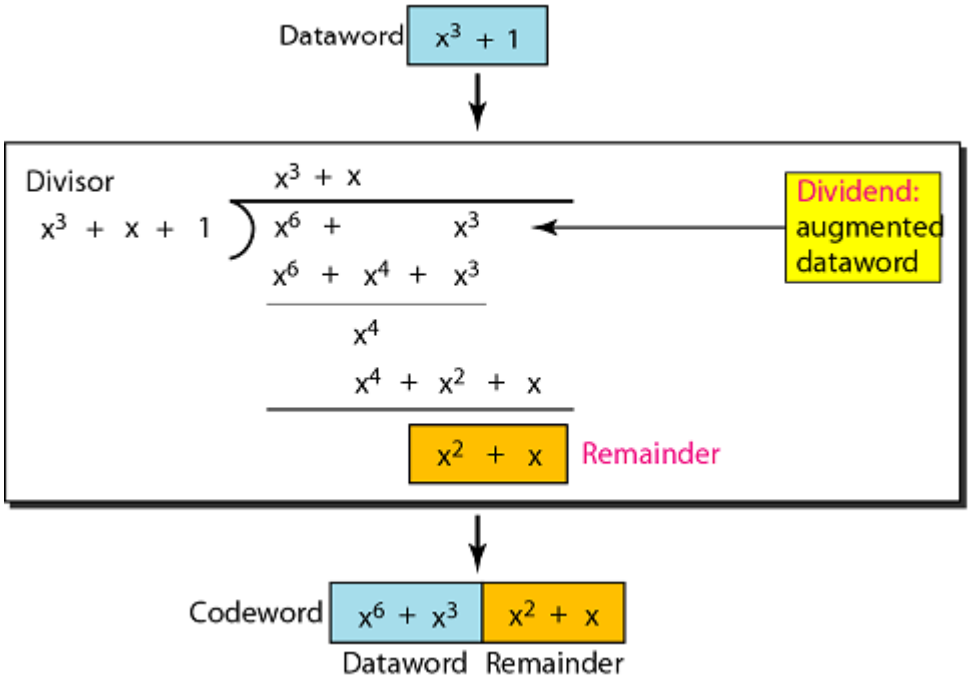


Рисунок 18.2 – Приклад кодування слова dataword за CRC

Нехай інформаційне повідомлення задано 14 бітами 11010011101100. Твірним поліномом є $x^3 + x + 1$, який подається у двійковій формі як $1*x^3 + 0*x^2 + 1*x + 1$.

Послідовне ділення інформаційного повідомлення на твірний (породжуючий) поліном відповідає таким перетворенням

Перший крок ділення

```
11010011101100 000 <--- повідомлення завершено 3 бітами справа
1011               <--- дільник (4 біти) G(x) = x³ + x + 1
------------------
01100011101100 000 <--- результат
```

Тобто ділення виконується шляхом зсуву дільника до наступної послідовності, отриманої у результаті зсуву даних перетворення

```
11010011101100 000 <---
1011               <---
01100011101100 000 <---
 1011              <---
00111011101100 000
  1011
00010111101100 000
   1011
00000001101100 000 <--- перехід до наступного старшого розряду 1
```

Оскільки наступні розряди є нульовими, то зсув відбувається не на один розряд, а на чотири розряди

```
00000001101100 000
       1011
00000000110100 000
        1011
00000000011000 000
         1011
00000000001110 000
          1011
00000000000101 000
```

```
           101 1
-----------------
00000000000000 100 <--- залишок (з біти) Завершення
```

Тобто залишок буде 100.

18.3 Принципи побудови кодів CRC

Особливості побудови кодів CRC визначаються степенем твірного полінома n. Відповідно довжина твірного полінома визначають як n+1. До найбільш вживаних поліномів відносять поліноми з довжинами 9 біт (CRC-8), 17 біт (CRC-16), 33 біт (CRC-32). 65 біт (CRC-64).

