МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

КРЕМЕНЧУЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ МИХАЙЛА ОСТРОГРАДСЬКОГО

Навчалньно-науковий інститут електричної інженерії та інформаційних технологій

Кафедра автоматизації та інформаційних систем

**ПРАКТИЧНА РОБОТА**

Виконав: студент групи КІ-23-1

Зеленський Богдан Сергійович

Перевірив:

Сидоренко Валерій Миколайович

м. Кременчук

2024 рік

**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**

**7.** Довести, що 𝑓(𝑛) = 3𝑛3 + 5 = 𝑂(𝑛3).

**12.** Задано функції 𝑓(𝑛) = 200𝑛3 + 15 і 𝑔(𝑛) = 𝑛3. Доведіть, що 𝑓(𝑛) = 𝑂(𝑔(𝑛)).

**Розв’язання**

**7.** 𝑓(𝑛) = 3𝑛3 + 5 = 𝑂(𝑛3)

𝑓(𝑛) = 3𝑛3 + 5 ≤ 3𝑛3 + 𝑛3 = 4𝑛3, n ≥ 1

𝑓(𝑛) = 3𝑛3 + 5 ≤ 4𝑛3, c = 4 i n0 = 1

**12.** 𝑓(𝑛) = 200𝑛3 + 15 = 𝑂(𝑔(𝑛)), де 𝑔(𝑛) = 𝑛3

𝑓(𝑛) = 200𝑛3 + 15

𝑔(𝑛) = 𝑛3

𝑓(𝑛) = 200𝑛3 + 15 ≤ 200𝑛3 + 𝑛3 = 201𝑛3, 𝑛 ≥ 1

200𝑛3 + 15 ≤ 201𝑛3, 𝑛0 = 1

𝑓(𝑛) = 200𝑛3 + 15 = 𝑂(𝑛3), с = 201, 𝑛0 = 1

**Контрольні питання**

1. Що таке асимптотична складність алгоритму?

Асимптотична складність алгоритму — це математичне поняття, яке описує, як змінюється час виконання або споживання ресурсів алгоритмом у залежності від розміру вхідних даних, коли цей розмір наближається до нескінченності. Іншими словами, це спосіб оцінки ефективності алгоритму при великих обсягах вхідних даних.

Основні види асимптотичної складності виражаються за допомогою нотацій:

* O (велике O) — верхня межа складності. Описує найгірший випадок, коли час виконання алгоритму не перевищить зазначену оцінку.
* Ω (велике Омега) — нижня межа складності. Описує найкращий випадок, коли алгоритм не може виконуватись швидше за цю межу.
* Θ (тета) — точна оцінка складності. Показує, що алгоритм працює приблизно з однаковою швидкістю як у найкращому, так і в найгіршому випадках.

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

O-нотація (велике O) — це стандартна нотація для визначення верхньої межі асимптотичної складності алгоритму. Її сутність полягає в тому, щоб описати, як швидко зростає кількість операцій або ресурсів, які потрібні алгоритму для виконання задачі, залежно від розміру вхідних даних nnn. O-нотація фокусується на найгіршому сценарії, тобто на тому, наскільки повільним може бути алгоритм при зростанні розміру вхідних даних.

Як визначається O-нотація:

* вибір домінуючого терміна: O-нотація відображає лише найбільший за швидкістю зростання термін у функції, яка описує кількість операцій або час виконання. Незначні члени або постійні множники відкидаються, оскільки при великих nnn вони стають несуттєвими;
* ігнорування констант: у великих O-нотаціях константи ігноруються, оскільки вони не мають суттєвого впливу на зростання часу виконання при великих обсягах даних;
* асимптотична поведінка: визначення O-нотації орієнтоване на аналіз поведінки алгоритму для дуже великих nnn. Навіть якщо для малих значень nnn поведінка може бути іншою, важливо, як змінюється час виконання при збільшенні обсягу вхідних даних.

Формальне визначення O-нотації:

Функція 𝑇 (𝑛) є 𝑂(𝑔(𝑛)), якщо існують константи 𝐶 > 0 і n0 ​ > 0, такі що для всіх 𝑛 ≥ 𝑛0 ​ виконується нерівність:

T(𝑛) ≤ C⋅𝑔(𝑛)

Це означає, що після деякої точки 𝑛0 функція T(𝑛) не перевищує множину 𝑔(𝑛), помножену на деяку константу C.

Сутність O-нотації:

* вона дозволяє порівнювати алгоритми за ефективністю незалежно від специфічних деталей їхньої реалізації чи обчислювальних потужностей;
* O-нотація зосереджена на масштабованості алгоритмів, тобто на тому, як вони справляються зі збільшенням розміру вхідних даних;
* вона допомагає оцінити найгірший випадок і визначити, наскільки ефективно працюватиме алгоритм на великих обсягах даних.

1. Які основні правила використання 𝑂-нотації при аналізі алгоритмів?

Основні правила використання O-нотації при аналізі алгоритмів:

* ігнорування констант: O-нотація відкидає константи;
* вибір домінуючого терміна: зберігається лише найшвидше зростаючий термін;
* аналіз найгіршого випадку: O-нотація описує верхню межу, тобто як алгоритм працює в найгіршому сценарії;
* розклад на етапи: якщо алгоритм складається з кількох етапів, сумарна складність визначається за найвищою складністю серед цих етапів;
* послідовні та вкладені цикли: послідовні цикли додаються, вкладені — перемножуються.

1. Що означають вирази 𝑂(1), 𝑂(𝑛), 𝑂(𝑛2) в контексті асимптотичної складності?

Вирази O(1), O(n) і O(n2) використовуються для опису асимптотичної складності алгоритмів. Вони означають:

* O(1) — константна складність: алгоритм виконує фіксовану кількість операцій, незалежно від розміру вхідних даних;
* O(n) — лінійна складність: час виконання алгоритму зростає пропорційно до розміру вхідних даних n;
* O(n2) — квадратична складність: час виконання алгоритму зростає як квадрат розміру вхідних даних nnn. Зазвичай це трапляється у випадках, коли є вкладені цикли, які проходять через всі елементи.

У контексті:

* O(1) вказує на швидкість і ефективність алгоритму для малих наборів даних;
* O(n) вказує на лінійне зростання ресурсів з збільшенням обсягу даних;
* O(n2) сигналізує про те, що алгоритм може стати повільним і неефективним для великих наборів даних через квадратичне зростання часу виконання.

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

Визначення асимптотичної складності алгоритму за його кодом або математичним виразом може бути здійснено за допомогою кількох кроків:

1. Аналіз структури алгоритму:

* цикли;
* умовні оператори;

1. Визначення кількості операцій;
2. Ігнорування менш значущих термінів:

* При складанні загальної формули часу виконання алгоритму зосереджуватись на найбільш значущих термінах;

1. Константи:

* Відкинути константи, оскільки вони не впливають на асимптотичну поведінку;

1. Математичні вирази;
2. Систематичний підрахунок;
3. Приклади:

* прямий перебір;
* бінарний пошук.