Алгоритми та рекурсія

Модуль 2 Лекція 1

План

- Поняття алгоритму
- Цикли та алгоритми для матриць
- Рекурсивні функції та алгоритми
- Складність алгоритмів
- Алгоритми сортування
- Префіксний та суфіксний запис
- Двійкові та шістнадцяткові числа
- Числа зі знаком
- Подальше вивчення матриць

Умовні позначення

- визначення

- приклад

- примітка

- важливо!

- теорема

Поняття алгоритму

Алгоритм - це скінченний набір інструкцій по перетворенню інформації (команд), виконання яких приводить до результату.

Властивості алгоритму:

1. Елементарність
2. Визначеність
3. Масовість
4. Результативність

Цикли і алгоритми для матриць

Необхідно знайти суму послідовності:

$$\sum_{i=1}^{n} p(i) = \underbrace{p1 + p2 + p3 + \ldots + pn}_{\text{n разів}}$$

$$S = \sum_{i=1}^{n} p(i)$$

$$S = p1$$
$$S = p1 + p2$$
$$S = p1 + p2 + p3$$
$$\ldots$$
$$S = p1 + p2 + p3 + \ldots + pn$$

$$S = p1$$
$$S = S + p2$$
$$S = S + p3$$
$$\ldots$$
$$S = S + pn$$

Загальна формула: $S = S + p(i)$

Отже, для знаходження суми послідовності

$$\sum_{i=1}^{n} p(i) = p1 + p2 + p3 + \ldots + pn$$

алгоритм має вигляд:

Покласти $S = 0$;

Цикл по i від 1 до n:

Замінити значення S на $S + p(i)$;

Кінець циклу.

Дані часткові впорядкування $\leq_1$ і $\leq_2$. Впорядкування $\leq_2$ називається ***топологічним сортуванням*** впорядкування $\leq_1$, якщо $\leq_2$ є повним впорядкуванням, і кожен раз, коли $a \leq_1 b$, тоді $a \leq_2 b$.

Алгоритм знаходження топологічного сортування для впорядкування $\leq_1$ на множині S.

Процедура ТС(S, ≤1):

Вибрати мінімальний елемент s із (S, ≤1);

Вилучити s із S;

Виконувати наступні кроки до тих пір, поки S не порожнє;

Вибрати мінімальний елемент t з (S, ≤1) і вилучити його;

Покласти $s \leq 1\ t$;

Перейменувати t в s;

Кінець процедури.

Добуток матриці на скаляр

Нехай $A = [Aij]$ і $B = [Bij]$ – матриці розміру $m \times n$.
Добуток матриці A на скаляр a є матриця $[Bij] = a[Aij]$.

Алгоритм знаходження B такий:

Процедура Скаляр(*a*, *A*, *m*, *n*):

Цикл по i від 1 до m:

Цикл по j від 1 до n:

$Bij = aAij$;

Кінець циклу;

Кінець циклу;

Кінець процедури.

Додавання матриць

Сума A + B є матриця $C = [C_{ij}]$ розміру $m \times n$, де $C_{ij} = A_{ij} + B_{ij}$.

Алгоритм знаходження суми двох матриць:

Процедура Додавання матриць(*A*, *B*, *m*, *n*):

Цикл по *i* від 1 до *m*:

Цикл по *j* від 1 до *n*:

$C_{ij} = A_{ij} + B_{ij}$;

Кінець циклу;

Кінець циклу;

Кінець процедури.

Множення матриць

Нехай $A = [A_{ij}]$ є матриця $m \times p$, а $B = [B_{ij}]$ є матриця $p \times n$.Тоді (матричний) ***добуток*** A і B, що позначається через AB, є матриця $C = [C_{ij}]$ розміру $m \times n$, де C_{ij} – скалярний добуток i-го рядка матриці A і j-го стовпця матриці B.

$$C_{ij} = [A_{i1}\ A_{i2}\ A_{i3}\ \dots\ A_{ip}] * \begin{bmatrix} B_{1j} \\ B_{2j} \\ \square \\ B_{pj} \end{bmatrix} = \sum_{k=1}^{p} A_{ik} B_{kj}$$

Алгоритм:

Процедура Множення матриць(A, B, m, p, n)

Цикл по i від 1 до m:

Цикл по j від 1 до n:

$C_{ij} = 0$;

Цикл по k від 1 до p:

C_{ij} замінити на $C_{ij} + A_{ik}B_{kj}$

Кінець циклу;

Кінець циклу;

Кінець циклу;

Кінець процедури.