Для заданої довжини n+1 поліному і залишку довжиною n можливим є застосування різних твірних поліномів. Тому код CRC зазвичай позначають як CRC-n-XXX, де XXX відповідає варіанту твірного полінома або поліномів. Визначальним при виборі поліному є максимальна довжина інформаційного блоку повідомлення, бажані характеристики виявлення помилок, продуктивність перетворень тощо.

Важливою характеристикою твірного полінома є його примітивність, тобто незвідність. А звідність поліному призводить до збільшення пропущених помилок у повідомленні.

Перевагою примітивних поліномів є збільшення максимальної припустимої довжини блоку повідомлення, оскільки всі одинарні бітові помилки мають різні залишки (синдроми). А це дозволяє виявляти всі помилки у двох бітах.

Для примітивного поліному степені r максимальна довжина блоку становить 2^r-1 для виявлення одинарних помилок. Однак для примітивного

полінома p(x) степеня r-1 з використанням полінома g(x)=p(x)(1+x), максимальна довжина блоку якого становить 2^{r-1}-1, код дозволяє виявляти одинарні, подвійні, потрійні і непарні помилки довільної кратності. Тобто можна балансувати довжини повідомлень з виявленням помилок шляхом факторизації твірного поліному.

Зазвичай значення r вибирають достатньо великим, а довжину блока наближену до максимального значення блока.

Якщо деякий поліном c(x) створено з використанням g(x), то

$$c(x) \bmod g(x)=0$$

Тому за наявності помилки e(x)

$$(e(x) + c(x)) \bmod g(x)= e(x) \bmod g(x)$$

Для одинарної помилки $e(x)=x^k$ для деякого значення k. Для виявлення цієї помилки необхідно, щоб g(x) не ділило націло X^k.

Для парної помилки відповідно для x^k і x^m .

$$e(x)=x^k + x^m = x^k (1+ x^{m-k}),\ k<m$$

то g(x) не повинно ділити $1+ x^j$, j=1,2,…,n за наявності парної помилки.

Стандарти різних кодів CRC передбачають застосування відповідних твірних поліномів. Зокрема, поширеними є застосування таких поліномів

$$g(x) = x^8 + x^2 + x + 1 \qquad \text{(CRC-8)}$$
$$g(x) = x^{10} + x^9 + x^5 + x^4 + x^2 + 1 \qquad \text{(CRC-10)}$$
$$g(x) = x^{16} + x^{12} + x^9 + x^5 + x^4 + x^2 + 1 \qquad \text{(CRC-16)}$$
$$g(x) = x^{16} + x^{15} + x^2 + 1 \qquad \text{(CRC-16)}$$
$$g(x) = x^{16} + x^{12} + x^5 + 1 \qquad \text{(CRC-CCITT)}$$
$$g(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1 \qquad \text{(CRC-32)}$$

Для побудови кодів використовують примітивні поліноми. Поширеними є такі примітивні поліномии

$p(x)$	$p(x)$
$x^2 + x + 1$	$x^{10} + x^3 + 1$
$x^3 + x + 1$	$x^{11} + x^2 + 1$
$x^4 + x + 1$	$x^{12} + x^6 + x^4 + x + 1$
$x^5 + x^2 + 1$	$x^{13} + x^4 + x^3 + x + 1$
$x^6 + x + 1$	$x^{14} + x^{10} + x^6 + x + 1$
$x^7 + x^3 + 1$	$x^{15} + x + 1$
$x^8 + x^4 + x^3 + x^2 + 1$	$x^{16} + x^{12} + x^9 + x^7 + 1$
$x^9 + x^4 + 1$	$x^{17} + x^3 + 1$

18.4 Згорткові коди

18.4.1 Загальна характеристика

Згорткові коди є кодами з пам’яттю. Вони належать до класу деревоподібних кодів, які є одним із найпоширеніших класів кодів, що виправляють помилки.

На відміну від блокових кодів, кодове слово деревоподібного коду n_0 (кадр кодового слова) формується в залежності не тільки від самого інформаційного блоку k_0 (інформаційного кадру), але також від m уже переданих інформаційних кадрів.

Величину $k=(m+1)k_0$ називають інформаційною довжиною слова. Кодова довжина блоку $n=(m+1)n_0$ є довжиною кодового слова, на якій зберігається вплив одного кадру інформаційних символів.

