

## Практичне заняття № 3 (за темою лабораторної роботи №2)

*Параметри алгоритму. Правило безпосереднього перероблення. Асимптотичні характеристики складності алгоритму. Алгоритми з поліноміальною та експоненціальною складністю.*

### Параметрична модель алгоритму

Деяка змінна величина, яка визначає значення характеристик математичного об'єкту, називається *параметром*. Прикладом може бути частотна характеристика RC – ланцюга. R і C – параметри, затримка прямокутного сигналу  $\tau = RC$

Характеристики алгоритму визначаються наступними параметрами Рис. 1 :

1. Правило початку.
2. Правило вводу даних.
3. Система вхідних даних.
4. Правило безпосереднього перероблення .
5. Система проміжних результатів.
6. Система кінцевих результатів.
7. Правило виводу.
8. Правило закінчення.

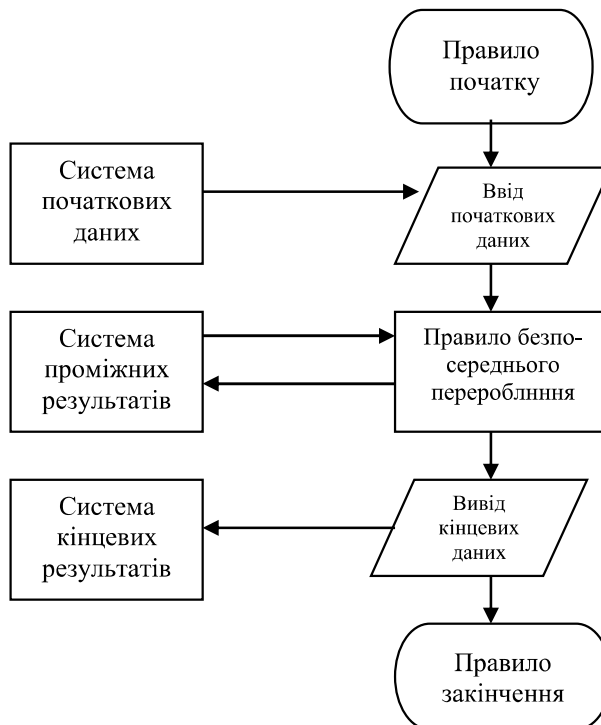


Рис. 1

Зміна будь-якого параметру алгоритму змінює часову складність та інші характеристики. Зміна параметру алгоритму з метою мінімізації часової складності алгоритму називається *параметричною оптимізацією алгоритму*.

До способів мінімізації часової складності відносяться:

1. Зміна правила початку визначає
  - вибір черговості використання даних в процесі обчислень
  - векторизація,
  - конкурентизація, тощо.
2. Зміна системи вхідних даних, наприклад, 10-вої, 16 річної тощо
3. Зміна системи проміжних результатів наприклад, використання двійкової системи,

4. Зміна правила вводу даних:

- генерування,
- читання,
- інкапсуляція

5. Зміна правил безпосереднього перероблення:

- розбиття масивів вхідних, вихідних даних проміжних,
- еквівалентні перетворення,
- апроксимація,
- використання попередніх обчислень

**Визначення** Параметрична модель алгоритму це сімка параметрів алгоритму об'єднаних зв'язками, які задають послідовність операцій виконання задачі  $\langle A, Q, q_0, q_f, I, O, P \rangle$ ,

де:

$A$  – множина символів зовнішнього алфавіту.  $A$  охоплює множини символів систем проміжних і кінцевих результатів,

$Q$  – множина символів алфавіту станів

$q_0, q_f$  – початковий та кінцевий стани роботи моделі алгоритму;  $q_0, q_f \in Q$

$I, O$  – операції вводу та виводу даних

$P$  – правило безпосереднього перероблення

Правило безпосереднього перероблення може бути задано деякою функцією, словесно, таблицею, графом, блок-схемою, тощо.

