КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА

Іванов Є.О., Ченцов О.І., Шевченко В. П.

ДИСКРЕТНА МАТЕМАТИКА

РОБОЧИЙ ЗОШИТ

з українсько-англійським тематичним словником

Теорія автоматів

Теорія кодування

КИЇВ-2011

Іванов Є.О., Ченцов О.І., Шевченко В. П.

Дискретна математика. Робочий зошит з українсько-англійським тематичним словником. Теорія автоматів. Теорія кодування. – К., 2011. – 60 с.

У книзі подано матеріали для вивчення шостого та сьомого модулів «Теорія автоматів» та «Теорія кодування» курсу дискретної математики студентами 1 курсу факультету кібернетики напрямків підготовки «Інформатика» та «Програмна інженерія».

Робочий зошит призначено для ведення у ньому конспекту з дискретної математики, для полегшення і прискорення конспектування тут наведені всі слайди лекцій, що позбавляє студента необхідності перемальовувати формули і рисунки, записувати з голосу означення і формулювання. Крім того у зошиті подані плани семінарських занять (теоретичні питання, умови аудиторних, домашніх і додаткових завдань), література до відповідного модулю курсу, українсько-англійський тематичний словник, який полегшить користування іноземною літературою з дискретної математики.

Всі матеріали зошиту можна знайти в електронній бібліотеці факультету кібернетики за адресою: ***http://www.unicyb.kiev.ua/Library/DM*** .

Рецензенти:

О.А.Летичевський, академік НАН України, д-р фіз.-мат. наук;

А.Ю.Дорошенко, д-р фіз.-мат. наук,

П.О.Бех, канд.філол.наук

Затверджено Радою факультету кібернетики
21 червня 2011 року, протокол № 11.

Друкується за авторською редакцією.

ЗМІСТ

МОДУЛЬ 6. ТЕОРІЯ АВТОМАТІВ

Історична довідка

Теорія автоматів, як розділ дискретної математики, вивчає абстрактні машини у вигляді математичних моделей, а також проблеми, які вони можуть розв’язувати. Теорія автоматів найбільш тісно пов’язана з теорією алгоритмів. Це пояснюється тим, що автомат перетворює дискретну інформацію по крокам у дискретні моменти часу і формує результат по крокам даного алгоритму. Ці перетворення можливі за допомогою як технічних, так і програмних засобів. Автомат можна уявляти, як деякий пристрій (чорну скриню), якій подають вхідні сигнали і одержують вихідні, пристрій може мати деякі внутрішні стани.

Абстрактна теорія автоматів — напрям в теорії автоматів, який характеризується тим, що при вивченні автоматів не беруть до уваги їх структурні особливості. За такого підходу, внутрішні стани автомата, його вхідні та вихідні сигнали розглядаються як деякі абстрактні символи, які утворюють відповідні алфавіти.

В абстрактній теорії автоматів основними конструктивними об'єктами для вивчення є скінченні автомати, а також реалізовані ними оператори та множини, які ними представляються (скінченно-автоматні оператори та множини). В абстрактній теорії автоматів широко застосовуються методи та поняття алгебри, математичної логіки та теорії алгоритмів.

Центральними проблемами абстрактної теорії, які породжені практичними завданнями конструювання та експлуатації обчислювальної техніки та отримали належний теоретичний розвиток, є проблеми синтезу, аналізу та мінімізації.

Проблема синтезу полягає в пошуку та побудові автомата, виходячи з умов, які висуваються до реалізованого ним оператора або до множини, яка ним представляється.

Проблема аналізу є оберненою до проблеми синтезу: за заданим автоматом потрібно описати його поведінку засобами певної мови. В певному сенсі аналіз та синтез можна розглядати як переклади з однієї мови на іншу, при цьому переклад, який відповідає аналізові, переважно легший. Розроблено багато алгоритмів синтезу та аналізу, головним чином для скінченних детермінованих автоматів.

Мінімізація автомату полягає в побудові за допомогою формальних перетворень еквівалентного за своїми функціональними можливостями автомата, який задовольняє деяким критеріям оптимальності, наприклад, у випадку скінченних автоматів має найменшу кількість станів.

Перші обчислювальні машини проектувалися на основі інженерної інтуїції. Проте дуже швидко основою для проектування ЕОМ стала теорія автоматів.

Перший, хто висловив думку про можливість застосування математичної логіки для проектування технічних пристроїв був Шенон - в США, а у СРСР - В.І.Шестаков і М.А.Гаврилов. Вони застосували найпростіший апарат формальної математичної логіки для конструювання перемикальних ланцюгів комутаторів телефонних станцій. Але виявилося, що він придатний і для простих електронних схем, тому у післявоєнні роки, коли почала розвиватися цифрова обчислювальна техніка, стали з'являтися

спроби застосування цього апарату для вирішення задач синтезу схем ЕОМ. Починаючи з 60-х років теорія автоматів спрямована на реальні завдання проектування машин стала інтенсивно розвиватись у Києві В.М.Глушковим та його учнями.

Теорія автоматів була обрана Глушковим не випадково. Як алгебраїст Глушков бачив, що поняття автомата, запропоноване Кліні, Муром та іншими вченими, являло собою багату можливостями математичну модель дискретного перетворювача інформації, для вивчення якої міг бути застосований потужний апарат сучасної математики. У той же час розробка прикладної теорії на основі красивого математичного апарату могла привернути увагу інженерів, яким у той час бракувало математичної теорії для розробки пристроїв, що містять запам'ятовуючі елементи.

У силу великої загальності і методологічної близькості до реальних пристроїв, теорія автоматів стала основою для розробки моделей кібернетичних систем у найрізноманітніших прикладних областях.

Глушков Віктор Михайлович (1923–1982) математик, кібернетик

Народився в Ростові-на-Дону. Закінчив Ростовський університет (1948 р.). У 1948—1956 рр. — асистент, доцент, завідуючий кафедрою теоретичної механіки Уральського лісотехнічного інституту (Свердловськ). У 1956—1957 рр. — завідуючий лабораторією обчислювальної техніки Інституту математики АН України. З 1957 р. — директор Обчислювального центру, з 1961 р. — Інституту кібернетики АН України. Опублікував ряд видатних праць у галузі вищої алгебри, після чого перейшов до розробки таких питань теоретичної кібернетики, як створення теорії проектування дискретних пристроїв. Серед вагомих результатів — створення загальної теорії автоматів та дискретних перетворювачів, розробка обчислювальних машин з інтерпретацією алгоритмічних мов високого рівня.

У 1962 р. надруковано монографію В.М.Глушкова «Синтез цифрових автоматів», яка дала фахівцям з розробки обчислювальної техніки сучасний математичний апарат. Ця праця містила практичну методику синтезу цифрових автоматів. В.М.Глушков створив формальний математичний апарат, що дозволяв ефективно застосувати абстрактно-автоматні та інші алгебраїчні методи для розв’язання завдань блокового проектування обчислювальних машин.

З ім’ям Глушкова пов’язано впровадження обчислювальних машин та відкриття принципово нового макроконвейєрного способу організації обчислень, розробка національної мережі обчислювальних центрів, створення засобів інтелектуалізації кібернетичних пристроїв та штучного інтелекту.

Кліні, Стівен Коул (*Kleene, Stephen Cole*; 1909–1994) — американський математик, праці якого заклали основи теоретичної інформатики. Закінчив Прінстонський університет, де у 1934 захистив дисертацію. З 1935 року на викладацькій роботі. У 1939-40 роках будує свою теорію рекурсії, яка стала справою всього його життя. Під час II Світової війни служив на американському флоті. З 1946 працював професором, завідувачем кафедри у Вісконсіні. З 1979 на пенсії.

Нейман, Джон фон (*Neumann, John Von*; 1903–1957) — видатний математик, один з засновників кібернетики. Зробив значний внесок у розвиток багатьох галузей

сучасної математики: теорія автоматів, функціональний аналіз, ергодична теорія, обґрунтування квантової механіки та інш.

Народився у Будапешті у сім'ї банкіра. Свою першу друковану працю опублікував у 18 років. У 1925 одержує у Цюріху диплом інженера-хіміка і одночасно захищає дисертацію на звання доктора філософії. У 1927-29 роках публікує цикл праць з теорії множин, теорії ігор та квантовій механіці. У 1931 році Нейман переїжджає до США, де викладає у Прінстонському університеті. У 1943-46 роках коли у Пенсільванському університеті було створено першу ЕОМ, Нейман запропонував її розробникам цілий ряд нових технічних рішень, які дозволили відокремити розробку логічної схеми машини від її технічної реалізації. За допомогою створених за його безпосередньою участю нових машин, зокрема, були зроблені розрахунки американської водневої бомби. Маючи значний практичний досвід у створенні обчислювальних машин у кінці 40-х років Нейман створює загальну математичну (логічну) теорію автоматів.

Шеннон, Клод Елвуд (*Shannon, Claude Elwood*; 1916–2001) — американський математик й електротехнік, один з творців математичної теорії інформації, значною мірою визначив своїми результатами розвиток загальної теорії дискретних автоматів.

З шкільних років проявляв рівний інтерес як до математики, так і до радіоелектроніки. У 1932 році вступив до університету штату Мічіган, у 1936 – до Массачусетського технологічного інституту, який закінчив у 1940 році. Одержав два наукових ступені – магістра з радіоелектроніки та доктора математики.

У 1941 р. поступив на роботу до відомої Bell Laboratory – лабораторії компанії Белл. У 1948 році надрукував фундаментальну роботу «A Mathematical Theory of Communication», в якій сформулював основи теорії інформації, зокрема, визначення кількості інформації через ентропію, а також запропонував одиницю виміру інформації, яку пізніше назвали бітом.. Значну цінність становить також його інша праця — «Communication Theory of Secrecy Systems» (1949), у якій сформульовані математичні основи криптографії.

У 1957 році Шеннон став професором Массачусетського технологічного інституту, звідки через 21 рік пішов на пенсію.

A={a_1, a_2, a_3,…. a_n} - алфавіт

$a_{i_1} a_{i_2} a_{i_3} .....a_{i_k}$ $k \geq 0$

слово довжини k в алфавіті A

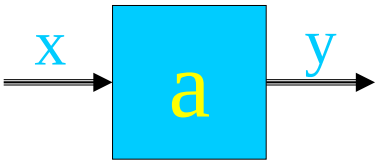
При $\boldsymbol{k}$=0 $\boldsymbol{e}$ - порожнє слово

A^* - множина всіх слів в алфавіті A

$l = a_{i_1} a_{i_2} .....a_{i_k}$, $m = a_{j_1} a_{j_2} .....a_{j_r}$

$l * m = a_{i_1} a_{i_2} .....a_{i_k} a_{j_1} a_{j_2} .....a_{j_r}$

Чорна скриня



t(0), t(1), t(2), …….

$$a(0) \xrightarrow[y(1)]{x(1)} a(1) \xrightarrow[y(2)]{x(2)} a(2) \xrightarrow[y(3)]{x(3)} \cdots\cdots$$

x(1), x(2), x(3), …→y(1), y(2), y(3), ...

$X^* \to Y^*$ згенероване алфавітне відображення

Автомат Мілі

<A,X,Y,δ,λ> $a_0 \in A$- початковий стан

A - множина станів
X - вхідний алфавіт
Y - вихідний алфавіт

δ:A×X→A
- функція переходів
λ:A×X→Y
- функція виходів

$$a_0 \xrightarrow[y_1=\lambda(a_0,x_1)]{x_1} a_1=\delta(a_0,x_1) \xrightarrow[y_2=\lambda(a_1,x_2)]{x_2}$$

$$a_2=\delta(a_1,x_2) \xrightarrow[y_3=\lambda(a_2,x_3)]{x_3} \cdots\cdots\cdots\cdots$$

Автомат Мілі

$< a_0, A, X, Y, \delta, \lambda >$ - ініціальний автомат

$x_1 x_2 x_3 x_4 \ldots\ldots x_n \in X^*$

$\Downarrow$

$a_1, a_2, a_3, a_4, \ldots\ldots a_n$

$y_1 y_2 y_3 y_4 \ldots y_n \in Y^*$

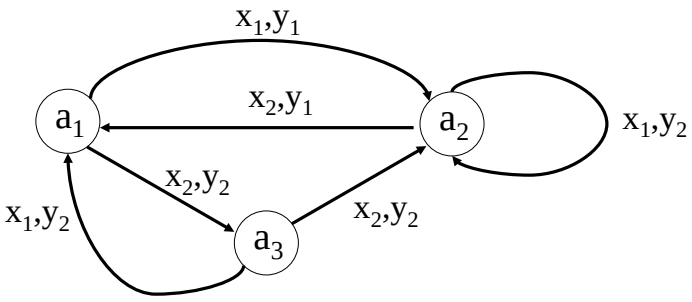
$a_i = \delta(a_{i-1}, x_i) \qquad y_i = \lambda(a_{i-1}, x_i)$

Вихідний символ в і-й момент часу залежить від попереднього стану в і-1-й момент часу та вхідного символу в і-й момент часу

$X=\{x_1,x_2\}\ Y=\{y_1,y_2\}\ A=\{a_1,a_2,a_3\}$

δ	a_1	a_2	a_3
x_1	a_2	a_2	a_1
x_2	a_3	a_1	a_2

λ	a_1	a_2	a_3
x_1	y_1	y_2	y_2
x_2	y_2	y_1	y_2



Автомат Мура

$<A, X, Y, \delta, \mu>$ $a_0 \in A$ - ініціальний стан

A - множина станів
X - вхідний алфавіт
Y - вихідний алфавіт

$\delta: A \times X \to A$
- функція переходів

$\mu: A \to Y$
- функція відміток

$$a_0 \xrightarrow[y_1=\mu(a_1)]{x_1} a_1=\delta(a_0,x_1) \xrightarrow[y_2=\mu(a_2)]{x_2}$$

$$a_2=\delta(a_1,x_2) \xrightarrow[y_3=\mu(a_3)]{x_3} \ldots\ldots\ldots\ldots$$

Автомат Мура

$< a_0, A, X, Y, \delta, \mu >$ - ініціальний автомат

$x_1 x_2 x_3 x_4 \ldots\ldots x_n \in X^*$

$\Downarrow$

$a_1, a_2, a_3, a_4, \ldots\ldots a_n$

$y_1 y_2 y_3 y_4 \ldots\ldots y_n \in Y^*$

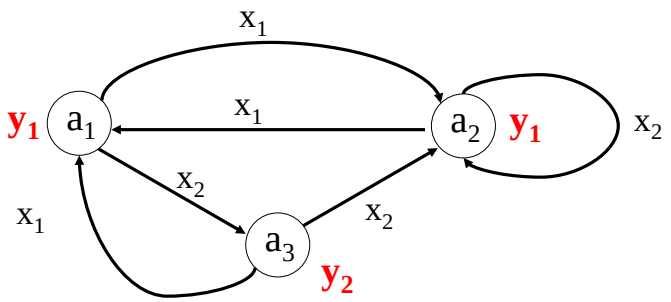
$a_i = \delta(a_{i-1}, x_i) \qquad y_i = \mu(a_i)$

Вихідний символ в і-й момент часу залежить тільки від стану в і-й момент часу (в який переходить автомат)

$X=\{x_1,x_2\}\ Y=\{y_1,y_2\}\ A=\{a_1,a_2,a_3\}$

δ	a_1	a_2	a_3
x_1	a_2	a_1	a_1
x_2	a_3	a_2	a_2

μ	a_1	a_2	a_3
	y_1	y_1	y_2



Теорема.