Основними напрямами застосування цих кодів є:

- радіозв'язок, у тому числі захищений спцуіального призначення (IMT'2000, GSM, IS'95);
- цифровий наземний і супутниковий зв'язок;
- радіомовні системи..

18.4.2 Коди Івадарі

Нехай λ і n0 – будь які позитивні числа. За визначенням кодом Івадарі називають двійковий систематичний згортковий код з виправленням пакетів помилок. Код задають твірною $(n_0 – 1)$ x n_0 матрицею поліномів

$$\mathbf{G}(x)=\begin{bmatrix} 1 & & & g_1(x) \\ & 1 & & g_2(x) \\ & & \ddots & \vdots \\ & & 1 & g_{(n_0-1)}(x) \end{bmatrix}$$

Тут для матричних елементів $g_{in_0}(x)$ використовують скорочене позначення $g_i(x)$

$$g_i(x)=x^{(\lambda+1)(2n_0-i)+i-3}+x^{(\lambda+1)(n_0-i)-1},\ i=1,\ldots,n_0-1\ldots$$

Найбільший степінь $(\lambda+1)(2n_0-1)-2$ має поліном $g_1(x)$.

Таким чином, коди Івадарі є згортковимит $((m+1)n_0,(m+1)(n_0-1))$ - кодами, що виправляють пакети помилок довжини не більш $\lambda*n_0$, з числом кадрів $m=(\lambda+1)(2n_0-1)-2$.

Оскільки $n_0-k_0=1$ для кодів Івадарі одержуємо тільки один синдромний поліном

$$s(x)=\sum_{i=0}^{n_0-1}g_i(x)e_i(x)+e_{n_0}(x)=$$
$$=e_{n_0}(x)+\sum_{i=0}^{n_0-1}\left[x^{(\lambda+1)(n_0-i)-1}+x^{(\lambda+1)(2n_0-i)+i-3}\right]e_i(x).$$

Побудова кодуючих пристроїв для кодів Івадарі є достаньо простою процедурою. Декодування передбачає однакову структуру декодерів цих кодів.

18.5 Сучасні методи стискання даних

18.5.1 Метод Deflate

Метод Deflate є комбінацією алгоритму LZ77 з алгоритмом кодування Хаффмана. Алгоритм LZ77 дозволяє описати потік даних як послідовність пар вказівник і відстань, та літералів. У традиційному описі LZ77 вихід повністю складається з трійок відстаней. Літерал у трійці позначає літерал, який надходить одразу після закінчення попередньої узгодженої послідовності. Коли не знайдено відповідності, перший символ перегляду все ще кодується як трійка з відстанню та довжиною нуля. Алгоритм LZ77 не визначає, як слід кодувати трійки.

Найбільш очевидним способом було б кодування кожного з елементів у трійці з фіксованою кількістю бітів залежно від кількості значень, які може прийняти елемент. Однак використання методу ентропійного кодування, як правило, призводить до кращих коефіцієнтів стиснення.

Deflate використовує кодування Хаффмана для кодування виходу LZ77. Літерали та довжини зібрані в один алфавіт, а відстані складають інший. Таким чином, і коди прямої довжини, і коди відстані кодуються змінною кількістю бітів відповідно до їх відносної частоти.

Кодування складається з генерування коду довжини та коду відстані. Коли не знайдено відповідності, створюється лише код для літералу. Дешифратор визначає, чи є код для прямолінійної чи довжини. Якщо є для літералу, він розшифровує це буквальне. В іншому випадку він прочитає наступний код, щоб отримати відстань відповідності, а потім зможе вивести відповідну послідовність. Незважаючи на те, що коди мають змінну кількість

бітів, декодер знає, де закінчується кожен код, оскільки коди Хаффмана є кодом префікса.

Замість використання одного коду для кожної з довжин та відстаней, що призведе до досить великих кодів Хаффмана, Deflate призначає діапазон довжин або відстаней кожному коду. Застосовується змінна кількість зайвих біт для розрізнення кількох довжин / відстаней, які потрапляють у кожен код.