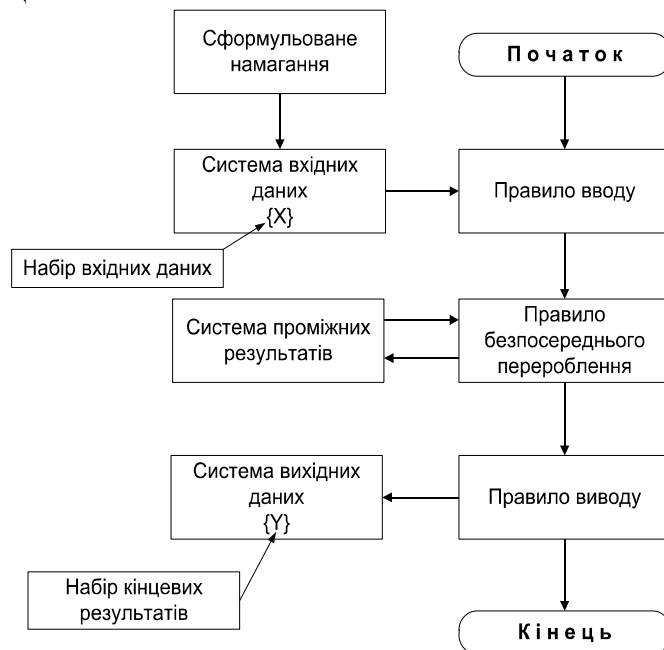


Рис.2

На Рис 2 зображено блок-схема параметричної моделі пари задача-алгоритм. Від блок-схеми алгоритму вона відрізняється доданим блоком "сформульоване намігання". Крім того в системі вхідних даних та системі кінцевих результатів виділений набір вхідних даних  $\{X\}$  та набір кінцевих результатів  $\{Y\}$ , які належать безпосередньо до задачі, яка розв'язується

## Асимптотичні співвідношення

Для опису швидкості зростання функцій використовується  $O$ -символіка. Функція  $f(n)$  має порядок зростання  $O(g(n))$ , якщо існують додатні константи  $C$  і  $n_0$  такі, що:

$$f(n) \leq C \cdot g(n), \quad \text{для } n > n_0.$$

Позначемо функцію яка виражає залежність часової складності від кількості вхідних даних ( $n$ ) через  $L(n)$ . Тоді, наприклад, коли говорять, що часова складність  $L(n)$  алгоритму має порядок(ступінь) зростання  $O(n^2)$  (читається як "О велике від  $n$  в квадраті", або просто як "о від  $n$  в квадраті", то вважається, що існують додатні константи  $c$  і  $n_0$  такі, що для всіх  $n$ , більших або рівних  $n_0$ , виконується нерівність  $L(n) \leq cn^2$ .

Наприклад, функція  $L(n) = 3n^3 + 2n^2$  має порядок зростання  $O(n^3)$ . Нехай  $n_0 = 0$  і  $c = 5$ . Очевидно, що для всіх цілих  $n \geq 0$  виконується нерівність  $3n^3 + 2n^2 \leq 5n^3$ .

Коли кажуть, що  $L(n)$  має ступінь зростання  $O(f(n))$ , то вважається, що  $f(n)$  є верхньою границею швидкості зростання  $L(n)$ . Щоби вказати нижню границю швидкості зростання  $L(n)$  використовують позначення  $\Omega(g(n))$ , що означає існування такої константи  $c$ , що для нескінченної кількості значень  $n$  виконується нерівність  $L(n) \geq c \cdot g(n)$ .

Теоретичне визначення порядку зростання функції є складною математичною задачею. На практиці визначення порядку зростання є задачею, що цілком вирішується за допомогою кількох базових принципів. Існують три правила для визначення складності:

1.  $O(c \cdot f(n)) = O(f(n))$

2.  $O(f(n) + g(n)) = O(\max(f(n), g(n)))$

3.  $O(f(n) \cdot g(n)) = O(f(n)) \cdot O(g(n))$

Перше правило декларує, що постійні множники не мають значення для визначення порядку зростання.

Друге правило називається "**Правило сум**". Це правило використовується для послідовних програмних фрагментів з циклами та розгалуженнями. Порядок зростання скінченної послідовності програмних фрагментів (без врахування констант) дорівнює порядку зростання фрагменту з найбільшою часовою складністю. Якщо алгоритм складається з двох фрагментів, функції часових складностей яких  $L_1(n)$  і  $L_2(n)$  мають ступені зростання  $O(f(n))$  і  $O(g(n))$  відповідно, то алгоритм має ступінь зростання  $O(\max(f(n), g(n)))$ .