Для будь-якого автомата Мура B можна побудувати автомат Мілі A, що генерує теж саме алфавитне відображення.

І навпаки,

для будь-якого автомата Мілі A можна побудувати автомат Мура B, що генерує теж саме алфавітне відображення

Мура ⇒ Мілі

$\langle B,X,Y,\delta,\mu\rangle$ $\langle A,X,Y,\delta',\lambda\rangle$

$$A = B$$
$$\delta'(a,x) = \delta(a,x)$$
$$\lambda(a,x) = \mu(\delta(a,x))$$

Мілі ⇒ Мура

$\langle a_0,A,X,Y,\delta,\lambda\rangle$ $\langle b_0,B,X,Y,\delta',\mu\rangle$

$$B=A\times X\cup\{a_0\}$$
$$b_0=a_0$$
$$\delta'((a,x),x') = (\delta(a,x),x')$$
$$\mu((a,x)) = \lambda(a,x)$$
$$\delta'(a_0,x)=(a_0,x),\ \ \mu(a_0)=y_1$$

$$A: a_0 \xrightarrow[y_1]{x_1} a_1 \xrightarrow[y_2]{x_2} a_2 \xrightarrow[y_3]{x_3} \cdots\cdots$$

$$B: a_0 \xrightarrow[y_1']{x_1} (a_0,x_1) \xrightarrow[y_2']{x_2} (a_1,x_2) \xrightarrow[y_3']{x_3} \cdots\cdots$$

$a_i=\delta(a_{i-1},x_i)$ $\quad y_i=\lambda(a_{i-1},x_i)$

$y_1'=\mu((a_0,x_1))=\lambda(a_0,x_1)=y_1$

Доведемо за індукцією, що $y_i'=y_i$,
а і-м станом автомату Мура В буде (a_{i-1},x_i)

A: $a_i \xrightarrow[y_{i+1}]{x_{i+1}} a_{i+1} \xrightarrow[y_{i+2}]{x_{i+2}} \cdots\cdots\cdots$

B: $(a_{i-1},x_i) \xrightarrow[y_{i+1}]{x_{i+1}} (a_i',x_{i+1}) \xrightarrow[y_{i+2}']{x_{i+2}} \cdots\cdots\cdots$

$a_i=\delta(a_{i-1},x_i) \quad y_i=\lambda(a_{i-1},x_i)$

$a_i' = \delta(a_{i-1},x_i)=a_i$

$y_{i+1}'=\mu((a_i',x_{i+1}))=\mu((a_i,x_{i+1}))=$

$=\lambda(a_i,x_{i+1})=y_{i+1}$

Умови автоматності алфавітного відображення

$$\varphi:X^*\rightarrow Y^*$$

1. $|\varphi(l)|=|l|$

довжина вихідного слова дорівнює довжині вхідного слова

2. $l=l_1*l_2 \Rightarrow \varphi(l)=\varphi(l_1)*q$

образ префіксу слова є префіксом образу слова

Теорема.

Для того, що б алфавітне відображення $\varphi:X^*\rightarrow Y^*$ генерувалося деяким, можливо нескінченим, автоматом, необхідно і достатньо щоб:

1. $|\varphi(l)|=|l|$
2. $l=l_1*l_2 \Rightarrow \varphi(l)=\varphi(l_1)*q$

$\varphi: X^* \to Y^*$

$\langle a_0, A, X, Y, \delta, \lambda \rangle$

$A = X^*$

$a_0 = e$

$\delta(l, x) = l * x$

$\lambda(l, x)$= останній літері $\varphi(l * x)$

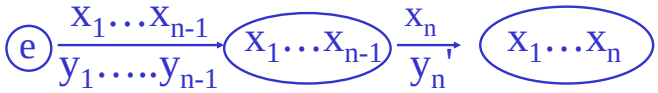
Будемо доводити за індукцією
по довжині вхідного слова l

$l = x_1$ $\varphi(l) = y_1$ оскільки $|\varphi(l)| = |l| = 1$

вихідне слово автомата = $\lambda(e, l) = \lambda(e, x_1) =$
$=$останній літері $\varphi(l) = y_1$

$l = x_1..x_n$ $\varphi(l) = y_1..y_n$ оскільки $|\varphi(l)| = |l| = n$

$\varphi(x_1 \ldots x_{n-1}) = y_1 \ldots .y_{n-1}$ за умовою 2



y_n'=останній літері $\varphi(x_1 \ldots x_n) =$
$=$останній літері $y_1..y_n = y_n$

Мінімізація автоматів

Мати
теж саме
за
меншу ціну

алфавітне
відображення

кількість
станів

$$a \xrightarrow[y]{x} a' \qquad x \in X,\ y \in Y$$

$$a \xrightarrow[r]{p} a' \qquad p \in X^*,\ r \in Y^*$$

Стан **a** під дією слова **p** видає слово **r**

$$b \xrightarrow[r']{p} b'$$

Стани a та b ззовні схожі,
якщо під дією однакових вхідних слів
видають однакові вихідні слова.

Стани a та b i-еквівалентні,
якщо під дією однакових вхідних слів
довжини не більше i,
видають однакові вихідні слова.

$$a \xrightarrow[r]{p} a', \quad b \xrightarrow[r']{p} b' \quad \Rightarrow \ r=r' \text{ для } |p| \le i$$

Стани, які i-еквівалентні для будь-якого i,
називаються еквівалентними
(∞-еквівалентними).

Лема 1. Відношення ∞-еквівалентності станів
є відношенням еквівалентності.

Теорема 2. Два автомата A та B генерують
одне й теж саме алфавітне відображення
тоді і тільки тоді, коли їх початкові стани
є еквівалентними (нескінченно еквівалентними).

Необхідність. Нехай $\varphi(p)= \varphi'(p)$

$$\text{A: } a_0 \xrightarrow[r]{p} a_1 \quad \text{B: } b_0 \xrightarrow[r']{p} b_1 \quad \Rightarrow \begin{matrix} r=\varphi(p) \\ r'=\varphi'(p) \end{matrix} \Rightarrow \quad r=r'$$

Достатність.

$$\text{A: } a_0 \xrightarrow[r]{p} a_1 \quad \text{B: } b_0 \xrightarrow[r']{p} b_1 \quad \Rightarrow r=r' \Rightarrow \begin{matrix} r=\varphi(p) \\ r'=\varphi'(p) \end{matrix} \Rightarrow \varphi(p)= \varphi'(p)$$

Теорема 3. *i*-еквівалентні стани
під дією одного й того ж слова довжини $j<i$
переходять у *(i-j)*-еквівалентні стани.

$$a \xrightarrow[r]{p} a_1 \xrightarrow[s_1]{q} a_2 \qquad |p|=j$$
$$b \xrightarrow[r]{p} b_1 \xrightarrow[s_2]{q} b_2 \qquad |q|\le i-j$$
$$|p*q|\le i$$
$$a \xrightarrow[r*s_1]{p*q} a_2 \quad b \xrightarrow[r*s_2]{p*q} b_2 \Rightarrow r*s_1=r*s_2 \Rightarrow s_1=s_2$$

Наслідок з теореми 3. Еквівалентні стани
під дією одного й того ж слова
переходять у еквівалентні стани.

$$a \xrightarrow[r]{p} a_1 \xrightarrow[s_1]{q} a_2$$
$$b \xrightarrow[r]{p} b_1 \xrightarrow[s_2]{q} b_2$$
$$a \xrightarrow[r*s_1]{p*q} a_2 \quad b \xrightarrow[r*s_2]{p*q} b_2 \Rightarrow r*s_1=r*s_2 \Rightarrow s_1=s_2$$

Канонічна мінімізація автомату

$A=<A,a_0,\delta,\lambda,X,Y>$

$B=<B,b_0,\delta',\lambda',X,Y>$

b=[a]={a':a' еквівалентно a в автоматі A}

$b_0=[a_0]$

$\delta'([a],x)=[\delta(a,x)]$

$\lambda'([a],x)=\lambda(a,x)$

де [s]={a':a' еквівалентно s в автоматі A}
– множина станів, еквівалентних s
– клас еквівалентності для s по відношенню
∞ еквівалентності станів

Доведемо коректність означень функцій δ та λ, тобто незалежність від вибору представника для класу еквівалентності [a]

$[a]=[a_1]$, тобто a еквівалентно a_1 в автоматі A

$$\lambda'([a],x) = \lambda(a,x) \;||\; \lambda'([a_1],x)=\lambda(a_1,x)$$

оскільки a та a_1 1-еквівалентні

$$\delta'([a],x) = [\delta(a,x)] \;||\; \delta'([a_1],x)=[\delta(a_1,x)]$$

оскільки $\delta(a,x)$ еквівалентно $\delta(a_1,x)$ за наслідком теореми 3

<u>Лема 4</u>. Стан *a* автомату A та стан [*a*] канонічної мінімізації A еквівалентні між собою.

$p=x_1x_2\ldots..x_k$

$$a \xrightarrow[\lambda(a,x_1)]{x_1} \delta(a,x_1) = a_1 \xrightarrow[\lambda(a_1,x_2)]{x_2}$$

$$[a] \xrightarrow[\lambda'([a],x_1)]{x_1} \delta'([a],x_1) = [a_1] \xrightarrow[\lambda'([a_1],x_2)]{x_2}$$

$\lambda'([a],x_1)=\lambda(a,x_1)$ $\delta'([a],x_1)=[\delta(a,x_1)]=[a_1]$

$\lambda'([a_1],x_2)=\lambda(a_1,x_2)$

Лема 5. В канонічній мінімізації автомату А довільні два різних стани не еквівалентні між собою.

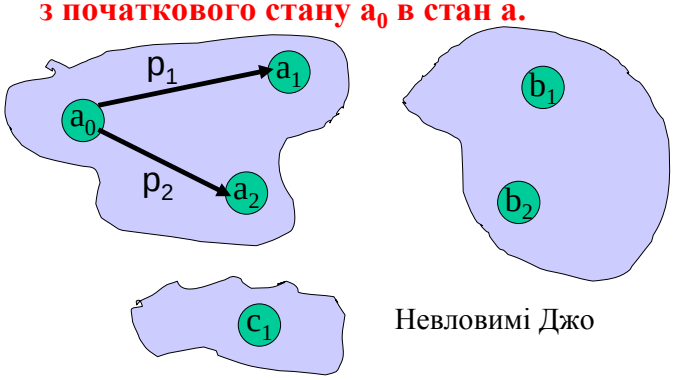
Нехай [a] еквівалентно [a_1]

[a] еквівалентно a

[a_1] еквівалентно a_1 $\Rightarrow$ a еквівалентно a_1 $\Rightarrow$

$$\Rightarrow [a] = [a_1]$$

Автомат A=<A,a_0,δ,λ,X,Y> називається зв'язним, якщо для довільного стану a існує слово p, що переводить автомат з початкового стану a_0 в стан a.



Невловимі Джо

Теорема 6. Канонічна мінімізація зв'язного автомату А містить найменшу можливу кількість станів серед усіх автоматів, що генерують те ж саме алфавітне відображення, що й автомат А.

Нехай А - зв'язний автомат
В - його канонічна мінімізація

Якщо А - зв'язний, то й В буде також зв'язний

$b_0 \overset{?}{\rightarrow} b$ $[a_0] \overset{?}{\rightarrow} [a]$ $a_0 \overset{p}{\rightarrow} a$ $[a_0] \overset{p}{\rightarrow} [a]$

$$b_0 \overset{p}{\rightarrow} b$$

Нехай A - зв'язний автомат
B - його канонічна мінімізація
D - деякий автомат. що генерує таке ж відображення

Побудуємо ін'єкцію ψ: B→D

$b_0 \xrightarrow{p} b \qquad d_0 \xrightarrow{p} \psi(b)$

Тоді b_0 еквівалентно d_0 за теоремою 2, оскільки B і D генерують одне і те ж алфавітне відображення, а тоді і b еквівалентно $\psi(b)$ за теоремою 3

Нехай ψ не є ін'єкцією, тобто $\psi(b)=\psi(b_1)=d$, тоді b еквівалентно $\psi(b)=d$, $d=\psi(b_1)$ еквівалентно b_1, а значить b еквівалентно b_1, а значить $b=b_1$ за лемою 5, то ж ψ є ін'єкцією

Оскільки ψ: B→D є ін'єкцією, то це і означає, що $|B|\le|D|$

Лема 7. Якщо в деякому автоматі відношення ***i***-еквівалентності співпадає з відношенням ***(i-1)***-еквівалентності, то це є відношення ∞-еквівалентності.

a *i*-еквівалентно b, $p=x_1x_2\ldots..x_k$

$a \xrightarrow[y_1]{x_1} a_1$, $b \xrightarrow[y_1]{x_1} b_1$ — ***i-1***-еквівалентні за теоремою 3; $a_1 \xrightarrow[y_2]{x_2} a_2$, $b_1 \xrightarrow[y_2]{x_2} b_2$ — ***i***-еквівалентні

a ∞-еквівалентно b

Лема 8. Обернення теореми 3.

Якщо *i*-еквівалентні стани a та b під дією одного й того ж довільного слова p довжини менше $j \le i$ переходять у *i*-еквівалентні між собою стани, то a та b є (i+j)-еквівалентними станами.