Для фази кодування можуть використовуватися два типи кодування Хаффмана: статистичне Хаффмана і динамічне (адаптивне) кодування Хаффмана. Останній змінюється на основі блоку. Тобто вихід розподіляється на блоки, кожен з яких кодується певним кодом Хаффмана. Код кожного блоку зберігається компактно на початку блоку. Новий блок запускається, коли новий код вважається необхідним для поліпшення коефіцієнта стиснення. В Deflate використовується динамічне кодування Хаффмана.

Формат, створений для Deflate, є простим. Перше, що слід зазначити, це те, що вихідний потік розділений на блоки. Кожен блок обмежений кінцем попереднього блоку (або початком потоку) і символ кінця блоку. Немає обмежень по довжині для кожного блоку: він може бути порожнім (тобто містити лише символ кінця блоку) або може бути довгим, скільки потрібно.

Кожен блок починається із заголовка в 3 біти. Перший біт у блоці вказує, чи є цей блок останнім блоком у потоці чи ні. Якщо він є, біт встановлюється на одиницю, інакше його значення дорівнює нулю. Два наступних біта вказують на тип блоку, який може бути одним із трьох: нестиснений блок (00), статистичний блок Хаффмана (01) або динамічний блок Хаффмана (10). Код 11 не використовується і призведе до помилки, якщо його знайде декодер.

Потік Deflate може містити комбінацію цих типів блоків. Наприклад, він може містити в основному динамічні блоки Хаффмана, але використовувати нестиснені блоки для частини потоку з особливо високою ентропією. Після

заголовка, що залишився в блоці складається з кодів довжини і відстані в прямому сенсі з відповідними додатковими бітами.

Літерали кодуються своїм кодом, тоді як для пар відстаней і довжина спочатку кодується, а потім її додаткові біти (якщо такі є), а потім код відстані та його зайві біти. Результатом виводу є байти, які заповнюються починаючи з їх найменш значущого біта (LSB) до найбільш значущого біта (MSB). Коди пакуються починаючи з MSB, тоді як додаткові біти пакуються починаючи з LSB.

На рисунку 18.3 показано приклад того, як певна послідовність літералів і збігів упаковується в блок «Deflate». Таблиці 3.1, 3.2 та 3.3 містять відповідні дані для цього прикладу. Перший кодований біт встановлюється, вказуючи, що це останній блок. Далі, блок 01 кодується, сигналізуючи про це як статистичний блок Хаффмана.

Літерал для символу A відповідає символу з номером 65 в алфавіті довжини прямолінійної форми, з таблиці 3.3 літерали в діапазоні 0–143 кодуються 8 бітами, код 00110000 (десятковий 48) відповідає символу 0. Отже, код для A дорівнює 01110001 (десятковий 113 = 48 + 65). Оскільки коди упаковані спочатку з MSB, це показано як приклад 10001110.

Далі кодується відповідність з відстані 1 та довжиною 21. Це означає, що частина самої головки пошуку повторюється. З таблиці 3.1 довжина 21 відповідає номеру символу 269, кодованому як 0001101 і упакованому як 1011000. Довжина 21 також потребує 2 додаткових біт: оскільки символ 269 містить довжини в діапазоні 19–22, а 21 - 19 = 2 зайвих біт буде 10. Той самий процес використовується для кодування відстані 1 як 00000 без зайвих бітів, згідно з таблицею 3.2 і за тим, що всі символи відстані кодуються 5 бітами. Решта літери та збіги кодуються таким же чином. Коли вхідний потік закінчується, Deflate також повинен закінчити блок. Таким чином, він виробляє код для символу кінця блоку, 0000000.