Третє правило називається "**Правило добутків**". Якщо  $L_1(n)$  і  $L_2(n)$  мають ступені зростання  $O(f(n))$  і  $O(g(n))$  відповідно, то добуток  $L_1(n) \cdot L_2(n)$  має ступінь зростання  $O(f(n)g(n))$ . Прикладом може бути фрагмент програми "цикл в циклі".

## Приклад.

Задані функції часової складності  $L(n)$  для чотирьох алгоритмів:

1.  $L_1(n) = n\sqrt{n}$     2.  $L_2(n) = 2^n + n$     3.  $L_3(n) = 3n^2 + 2n^3$     4.  $L_4(n) = n + \log_2 n$

Використавши правило сум і правило добутків знайдемо  $O(n)$  :

$$O_1(n) = n\sqrt{n} \quad O_2(n) = 2^n \quad O_3(n) = n^3 \quad O_4(n) = n$$

Розташуємо функції  $O_i(n)$  у порядку зростання:

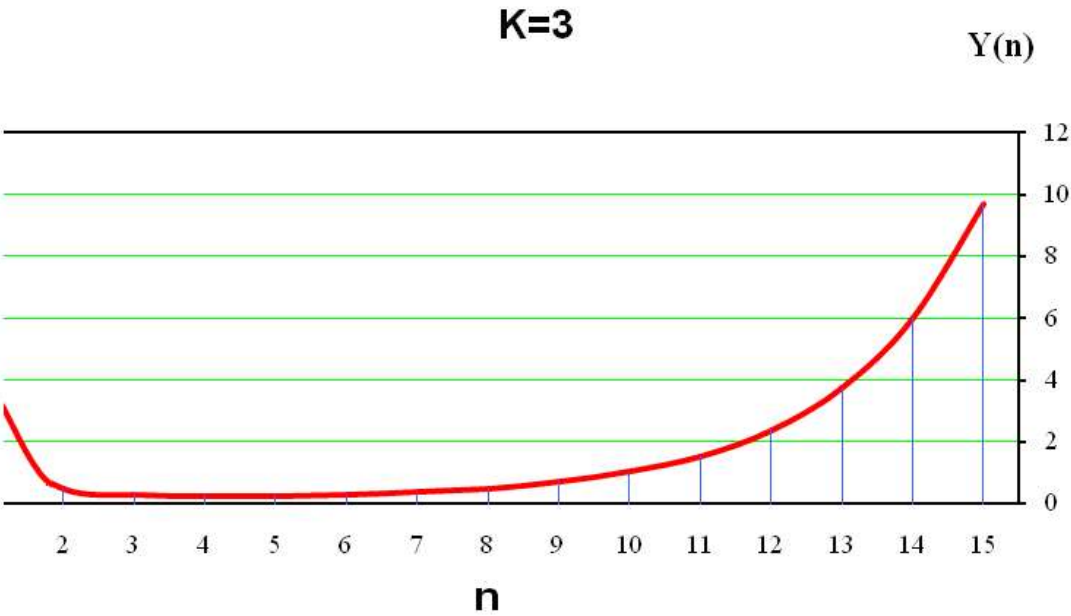
1.  $O_4(n) = n$     2.  $O_1(n) = n\sqrt{n}$     3.  $O_3(n) = n^3$     4.  $O_2(n) = 2^n$

Функція  $O_2(n) = 2^n$  має найбільший ступінь зростання.