Рекурсивні функції та алгоритми

Іноді функцію буває зручніше або навіть необхідніше задавати за допомогою так званого «рекурсивного методу». З принципу індукції випливає, що якщо:

1. функція задана для деякого даного початкового значення a, і
2. коли функція задана для деякого значення k, більшого за a, вона задана також для значення $k + 1$,

то, тим самим, функція визначена для всіх цілих чисел, більших a.

Знаходження Факторіалу $f(n) = n! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot n$

Процедура Факторіал(n):

Якщо $n = 0$, то Факторіал(n) = 1;
Покласти Факторіал(n) = 1;
Цикл по k від 1 до n:
Значення Факторіал(n) замінити на $k \cdot$ (Факторіал(n));
Кінець циклу;
Кінець процедури.

Рекурсивне визначення функції «факторіал»:

Факторіал(0) = 1;

Факторіал ($k + 1$) = ($k + 1$) $\cdot$ Факторіал(k).

Процедура Факторіал(n):

Якщо $n = 0$, то Факторіал(n) = 1;
Якщо $n > 0$, то Факторіал(n) = $n \cdot$ Факторіал($n - 1$);
Кінець процедури.

План

Послідовність Фібоначі

1, 1, 2, 3, 5, 8, 13, 21 і 34 (кожен наступний є сумою двох попередніх)

$a_1 = 1, a_2 = 1, a_3 = a_1 + a_2, \ldots, a_n = a_{n-2} + a_{n-1}$

Алгоритм

Процедура *Фіб*(*n*):

Якщо $n = 1$, то *Фіб*(*n*) = 1;

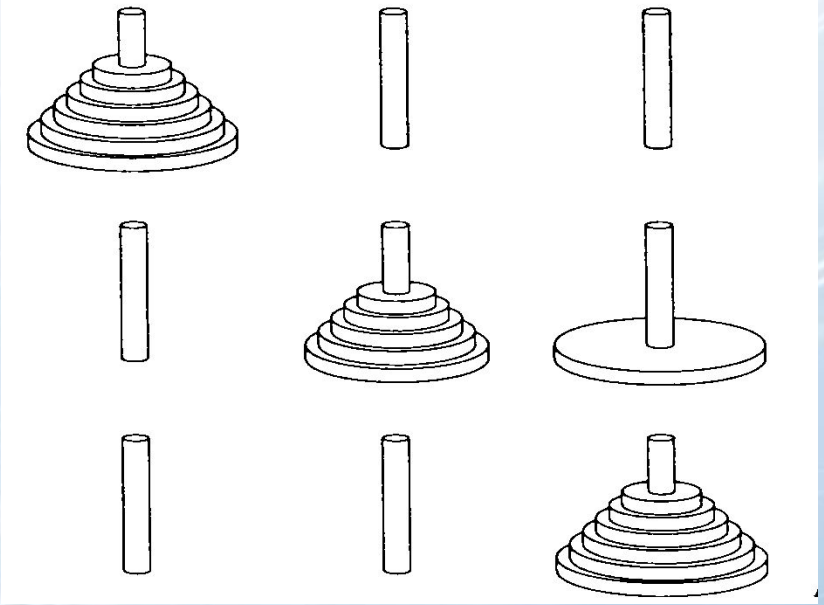
Якщо $n = 2$, то *Фіб*(*n*) = 1;

Якщо $n > 2$, то *Фіб*(*n*) = *Фіб*(*n*-1) + *Фіб*(*n*-2);

Кінець процедури.

Ханойська башня.

В задачі є три стержні та набір дисків різного діаметру. Диски нанизані на один зі стержнів в порядку спадання їх діаметра – найбільший знаходиться знизу, найменший – зверху. Задача полягає в тому, щоб перемістити всі диски на інший стержень зі збереженням їх порядку. При цьому диски можна переміщати по одному за раз, і диск більшого діаметру не можна поміщати на диск меншого діаметру.