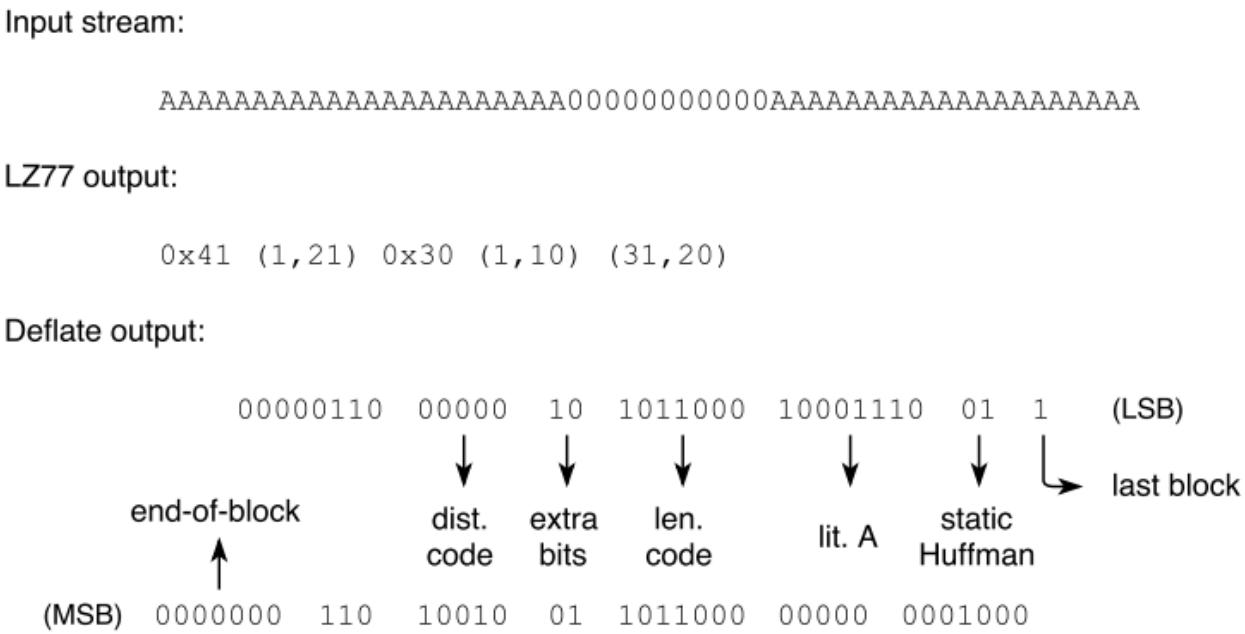


Рисунок 18.3 - Приклад кодування вхідного потоку у вигляді блоку Deflate

18.5.2 Алгоритм Gzip

Потоки Deflate зазвичай упаковуються в інший формат з більшою кількістю функцій. Прикладом програмної реалізації Deflate є програма Gzip. Програма призначена для стиснення відповідного формату, в якому використовується Deflate.

Потоки Gzip додають заголовок та колонтитул до Deflate. Заголовок підтримує такі функції, як вказівка початкового імені файлу, час його модифікації, додавання опису файлу та операційної системи, місце створення.

Заголовок містить поле із зазначенням методу стиснення. Це означає, що файли Gzip можуть підтримувати інші типи корисних завантажень, але використовується лише Deflate.

Нижній колонтитул містить циклічну перевірку нестиснених даних. Це дозволяє виявити помилки в нестиснених даних, а також вихідний розмір стисненого потоку..

На рисунку 18.4 показано приклад заголовку файлу Gzip. Цей файл використовує найкоротший дійсний заголовок Gzip (тобто не включаються необов'язкові поля), а корисним завантаженням є порожній блок "Зняти".

Перші два байти складають ідентифікатор, який є "магічним числом", що ідентифікує файл Gzip. Наступний байт CM визначає метод стиснення, що використовується для корисного завантаження.

Деякі біти поля FLG можна встановити, щоб вказати, що додаткові поля для заголовка є. Наприклад, найбільш часто вживаний біт 3 вказує на те, що присутнє необов'язкове поле з оригінальним іменем файлу.

Наступні чотири байти зберігають час модифікації стисненого файлу в секундах відносно епохи Unix (1 січня 1970 р.). Нульове значення модифікації означає, що час модифікації не було включено.

Байт XFL дозволяє вказувати додаткові прапорці для використовуваного методу стиснення. Байт ОС вказує операційну систему, де був стиснутий файл (наприклад, 03 призначено для Unix, а FF означає, що система "невідома").

