МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

КРЕМЕНЧУЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ МИХАЙЛА ОСТРОГРАДСЬКОГО

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ЕЛЕКТРИЧНОЇ ІНЖЕНЕРІЇ
ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА АВТОМАТИЗАЦІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ

НАВЧАЛЬНА ДИСЦИПЛІНА
«**АЛГОРИТМИ І СТРУКТУРИ ДАНИХ**»

ЗВІТ
З ПРАКТИЧНОЇ РОБОТИ №3

Виконав:
студент групи КН-24-1
Соломка Б. О.

Перевірила:
доцент кафедри АІС
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*.

Хід роботи

1. Теоретичні відомості

O-нотація використовується для опису асимптотичної верхньої межі швидкості зростання функції. Формально, якщо функції f та g відображають натуральні числа у простір дійсних чисел, то f(n) = O(g(n)), якщо існують константи c > 0 та n_0, такі що $f(n) \leq c \cdot g(n)$ для всіх $n \geq n_0$.

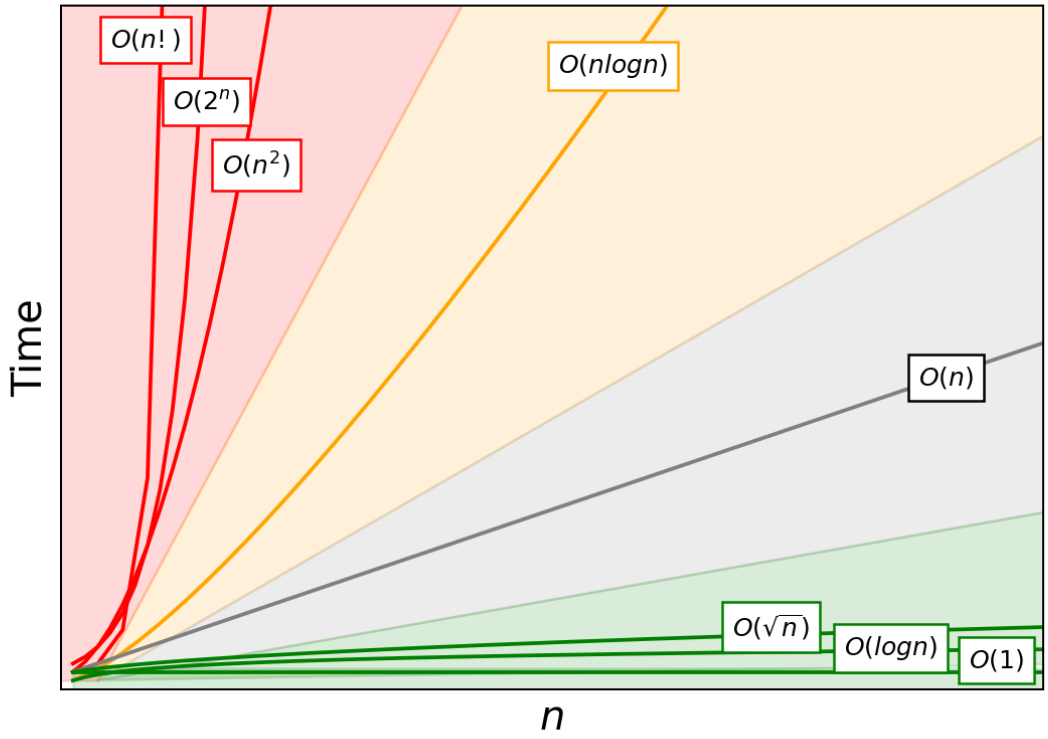

Рисунок 1 – O-нотація

Для спрощення O-нотації використовуються наступні правила:

Якщо функція представлена у вигляді суми кількох членів, то член, який

росте швидше за інші, визначає порядок функції.

Константні множники можна опускати при визначенні O-нотації.

2. Розв'язання завдань

Відповідно до індивідуального завдання, необхідно розв'язати дві задачі.