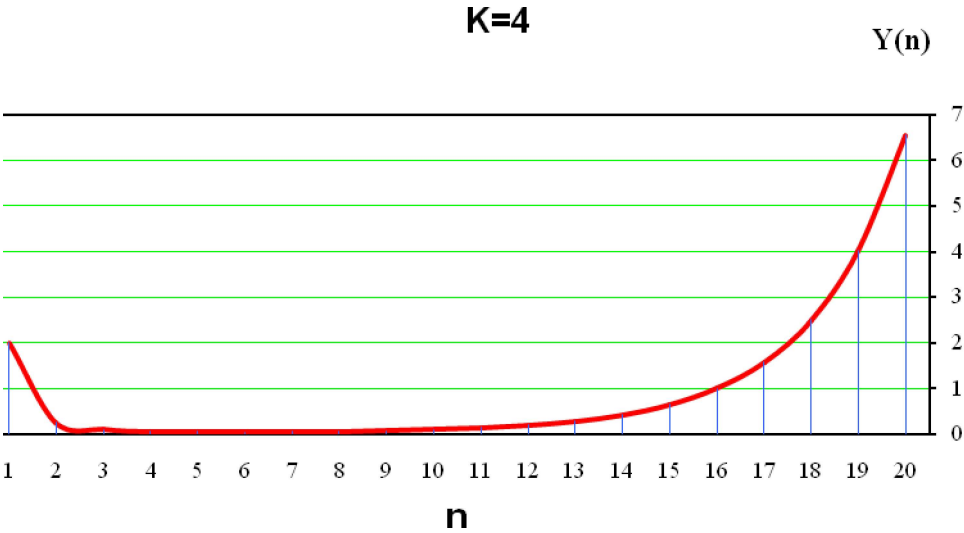
Побудуємо графіки  $Y(n) = \frac{O_2(n)}{P_k(n)}$  для  $n = (1, 2, \dots, 10)$ ;  $k = 3, 4, 5$

Для спрощення будемо вважати що поліном для відповідних значень  $K$  буде прирівнюватися до  $n^3$ ,  $n^4$  та  $n^5$ , оскільки ці значення є тою адитивною складовою в поліномі, яка найшвидше зростає.

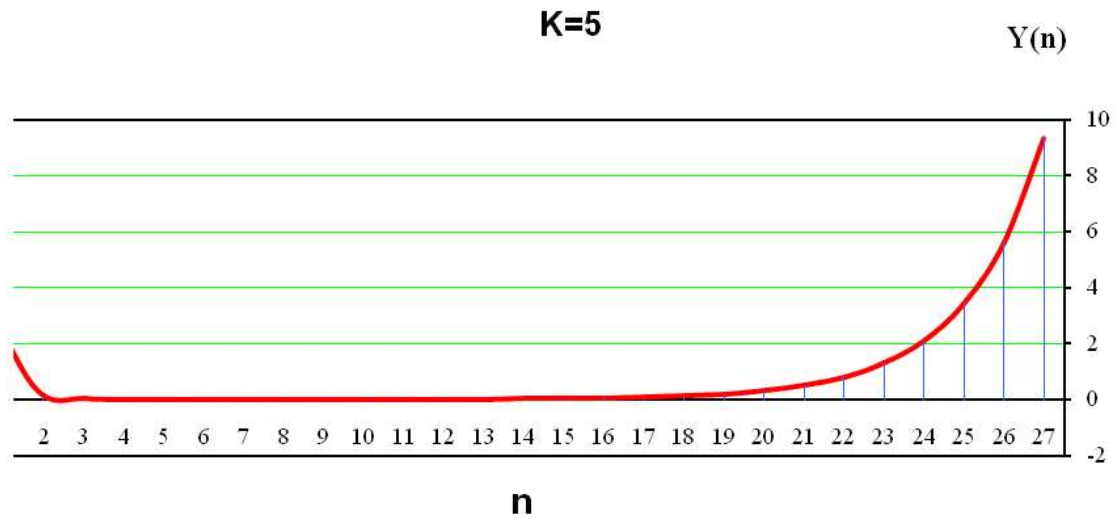
$$Y(n) = \frac{2^n}{n^3} :$$



$$Y(n) = \frac{2^n}{n^4} :$$



$$Y(n) = \frac{2^n}{n^5} :$$



Графіки показують, що існують такі значення  $n_0$  (при зростанні  $K$  значення  $n_0$  теж зростає), починаючи з яких значення функції порядку зростання часової складності буде приймати більші значення ніж значення відповідного поліному. Це ілюструє приналежність алгоритму до класу алгоритмів з експоненціальною складністю.