МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КРЕМЕНЧУЦЬКИЙ НАЦІОНАЛЬНИЙ
УНІВЕРСИТЕТ ІМЕНІ МИХАЙЛА
ОСТРОГРАДСЬКОГО

Кафедра комп’ютерної інженерії та електроніки

ЗВІТ З ПРАКТИЧНОЇ РОБОТИ №1

з навчальної дисципліни
«Алгоритми та методи обчислень»

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

Студент гр. КІ-24-1, Варакута О.О
Викладач, Сидоренко В. М

Кременчук 2025

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

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

Короткі теоретичні відомості:

Припустимо, що ми написали алгоритм і перед нами стоїть задача оцінки його ефективності [41]. Коли справа доходить до аналізу ефективності алгоритму, ми зазвичай пишемо функцію, скажімо $f(n)$, яка представляє кількість часу, кількість кроків або кількість простору, необхідного для вирішення проблеми розміру n. У цьому випадку нас не цікавить точне значення f для певного n, але нас цікавить, як швидко зростає $f(n)$ зі збільшенням n. Іншими словами, ми хочемо знатишвидкість зростання функції f. Для опису зростання функцій із збільшенням розміру входу, використовуються так звані асимптотичні нотації. Вони описують граничнуповедінку функції, коли її аргумент прагне до нескінченності: $n \to \infty$.Перша важлива з нотацій це так звана велика О-нотація (велика літера O,а не нуль), яке також називають символом Ландау. Буква O використовується тому, що швидкість зростання функції також може називатися її порядком. На прикладі часу виконання алгоритму велике O використовується як асимптотична верхня межа часу виконання.

Правила спрощення Щоб виразити швидкість зростання функції за допомогою великого O, можна застосувати такі правила:

Правило 1. Якщо функція f представлена у вигляді суми кількох членів, то член, який росте швидше за інші, визначатиме порядок f. Іншими словами, коли аргумент прагне до нескінченності, член із найвищим темпом зростання починає домінувати, роблячи незначним внесок решти членів у темп зростання всієї функції.

Правило 2. Фактори, які не залежать від аргументу функції, або просто константи, можна опускати. Тобто ми не пишемо $O(4n\ 3\ )$ попри те, що це не

порушує визначення. Замість цього ми пишемо $O(n\ 3\ )$. Мета полягає у тому, щоби вираз для $g(n)$ максимально спростити.

Задачі для самостійного розв'язання

Виконати індивідуальне завдання. Завдання полягає у розв'язанні двох задач, які потрібно вибрати зі списку, наведеного нижче. Правило вибору номерів наступний: n, n + 5, де n – номер студента в списку групи. У разі, якщо було досягнуто кінця списку задач, потрібно циклічно повернутися на його початок.

Завдання

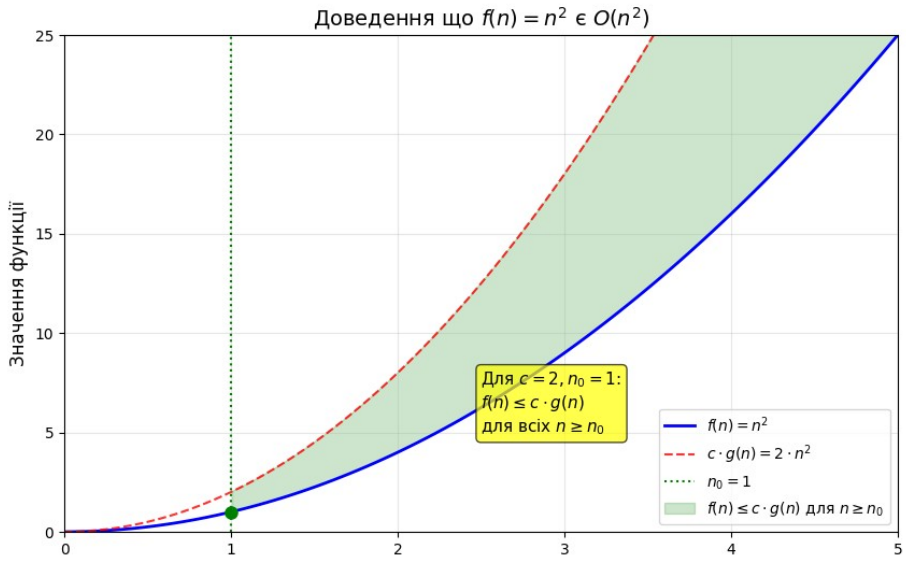
5. Дано функцію $fg(n) = n\ 2$. Знайти асимптотичну складність у O-нотації.

Оскільки $f(n)=n^2$, то вона зростає за квадратичним законом.

Для будь-яких $c>1, i\, n_0=1$ виконується:

$f(n)=n^2\le C\cdot n^2, \forall\, n\ge n_0$

$f(n)=O(n^2)$



10. Дано функції $f(n)$ = 150n 2 + 11 та $g(n)$ = n 2 . Покажіть, що $f(n)$ = $O(g(n))$.