Нижній колонтитул слідує за полегшеним завантаженням Deflate, яке, за потреби, заповнено нульовими бітами. Перші чотири байти колонтитула зберігають CRC32 нестиснених даних, отриманих з використанням полінома CRC-32, зворотним поданням якого є EDB88320. Останні чотири байти мають розмір вихідного потоку за модулем 2^{32}.

```
Header:     1F8D  08   00   00000000  00   03
             ID   CM  FLG    MTIME    XFL  OS

Payload:    0300

Footer:     00000000  00000000
             CRC32      ISIZE
```

Рисунок 18.4 - Заголовок і колонтитул для мінімального файлу Gzip

Швидкість Gzip частково залежить від процедур пошуку відповідності. Коли Gzip читає вхід, то хешує кожні 3 байти і зберігає місця, які відповідають кожному хеш-значенню.

Коли Gzip починає пошук нового збігу, то хешує перші 3 байти буфера пошуку. Потім програма отримує доступ до кожного з місць у вікні, хеш яких однаковий. Для кожного з цих місць Gzip: 1) перевіряє, чи дійсно 3 байти відповідають, що може не відбутися через хеш-колізії; 2) порівнює вікно з підказкою, щоб знайти довжину відповідності.

Хеш-функція, яку використовує Gzip, є рекурсивним хешем, тобто хеш, який обчислюється для n-граму на основі хеша для попереднього n-грама та нового символу в цьому n-грамі. У Gzip використовуються триграми (n = 3), оскільки мінімальна припустима довжина збігу - три байти.

Тип рекурсивного хешування, що використовується, називається хешированием циклічним многочленом. У цьому типі хешів множення можна обчислити зі зміщеннями та доповненнями з ексклюзивними виборками, що призводить до скорочення часу обчислення.

На рисунку 18.5 показано приклад хеш-ланцюга. Розмір вікна становить 32 кБ. Для позначення позицій у ньому потрібно принаймні 15 біт. Головний і попередній зберігають 2^{15} позицій. Тому вони оголошені з розміром 64 кБ.

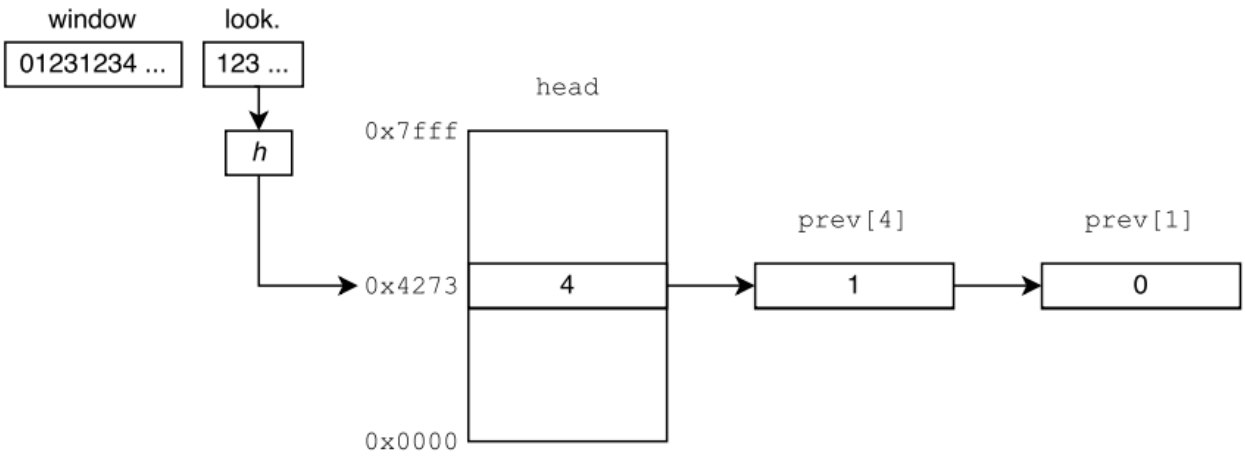


Рисунок 18.5 – Приклад хеш-ланцюга, що містить дві позиції з тим же хешем, що і поточний вигляд.