$a \xrightarrow[m_1]{l} a_2$ якщо $|l| \le i+j$, то $m_1=m_2$ $\quad l=p*q$, $|p| \le j$, $|q| \le i$, $|p*q| \le i+j$

$b \xrightarrow[m_2]{l} b_2$

$a \xrightarrow[r_1]{p} a_1 \xrightarrow[s_1]{q} a_2$ $\quad m_1=r_1*s_1$

$b \xrightarrow[r_2]{p} b_1 \xrightarrow[s_2]{q} b_2$ $\quad m_2=r_2*s_2$

$r_1=r_2$, оскільки a і-еквівалентно b, $|p| \le j \le i$

$s_1=s_2$, оскільки a_1 і-еквівалентно b_1, $|q| \le i$

звідси $m_1= r_1*s_1=r_2*s_2= m_2$

Алгоритм Ауфенкампа-Хона

побудови розбиття множини станів на ∞-еквіалентні класи

Крок 1. ***1***-еквівалентними вважаємо стани, для яких співпадають стовпчики таблиці виходів. ***I***:=1

Крок 2. Будуємо модифіковану таблицю переходів: у таблиці переходів стани замінюємо на класи, до яких належать ці стани

Крок 3. ***(I+1)***-еквівалентними вважаємо стани, які є ***I***-еквівалентними і для яких співпадають стовпчики модифікованої таблиці переходів.

Крок 4. Якщо розбиття на ***I***-еквівалентні класи співпадає з розбиттям на ***(I+1)***-еквівалентні, то це є розбиття на ∞-еквівалентні класи і роботу закінчено, інакше знову крок 2.

δ	a_0	a_1	a_2	a_3	a_4	a_5
0	a_2	a_1	a_0	a_3	a_4	a_5
1	a_1	a_3	a_4	a_5	a_5	a_3

λ	a_0	a_1	a_2	a_3	a_4	a_5
0	0	0	0	0	0	0
1	1	0	1	1	0	1

Класи ***1***-еквівалентності

$b_0=\{a_0,a_2,a_3,a_5\}$; $b_1=\{a_1,a_4\}$

Класи ***1***-еквівалентності $b_0=\{a_0,a_2,a_3,a_5\}$; $b_1=\{a_1,a_4\}$

δ	a_0	a_1	a_2	a_3	a_4	a_5
0	a_2	a_1	a_0	a_3	a_4	a_5
1	a_1	a_3	a_4	a_5	a_5	a_3

M_1	a_0	a_1	a_2	a_3	a_4	a_5
0	b_0	b_1	b_0	b_0	b_1	b_0
1	b_1	b_0	b_1	b_0	b_0	b_0

Класи ***2***-еквівалентності $c_0=\{a_0,a_2\}$; $c_1=\{a_1,a_4\}$; $c_3=\{a_3,a_5\}$

Класи ***2***-еквівалентності $c_0=\{a_0,a_2\}$; $c_1=\{a_1,a_4\}$; $c_3=\{a_3,a_5\}$

δ	a_0	a_1	a_2	a_3	a_4	a_5
0	a_2	a_1	a_0	a_3	a_4	a_5
1	a_1	a_3	a_4	a_5	a_5	a_3

M_2	a_0	a_1	a_2	a_3	a_4	a_5
0	c_0	c_1	c_0	c_3	c_1	c_3
1	c_1	c_3	c_1	c_3	c_3	c_3

Класи ***3***-еквівалентності $d_0=\{a_0,a_2\}$; $d_1=\{a_1,a_4\}$; $d_3=\{a_3,a_5\}$

δ	a_0	a_1	a_2	a_3	a_4	a_5
0	a_2	a_1	a_0	a_3	a_4	a_5
1	a_1	a_3	a_4	a_5	a_5	a_3

λ	a_0	a_1	a_2	a_3	a_4	a_5
0	0	0	0	0	0	0
1	1	0	1	1	0	1

$d_0=\{a_0,a_2\}$
$d_1=\{a_1,a_4\}$
$d_3=\{a_3,a_5\}$

δ'	d_0	d_1	d_3
0	d_0	d_1	d_3
1	d_1	d_3	d_3

λ'	d_0	d_1	d_3
0	0	0	0
1	1	0	1

Підмножину слів в алфавіті X
будемо називати
подією в алфавіті X

Представлення подій в автоматах Мілі

$$\mathbf{A=<A,a_0,\delta,\lambda,X,Y>}$$

φ: $X^* \to Y^*$ згенероване відображення

Подія R_y, представлена в A
символом вихідного алфавіту $y \in Y$

$$R_y=\{l \in X^* \mid \text{ост.літ.}(\varphi(l))=y\}$$

Подією, представленою множиною M вихідних символів
будемо називати об'єднання подій,
представлених символами множини M.

Теорема 1. Автомат $\mathbf{A=<A,a_0,\delta,\lambda,X,Y>}$ однозначно породжує розбиття непорожніх слів алфавіту X на події, представлені в A символами вихідного алфавіту. І навпаки, розбиття множини непорожніх слів в алфавіті X на події, асоційовані з символами вихідного алфавіту однозначно задає відповідне алфавітне автоматне відображення.

Доведемо, що $\{R_y\}$ - розбиття

1. $R_y \cap R_z \neq \varnothing \Rightarrow l \in R_y, l \in R_z \Rightarrow$
   $\Rightarrow$ ост.літ.$(\varphi(l))=y$, ост.літ.$(\varphi(l))=z$
2. $l \in X^*$, $l \neq e \Rightarrow \varphi(l) \neq e \Rightarrow$ ост.літ.$(\varphi(l))=y' \Rightarrow$
   $\Rightarrow l \in R_{y'} \Rightarrow l \in \cup R_y$

Нехай $\{R_y\}$ розбиття $X^* \backslash \{e\}$ побудуємо φ: $X^* \to Y^*$

$$\varphi(e)=e, \quad l=x_1x_2\ldots x_k$$

$x_1 \in R_{y_1} \Rightarrow$ ост.літ.$\varphi(x_1)=y_1 \Rightarrow$ перша літ. $\varphi(l)=y_1$
$x_1x_2 \in R_{y_2} \Rightarrow$ ост.літ.$\varphi(x_1x_2)=y_2 \Rightarrow$ друга літ.$\varphi(l)=y_2$
……
$x_1x_2\ldots x_k \in R_{y_k} \Rightarrow$ ост.літ.$\varphi(x_1x_2\ldots x_k)=y_k \Rightarrow$
$\Rightarrow$ k-та літ.$\varphi(l)=y_k$

$$\varphi(l)=y_1\, y_2 \ldots\ldots\, y_k$$

Представлення подій в автоматах Мура

$$\mathbf{A=<A,a_0,\delta,\mu,X,Y>}$$

Подія, представлена в А множиною станів $K \subset A$,
якщо вона складається з усіх слів,
які переводять автомат з початкового стану
у стани множини К.

$$R_K=\{l \in X^* \mid a_0 \xrightarrow{l} a \in K\}$$

Теорема 2. Будь-яка подія, представлена в автоматі Мілі множиною вихідних символів М може бути представлена у деякому автоматі Мура множиною станів К.
І навпаки, будь-яка подія, представлена в автоматі Мура множиною станів К може бути представлена у деякому автоматі Мілі множиною вихідних символів М.

Мура $\Rightarrow$ Мілі

$\mathbf{A=<A,a_0,\delta,\mu,X,Y>}$ Подія представлена множиною станів К

$\mathbf{B=<A,a_0,\delta,\lambda,X,\{0,1\}>}$ Подія представлена множиною вихідних символів М={1}

$$\lambda(a,x)=\begin{cases}1 \text{ якщо } \delta(a,x)\in K\\ 0 \text{ якщо } \delta(a,x)\notin K\end{cases}$$

Мілі $\Rightarrow$ Мура

$\mathbf{A=<A,a_0,\delta,\lambda,X,Y>}$ Подія R_M представлена множиною вихідних символів М

$$\mathbf{B=<A\times X\cup\{a_0\},a_0,\delta',\mu,X,Y>}$$

Візьмемо автомат Мура В, який генерує те ж саме відображення, що й автомат Мілі А.
Розглянемо подію R_K представлену множиною станів

$$K=\{(a,x)\in A\times X \mid \lambda(a,x)\in M\}$$

Доведемо, що $R_M=R_K$

$R_K \subset R_M$

$l \in R_K \Leftrightarrow a_0 \xrightarrow{l} (a,x) \in K$ (1)

доведемо, що ост.літ. $\varphi_A(l) \in M$

$(a,x) \in K$, тоді за побудовою K: $\lambda(a,x) \in M$

Оскільки $\varphi_A(l)=\varphi_B(l)$ за побудовою автомата B

ост.літ.$(\varphi_A(l))$=ост.літ.$(\varphi_B(l))$ = $\mu((a,x))$ =
$=\lambda(a,x) \in M \Rightarrow l \in R_M$

$R_M \subset R_K$

$l \in R_M \Rightarrow$ ост.літ.$\varphi_A(l)=y \in M$ за означенням R_M

$\varphi_A(l)=\varphi_B(l)$, за побудовою автомату B

Нехай під дією слова l автомат Мура B переходить зі стану a_0 в стан (a,x)
Тоді ост.літ.$(\varphi_B(l))$= .$(\varphi_A(l))$= $y=\mu((a,x))$=
$=\lambda(a,x) \in M \Rightarrow (a,x) \in K \Rightarrow l \in R_K$

Алгебра подій

Алфавіт $X=\{x_1,x_2,\ldots x_n\}$

1. Елементарні події: e, $x_1,x_2,\ldots x_n$
2. $S \vee R=\{l \in X^* | l \in S \vee l \in R\}$ – діз'юнкція
3. $S \bullet R=\{l \in X^* | l=p*q, p \in S, q \in R\}$ - добуток
4. $\{S\}=e \vee S \vee S \bullet S \vee S \bullet S \bullet S \vee \ldots..$ - ітерація

Регулярною подією в алфавіті X будемо називати подію, яку можна одержати з елементарних подій за допомогою скінченої кількості операцій диз'юнкції, добутку та ітерації.

Теорема 3. Регулярні події і тільки вони можуть бути представлені в скінчених автоматах.

Теорема 4. Об'єднання, перетин та доповнення регулярних подій є регулярними подіями.

1. Об'єднання: $R \cup S \leftrightarrow R \vee S$

Нехай регулярній події R відповідає регулярний вираз Φ_R, а регулярній події Q відповідає регулярний вираз Φ_Q

Тоді події $R \cup Q$ відповідатиме регулярний вираз $\Phi_R \vee \Phi_Q$

2. Перетин

Нехай

R представлено K_1 в автоматі $<A_1,a_1,\delta_1,\mu_1,X,Y>$

S представлено K_2 в автоматі $<A_2,a_2,\delta_2,\mu_2,X,Y>$

Побудуємо декартовий добуток автоматів A_1 та A_2

$<A_1 \times A_2,(a_1,a_2),\delta,\mu,X,Y>$

$\delta((a',a''),x)=(\delta_1(a',x),\delta_2(a'',x))$

$\mu((a',a''))=\mu(a')$

Тоді $R \cap S$ представлено в декартовому добутку автоматів множиною станів $K_1 \times K_2$

$$(a_1,a_2) \xrightarrow{l} (a',a'') \in K_1 \times K_2 \Rightarrow$$

$$\Rightarrow \begin{matrix} a_1 \xrightarrow{l} a' \in K_1 \Rightarrow l \in R \\ a_2 \xrightarrow{l} a'' \in K_2 \Rightarrow l \in S \end{matrix} \Rightarrow l \in R \cap S$$

2. Доповнення

R представлено в автоматі Мура A множиною станів K
Тоді $X^*\backslash R$ буде представлено в цьому ж автоматі множиною станів $A\backslash K$

Застосування теорії автоматів для побудови ефективних алгоритмів.

Розпізнавання входження ланцюжків

Постановка задачі

$y=b_1b_2b_3\ldots b_L$ – слово (ланцюжок)

$x=a_1a_2a_3\ldots a_N$ – текст (послідовність символів)

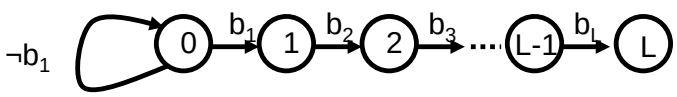
чи входить слово y в текст x ?

$x=z_1yz_2$

Звичайний алгоритм пошуку

```
for i:=L to N do                    N-L+1 повторення
begin
      result:=true; j:=1;
      while result and j<=L do  максимум L повторень
      begin
            if bj≠ai-L+j then result:=false;
            i:=i+1; j:=j+1
      end;
      if result return("YES");
end;                    Всього максимум L·(N-L+1) операція
result return("NO")     O(L·N) операцій
```

Базовий автомат



Функція відмов

$$f(j) = \max_{s<j,\, b_1 b_2 \ldots b_s = b_{j-s+1} b_{j-s+2} \ldots b_j} s$$

f(j)<j – найдовший префікс y, який є суфіксом $b_1 b_2 \ldots b_j$

Приклад функції відмов

для y=aabbaab

j	f(j)	$b_1 \ldots b_j$	$b_1 \ldots b_s$
1	0	a	
2	1	aa	a
3	0	aab	
4	0	aabb	
5	1	aabba	a
6	2	aabbaa	aa
7	3	aabbaab	aab

Узагальнена функція відмов $f^m(j)$

1)$f^1(j)=f(j)$

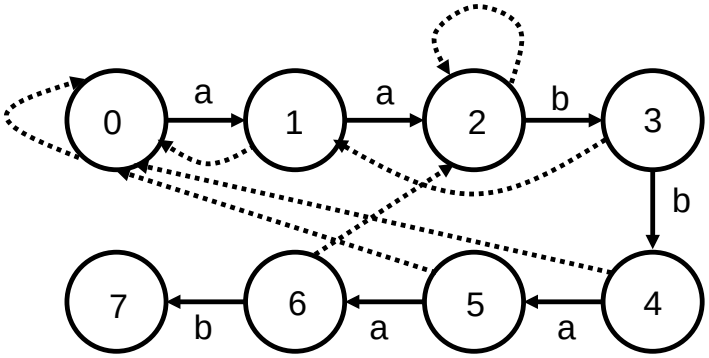
2)$f^m(j)=f(f^{m-1}(j)$ для m>1

$$f^m(j) = \underbrace{f(f(f(....f}_{m\ разів}(j))....)$$

Функція переходів автомату

$$\delta(j,b) = \begin{cases} j+1 & якщо \quad b = b_{j+1} \\ k+1 & якщо \quad k = \max\limits_{b=b_{k+1}} f^m(j) \\ 0 & якщо \quad \forall m\, b \neq b_{f^m(j)+1}\ ma\ b \neq b_1 \end{cases}$$

Автомат перевірки y=aabbaab



Алгоритм обчислення функції відмов

```
f(1):=0;
for j:=2 to L do   L-1 повторення
begin
      i:=f(j-1);                                  зменшення f
      while b_j≠b_{i+1} and i>0 do i:=f(i);        f≥0
      if b_j≠b_{i+1} and i=0
            then f(j):=0
            else f(j):=i+1     максимум L збільшень
                                  f на 1
end
```

Всього O(L)

Практичні заняття

ТЕМА 1: СКІНЧЕНІ АВТОМАТИ, АВТОМАТНІ ВІДОБРАЖЕННЯ.