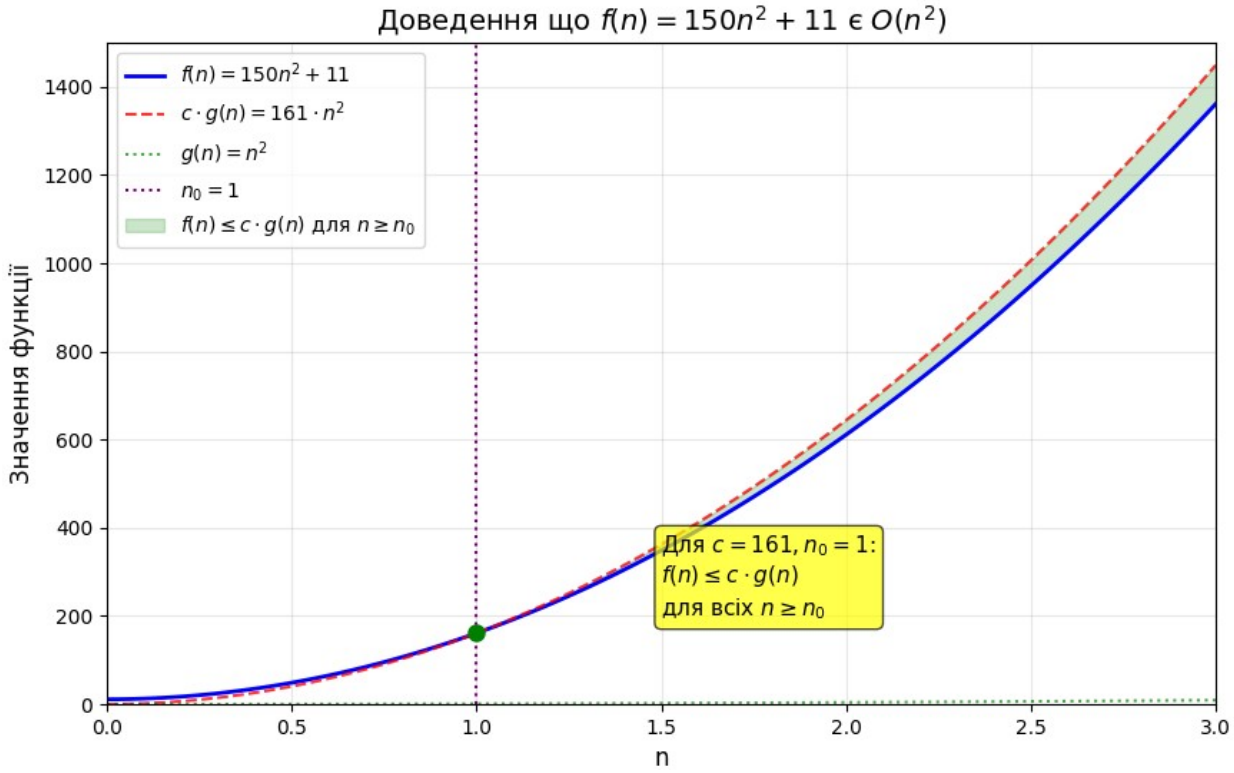
$150\,n^2+11\le C\cdot n^2, \forall\, n\ge n_0$

Нехай n_0 = 1:

$150\,n^2+11\le 161\,n^2\, при\, n\ge 1$

Отже, c = 161 n_0 = 1

$f(n)=O(n^2)$



Контрольні питання:

1. Що таке асимптотична складність алгоритму?

Асимптотична складність — це міра ефективності алгоритму, яка описує, як його час виконання, або обсяг пам'яті зростають при збільшенні розміру вхідних даних *n* до нескінченності.

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

О-нотація визначає асимптотичну верхню межу швидкості зростання функції.

Математичне визначення:

$f(n)=O(g(n))$якщо $c>o, n_0>0: \forall n \geq n_0, f(n) \leq c \cdot g(n)$

Сутність: Описує найгірший випадок роботи алгоритму, Дає гарантію, що алгоритм не буде працювати гірше за певну межу ,Не є точною оцінкою — це верхня межа

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

Правило найшвидшого члена :з усіх доданків вибирається той, що найшвидше зростає.

Відкидання констант: Множники-константи ігноруються

.Сума функцій: $O(f(n))(g(n))=O\left(max(f(n), g(n))\right)$

Добуток функцій:$O(f(n)) \cdot O(g(n))=0(f(n) \cdot g(n))$

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

$O(1)$ — Постійна складність: час виконання не залежить від розміру вхідних даних. Наприклад : доступ до елементу масиву за індексом, арифметичні операції

$O(n)$ —Лінійна складність: час виконання пропорційний розміру вхідних даних. Наприклад: пошук в невідсортованому масиві, обхід списку

$O(n^2)$ Квадратична складність: час виконання пропорційний квадрату розміру вхідних даних. Наприклад : сортування вибором, бульбашкове сортування, два вкладені цикли.

5. Яким чином визначити асимптотичну складність алгоритму за його

кодом або математичним виразом?

Проаналізувати структуру алгоритму — визначити, які частини виконуються послідовно, які — у циклах або рекурсивно.

Порахувати кількість базових операцій (порівнянь, присвоєнь, обчислень тощо) у залежності від розміру вхідних даних nnn.

Встановити функціональну залежність — записати кількість операцій у вигляді функції f(n).

Залишити тільки домінуючий член (той, що найшвидше зростає при збільшенні n), відкинувши константи та менш значущі доданки.

Записати оцінку у нотації “O”