Практичне заняття № 4
(за темою лабораторної роботи №2)

Параметри алгоритму. Правило безпосереднього переоблення. 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 **τ** = RC

Характеристики алгоритму визначаються наступними параметрами Рис. 1 :

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

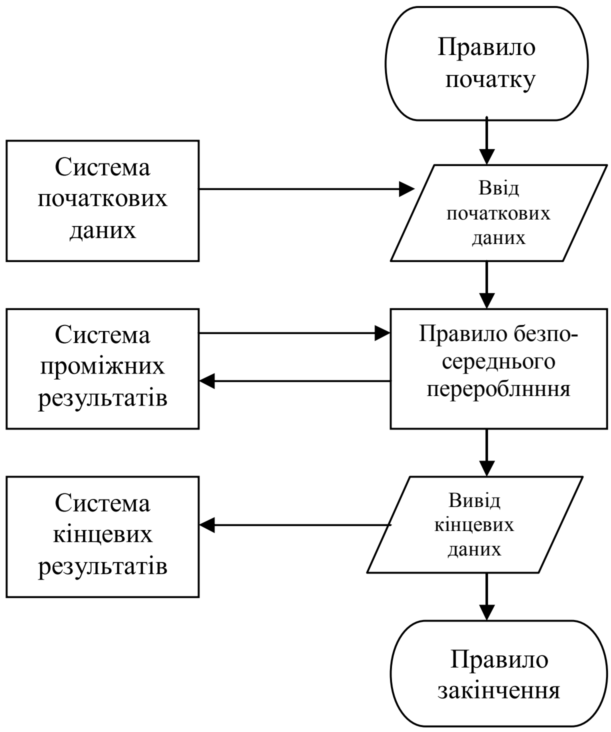


Рис. 1

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 переороблення:
- 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* $< A, Q, q_0, q_f, I, O, P >$,

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 переороблення

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

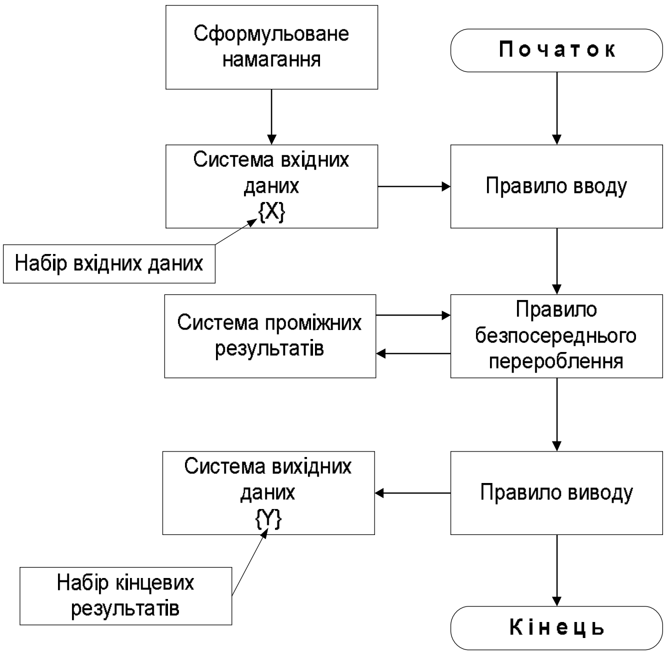


Рис.2

На Рис 2 зображено блок-схема параметричної моделі пари задача-алгоритм. Від блок-схеми алгоритму вона відрізняється доданим блоком "сформульоване намагання". 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 $\mathbf{n}_0$ такі, що:

$$f(n) <= C*g(n), \quad \text{для } n > n_0.$$

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(степінь) зростання $O(n^2)$ (читається як "O велике від **n** в квадраті", або просто як "о від **n** в квадраті", то вважається, що існують додатні константи **c** і $\mathbf{n_0}$ такі, що для всіх **n**, більших або рівних $\mathbf{n_0}$, виконується нерівність $\mathbf{L(n)<=cn^2}$.

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

Коли кажуть, що **L(n)** має степінь 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 **Ω(g(n))**, що означає існування такої константи **c**, що для нескінченої кількості значень **n** виконується нерівність **L(n)>=c*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. $\mathbf{O(c* f(n))=O(f(n))}$
2. $\mathbf{O\big(f(n) + g(n)\big) = O\big(max(f(n), g(n))\big)}$
3. $\mathbf{O\big(f(n) * g(n)\big) = O(f(n)) * O(g(n))}$

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 скінченої послідовності програмних фрагментів (без врахування констант) дорівнює порядку зростання фрагменту з найбільшою часовою складністю. Якщо алгоритм складається з двох фрагментів, функції часових складностей яких $\mathbf{L_1(n)}$ і $\mathbf{L_2(n)}$ мають ступені зростання **O(f(n))** і **O(g(n))** відповідно, то алгоритм має степінь зростання **O(max(f(n),g(n))).**

Третє правило називається ***"Правило добутків"***. Якщо $\mathbf{L_1(n)}$ і $\mathbf{L_2(n)}$ мають ступені зростання **O(f(n))** і **O(g(n))** відповідно, то добуток $\mathbf{L_1(n)\ L_2(n)}$ має степінь 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) :