Мета: Засвоєння властивостей автоматів та згенерованих ними відображень. Різниця та подібність між автоматами Мілі та Мура. Побудова автомату Мура по автомату Мілі.

Теоретичні питання: Алфавіт, слова в алфавіті. Чорна скринька. Автомат Мілі, автомат Мура, згенеровані автоматом алфавітні відображення. Відповідність між автоматами Мілі та Мура.

Аудиторне завдання:

1. Побудувати автомат, що генерує алфавітне відображення $f_1 : \{a,b\}^* \to \{0,1\}^*$ таке що, $f_1(l)$ закінчується на 1, якщо останні 4 літери слова l є літери a.

2. Побудувати автомат, що генерує алфавітне відображення $f_2 : \{0,1,2,3,5\}^* \to \{+,-\}^*$ таке що, $f_2(l)$ закінчується +, якщо сума цифр після останнього + більше або дорівнює 5.

3. Побудувати автомат Мура по автомату Мілі:

$\mathbf{A}_1$

δ	a_1	a_2
0	a_1	a_1
1	a_2	a_2

λ	a_1	a_2
0	0	1
1	0	1

$\mathbf{A}_2$

δ	a_1	a_2	a_3	a_4
0	a_3	a_1	a_1	a_3
1	a_2	a_4	a_2	a_2

λ	a_1	a_2	a_3	a_4
0	0	1	0	1
1	0	1	0	1

Домашнє завдання:

1. Побудувати автомат, що генерує алфавітне відображення $f_3 : \{a,b\}^* \to \{-1,0,1\}^*$ таке що, остання літера $f_3(l)$ дорівнює $sign(n_a(l) - n_b(l))$, де $n_x(l)$ кількість літер x в слові l. Чи можна це відображення реалізувати скінченним автоматом?

2. Побудувати автомат Мура по автомату Мілі

A3

δ	a1	a2	a3	a4
0	a2	a2	a4	a2
1	a4	a3	a3	a3

λ	a1	a2	a3	a4
0	1	1	1	0
1	1	0	1	1

A4

δ	a1	a2	a3	a4	a5	a6	a7
0	a1	a2	a4	a5	a4	a5	a2
1	a7	a3	a5	a4	a5	a4	a6

λ	a1	a2	a3	a4	a5	a6	a7
0	0	0	1	0	0	1	0
1	1	0	1	1	1	1	0

Додаткове завдання:

1. Побудувати автомат Мілі, що обчислює мінімум (максимум) двох додатних цілих двійкових чисел. Вважати: що довжини вирівнюються дописуванням "0", на вхід числа подаються починаючи з старших розрядів, на виході повинні отримати двійкове представлення мінімального (максимального) числа.

2. Побудувати автомат Мілі, який працює як накопичуючий суматор послідовної дії. На вхід подаються два цілих додатних двійкових числа, починаючи з молодших розрядів, на виході отримуємо суму.

ТЕМА 2: МІНІМІЗАЦІЯ СКІНЧЕНИХ АВТОМАТІВ.

Мета: Засвоєння понять еквівалентності станів, канонічної мінімізації автоматів, алгоритму Ауфенкампа – Хона.

Теоретичні питання: Еквівалентні стани, канонічна мінімізація, зв'язний автомат.

Аудиторне завдання:

1. Мінімізувати автомати:

A_3

δ	a_1	a_2	a_3	a_4
0	a_2	a_2	a_4	a_2
1	a_4	a_3	a_3	a_3

λ	a_1	a_2	a_3	a_4
0	1	1	1	0
1	1	0	1	1

A_4

δ	a_1	a_2	a_3	a_4	a_5	a_6	a_7
0	a_1	a_2	a_4	a_5	a_4	a_5	a_2
1	a_7	a_3	a_5	a_4	a_5	a_4	a_6

λ	a_1	a_2	a_3	a_4	a_5	a_6	a_7
0	0	0	1	0	0	1	0
1	1	0	1	1	1	1	0

Домашнє завдання:

1. Мінімізувати автомати:

A_5

δ	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7
0	a_1	a_0	a_3	a_2	a_5	a_4	a_7	a_6
1	a_2	a_3	a_4	a_5	a_6	a_7	a_0	a_1

λ	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7
0	1	1	0	0	0	0	1	1
1	1	1	0	0	0	0	1	1

A_6

δ	a_0	a_1	a_2	a_3	a_4
0	a_1	a_2	A_0	a_4	a_0
1	a_4	a_0	A_1	a_0	a_{31}

λ	a_0	a_1	A_2	a_3	a_4
0	1	1	0	1	0
1	0	0	0	0	0

Додаткове завдання:

1. Вважаючи, що вхідний алфавіт X={0, 1}, навести автомат, який допускає:
   a) Всі вхідні ланцюжки;
   b) Не допускає жодного вхідного ланцюжка;
   c) Всі вхідні ланцюжки, крім порожнього;
   d) Ланцюжки, що починаються "0" та закінчуються "1";
   e) Ланцюжки з непарною кількістю "1";
   f) Ланцюжки, де кількість "1" - парна, а кількість "0" - непарна;
   g) Ланцюжки, де між входженнями "1" парна кількість "0";
   h) Ланцюжки, довжина яких ділиться на 3.
2. Для наведених вище прикладів (1.) записати відповідні події у вигляді регулярних виразів.
3. Побудувати скінчений автомат, що розпізнає парні числа, які записані у двійковій формі та зчитуються зліва направо.
4. Побудувати скінчений автомат, здатний розпізнавати десяткові числа, що записані у формі з плаваючою крапкою (±dd*.d*E±dd, де d∈{0, 1, 2, …, 9}) .
5. Показати неможливість побудови скінченого автомата, на вхід якого подаються ланцюжки над алфавітом {a, b} й вважаються допустимими з них тільки, що мають однакову кількість символів "a" та "b".

Література до модуля 6.

ОСНОВНА

1. Глушков В.М. Введение в кибернетику. – К.:Изд-во Академии наук Украинской ССР, 1964. – 324 с.
2. Глушков В.М. Синтез цифровых автоматов. – М.: Наука, 1962. – 476 с.
3. Горшков П.В., Родимин С.П., Шевченко В.П. Методические рекомендации по изучению раздела "Конечные автоматы" курса "Дискретная математика" для студентов первого курса факультета кибернетики. – К.: КГУ, 1986. – 46 с.
4. Брауэр В. Введение в теорию конечных автоматов. – М.: Радио и связь, 1987. – 392 с.
5. Чирков М.К. Основы общей теории конечных автоматов. - Л.: Изд-во Ленингр. ун-та, 1975. – 280 с.
6. Кузнецов О.П., Адельсон-Вельский Г.М. Дискретная математика для инженера. – М.: "Энергоатомиздат", 1988. – 480 с.
7. Белоусов А.И., Ткачев С.Б. Дискретная математика. – М.: Издательство МГТУ им. Н.Э. Баумана, 2001. – 744 с.

ДОДАТКОВА

1. Глушков В.М. Абстрактная теория автоматов. – Успехи математических наук, 1961, Т.14,№5, с.1-62.
2. Ахо А., Ульман Дж. Теория синтаксического анализа, перевода и компиляции. - М.: Мир, 1978. Т.1. – 612 с.
3. Льюис Ф., Розенкранц Д., Стирнз Р. Теоретические основы проектирования компиляторов. – М.: Мир, 1979. - 654 с.
4. Ахо А., Хопкрофт Дж., Ульман Дж. Построение и анализ вычислительных алгоритмов. – М.:Мир, 1979. - 536 с.
5. Горбатов В.А. Основы дискретной математики: Учебное пособие для студентов вузов. – М.:"Высшая школа", 1986. - 311 с.
6. Гаврилов Г.П., Сапоженко А.А. Сборник задач по дискретной математике. – М.: "Наука", 1977. – 368 с.
7. Глушков В.М., Цейтлин Г.Е., Ющенко Е.Л. Алгебра. Языки. Программирование. – К.: "Наукова думка", 1989. – 376 с.

МОДУЛЬ 7. ТЕОРІЯ КОДУВАННЯ

Для самостійного вивчення

Історична довідка

Теорія кодування має тривалу історію. Системи числення для представлення чисел були, фактично, першими кодами. Теорія кодування – одна з тих важливих областей прикладної математики, яка значним чином вплинула на розвиток комп'ютерної справи.

Кодування (рос. кодирование, англ. encoding, нім. Kodieren n) – процес перетворення інформації з форми зручної для безпосереднього використання у форму зручну для передачі, збереження та автоматичної обробки. Необхідність у кодуванні виникає насамперед через потребу налаштувати форму повідомлення під певний канал зв'язку або під певний пристрій зберігання чи перетворення інформації.

Переважна більшість населення вважає, коди або шифри використовуються виключно для втаємничення інформації. Однак у наші часи коди і кодування, насамперед, - це засоби економічної, зручної і надійної передачі або збереження повідомлень. Сьогоднішні застосування кодування пов'язані, насамперед, з розвитком засобів зв'язку, передачею все більшого обсягу інформації по каналах, без яких (телеграфу, телефону, радіо, телебачення) не можна уявити наше існування.

Необхідність кодування даних вперше виникла з появою телеграфу. Перший код для передачі повідомлень названий ім'ям винахідника телеграфного апарату і відомий всім як азбука Морзе. В цьому коді кожному символу ставилась у відповідність своя послідовність з коротких (крапка) і тривалих (тире) сигналів, які розділювалися паузами. Інший широко поширений у телеграфії код Бодо використовує для кодування два типи сигналів – імпульс і паузу, кожному символу ставиться у відповідність послідовність з п'яти таких сигналів.

Телеграфні канали у XIX столітті були дорогими і ненадійними, що змусило розв'язувати задачу мінімізації вартості і підвищення надійності передачі телеграм. Проблема ще більше загострилась з прокладанням трансатлантичних кабелів. Тоді ж для перевірки правильності передачі став застосовуватись метод контролю парності, який застосовувався також і для контролю правильності вводу перфокарт. Нажаль, цей метод був не лише незручним, а й пропускав подвійні помилки, які зберігали парність контрольної суми. З розвитком каналів зв'язку виникла потреба у більш ефективних методах контролю.

Першим теоретичне розв'язання проблеми передачі інформації по неякісним каналам запропонував Клод Шеннон, засновник статистичної теорії інформації. Перехід від теоретичних досліджень Шеннона до практичних застосувань став можливим завдяки Ричарду Геммінгу, колеги Шеннона по Bell

Labs. Він відкрив клас кодів, які дістали назву «коди Геммінга». Існує легенда, що на створення цих кодів його підштовхнула незручність роботи з перфокартами, які постійно не хотіли правильно вводитись у комп'ютер, а оскільки працювати йому доводилось, як правило, по вихідних у відсутність операторів, то долати цю проблему доводилось самому. Геммінг запропонував коди, які корегували помилки в каналах зв'язку, в тому числі і всередині комп'ютера.

Сучасні модифікації цих кодів використовуються майже в усіх системах збереження і передачі інформації, від програвачів компакт дисків до систем космічного зв'язку.

Інший стародавній напрямок у теорії кодування – це криптографія, або утаємнене кодування. Криптографія бере свій початок 4 тисячі років тому в Єгипті, коли єгиптяни використовували ієрогліфічний код для написів на могилах.

Криптогра́фія (від грецького kryptós — прихований і gráphein — писати) — наука про математичні методи забезпечення конфіденційності (неможливості прочитання інформації стороннім) і автентичності (цілісності і справжності авторства) інформації. Розвинулась з практичної потреби передавати важливі відомості найнадійнішим чином.

Для сучасної криптографії характерне використання відкритих алгоритмів шифрування, що припускають використання обчислювальних засобів. Відомо більш десятка перевірених алгоритмів шифрування, які, при використанні ключа достатньої довжини і коректної реалізації алгоритму, роблять шифрований текст недоступним для криптоаналізу.

До нашого часу, криптографія займалася виключно забезпеченням конфіденційності повідомлень (тобто шифруванням) — перетворенням повідомлень із зрозумілої форми в незрозумілу і зворотнє відновлення на стороні одержувача, роблячи його неможливим для прочитання для того, хто перехопив або підслухав без секретного знання (а саме ключа, необхідного для дешифровки повідомлення). В останні десятиліття сфера застосування криптографії розширилася і включає не лише таємну передачу повідомлень, але і методи перевірки цілісності повідомлень, ідентифікування відправника/одержувача (аутентифікація), цифрові підписи, інтерактивні підтвердження, та технології безпечного спілкування, тощо.

Найперші форми тайнопису вимагали не більше ніж аналог олівця та паперу, оскільки в ті часи більшість людей не могли читати. Поширення писемності, або писемності серед ворогів, викликало потребу саме в криптографії. Ще в V столітті до нашої ери грецький історик Геродот наводив приклади листів, зміст яких був зрозумілим лише адресату. Спартанці мали спеціальний пристрій, за допомогою якого писали важливі повідомлення зі збереженням таємниці.

Основними типами класичних шифрів є перестановочні шифри, які змінюють порядок літер в повідомленні, та підстановочні шифри, які систематично замінюють літери або групи літер іншими літерами або групами літер. Прості варіанти обох типів пропонували слабкий захист від досвідчених

супротивників. Одним із ранніх підстановочних шифрів був шифр Цезаря, в якому кожна літера в повідомленні замінювалась літерою через декілька позицій із абетки. Цей шифр отримав ім'я Юлія Цезаря, який його використовував, зі зсувом в 3 позиції, для спілкування з генералами під час військових кампаній.

