МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

КРЕМЕНЧУЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ МИХАЙЛА ОСТРОГРАДСЬКОГО

Кафедра комп’ютерної інженерії та електроніки

Виконала: Групи

Пугачова Д. В. КІ-23-1

Кременчук 2024

Практична робота №1

**Тема**. Асимптотична складність алгоритмів. 𝐎-нотація

**Мета:** набути практичних навичок у розв’язанні задач на оцінку асимптотичної складності алгоритмів у 𝑂.

Завдання

№5

fg(n)=n² O(n)=?

f(n)≤C×(n)

f(n)= n²

g(n)= n²

C=1 n0=1 n≥1

f\*(n)= n²≤1× n²=C×g(n)

f(n)=O(n²)

№15

f(n)=50n³-25

g(n)= n³

f(n)=O(g(n))

f(n)=50n³-25≤50n³ n≥1

f(n)≤50n³

g(n)= n³

f(n)≤50g(n) n≥1

C=50 n0=1

f(n)≤50g(n)

f(n)=O(g(n))

f(n)=50n³-25 є O(n³)

**Контрольні питання**

1. Що таке асимптотична складність алгоритму?

***Асимптотична складність алгоритму*** *— це спосіб оцінки того, як змінюється час виконання або потреби в пам'яті алгоритму з ростом розміру вхідних даних. Вона дозволяє визначити ефективність алгоритму в умовах великих значень nnn (розміру вхідних даних), незалежно від конкретних деталей реалізації або апаратного забезпечення.*

*Асимптотична складність зазвичай виражається в* ***O-нотації*** *(від "Order" — порядок), яка описує найгіршу поведінку алгоритму при великих значеннях вхідних даних.*

1. Яким чином визначається 𝑂-нотація і яка її сутність?

***О-нотація*** *(Big-O notation) — це математичний спосіб опису асимптотичної складності алгоритму, який вказує на верхню межу часу виконання або кількості операцій, необхідних для виконання алгоритму у найгіршому випадку в залежності від розміру входу.*

*Сутність О-нотації полягає в тому, що вона дозволяє оцінити, як змінюється поведінка алгоритму при збільшенні розміру вхідних даних. Вона використовується для:*

1. ***Оцінки швидкодії****: О-нотація дає уявлення про те, як швидко або повільно буде виконуватись алгоритм у залежності від розміру вхідних даних.*
2. ***Визначення ефективності****: З її допомогою можна порівнювати різні алгоритми за їх ефективністю, особливо коли мова йде про великі обсяги даних.*
3. Які основні правила використання 𝑂-нотації при аналізі алгоритмів?

*Основні правила використання* ***О-нотації*** *при аналізі алгоритмів:*

1. ***Ігнорування констант та незначних доданків****: О-нотація зберігає лише найбільш значущі члени складності, ігноруючи множники та дрібні доданки. Наприклад, O(3n2+5n)O(3n^2 + 5n)O(3n2+5n) стає O(n2)O(n^2)O(n2).*
2. ***Аналіз у найгіршому випадку****: О-нотація зазвичай описує складність алгоритму в найгіршому випадку, коли час виконання максимальний.*
3. ***Асимптотична оцінка****: О-нотація описує, як змінюється час виконання при великих розмірах вхідних даних, без точного вимірювання часу.*
4. ***Множення при вкладених циклах****: Якщо алгоритм має вкладені цикли, їх складності множаться. Наприклад, O(n)O(n)O(n) та O(m)O(m)O(m) дають O(n⋅m)O(n \cdot m)O(n⋅m).*
5. ***Аналіз рекурсії****: Для рекурсивних алгоритмів використовуються рівняння для визначення складності, наприклад, T(n)=2T(n/2)+O(n)T(n) = 2T(n/2) + O(n)T(n)=2T(n/2)+O(n).*
6. ***Ігнорування дрібниць****: Малі доданки (наприклад, +5+5+5) або множники (наприклад, 3n3n3n) не враховуються.*
7. ***Порівняння складностей****: Алгоритми з меншою О-нотацією (наприклад, O(nlog⁡n)O(n \log n)O(nlogn) замість O(n2)O(n^2)O(n2)) ефективніші на великих даних.*

*О-нотація допомагає швидко оцінити ефективність алгоритмів, зосереджуючись на найбільш важливих факторах.*

1. Що означають вирази 𝑂(1), 𝑂(𝑛), 𝑂(𝑛 2 ) в контексті асимптотичної складності?

** ***O(1)*** *— Постійний час: Час виконання не залежить від розміру входу.*

* ***Приклад****: Доступ до елемента масиву за індексом.*

** ***O(n)*** *— Лінійний час: Час виконання пропорційний розміру входу.*

* ***Приклад****: Лінійний пошук в масиві.*

** ***O(n²)*** *— Квадратичний час: Час виконання пропорційний квадрату розміру входу.*

* ***Приклад****: Бульбашкове сортування.*

1. Яким чином визначити асимптотичну складність алгоритму за його кодом або математичним виразом?

** ***Цикли****:*

* *Один цикл O(n)O(n)O(n).*
* *Вкладені цикли: складність множиться. Наприклад, два вкладені цикли O(n2)O(n^2)O(n2).*

** ***Рекурсія****: Використовуємо рекурсивне рівняння. Наприклад, T(n)=2T(n/2)+O(n)T(n) = 2T(n/2) + O(n)T(n)=2T(n/2)+O(n) дає O(nlog⁡n)O(n \log n)O(nlogn).*

** ***Послідовні операції****: Складність сумується, наприклад, O(n)+O(n)=O(n)O(n) + O(n) = O(n)O(n)+O(n)=O(n).*

** ***Математичні вирази****: Беремо найбільший член. Наприклад, 3n2+5n3n^2 + 5n3n2+5n — це O(n2)O(n^2)O(n2).*

** ***Ігноруємо константи****: O(3n)O(3n)O(3n) скорочується до O(n)O(n)O(n).*