Процедура ***Хан(A, C, n)***:

Якщо $n = 1$, перемістити єдиний диск із стрижня A на C;

Покласти $B = 6 - A - C$;

Якщо $n > 1$, тоді

Змістити верхні $n - 1$ дисків на B, використовуючи *Хан*(A, B, n-1);

Змістити останній диск з A на C, використовуючи *Хан*(A, C, 1);

Змістити верхні $n - 1$ дисків на C, використовуючи *Хан*(B, C, n-1);

Кінець процедури.

Головоломку Ханойська вежа придумав французький математик Едуард Люка в 1883 р. Він зв'язував свою іграшку з міфічною легендою про значно більшу вежу Брами, яка, як стверджувалося, складалась з 64 дисків чистого золота, а стержні представляють собою три діамантових шпиля. При створенні світу Всевишній помістив диски на перший шпиль та велів, щоб жреці перемістили їх на третій у відповідності з визначеними правилами. За наявними відомостями жреці трудяться над цією задачею вдень і вночі – і як тільки вони завершать, башня розсиплеться в прах і наступить кінець світу.

Складність алгоритмів

Нехай область визначення функцій f і g - множина цілих додатних чисел, а множина їх значень - множина дійсних чисел.

Функція g **мажорує** функцію f, якщо існує дійсне число k і ціле додатне число m таке, що $|f(n)| \le k|g(n)|$ для всіх $n \ge m$.

Якщо g мажорує f, це позначається як $f(n) = O(g(n))$. Символ $O(g(n))$ читається як "O велике" від $g(n)$; при цьому говорять, що $f(n)$ має порядок O велике від $g(n)$.

ТЕОРЕМА 4.1. Якщо r і s - дійсні числа, $r \le s$ і $n > 1$, тоді $n^r \le n^s$. Отже, $n^r = O(n^s)$.

☑ **ТЕОРЕМА 4.2.** Якщо $f(n) = O(g(n))$, то $cf(n) = O(g(n))$.

☑ **ТЕОРЕМА 4.3.** Якщо $f(n) = O(g(n))$ і $h(n) = O(g(n))$, то $(f+h)(n) = O(g(n))$

☑ **ТЕОРЕМА 4.4.** Якщо $f(n) = O(g(n))$ і $h(n) = O(e(n))$, то $(f \cdot h)(n) = O((g \cdot e)(n))$.

Розглянемо алгоритм додавання матриць

Процедура Додавання матриць(*A*, *B*, *m*, *n*):

Цикл по *i* від 1 до *m*:

Цикл по *j* від 1 до *n*:

$C_{ij} = A_{ij} + B_{ij}$;

Кінець циклу;

Кінець циклу;

Додавання відбувається для кожного i та j. Оскільки *i* приймає *m* значень, а *j* приймає *n* значень, то виконується *mn* операцій додавання. Нехай $N = \max(m, n)$. Тоді число виконуваних арифметичних операцій має порядок $O(N^2)$.

Розглянемо алгоритм

Процедура Множення матриць(A, B, m, p, n)

Цикл по i від 1 до m:

Цикл по j від 1 до n:

$C_{ij} = 0$;

Цикл по k від 1 до p:

C_{ij} замінити на $C_{ij} + A_{ik}B_{kj}$

Кінець циклу;

Кінець циклу;

Кінець циклу;

Кінець процедури.

Виконується mnp операцій додавання і mnp операцій множення. Отже, всього виконується $2mnp$ операцій. Нехай $N = \max(m, n, p)$. Порядок $O(N^3)$.

Алгоритми сортування

Сортування вибором

Процедура СВ:

Цикл по *i* від 1 до *n*-1:

Покласти *min* = *i*;

Цикл по *j* від 1 до *n*:

Якщо *ai* < *amin*, то нехай *min* = *j*;

Кінець циклу;

Поміняти місцями *ai* і *amin*;

Кінець циклу;

4	9	7	6	2	3

Задана послідовність

2	9	7	6	4	3

Крок 0: 2 ↔ 4

2	3	7	6	4	9

Крок 1: 3 ↔ 9

2	3	4	6	7	9

Крок 2: 4 ↔ 7

2	3	4	6	7	9

Крок 3: 6 ↔ 6

2	3	4	6	7	9

Крок 4: 7 ↔ 7

Бульбашкове сортування

Процедура БС:

Цикл по i від 2 до n:

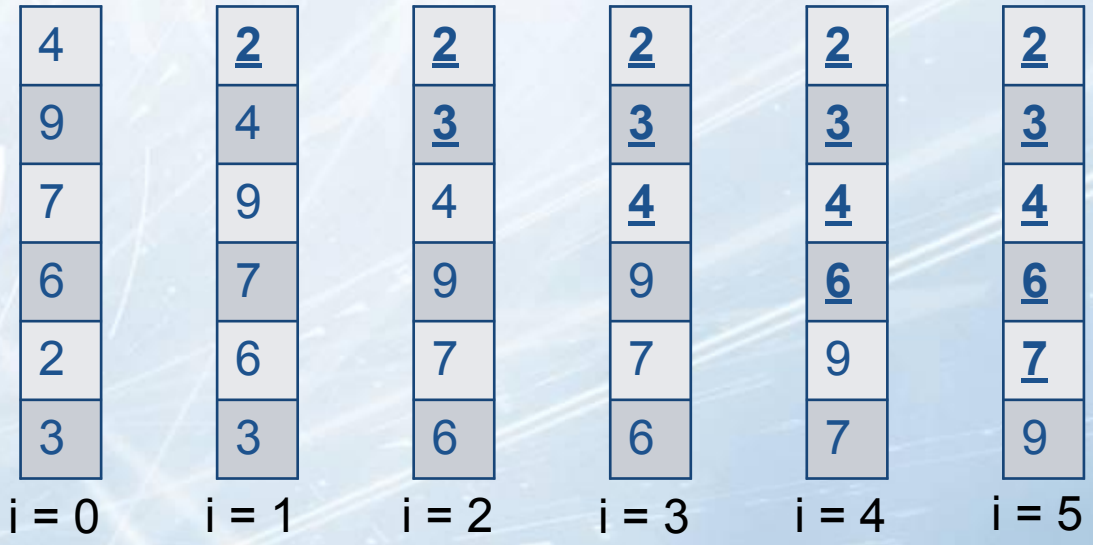
Цикл по j від n до i з кроком -1:

Якщо $a_j < a_{j-1}$, то поміняти місцями a_j і a_{j-1};

Кінець циклу;

Кінець циклу;

Кінець процедури.



Швидке сортування

Процедура ШС(1, *n*):

Покласти $l = 1$ і $r = n$;

Якщо $l \neq r$, то

Покласти $m = l$ і розбити список $(a_l, a_{l+1}, a_{l+2}, ..., a_r)$ на

S_1, отриманий як $\{a_i : a_i \leq a_l, i \neq l\}$ і

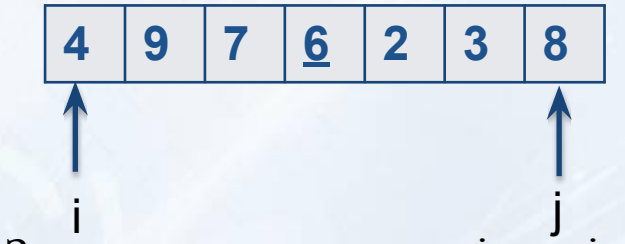
S_2, отриманий як $\{a_i : a_i > a_l\}$, так що

$S_1 = (a_l, a_{l+1}, a_{l+2}, \cdots, a_j)$ і $S_2 = (a_{j+1}, a_{j+2}, a_{j+3}, \cdots, a_r)$ після перепозначення;

Провести конкатенацію ШС$(1, j)$, a_m і ШС$(j + 1, r)$;

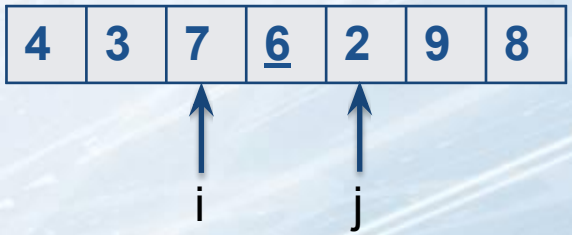
Кінець процедури.