В середні віки та в епоху Відродження над створенням таємних шифрів працювали такі видатні люди як філософ Френсіс Бекон, математики Франсуа Вієт, Джероламо Кардано. З плином часу стали з’являтись справді складні шифри. Різноманітні методи шифрування розроблювались у королівських дворах і в університетах. Одночасно з мистецтвом шифрування розвивалося і мистецво дешифрування, яке називають криптоаналізом. Таємні шифри є сьогодні необхідним атрибутом багатьох детективних романів і серіалів ☺.

Поява цифрових комп'ютерів та електроніки після другої світової війни зробило можливим появу складніших шифрів. Більше того, комп'ютери дозволяли шифрувати будь-які дані, які можна представити в комп'ютері у двійковому виді, на відміну від класичних шифрів, які розроблялись для шифрування письмових текстів. Це зробило непридатними для застосування лінгвістичні підходи в криптоаналізі. Багато комп'ютерних шифрів можна характеризувати за їхньою роботою з послідовностями бінарних бітів (інколи в блоках або групах), на відміну від класичних та механічних схем, які, зазвичай, працюють безпосередньо з літерами. Однак, комп'ютери також знайшли застосування у криптоаналізі, що, в певній мірі, компенсувало підвищення складності шифрів. Тим не менше, гарні сучасні шифри залишались попереду криптоаналізу; як правило, використання якісних шифрів дуже ефективне (тобто, швидке і вимагає небагато ресурсів), в той час як злам цих шифрів потребує набагато більших зусиль ніж раніше, що робить криптоаналіз настільки неефективним та непрактичним, що злам стає практично неможливим.

Шеннон, Клод Елвуд (*Shannon, Claude Elwood*; 1916–2001) — американський математик й електротехнік, один з творців математичної теорії інформації, значною мірою визначив своїми результатами розвиток загальної теорії дискретних автоматів.

З шкільних років проявляв рівний інтерес як до математики, так і до радіоелектроніки. У 1932 році вступив до університету штату Мічиган, у 1936 – до Массачусетського технологічного інституту, який закінчив у 1940 році. Одержав два наукових ступені – магістра з радіоелектроніки та доктора математики.

У 1941 р. поступив на роботу до відомої Bell Laboratory – лабораторії компанії Белл. У 1948 році надрукував фундаментальну роботу «A Mathematical Theory of Communication», в якій сформулював основи теорії інформації, зокрема, визначення кількості інформації через ентропію, а також запропонував одиницю виміру інформації, яку пізніше назвали бітом. Значну цінність становить також його інша праця — «Communication Theory of Secrecy Systems» (1949), у якій сформульовані математичні основи криптографії.

У 1957 році Шеннон став професором Массачусетського технологічного інституту, звідки через 21 рік пішов на пенсію.

Ричард Геммінг (1915–1998) почав свою освіту у Чиказькому університеті, де у 1937 одержав ступінь бакалавра, в 1939 одержав ступінь магістра в університеті Небраски, а ступінь доктора математики – в університеті Іллінойса. З 1945 році Геммінг бере участь у Манхеттенському проекті по створенню атомної бомби. У 1946 влаштувався на роботу до Bell Telephone Laboratories, де працював в тому числі з Клодом Шенноном.

У 1950 році опублікував результати про коди виявлення та виправлення помилок, які зробили його всесвітньо відомим. У 1956 брав участь в розробках одного з перших мейнфреймів IBM 650.

Основи теорії кодування

Кодування – це перехід до іншого способу подання інформації, або представлення інформації у вигляді послідовності певних символів, сигналів за визначеними правилами. Така послідовність символів називається **кодом.**

- Позиційна система числення – кодування чисел;
- Римська система числення – кодування натуральних чисел;
- Декартові координати – кодування точок простору.

2

Застосування кодування

- Представлення даних різної природи (текст, графіка, відео, …) в зручній для обробки чи передачі формі;
- Захист інформації від несанкціонованого доступу;
- Забезпечення захисту від помилок при передачі інформації по каналах зв`язку;
- Стиснення інформації.

3

Постановка задачі кодування

$A = \{a_1, a_2, \ldots, a_n\}$, $B = \{b_1, b_2, \ldots, b_m\}$ - алфавіти
$F : S \to B^*$ для $S \subseteq A^*$.

- F – функція ***кодування***;
- $\alpha \in S$ – ***повідомлення***;
- $\beta = F(\alpha)$ - ***коди***;
- F^{-1} (якщо існує) – функція ***декодування***.

Якщо $|B| = m$ - F – ***m-ічне кодування***.
$B = \{0, 1\}$ – ***двійкове кодування***.

4

Основні проблеми теорії кодування

При заданих алфавітах A, B й множині повідомлень S знайте кодування F, що має певні властивості:

- ***Оптимальність -*** як правило, мінімізацієя довжин кодів;
- ***Існування декодування*** – природна вимога, але потрібна не завжди (наприклад, трансляція програм – кодування з мов високого рівня в машинні команди);
- ***Стійкість від помилок –*** виправлення помилок передачі інформації при декодуванні (функція декодування F^{-1} має властивість, що $F^{-1}(\beta) = F^{-1}(\beta')$, де β' в певному сенсі близька до β;
- ***Потрібна складність або,*** *навпаки****, простота*** кодування та декодування (криптографія).

5

Алфавітне кодування

$A=\{a_1,a_2,a_3,\ldots\ldots a_n\}$ B – алфавіти
Схема (таблиця) кодування σ: $A \to B^*$
Символу $a_i \in A$ ставиться у відповідність слово $\beta_i \in B^*$
V = $\sigma(A)=\{\beta_1,\beta_2,\ldots\beta_n\}$ $(i \neq j \Rightarrow \beta_i \neq \beta_j)$ – множина елементарних кодів

Тоді функція алфавітного кодування за схемою σ для слова $\alpha_1\alpha_2\alpha_3\ldots\alpha_k$ $\alpha_i \in A$
визначається як слово утворене елементарними кодами для символів α_i:
$F_\sigma(a_1a_2a_3\ldots a_k)=\beta_1*\beta_2*\beta_3*\ldots.*\beta_k \in B^*$

Алфавітне кодування може бути застосовано для довільного повідомлення $\alpha = a_1a_2a_3\ldots a_k \in A^*$

6

Приклад таблиці кодів

Алфавіти : A = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}, B = {0, 1}
Схема : σ = {0→0, 1→1, 2→10, 3→11, 4→100, 5→101, 6→110, 7→111, 8→1000, 9→1001}.

Схема є однозначною, але кодування **не є взаємно однозначним:**
$F_\sigma(333) = 111111 = F_\sigma(77)$.

В той же час схема :
σ` = {0→0000, 1→0001, 2→0010, 3→0011, 4→0100, 5→0101, 6→0110, 7→0111, 8→1000, 9→1001}
є взаємно однозначною (двійково-десяткове кодування) й допускає декодування.

7

Роздільні схеми

Схему алфавітного кодування називають ***роздільною***, якщо $\beta_{i1} \dots \beta_{ik} = \beta_{j1} \dots \beta_{jt} \Rightarrow k = t \ \& \ \forall q \in 1..k \ i_q = j_q$.

Тобто довільне слово, що складається з елементарних кодів, єдиним чином розкладається на елементарні коди (попередній приклад двійково-десяткового кодування).

Алфавітне кодування з роздільною схемою завжди допускає декодування (є взаємно однозначним).

8

Префіксні схеми

Схема σ є ***префіксною***,
якщо елементарний код однієї букви
не є префіксом елементарного коду іншої букви:

$$(\forall\, i \neq j \ \ \beta_i, \beta_j \in \mathbf{V}) \Rightarrow$$
$$\Rightarrow (\forall\, \beta \in B^* \ \beta_i \neq \beta_j\beta) \ .$$

9

Теорема. Префіксна схема є роздільною

Нехай схема σ не є роздільною.
Тоді існує таке слово $\beta \in F_\sigma(A^*)$, що
$\beta = \beta_{i1}, \ldots, \beta_{ik} = \beta_{j1}, \ldots, \beta_{jt}$ & $(\exists\, q\ \beta_{iq} \neq \beta_{jq})$.

Виберемо найменше таке q, тоді скоротивши однакові префікси маємо $\beta_{iq}, \ldots, \beta_{ik} = \beta_{jq}, \ldots, \beta_{jt}$

$|\beta_{iq}|$ не може дорівнювати $|\beta_{jq}|$ то ж нехай $|\beta_{iq}| > |\beta_{jq}|$, тоді $\beta_{iq} = \beta_{jq}\delta$ де δ префікс $\beta_{jq+1}, \ldots, \beta_{jt}$,
але це протирічить тому, що схема префіксна.

10

Зауваження до теореми

Властивість бути префіксною є достатньою, але не є необхідною для того, щоб схема була роздільною.

Приклад

Схема:
$A = \{a, b\}$, $B = \{0, 1\}$, $\sigma = \{a \to 0, b \to 01\}$
є роздільною, але не є префіксною.

11

Ефективне кодування

Найпоширеніший критерій ефективності кодування – щоб коди повідомлень мали найменшу (якомога меншу) довжину.

Якщо про множину можливих повідомлень S ($S \subseteq A^*$) **не маємо додаткової інформації** – розв`язувати задачу оптимізації кодування **неможливо**.

Якщо ж доступна додаткова, наприклад, статистична інформація про частоту входження букв у повідомлення, то можна побудувати оптимальне кодування, в якому на одну букву повідомлення в середньому припадає менша кількість символів алфавіту **B**.

12

Мінімізація довжини коду повідомлень

Якщо задана схема алфавітного кодування, що є роздільною $\sigma = \{a_1 \to \beta_1, \ldots, a_n \to \beta_n\}$, то довільна схема $\sigma` = \{a_1 \to \beta`_1, \ldots, a_n \to \beta`_n\}$, де $(\beta`_1, \ldots, \beta`_n)$ є перестановкою $(\beta_1, \ldots, \beta_n)$ також буде роздільною.

Якщо довжини елементарних кодів рівні, то перестановка елементарних кодів у схемі не впливає на довжину коду повідомлення.

Для конкретних повідомлення й схеми кодування існує така перестановка елементарних кодів у схемі, при якій довжина коду повідомлення буде мінімальною.

Нехай $k_1, \ldots, k_n$ – кількості входження букв $a_1, a_2, \ldots, a_n$ до повідомлення **s**,
$l_1, \ldots, l_n$ – довжини елементарних кодів $\beta_1, \ldots, \beta_n$ відповідно.

Тоді, якщо $k_i \le k_j$ та $l_j \le l_i$, то $k_i * l_i + k_j * l_j \le k_i * l_j + k_j * l_i$.
Для мінімізації довжини коду повідомлення **s** потрібно:

- впорядкувати букви за спаданням кількості входжень;
- елементарні коди впорядкувати за зростанням довжини;
- призначити буквам коди згідно отриманих порядків.

Цей простий метод спрацьовує тільки для фіксованого повідомлення **s** та фіксованої схеми σ.

Частотні характеристики

$A = \{a_1, a_2, \ldots a_n\}$ – алфавіт
$S = \{s_1, s_2, \ldots s_k\}$ – множина можливих повідомлень
$q_1, q_2, \ldots q_n$ – кількість входжень відповідних букв у повідомлення множини S, тоді
$p_i = q_i / (|s_1| + |s_2| + \ldots + s_k)$ – частота (ймовірність) появи букви a_i в повідомленнях множини S
$p_1 + p_2 + \ldots + p_n = 1$

Частотна характеристика $P = \{p_1, p_2, \ldots p_n\}$ для алфавіту A і множини повідомлень S є дуже важливою для вирішення багатьох задач теорії кодування

Вартість кодування

Нехай A = {a_1, …, a_n} – алфавіт

P = {p_1, …, p_n} – частотна характеристика алфавіту A

Для кожної схеми алфавітного кодування

σ = {$a_1 \to \beta_1$, …, $a_n \to \beta_n$}

$$l_\sigma(P) = \sum_{i=1}^{n} p_i \cdot |\beta_i|$$

- назвемо середньою вартістю (або довжиною) кодування σ для частотної характеристики P.

Приклад

Для схеми
A = {a, b}, B = {0, 1}, σ = {a→0, b→01}, P = {0.5, 0.5}
вартість кодування становить 0.5*1 + 0.5*2 = 1.5,
а при P = {0.9, 0.1} вартість дорівнює 0.9*1 + 0.1*2 = 1.1.

Оптимальне кодування

Алфавітне кодування σ
для якого $l_\sigma(P)$ приймає мінімальне значення,
називається ***оптимальним кодуванням***
для частотної характеристики P .

Відомі декілька методів для побудови оптимальних, або близьких до оптимального кодів: алгоритми ***Фано***, ***Хаффмена***, ***Шеннона***.

Алгоритм ***Фано*** дозволяє будувати префіксну схему (що є роздільною) близьку до оптимальної й відрізняється простотою.

Алгоритм ***Хаффмена*** дозволяє будувати оптимальну префіксну схему.

Метод ***Шеннона*** дозволяє отримувати досить точні оцінки.

Алгоритм Фано

Алгоритм: Побудова схеми кодування, близької до оптимальної.
Вхід: P : array [1..n] of real – масив ймовірностей появи букв у повідомленні, впорядкований за не зростанням; $0 < p_n \le \ldots \le p_1 \le 1$, $p_1 + \ldots + p_n = 1$.
Вихід: C : array [1..n, 1..L] of 0..1 – масив елементарних кодів.

Основну роботу виконує рекурсивна процедура Fano(b, e, k).
Вхідні параметри процедури Fano(b, e, k) : b, e – індекси початку й кінця частини масиву P, k – довжина вже побудованих кодів в масиві C.
Результат процедури Fano : заповнена частина в масиві C.
Початкове звернення: Fano(1, n, 0).

```
procedure Fano(b, e, k);
  if e > b then begin
     k := k +1; {місце для чергового розряду}
     m := Med(b, e); {ділення масиву на дві частини}
```

```
    for i := b to e do  C[i, k] :=i > m; {перша частина заповнюється 0, друга 1}
    Fano(b, m, k);  {обробка першої частини}
    Fano(m+1, e, k); {обробка другої частини}
  end {if}
```

Функція **Med(b, e)** знаходить медіану вказаної частини масиву P[b..e], тобто визначає індекс *m* (b ≤ m < e), що сума елементів P[b..e] найбільш близька до суми елементів P[m+1..e].

```
function Med(b, e) : integer;
  Sb := 0; {сума елементів першої частини}
  for i := b to e-1 do Sb := Sb + P[i]; {спочатку всі, крім останнього}
  Se := P[e]; {початкова сума елементів другої частини}
  m := e; {починаємо шукати медіану з кінця}
  repeat
    d := Sb – Se;  m := m -1;
    Sb := Sb – P[m];  Se := Se + P[m];
  until d ≤ |Sb -Se|;
  return m;
```

Обґрунтування

При кожному збільшенні довжини кодів в одній частині коди доповнюються 0, а в іншій частині – 1. Тому, коди однієї частини не можуть бути префіксами іншої. Збільшення довжини кодів завершується коли довжина частини масиву дорівнює 1, тобто залишається один код. Тому, за побудовою, схема є префіксною, а це означає - роздільною.