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

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. $O_2(n) = 2^n$

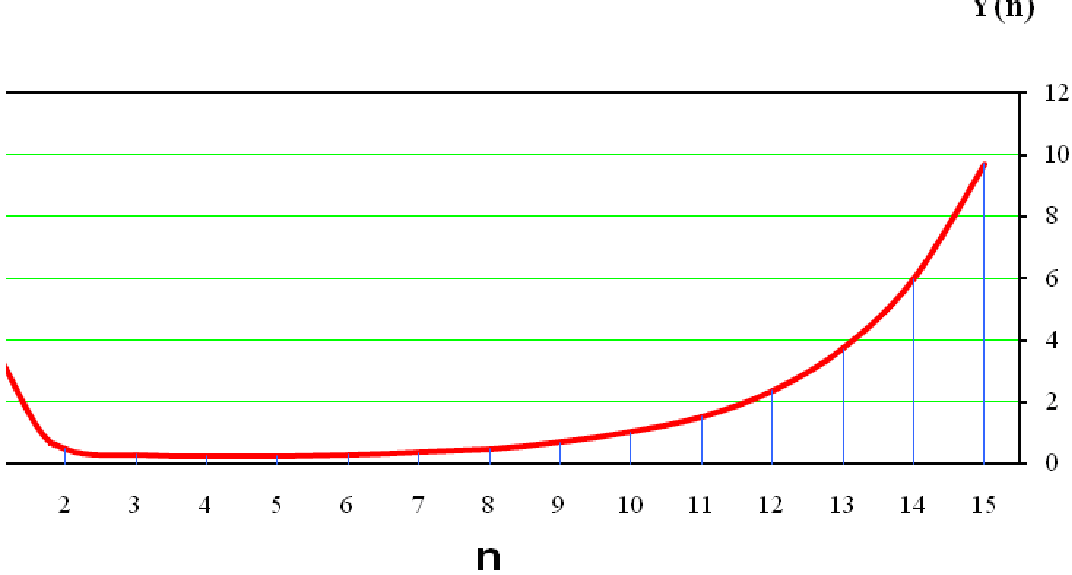
Функція $O_2(n) = 2^n$ має найбільший степінь зростання.

Побудуємо графіки $Y(n) = \dfrac{O_2(n)}{P_k(n)}$ для n = (1,2,…,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.

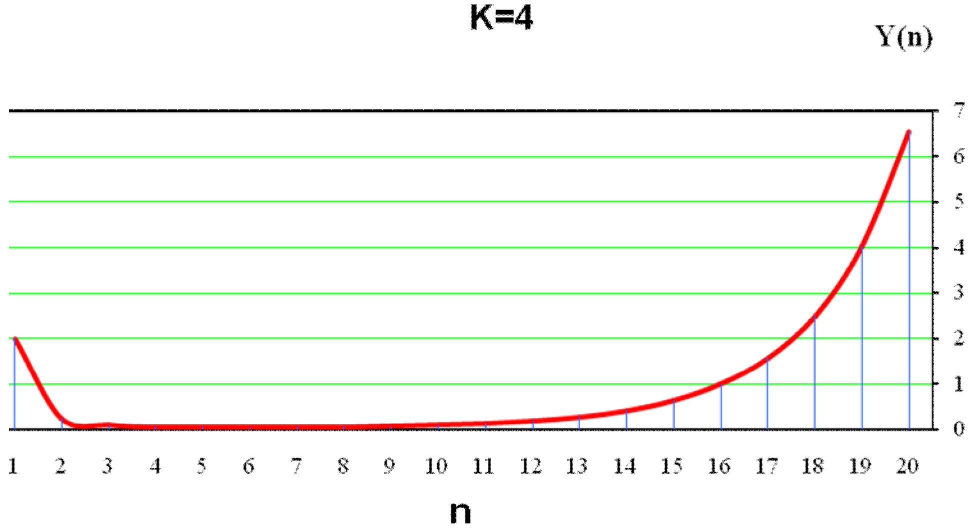
$$Y(n) = \frac{2^n}{n^3}:$$

K=3

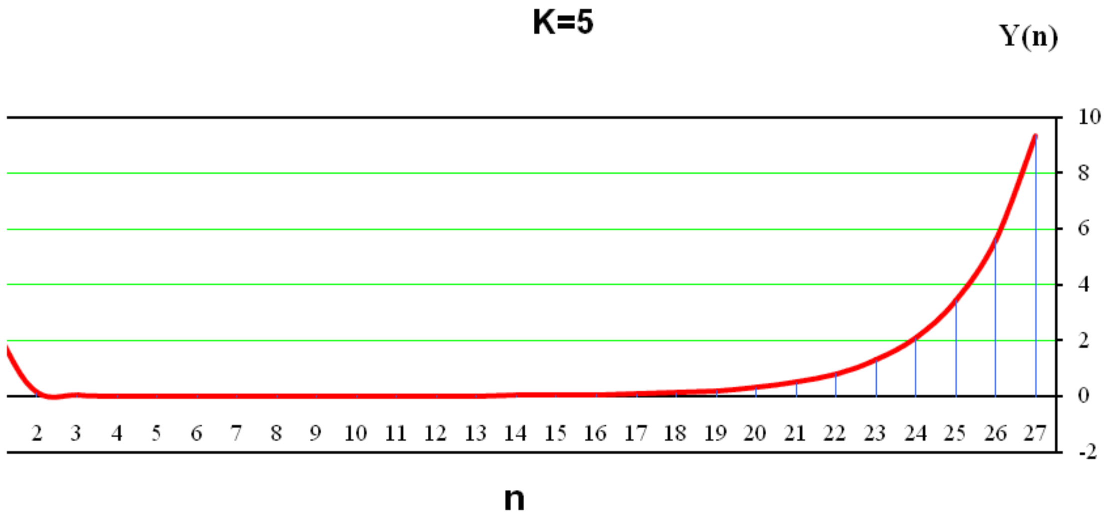


$$Y(n) = \frac{2^n}{n^4} :$$

K=4



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



Графіки показують, що існують такі значення $\mathbf{n}_0$ (при зростанні **K** значення $\mathbf{n}_0$ 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.