Швидке сортування

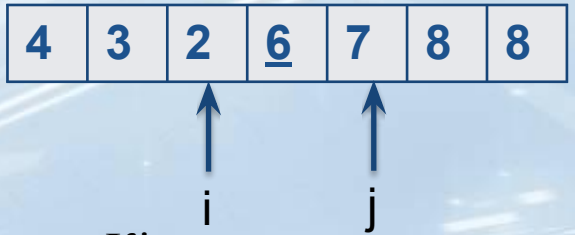


Задане положення вказівників

положення 1-го обміну



положення 2-го обміну



Кінець процедури

План

Префіксний та суфіксний записи

Вираз знаходиться в ***префіксному*** (***польському***) ***запису***, якщо в ньому знак операції безпосередньо передує операндам, на які вона діє.

Вираз знаходиться в суфіксному (***постфіксному***, або ***зворотному польському запису***), якщо в ньому знак операції слідує безпосередньо за операндами, на які вона діє. Вираз знаходиться в ***інфіксному запису***, якщо в ньому знак операції знаходиться між операндами, на які вона діє.

Арифметичний вираз знаходиться в ***повному дужковому запису***, якщо кожному оператору в ньому відповідає пара дужок, в які вкладені оператор і ті операнди, на які він діє.

Перехід від інфіксного виразу до постфіксного може бути здійснено за допомогою наступної процедури:

1. Додавати дужки у вираз, поки він не стане повнодужковим.

2. Починаючи з самих внутрішніх дужок, видалити пару дужок і помістити оператор, що знаходиться усередині дужок на місце відповідне його правій дужці. За наявності більше однієї самої внутрішньої пари дужок починати потрібно з лівої пари.

3. Перейти до наступної пари дужок, вилучити її і помістити оператор, що знаходиться усередині дужок, на місце відповідне його правій дужці.

4. Продовжувати крок (3), поки всі дужки не будуть видалені.

1. Нехай задано інфіксний вираз $(a + b) \times (c + d) \wedge x$. Додаємо дужки, поки вираз не стане повнодужковим. Одержуємо вираз $((a + b) \times ((c + d) \wedge x))$. Виконавши крок (2), отримаємо $(ab + \times (cd + \wedge x))$. Виконавши крок (3), одержуємо $(ab + \times cd + x\wedge)$. Повторюючи крок (3), приходимо до запису $ab + cd + x \wedge \times$.

2. Нехай задано інфіксний вираз $a \times b + c \times d$. Додаємо дужки, поки не одержимо повнодужковий вираз, тобто вираз $((a \times b) + (c \times d))$. Виконавши крок (2) одержимо $(ab\times + cd\times))$. Повторюючи крок (3), одержуємо $ab \times cd\times +$.

Постфіксний вираз може бути замінено інфіксним за допомогою наступної процедури:

1. Починати зчитування виразу зліва направо.

2. Якщо зчитаний символ не є символом операції, тоді помістити його в стек і продовжувати читання.

3. Якщо зчитаний символ є символом операції, то:

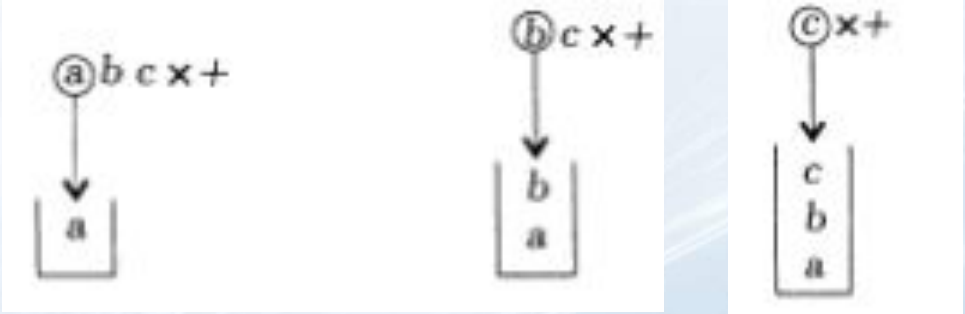
а) виштовхнути зі стека два верхні елементи;

б) помістити символ операції між другим і першим елементом із стека і поставити дужки;

в) помістити вираз, одержаний на кроці (б), знову в стек.

4. Продовжувати читання зліва направо і виконувати кроки (2) і (3) до завершення.