Приклад

Коди, побудовані за алгоритмом Фано для заданого розподілу ймовірностей (n=7):

p_i	C[i]	l[i]
0.20	00	2
0.20	010	3
0.19	011	3
0.12	100	3
0.11	101	3
0.09	110	3
0.09	111	3
$l_\sigma(P)$		2.80

Оптимальні коди

Оптимальне кодування має певні властивості, які можна використати при його побудові.

Лема 1. Нехай $\sigma = \{a_1 \to \beta_1, \ldots, a_n \to \beta_n\}$ схема алфавітного кодування для розподілу ймовірностей $P = \{p_1, \ldots, p_n\}$ $(0 < p_n \le \ldots \le p_1)$. Тоді якщо $p_i > p_j$, то $l_i \le l_j$.

Лема 2. Якщо $\sigma = \{a_1 \to \beta_1, \ldots, a_n \to \beta_n\}$ схема оптимального префіксного кодування для розподілу ймовірностей $P = \{p_1, \ldots, p_n\}$ $(0 < p_n \le \ldots \le p_1)$, то серед елементарних кодів, що мають максимальну довжину, є два, які відрізняються тільки в останньому розряді.

Теорема 3. Якщо $\sigma = \{a_1 \to \beta_1, \ldots, a_{n-1} \to \beta_{n-1}\}$ схема оптимального префіксного кодування для розподілу ймовірностей $P = \{p_1, \ldots, p_{n-1}\}$ $(0 < p_{n-1} \le \ldots \le p_1)$ й $p_j = q` + q``$, причому

$$p_1 \ge \ldots \ge p_{j-1} \ge p_{j+1} \ldots \ge p_{n-1} \ge q` \ge q`` > 0 ,$$

то кодування за схемою

$$\sigma = \{a_1 \to \beta_1, \ldots, a_{j-1} \to \beta_{j-1}, a_{j+1} \to \beta_{j+1}, \ldots, a_{n-1} \to \beta_{n-1}, a_j \to \beta_j 0, a_n \to \beta_j 1\}$$

є оптимальним префіксним кодуванням для розподілу ймовірностей $P = \{p_1, \ldots, p_{j-1}, p_{j+1}, \ldots, p_{n-1}, q`, q``\}$.

Алгоритм Хаффмена

Алгоритм Huffman: Побудова оптимальної схеми кодування.
Вхід: n – кількість букв алфавіту, P : array [1..n] of real – масив ймовірностей появи букв у повідомленні, впорядкований за спаданням $(0 < p_n \le \ldots \le p_1 \le 1, p_1 + \ldots + p_n = 1)$.
Вихід: C : array [1..n, 1..L] of 0..1 – масив елементарних кодів, lc : array [1..n] of 1..L - масив довжин елементарних кодів схеми оптимального префіксного кодування.

```
procedure Huffman(P, n);
 if n =2 then begin
  C[1,1] := 0; lc[1] := 1;  {перший елемент}
  C[2,1] := 0; lc[2] := 1;  {другий елемент}
else
  q := P[n-1] + P[n];  {сума двох останніх}
  j := Up(n, q);  {пошук місця та вставка суми}
  Huffman(P, n-1);
  Down(n, j);  {добудова кодів}
 end {if}
```

Функція **Up(n, q)** знаходить у масиві P місце, в якому повинне знаходитись число ***q*** та вставляє це число, здійснюючи зсув вниз інших елементів. Параметр ***n*** – довжина частини масиву P, що розглядається.

```
function Up(n, q) : integer;
for i := n-1 downto 2 do
  if P[i-1] ≤ q then  P[i] := P[i-1]  {зсув елементу масиву}
  else begin  j := i - 1;  {визначення місця елемента для вставки}
```

```
            break {for i  - все зроблено, цикл не потрібно продовжувати} end
{if}
   P[j] := q;  {вставка елементу}
return j;
```

Процедура **Down(n, j)** будує оптимальний код для ***n*** букв, виходячи з побудованого оптимального коду для ***n-1*** букви. Для цього код букви з номером ***j*** тимчасово вилучається з масиву ***C*** шляхом зсуву вгору кодів букв з номерами більшими за ***j*** , а потім в кінець розглядаємої частини масиву ***C*** додається пара кодів, що отримують з коду букви з номером ***j*** дописуванням 0 та 1, відповідно.

Вхідні параметри процедури Down(n, j) : n – довжина розглядаємої частини масиву P, j – номер «розділяємої» букви.

Далі **C[j, *] - j –** рядок масиву **C.**

```
procedure Down(n, j);
 d := C[j, *];  {збереження коду букви j }
 t := lc[j];  {та довжини цього коду}
 for i := j to n-2 do
    begin
     C[i, *] := C[i+1, *];  {зсув коду}
      lc[i] := lc[i+1]  {та його довжини}
    end {for};
 C[n-1, *] := d;  C[n, *] := d; {код букви j}
 C[n-1, t+1] := 0;  C[n, t+1] := 1;  {подовження кодів}
 lc[n-1] := t+1;  lc[n] := t+1;  {збільшення довжин}
```

Обґрунтування

Для пари букв при довільному розподілі ймовірностей оптимальне кодування є очевидним: для першої букви призначається код 0, для другої – 1. Саме це й здійснюється в першій частині умовного операторі (**if** n =2) процедури Huffman . Рекурсивна частина алгоритму в точності слідує теоремі 3 з попереднього розділу. За допомогою функції Up у впорядкованій частині масиву P вилучаються дві останні (найменші) ймовірності та їх сума вставляється до масиву P, так щоб масив (на одиницю меншої довжини) залишився впорядкованим. Місце вставки при цьому зберігається в локальній змінній j . Так відбувається, доки не залишиться масив з двох елементів, для якого оптимальний код відомий. Після цього в оберненому порядку будуються оптимальні коди для трьох, чотирьох й т.д. елементів. Зауважимо, що при цьому масив ймовірностей P вже не потрібен, потрібна тільки послідовність номерів кодів, які повинні бути вилучені з масиву кодів й продубльовані в кінці з додаванням розряду. А ця послідовність зберігається в екземплярах локальної змінної j, що відповідають рекурсивним викликам процедури Huffman .

Приклад

Побудова оптимального коду Хаффмена для (n=7).

Ліва частина таблиці показує зміни масиву P, а права частина - масиву C. Позиція, що відповідає поточному значенню змінної j, виділена жирним шрифтом.

P						C					
0.20	0.20	**0.23**	**0.37**	**0.40**	**0.60**	**0**	**1**	**00**	**01**	10	10
0.20	0.20	0.20	0.23	0.37	0.40	1	00	01	10	11	11
0.19	0.19	0.20	0.20	0.23			01	10	11	000	000
0.12	**0.18**	0.19	0.20					11	000	**001**	010
0.11	0.12	0.18							001	010	011
0.09	0.11									011	0010
0.09											0011

Вартість отриманого кодування складає:

0.20 * 2 + 0.20 * 2 + 0.19 * 3 + 0.12 * 3 + 0.11* 3 + 0.09 * 4 + 0.09 * 4 = 2.78

Цей результат дещо кращий ніж, отриманий алгоритмом Фано.

Завадостійке кодування
(Кодування стійке від помилок)

Хоча невпинно зростають якість та надійність електронних пристроїв та каналів зв`язку, але й у їх роботі можливі збої, як систематичні, так й випадкові. Результатом вказаних проблем можуть бути відповідні втрати, або спотворення інформації. Але за деяких умов можна застосовувати методи кодування, що дозволяють правильно декодувати повідомлення, навіть при наявності помилок у коді. В якості моделі, що досліджується, будемо розглядати канал зв`язку з перешкодами, оскільки інші проблемні випадки можна до нього звести.

Кодування з виправленням помилок

Канал зв`язку ***C*** із спотвореннями можна розглядати як відображення ***C: B*→B**** відмінне від тотожного.

Тоді використання кодування ***F: S→B**** при передачі інформації каналом ***C*** можна представити діаграмою:

$$S \xrightarrow{F} K \xrightarrow{C} K` \xrightarrow{F^{-1}} S, \quad \text{K} \subset \text{B}^*$$

Кодування F, що має наступні властивості:

$$\forall s \in S, \quad F^{-1}(C(F(s))) = s,$$

називають ***стійким від помилок*** (або ***кодуванням з виправленням помилок***, таким що ***здатне до самостійної корекції***).

Без обмежень загальності можна вважати, що A = B = {0, 1}, а також що саме кодування вільне від помилок.

18

Класифікація помилок

Основні помилки можна звести до наступних типів:

- $0 \to 1$, $1 \to 0$ - ***заміщення символів***;
- $0 \to \lambda$, $1 \to \lambda$ - ***вилучення символів***;
- $\lambda \to 1$, $\lambda \to 0$ - ***вставки символів***.

Канал характеризується верхніми оцінками кількості помилок кожного типу, які можливі при передаванні каналом повідомлення певної довжини. Загальна характеристика помилок каналу (тобто їх кількість та типи) позначається δ.

19

Поганий приклад

Припустимо, що є канал з характеристикою $\delta = (1/n, 0, 0)$, тобто можлива одна помилка типу ***заміщення символів*** при передаванні повідомлення довжини n.

Розглянемо кодування: $F(\alpha) = \alpha\alpha\alpha$, (кожний розряд у повідомленні повторюється тричі) та
декодування
$F^{-1}(abc) = (a + b + c > 1)$, тобто розряд відновлюється методом «голосування».

Хоча наведене кодування здається захищеним для даного каналу, але насправді це не так. Для повідомлення довжини 3n можливо не більше 3 помилок типу заміщення символів, але місця цих помилок не обов`язково розподілені рівномірно.

Нехай канал ***C*** має характеристику $\delta(1/3,0,0)$

$$010 \xrightarrow{F} 000111000 \xrightarrow{C} 111111000 \xrightarrow{F^{-1}} 110$$

20

Контроль парності

Даний метод **дозволяє виявити помилку** в передачі коду, якщо вона сталася в одному єдиному розряді.
Однак, цей метод **не дозволяє визначити** де сталася помилка і виправити її.

$\alpha_1\alpha_2\alpha_3\ldots\alpha_n$ – двійкове число
$\alpha_{контр}$ - додатковий контрольний розряд, такий що
$\alpha_1 \oplus \alpha_2 \oplus \alpha_3 \oplus \ldots \oplus \alpha_n \oplus \alpha_{контр} = 0$,
де $\oplus$ - сума за модулем 2
По каналу зв'язку передається n+1 розрядний код

$$\alpha_1\alpha_2\alpha_3\ldots\alpha_n\ \alpha_{контр}$$

21

Контрольна сума

Нехай треба передати ***n k***-значних чисел в ***p***-ічній системі числення: $a_1,a_2,a_3,\dots a_n$, $0 \le a_i < p^k$

$a_{контр}$ – контрольна сума цієї послідовності

$$a_{контр} = \sum_{i=1}^{n} a_i \bmod p^k$$

Як і $a_1,a_2,a_3,\dots a_n$, $a_{контр} < p^k$

По каналу зв'язку передається n***+1 k***-значних чисел:

$a_1,a_2,a_3,\dots a_n, a_{контр}$

22

Можливість виправлення всіх помилок

Нехай E_s^δ - множина слів, які можна отримати з слова s в результаті всіх можливих комбінацій помилокδ, що можуть мати місце в каналі, тобто $s \in S \subseteq A^*$, $E_s^\delta \subseteq B^*$. Якщо $s` \in E_s^\delta$, то ту конкретну послідовність помилок, що дозволяє отримати з слова s слово s`, позначимо $E^\delta(s, s`)$.

Теорема. Для існування стійкого до помилок кодування з виправленням всіх помилок, необхідно й достатньо, щоб $\forall s_1, s_2 \in S$ $(s_1 \neq s_2)$ $E_{s_1}^\delta \cap E_{s_2}^\delta = \varnothing$, тобто необхідно й достатньо, щоб існувало розбиття множини B^* на множини B_s $(\cup B_s = B^*,\ \cap B_s = \varnothing)$, таке що $\forall s \in S$ $E_s^\delta \subseteq B_s$.

Доведення.

Необхідність випливає з того, що з стійкості кодування слідує $E_{s_1}^\delta \cap E_{s_2}^\delta = \varnothing$

Достатність. Навпаки – по розбиттю $\cup B_s$ будується функція F^{-1}: $\forall s` \in B_s$ $F^{-1}(s`) = s$.

Приклад

Розглянемо канал, в якому в довільному розряді відбувається помилка типу заміщення з ймовірністю p (0 < p < 1/2), причому заміщення різних розрядів статистично незалежні. Такий канал називають ***двійковим симетричним***. В цьому випадку довільне слово s ∈ A* може бути перетворено в довільне інше слово s` ∈ A* заміщеннями розрядів. Таким чином, $\forall s \in A^*$ $E_s^\delta = A^*$ й виправити всі помилки в двійковому симетричному каналі неможливо.

Кодова відстань

Розглянемо

$$d_\delta(\beta`, \beta``) = \begin{cases} \min\limits_{E^\delta(\beta,\beta``)} |E^\delta(\beta`, \beta``)| & , \beta \in E_{\beta`}^\delta \\ +\infty & , \beta`` \notin E_{\beta`}^\delta \end{cases}$$

Ця функція називається відстанню Геммінга, й справді задовольняє всім вимогам відстані (метрики):

1. $d_\delta(\beta`,\beta``)=0 \Leftrightarrow \beta`=\beta``$;
2. $d_\delta(\beta`,\beta``)=d_\delta(\beta``,\beta`)$;
3. $d_\delta(\beta`,\beta``)\le d_\delta(\beta`,\beta```)+d_\delta(\beta```,\beta``)$

Нехай $\sigma = \{a_1\to\beta_1, \ldots, a_n\to\beta_n\}$ схема алфавітного кодування, а d – деяка метрика на B^*. Тоді мінімальна відстань між елементарними кодам

$$d_\delta(\sigma)=\min_{1\le i<j\le n} d_\delta(\beta_i,\beta_j)$$

називається кодовою відстанню схеми σ.