У Gzip вхід зчитується до буфера з подвоєним розміром вікна. Коли буфер стає повним, друга половина переміщується до першої. Коли це відбувається, значення, збережені в head і prev, повинні бути оновлені, щоб залишатися дійсними: розмір вікна WSIZE віднімається від кожного значення. Значення, які стали б негативними, встановлюються на 0, фактично виводячи із занадто старих позицій хеш-ланцюга. Введення в хеш-ланцюг є постійною операцією в часі.

Gzip підтримує дев'ять рівнів стиснення, які дозволяють міняти швидкість стиснення для поліпшення коефіцієнта стиснення з обмеженням часу на пошук відповідностей. Рівні коливаються від 1 (найшвидший) до 9 (найкращий коефіцієнт стиснення). Виконання gzip -6 з рівнем стиснення 6 є типовим. Для рівнів 1–3 пропускається введення послідовностей, що, ймовірно, не призведе до значного стиснення. Це також запобігає збільшенню довжини хеш-ланцюга за рахунок цих послідовностей і скорочує час, витрачений на вставки та майбутні пошуки відповідностей у цьому ланцюжку. Для рівнів 4–9 параметр дозволяє пропустити оцінку відповідності для деяких співпадінь, що може скоротити час виконання.

БАЗОВА ЛІТЕРАТУРА

1. Коваленко А.Є. Побудова кодів на основі типових алгоритмів кодування даних : методичні вказівки із самостійної роботи студентів з дисципліни «Теорія інформації і кодування» / Уклад. А.Є.Коваленко. Київ: ННК «ІПСА» НТУУ «КПІ», 2012.71 с.

2. Теорія інформації і кодування : Методичні вказівки до проведення практичних занять для студентів напрямів підготовки 6.040302 «Інформатика», 6.040303 «Системний аналіз» / Уклад. А.Є.Коваленко. Київ: ННК «ІПСА» НТУУ «КПІ», 2013. 42 с.

3. Теорія інформації і кодування : Методичні рекомендації до виконання модульної контрольної роботи та залікової роботи для студентів напрямів підготовки 6.040302 «Інформатика», 6.040303 «Системний аналіз / Уклад. А.Є.Коваленко. Київ.: ННК «ІПСА» НТУУ «КПІ», 2013. 22 с.

4. Коваленко А.Є. Побудова кодів на основі типових алгоритмів кодування даних : методичні вказівки із самостійної роботи для студентів з дисципліни «Теорія інформації і кодування» підготовки бакалаврів за спеціальністю “Системний аналіз”2-ге вид., розшир. та доповн. / Уклад. А.Є.Коваленко. Київ.: ІПСА НТУУ «КПІ імені Ігоря Сікорського», 2017. 151 с.

5 Коваленко А.Є. Теорія інформації та кодування: Практикум для студентів напряму підготовки 6.040303 «Системний аналіз». Київ:НТУУ «КПІ», 2014.198 с.

6. Теорія інформації і кодування: Контрольні завдання до модульної контрольної роботи [Електронний ресурс] : навч. посіб. для студ. спеціальності 124 «Системний аналіз», спеціалізацій «Системний аналіз та управління», «Системний аналіз фінансового ринку». 2-ге вид., допов. і переробл./ КПІ ім. Ігоря Сікорського ; уклад.: А.Є.Коваленко. Електронні текстові дані (1 файл: 1,02 Мбайт). Київ : КПІ ім. Ігоря Сікорського, 2018. 54с.

ДОПОМІЖНА ЛІТЕРАТУРА

7. Shannon C.E. A Mathematical Theory of Communication. The Bell System Technical Journal.- 1948. July, October,Vol. 27.p. 379–423, 623–656.

8. Hamming R.W. Coding and information theory Second ed.Englewood Cliffs NJ: Prentice Hall, 1986. 260 p.

9. Тулякова Н.О. Теорія інформації: Навчальний посібник. Суми: Вид-во СумДУ, 2008. 212 с. 2010. 248с.

10. Mauro Barni, Benedetta Tondi Lecture notes on Information Theory and Coding. Siena: Universit` a degli Studi di Siena Facolt`a di Ingegneria, 2012. 156p