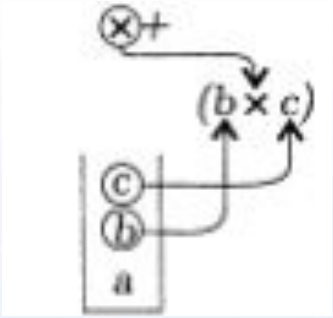
Нехай задано вираз *abc*×+. Спочатку зчитуємо *a* і, оскільки це не символ операції, поміщаємо цей елемент в стек, як показано на мал. 1. Потім прочитуємо *b* і, оскільки це не символ операції, поміщаємо його в стек, як показано на мал. 2.



мал.1 мал.2 мал.3

Далі зчитуємо *c* і, оскільки це не символ операції, поміщаємо його в стек, як показано на мал. 3.

Потім прочитуємо × і, оскільки це символ операції, виштовхуємо із стека (видаляємо) с (верхній елемент) і *b* (наступний елемент), поміщаємо × між *b* і с, як показано на мал.4.



мал. 4

Поміщаємо (*b* × с) у стек, як показано на мал. 5. Далі зчитуємо + і, оскільки це символ операції, виштовхуємо із стека (видаляємо) (*b* × *c*) і *a* (наступний елемент). Поміщаємо + між *a* і (*b* × *c*), як показано на мал. 6.

Поміщаємо (*a* + (*b* × *c*)) в стек. Одержуємо шуканий вираз.

мал. 5 мал.6

План

Двійкові та шістнадцяткові числа

Якщо число n представляється в ***десятковій системі числення***, в основі якої лежить число 10, як

$a_m a_{m-1} a_{m-2} \ldots a_2 a_1 a_0$

то

$n = a_m \cdot 10^m + a_{m-1} \cdot 10^{m-1} + \ldots + a_2 \cdot 10^2 + a_1 \cdot 10 + a_0,$

де a_j - цифри від 0 і 9 для всіх $0 \le i \le m$.

$3124 = 3(10)^3 + 1(10)^2 + 2(10) + 4.$

Аналогічно, якщо число n подане в ***двійковій системі числення***, в основі якої лежить число 2, то кожна цифра повинна бути або 1, або 0, і її значення визначається позицією в записі числа.

$n = a_m \cdot 2^m + a_{m-1} \cdot 2^{m-1} + \ldots + a_2 \cdot 2^2 + a_1 \cdot 2 + a_0$

$10101_2 = 21_{10}$

21	2			
20	10	2		
1	10	5	2	
	0	4	2	2
		1	2	**1**
			0	

2^4	2^3	2^2	2^1	2^0	
1	0	1	0	1	$= 1\cdot 2^4 + 0\cdot 2^3 + 1\cdot 2^2 + 0\cdot 2^1 + 1\cdot 2^0 =$
					$= 2^4 + 2^2 + 1 = 16 + 4 + 1 = 21$

В основі шістнадцяткової системи числення лежить число 16.

10_{16} еквівалентно 16, а букви A, B, C, D, E, F позначають числа 10, 11, 12, 13, 14, 15 відповідно.

Число в шістнадцятковій системі представлено у вигляді $a_m a_{m-1} a_{m-2} \ldots a_2 a_1 a_0$

то $n = a_m \cdot 16^m + a_{m-1} \cdot 16^{m-1} + \ldots + a_2 \cdot 16^2 + a_1 \cdot 16 + a_0$

де для всіх $0 \le i \le m$, $0 \le a_i \le 16$

$n = 2A3B4_{16}$ відповідає десяткове $n = 2 \cdot 16^4 + A \cdot 16^3 + 3 \cdot 16^2 + B \cdot 16^1 + 4 = 130072 + 40960 + 768 + 176 + 4 = 171980$

Перетворити 1011.1111_2 в десякову форму.

$$1011.1111_2 = 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 + 1 \cdot 2^{-1} + 1 \cdot 2^{-2} + 1 \cdot 2^{-3} + 1 \cdot 2^{-4}$$

$$= 8 + 0 + 2 + 1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} = 11 + 0.5 + 0.25 + 0.125 + 0.625 = 11.9375$$

Числа зі знаком

Представлення додатніх і від'ємних чисел у двійковій та шістнадцятковій системах будується за допомогою **методу двійкового доповнення.**

Цей метод вимагає, щоб цілі числа були записані з використанням фіксованої кількості бітів (двійкових розрядів).