Теорема. Алфавітне кодування зі схемою $\sigma = \{a_1\to\beta_1, \ldots, a_n\to\beta_n\}$ та кодовою відстанню $d_\delta(\sigma)$ є кодуванням з виправленням p помилок типу δ тоді й тільки тоді, коли $d_\delta(\sigma)>2p$.

Код Геммінга для виправлення одного заміщення

Розглянемо побудову *коду Геммінга,* який дозволяє виправляти одиночні помилки типу заміщення. Цей випадок є найпростішим, але й одночасно дуже важливим, оскільки внутрішні шини передачі даних сучасних комп`ютерів мають подібну властивість. Зрозуміло, що для виправлення помилки разом з основним повідомленням потрібно передавати деяку додаткову інформацію.

Нехай повідомлення $\alpha=a_1 \ldots a_m$ кодується словом $\beta=b_1 \ldots b_n$, $n > m$. Позначимо $k = n - m$. Вважаємо, що канал допускає не більше однієї помилки типу заміщення в слові довжини n.

Для заданого n кількість додаткових розрядів k підбирають таким чином, щоб $2^k \ge n + 1$ $2^k \ge n+1 \Rightarrow \frac{2^n}{n+1} \ge 2^{n-k} \Rightarrow \frac{2^n}{n+1} \ge 2^m$.

Приклад

Для повідомлення довжиною m = 32 потрібно k = 6 додаткових розрядів.

Розглянемо принципи побудови *коду Геммінга*. Для цього визначимо підмножини натуральних чисел у відповідності з їх представленням у двійковій системі числення:

- $V_1 = \{1, 3, 5, 7, 9, 11, \ldots\}$ – всі числа, у яких розряд №1 дорівнює 1;
- $V_2 = \{2, 3, 6, 7, 10, 11, \ldots\}$ – всі числа, у яких розряд №2 дорівнює 1;
- $V_3 = \{4, 5, 6, 7, 12, 13, \ldots\}$ – всі числа, у яких розряд №3 дорівнює 1;
- …

За визначенням V_k починається з числа 2^{k-1}.

Розряди в слові $\beta=b_1 \ldots b_n$ з номерами $2^0 = 1, 2^1 = 2, 2^2 = 4, \ldots, 2^{k-1}$ вважаємо *контрольними*, інші розряди – *інформаційними*. Запишемо в інформаційні розряди всі розряди слова $\alpha=a_1 \ldots a_m$ без змін. Контрольні розряди визначаються наступним чином:

- $b_1 = b_3 \oplus b_5 \oplus b_7 \oplus \ldots$ - сума за модулем 2 розрядів з номерами з V_1, крім першого;

- $b_2 = b_3 \oplus b_6 \oplus b_7 \oplus \ldots$ - сума за модулем 2 розрядів з номерами з V_2, крім першого;
- $b_3 = b_5 \oplus b_6 \oplus b_7 \oplus \ldots$ - сума за модулем 2 розрядів з номерами з V_3, крім першого;
- …
- $b_{2^{j-1}} = \bigoplus_{i \in V_j \setminus \{2^{j-1}\}} b_i$.

Нехай після проходження через канал отримали код $c_1 \ldots c_n = C(b_1 \ldots b_n)$, причому

$$\exists I \;\; c_I = \begin{cases} b_I \\ \overline{b_I} \end{cases}, \quad \forall i \neq I \;\; c_i = b_i \;\; .$$

I – номер розряду, в якому, можливо, є помилка заміщення. Нехай $I = i_t \ldots i_1$ – двійкове представлення I.

Визначимо число $J = j_t \ldots j_1$ наступним чином:

- $j_1 = c_1 \oplus c_3 \oplus c_5 \oplus c_7 \oplus \ldots$ - сума за модулем 2 розрядів з номерами з V_1;
- $j_2 = c_2 \oplus c_3 \oplus c_6 \oplus c_7 \oplus \ldots$ - сума за модулем 2 розрядів з номерами з V_2;
- $j_3 = c_4 \oplus c_5 \oplus c_6 \oplus c_7 \oplus \ldots$ - сума за модулем 2 розрядів з номерами з V_3;
- …
- $j_p = \bigoplus_{q \in V_p} c_q$

Теорема. $I = J$.

Ці числа рівні тому, що порозрядно рівні їх двійкові представлення. Дійсно, якщо $i_1 = 0$, тоді $I \notin V_1$, а це означає

$$j_1 = c_1 \oplus c_3 \oplus c_5 \oplus c_7 \oplus \ldots = b_1 \oplus b_3 \oplus b_5 \oplus b_7 \oplus \ldots = 0$$

за визначенням b_1. Якщо ж $i_1 = 1$, тоді $I \in V_1$, а це означає

$$j_1 = c_1 \oplus c_3 \oplus c_5 \oplus c_7 \oplus \ldots = b_1 \oplus b_3 \oplus b_5 \oplus b_7 \oplus \ldots \oplus \neg b_x \oplus \ldots = 1,$$

так як, якщо в сумі за модулем 2 змінити рівно один розряд, то поміняється й значення всієї суми.

Тому завжди $i_1 = j_1$. Аналогічно (розглядаючи V_2, V_3, …) маємо $i_2 = j_2$, $i_3 = j_3$, … . Тобто $I = J$.

Звідси й випливає метод декодування з виправленням помилок:

- Обчислюємо J;
- Якщо $J = 0$, то помилок немає;
- Якщо $J \neq 0$, то $c_J \Leftarrow \neg c_J$;
- Виділяємо *інформаційні розряди* повідомлення, які вже не містять помилок.

Зауваження

Звісно, що метод кодування Геммінга не обмежується тільки розглянутим випадком виправлення одиночних помилок заміщення. Питання побудови кодів

Геммінга з виправленням помилок інших типів розглянуті, наприклад, в [ДМ и МВК Яблонского, Лупанова].

Стиснення даних

Методи кодування, які дозволяють побудувати (без втрати інформації) коди повідомлень, що мають меншу довжину в порівнянні з первісним повідомленням, називають методами ***стиснення*** (або ***упаковки***) даних.

Якість стиснення визначається ***коефіцієнтом стиснення***,

$$1 - \frac{\textit{довжина коду}}{\textit{довжина повідомлення}}$$

який вимірюється найчастіше у відсотках.

Відомі програми для стиснення (arj , zip та інші) мають коєфіцієнти стиснення ≥70% й застосовують неалфавітне кодування.

23

Стиснення текстів

Розглянемо наступний спосіб кодування:

- Повідомлення за певним алгоритмом розбивається на послідовності символів – *слова* (слово може мати декілька входжень до повідомлення).
- Отриману множину слів розглядають як множину букв нового алфавіту. Для цього алфавіту будують схему алфавітного кодування, що є роздільною, її, зазвичай, називають *словником*.
- Код повідомлення будують як пару – код словника й послідовність кодів слів.
- При декодуванні повідомлення використовують словник.

24

Приклад

Для кодування тексту українською мовою, *слово* розглядаємо традиційно (те, що виокремлюється розділяючими символами).

Можна прийняти припущення, що в тексті використовується не більш ніж $2^{16} = 65\,536$ різних слів (зазвичай суттєво менше).

Словам можна спів ставити номери – цілі числа з довжиною представлення 2 байти (рівномірне кодування).

Враховуючи, що в середньому слова містять більше 2 букв, таке кодування дає досить суттєве стискання тексту.

Для великих текстів додаткові витрати на збереження словника не є суттєвими.

25

Вказаний вище спосіб кодування можна вдосконалити, наприклад, наступним чином:

- На кроці 2 застосувати алгоритм Хаффмена для побудови оптимального коду.
- На кроці 1 розв'язати екстремальну задачу розбиття тексту на слова таким чином, щоб серед можливих розбиттів обрати те, що на кроці 2 дає найменшу вартість кодування. Таке кодування буде – «*абсолютно*» оптимальним. На жаль, дуже великою є складність задачі й відповідно час на її розв'язання, тому на практиці це не використовують.

Алгоритм Лемпеля – Зіва

На практиці застосовують наступний підхід, відомий як – *адаптивне стиснення*. За один перегляд тексту одночасно динамічно будують словник й кодують текст. При цьому словник не зберігають – за рахунок використання відповідного алгоритму, при декодуванні словник динамічно відновлюють.

Розглянемо найпростішу реалізацію цього підходу, відому як *алгоритм Лемпеля – Зіва*. Спочатку словник – масив D містить лише порожнє слово, з кодом 0. Далі в тексті послідовно виділяють слова. Чергове слово – максимальне за довжиною слово з наявних в словнику плюс ще один символ. В стисле представлення записується знайдений код слова та додаткова буква, а словник поповнюється «розширеним» словом.

Алгоритм Lmp_Zv_p: Упаковка за методом Лемпеля – Зіва
Вхід: вхідний текст, що заданий масивом кодів символів f : array[1..n] of char.
Вихід: стиснений текст, що представлений послідовністю пар (p, q), де p – номер слова в словнику, q – код додаткової букви – массив g : array[1..m] of record pp : integer; qq : char end.

```
program Lmp_Zv_p;
  D[0] := “”; dk := 0; {початковий стан словника}
  k := 1; m:= 0; {положення поточної букви тексту}
  while k ≤ n do begin
     p := FD(k);  {індекс знайденого слова в словнику}
     t := Length(D[i]); {довжина знайденого слова}
     m := m + 1; {рух по упакованому тексту}
     g[m].pp := p; g[m].qq := f[k+t]; {додати до результату пару - код знайденого
                                       слова й букву}
     dk := dk + 1;  D[dk] := D[p] + f[k+t];  {поповнення словника, + це
                                       конкатенація}
      k := k + t +1;  {рух по тексту}
   end {while}
```

Пошук слова в словнику здійснюють за допомогою функції ***FD(к)***. Параметр ***к*** – номер символу в тексті, починаючи з якого потрібно шукати в тексті слова словника. Повертає функція індекс самого довгого слова в

словнику, що співпадає з символами f[k] … f[k+t] тексту. Якщо такого слова в словнику немає, то повертає 0.

```
function FD(к) : integer;
  t := 0; p := 0;  {початковий стан}
   for i := 1 to  dk  do
      if  D[i] = f[k .. (k + Length(D[i]) + 1)] & Length(D[i]) > t  then begin
        p := i; t := Length(D[i]);  {знайшли більше слово}
      end {if}
   return p;
```

Розпаковка здійснюється наступним алгоритмом.

Алгоритм Lmp_Zv_u: Розпаковка за методом Лемпеля – Зіва
Вхід: стиснений текст, представлений масивом пар g : array[1..m] of record pp : integer; qq : char end.
Вихід: розпакований текст.

```
program Lmp_Zv_u;
   D[0] := “”; dk := 0; {початковий стан словника}
   for k := 1 to  m  do begin
      p := g[k].pp; {індекс слова в словнику}
      q := g[k].qq; {додаткова буква}
       write(D[p] + q); {виведення розширеного слова, + це конкатенація}
       dk := dk + 1; D[dk] := D[dk] + q; {поповнення словника, + це конкатенація}
   end {for}
```

Зауваження

На практиці застосовують різні удосконалення цієї схеми:

- Словник можна з самого початку ініціювати, наприклад, кодами символів, тобто вважати, що слова з однієї букви вже відомі.
- В текстах досить часто зустрічаються регулярні послідовності: пробіли, табуляції та інші. Зберігати подібні підпослідовності в словнику не раціонально. Краще застосовувати спеціальні прийоми, наприклад, кодувати подібну послідовність парою (k, s), де s – код символу, а k – їх кількість.

Шифрування

Шифрування – кодування даних з метою захисту від несанкціонованого доступу.

Процес кодування називають ***шифруванням***, декодування - ***розшифруванням***, кодоване повідомлення – ***шифрованим***, а метод (що застосовується) – ***шифром***.

Основна вимога до шифру полягає в тому, що розшифрування можливе лише при наявності деякої додаткової інформації - ***ключа*** шифру. Процес декодування шифрованого повідомлення без ключа називають ***дешифруванням*** (розкриття шифру).

26

Шифрування

Область знань про шифри, методи їх створення та розкриття називають ***криптографією***. Властивість шифру протистояти розкриттю називають ***криптостійкістю*** (або надійністю) й зазвичай оцінюється складністю алгоритму дешифрування.

На практиці надійність шифру оцінюють з економічних міркувань. Якщо розкриття шифру **коштує більше** ніж сама шифрована інформація, шифр вважають достатньо надійним.

Враховуючи насамперед важливість «*комп'ютерного застосування*» будемо розглядати шифри в алфавіти A = B = {0, 1}.

27

Простий шифр

а б в г ґ д е є ж з и і ї й к л м н о п р с т у ф х ц ч ш щ ь ю я

↓ ↓

д е є ж з и і ї й к л м н о п р с т у ф х ц ч ш щ ь ю я а б в г ґ

факультет ⇒ щдпшрвчіч

$$F(a_i)=a_i+T$$

Для даного шифру T=5

28

Шифрування за допомогою псевдовипадкових чисел

Як джерело (датчик) псевдовипадкових чисел часто використовують рекурентні співвідношення:

$T_{i+1} = (a*T_i + b) \bmod c$, де T_i – попереднє псевдовипадкове число, а числа a, b, c – сталі.

Досить часто їх обирають з наступних міркувань: $c = 2^n$, де n – розрядність процесора, $a \bmod 4 = 1$, b – непарне. В цьому випадку послідовність псевдовипадкових чисел має період c.

Для **A=B={0,1}** процес шифрування визначають так:

- Повідомлення для шифрування розбивають на послідовність слів $S_0, S_1, \dots$, кожне довжиною n.
- До цих слів додають порозрядно за модулем 2 слова послідовності $T_0, T_1, \dots$ (послідовність називають *гаммою шифру*). Тобто $C_i = S_i \oplus T_i$.

29

Шифрування за допомогою псевдовипадкових чисел

Процес *розшифрування* полягає в тому, що ще раз потрібно до слів шифрованого повідомлення $C_0, C_1, \dots$ додати за модулем 2 слова послідовності $T_0, T_1, \dots$. Тобто

$S_i = C_i \oplus T_i$.

Ключем шифру є початкове значення послідовності T_0 .

Шифри, що використовують для шифрування та дешифрування один ключ називають *симетричними*.

З ростом **n** експоненційно зростає й криптостійкість шифру.

30

Недоліки шифрування з псевдовипадковими числами

Наведений вище метод шифрування має суттєвий недолік. Якщо відома хоча б частина первісного повідомлення, то все повідомлення можна легко розшифрувати.

Дійсно, нехай відомо одне первісне слово S_i . Тоді $T_i = S_i \oplus C_i$, а далі всю праву частину *гамма шифру* можна визначити за формулою датчика псевдовипадкових чисел.

На практиці цілком ймовірно, що частина повідомлення є відомою. Наприклад, текстові редактори часто вставляють в початок файлу одну й ту ж саму службову інформацію.

31

Способи покращення шифрування

Для підвищення криптостійкісті симетричних шифрів застосовують різні прийоми:

- Обчислення гамми шифру за ключем, застосовуючи більш складний (секретний) спосіб;
- Застосування замість $\oplus$ більш складної операції (але потрібно забезпечити оберненість);
- Попереднє перемішування бітів первісного повідомлення за певним алгоритмом.

Одним з найбільш надійних симетричних шифрів вважається DES (Data Encryption Standard), який поєднує декілька методів підвищення криптостійкісті.

32

Асиметричні алгоритми шифрування

Асиметричні алгоритми шифрування використовують різні ключі для шифрування та розшифрування даних тому їх і назвали асиметричними.
Ключ для **шифрування** може бути відкритим, а ключ для **розшифрування** – інший і завжди таємний, тому такі системи ще називають криптосистемами з відкритим ключем.
Хоча ключова пара математично зв'язана, визначення закритого ключа з відкритого в практичному плані нездійсненне. Кожний, у кого є відкритий ключ, **зможе зашифрувати** дані, але **не зможе їх розшифрувати.** Тільки людина, яка володіє відповідним закритим ключем може розшифрувати інформацію.

33

Ідея асиметричного шифрування

Функцію **F(x)** називають односторонньою, якщо **F(x)** по **x** може бути алгоритмічно легко обчислено, а обчислення **x** по **F(x)** алгоритмічно неможливо.

Приклад: Функція кодування паролю користувача для зберігання в комп'ютерних системах.

Для забезпечення можливості розшифрування в односторонні функції закладається лазівка (відмичка, секрет), знаючи яку можна відносно легко провести розшифрування. Тобто існує алгоритм **G**, який по **F(x)** та секретному коду α визначає **x**: **G(F(x),α)=x**.

34

Практичні заняття

ТЕМА 1: АЛФАВІТНЕ КОДУВАННЯ ТА ЕФЕКТИВНІ КОДИ.

Мета: Засвоїти основні поняття, результати та методи з побудови ефективних алфавітних кодувань.

Теоретичні питання: Префіксні та роздільні схеми. Умови існування. Оптимальне кодування. Алгоритми Фано й Хаффмена.

Аудиторне завдання:

1. Чи є наведена схема алфавітного кодування

$$\sigma = \{a \to 0,\ b \to 10,\ c \to 011,\ d \to 1011,\ e \to 1111\}$$

префіксною, роздільною? (3-6.1)

2. Для наведеної схеми алфавітного кодування, що є роздільною, побудувати префіксний код з тим же набором довжин елементарних кодів $\sigma = \{a \to 01,\ b \to 10,\ c \to 100,\ d \to 111,\ e \to 011\}$. (7-5.3.6.1).

3. Для алфавітного кодування

$$A = \{a_1, a_2\},\ B = \{b_1, b_2\},\ \sigma = \{a_1 \to b_1,\ a_2 \to b_1\, b_2\}$$

з`ясувати, чи буде воно взаємно однозначним? (8-6.8.1)

4. Побудувати оптимальне префіксне алфавітне кодування для алфавіту {a, b, c, d} з наступним розподілом ймовірностей появи букв: $p_a = 1/2$, $p_b = 1/4$, $p_c = 1/8$, $p_d = 1/8$. (3-6.2)

5. Показати, що максимальна довжина елементарних кодів в оптимальному кодуванні для алфавіту з n букв не перевищує n-1. (7-5.3.18.1)

Домашнє завдання:

1. Чи є наведена схема алфавітного кодування

$$\sigma = \{1 \to ab,\ 2 \to dc,\ 3 \to a,\ 4 \to bcadd,\ 5 \to ca\}$$

префіксною, роздільною? (7-5.3.1.1)

2. Для алфавітного кодування

$$A = \{a_1, a_2, a_3\},\ B = \{b_1, b_2\},\ \sigma = \{a_1 \to b_1b_1,\ a_2 \to b_2b_1b_1,\ a_3 \to b_1b_1b_2\}$$

показати, що воно не є взаємно однозначним. (8-6.8.2)

3. Для заданого розподілу ймовірностей появи букв побудувати оптимальний код за методом Хаффмена P = {0.34, 0.18, 0.17, 0.16, 0.15}. (7-5.3.12.1)

4. Для розподілу ймовірностей з попередньої задачі побудувати код за методом Фано. (7-5.3.13.1)

5. Навести приклад розподілу ймовірностей появи букв, при якому код, побудований за методом Фано не є оптимальним. (7-5.3.13.2)

Додаткове завдання:

1. Для алфавітного кодування

$A = \{a_1, a_2, a_3\}$, $B = \{b_1, b_2, b_3\}$, $\sigma = \{a_1 \to b_1, a_2 \to b_1b_2, a_3 \to b_3b_1\}$
показати, що ні σ, ні σ^{-1} не є префіксною, але кодування є взаємно однозначним. (8-6.8.4)

2. Для алфавітного кодування
$A = \{a_1, a_2, a_3, a_4, a_5\}$, $B = \{b_1, b_2, b_3\}$,
$\sigma = \{a_1 \to b_1b_2, a_2 \to b_1b_3b_2, a_3 \to b_2b_3, a_4 \to b_1b_2b_1b_3, a_5 \to b_2b_1b_2b_2b_3\}$
з`ясувати, чи буде воно взаємно однозначним (використовуючи загальний критерій)? (8-6.8.6)

3. Нехай схема алфавітного двійкового коду σ містить 2^{n+1} елементарних кодів з довжиною, що не перевищує n . (7-5.3.11)
   a. Довести, що схема кодування σ не є префіксною.
   b. Чи може схема кодування σ бути роздільною?

4. Показати, що якщо n не є степінню 2, то при довільному розподілі ймовірностей появи букв $P = \{p_1, \dots, p_n\}$ в оптимальному коді знайдуться елементарні коди різної довжини. (7-5.3.15):

5. Пояснити, чому наведений код не є оптимальним $P = \{0.6, 0.2, 0.2, 0.1, 0.1\}$, $\sigma = \{a \to 0, b \to 10, c \to 11, d \to 01\}$. (7-5.3.16)

6. Знайти найменше n та такий розподіл ймовірностей появи букв $P = \{p_1, \dots, p_n\}$, при яких існують оптимальне кодування з різними довжинами елементарних кодів. (7-5.3.17)

7. Схема кодування називається *майже рівномірною*, якщо довжини елементарних кодів відрізняються не більш ніж на 1. Показати, що для довільного натурального n *майже рівномірне* кодування є оптимальним при розподілі ймовірностей $P = \{1/n, 1/n, \dots, 1/n\}$. Чи вірним є обернене твердження? (7-5.3.20)

8. Чи вірним є твердження, що в оптимальному кодуванні кількість елементарних кодів максимальної довжини є парним числом? Чи вірно, що це число є степінню 2? (7-5.3.23)

ТЕМА 2: ЗАВАДОСТІЙКЕ КОДУВАННЯ. СТИСНЕННЯ ДАНИХ. ШИФРУВАННЯ.

Мета: Засвоїти основні поняття, результати та методи з побудови завадостійкого кодування, стиснення даних, шифрування.

Теоретичні питання: Кодування з виправленням помилок. Класифікація помилок. Кодова відстань. Код Геммінга. Стиснення текстів. Алгоритм Лемпеля–Зіва. Криптографія. Шифрування за допомогою випадкових чисел.

Аудиторне завдання:

1. Для кодового слова $\alpha = 01100$ побудувати множину всіх слів, які можуть бути отримані при допущенні однієї помилки заміщення.

2. За методом Геммінга побудувати кодове слово для повідомлення $\alpha = 1011$. (8-6.9.1)

3. Декодувати слово $\beta^{`}$ = 1001110 , де можлива помилка не більш ніж в одному розряді. (8-6.9.3)

Домашнє завдання:

1. Показати, що для несиметричних помилок наступна функція є відстанню: $d_\delta(\beta^{`},\beta^{``}) = 2\min\limits_{\beta^{```}\in B^*}\max(\min\limits_{\{E^\delta(\beta^{`},\beta^{```})\}}|E^\delta(\beta^{`},\beta^{```})|, \min\limits_{\{E^\delta(\beta^{```},\beta^{``})\}}|E^\delta(\beta^{```},\beta^{``})|)$. (3-6.3)

2. Відслідкувати роботу алгоритму Лемпеля – Зіва на прикладі наступного первісного тексту: abaabaaab. (3-6.4)

3. За методом Геммінга побудувати кодове слово для повідомлення α = 10101011. (8-6.9.2.5)

4. Декодувати слово $\beta^{`}$ = 001011110111111 , де можлива помилка не більш ніж в одному розряді. (8-6.9.4)

5. Для невеликого повідомлення (з 2, 3 слів) здійснити шифрування та розшифрування за допомогою псевдовипадкових чисел, при невеликих значеннях сталих a, b, c та ключа шифру.

Додаткове завдання:

1. Записати алгоритм кодування та декодування за методом Геммінга, що виправляє не більш ніж одну помилку.

2. Реалізувати стиснення даних за алгоритмом Лемпеля – Зіва.

3. Реалізувати шифрування даних за допомогою псевдовипадкових чисел.

Література до модуля 7.

ОСНОВНА

1. Дискретная математика и математические вопросы кибернетики. Т.1 /Под общ. редакцией С.В.Яблонского и О.Б.Лупанова. - М.: "Наука". 1974. – 312 с.
2. Хемминг Р.В. Теория кодирования и теория информации. – М.: "Радио и связь". 1983.– 176 с.
3. Новиков Ф.А. Дискретная математика для программистов. - СПб.: "Питер", 2001. - 304 с.
4. Нікольський Ю.В., Пасічник В.В., Щербина Ю.М. Дискретная математика. – К.: Видавнича группа BHV, 2007. – 368 с.
5. Фомичев В.М. Дискретная математика и криптология. Курс лекций / Под общ. ред. д-ра физ.-мат. н. Н.Д. Подуфалова. - М.: ДИАЛОГ-МИФИ, 2003. - 400 с.
6. Марков А.А. Введение в теорию кодирования. - М.: "Наука". 1982. – 192 с.

7. Гаврилов Г.П., Сапоженко А.А. Сборник задач по дискретной математике. - М.: "Наука", 1977. - 368 с.
8. Галушкина Ю.И., Марьямов А.Н. Конспект лекций по дискретной математике.- М.: Айрис_пресс, 2007. – 176с.

ДОДАТКОВА

1. Введение в криптографию / Под общ. ред. В.В. Ященко. - М.: МЦНМО: "ЧеРо", 1999. - 272 с.
2. Анин Б.Ю. Защита компьютерной информации. - СПб.: БХВ-Петербург, 2000. - 384 с.
3. Домашев А.В., Грунтович М.М., Попов В.О., Правиков Д.И., Прокофьев И.В., Щербаков А.Ю. Программирование алгоритмов защиты информации. Учебное пособие. - М.: "Нолидж", 2002. - 416 с.
4. Аршидов М.Н., Садовский Л.Е. Коды и математика (рассказы о кодировании). – М.: Наука, 1983. – 144 с.
5. Брассар Ж. Современная криптология. – М.: Издательско-полиграфическая фирма ПОЛИМЕД, 1999. – 176 с.
6. Коблиц Н. Курс теории чисел и криптографии. – М.: Научное изд-во ТВП, 2001. – 260 с.

Українсько-англійський тематичний словник з дискретної математики

Розділи «Теорія автоматів» та «Теорія кодування».

автомат	automaton
— зв'язний	connected automaton
— Мілі	Mealy machine
— Мура	Moore machine
автоматне відображення	automaton mapping
алгебра подій	event algebra
алгоритм Лемпеля – Зіва	LZ77 and LZ78
алгоритм Лемпеля – Зіва – Уелча	Lempel – Ziv – Welch (LZW)
алгоритм мінімізації Ауфенкампа – Хона	Aufenkamp – Hohn minimization algorithm
алгоритм мінімізації Хопкрофта	Hopcroft minimization algorithm
алфавіт	alphabet
— вихідний	output alphabet
— вхідний	input alphabet
алфавітне відображення	mapping of words
відстань Геммінга	Hamming distance
декодування	decoding
детермінований скінчений автомат (ДСА)	deterministic finite automaton
— — —, мінімізація	DFA minimization
дешифрування	decryption
діаграма станів	state diagram
згенероване (автоматом) відображення	mapping induced (by automaton)
ключ	key
код	code
— Геммінга	Hamming code
— префіксний	prefix code

кодування	encoding
— алфавітне	non-singular code
— арифметичне	arithmetic coding
— ентропійне	entropy encoding
— з виправленням помилок	error-correcting codes
— завадостійке	error-resistant encoding
— Хаффмена	Huffman coding
криптографія	cryptography
криптостійкий	cryptographically strong, strong cryptography
криптостійкість	cipher strength
мінімізація автоматів	automata minimization
недетермінований скінчений автомат	nondeterministic finite automaton
повідомлення	message
подія	event
— елементарна	elementary event
представлення подій	representation of events
пряме виправлення помилок	forward error correction (FEC), channel coding
регулярна множина	regular set
— мова	regular language
— подія	regular event
— вираз	regular expression
розпізнає (мову)	accept (language)
скінченний автомат	finite-state machine (FSM), finite automaton
скінченний перетворювач	finite-state transducer
стан	state
— заключний	final state, accepting state
— початковий	initial state, start state
стиснення	compression
— без втрат	lossless compression
— даних	data compression, source coding
— з втратами	lossy compression
схема роздільна	uniquely decodable code

теорія автоматів	Automata theory
теорія кодування	coding theory
функція	function
— виходів	output function
— переходів	state-transition function
чорна скриня	black box
шифр	cipher
шифрування	encryption
— імовірнісне	probabilistic encryption