Для представлення цілого додатнього числа за допомогою **8 біт** перед ним при необхідності дописується потрібна кількість нулів.

Ціле додатнє число повинне починатися з 0, а ціле від'ємне – з 1. Оскільки при такому способі запису найбільше можнливо ціле додатнє число є 01111111, то всього є 127 цілих додатніх чисел. Якщо дано додатнє ціле число, представлене з використанням n біт, тоді відповідне йому від'ємне число знаходять шляхом віднімання цього числа з 2^n, вираженому в двійковій формі. Це здійснюється відніманням числа з 2^n-1 (знаходження 1-го доповнення) і додаванням 1.

Знайти представлення числа -56 у 8-бітному двійковому доповненні. Спочатку число 56 представляється як 00111000. Віднімачи його з 11111111 отримуємо 11000111. Додавання 1 дає 11001000, так що -56 має представлення 11001000.

Знайти представлення в шістнадцятковій системі числення заперечення шістнадцяткового числа 78F з використанням 16 біт. Спочатку замінимо запис числа на 16-бітне число 078F. Потім віднімемо 078F з FFFF, отримаємо F870, і далі додаємо 1. В результаті отримуємо представлення F871.

Подальше вивчення матриць

Детермінант матриці розміру ***n x n*** позначається через det (A), $|A|$ або

$$\begin{vmatrix} A_{11} & A_{12} & \cdots & A_{1n} \\ A_{21} & A_{22} & \cdots & A_{2n} \\ \vdots & & & \vdots \\ A_{n1} & A_{n2} & \cdots & A_{nn} \end{vmatrix}$$

і визначається наступним чином

а) якщо $n = 1$, то $\det(A) = |A_{11}| = A_{11}$;

б) якщо $n \geq 2$, то $\det(A) = A_{1j}(-1)^{1+j} \det(\square A_{1j})$

Нехай $M_{ij} = \det(A_{ij})$. Число M_{ij} називається ***мінором*** елемента A_{ij}, а число $Ci = (-1)i+j \det(A_{ij})$ - ***алгебраїчним доповненням*** елемента A_{ij}. Отже $\det(A) = A_{1j}(-1)\,1+j \det(\square A_{1j}) = A_{1j}\,C_{1j}$

Нехай I_n є матриця $[I_{ij}]$ розміру $n \times n$, де $I_{ij} = \delta_{ij}$, тобто на діагоналі матриці I_n стоять одиниці, а вся решта її елементів дорівнюють нулю. Матриця I_n називається одиничною матрицею розміру $n \times n$, або просто ***одиничною матрицею***. Одинична матриця володіє тією властивістю, що для будь-якої матриці A розміру $n \times n$ має місце рівність $AI_n = I_nA = A$.

Нехай A - матриця розміру $n \times n$. ***Мультиплікативно оберненою***, або просто ***оберненою*** до матриці A називається матриця A^{-1}, яка володіє (якщо вона існує) тією властивістю, що $AA^{-1} = A^{-1}A = I_n$.

Алгоритм знаходження матриці, оберненої до матриці розміру n x n

1. Знайти det(A). Якщо він рівний нулю, обернена матриця не існує.

2. Для кожного A_{ij} знайти доповнення алгебри C_{ij} і сформувати матрицю $C = [C_{ij}]$, що називається ***матрицею алгебраїчних доповнень***.

3. Знайти матрицю C^t, транспоновану до матриці C, так що $C^t_{ij} = C_{ji}$. Матриця C^t називається ***спряженою*** до матриці A і позначається adj (A).

4. Розділити кожен елемент спряженої матриці adj(A) на det(A).

5. $A^{-1} = \text{adj}(A)$.

Література до лекції

- Андерсон Д.А. Дискретная математика и комбинаторика: Пер. с англ.. – М.: Изд. дом «Вильямс», 2003. – 960 с.
- Грэхем Р., Кнут Д., Паташник О., Конкретная математика. Основание информатики: Пер. с англ. – М.: Мир, 1998. – 703 с.
- Співаковський О.В., Львов М. С. Основи алгоритмізації та програмування: Навчальний посібник. - Херсон, 1997.- 140 с.
- Матеріали сайту: http://algolist.manual.ru/

Дякую за увагу