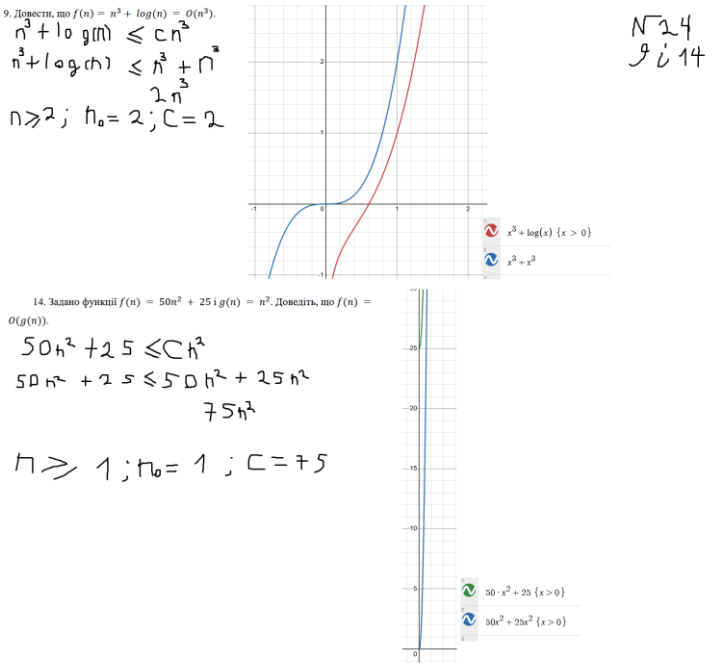

Рисунок 2 – Хід розв’язання вправ

Задача 9. Довести, що $f(n) = n^3 + \log(n) = O(n^3)$.

Розв'язання: За визначенням O-нотації, треба довести, що існують константи $c > 0$ та n_0, такі що $f(n) \leq c \cdot n^3$ для всіх $n \geq n_0$.

Маємо:

$f(n) = n^3 + \log(n)$

Відомо, що для достатньо великого n, $\log(n) < n$. Зокрема, для $n \geq 2$, $\log(n) < n$. Також, для $n \geq 1$, $n < n^3$ (оскільки $n^2 > 1$).

Тому для $n \geq 2$ справедливим є твердження: $\log(n) < n < n^3$

Отже: $n^3 + \log(n) < n^3 + n^3 = 2n^3$

Таким чином, можемо взяти $c = 2$ та $n_0 = 2$, і умова $n^3 + \log(n) \leq 2n^3$ буде виконуватись для всіх $n \geq 2$.

Отже, $f(n) = n^3 + \log(n) = O(n^3)$.

Задача 14. Задано функції $f(n) = 50n^2 + 25$ і $g(n) = n^2$. Доведіть, що $f(n) = O(g(n))$.

Розв'язання: За визначенням О-нотації, треба довести, що існують константи $c > 0$ та n_0, такі що $f(n) \leq c \cdot g(n)$ для всіх $n \geq n_0$.

Маємо:

$f(n) = 50n^2 + 25$ і $g(n) = n^2$.

Для $n \geq 1$ справедливим є: $25 \leq 25n^2$ (оскільки $n^2 \geq 1$ при $n \geq 1$)

Тому: $50n^2 + 25 \leq 50n^2 + 25n^2 = 75n^2$

Отже, можемо взяти $c = 75$ та $n_0 = 1$, і умова $50n^2 + 25 \leq 75n^2$ буде виконуватись для всіх $n \geq 1$.

Таким чином, $f(n) = 50n^2 + 25 = O(g(n))$, де $g(n) = n^2$.

Висновки

В ході виконання практичної роботи я:

Ознайомився з поняттям асимптотичної складності алгоритмів та О-нотацією, що використовується для оцінки верхньої межі швидкості зростання функцій.

Вивчив основні правила спрощення при роботі з О-нотацією:

Член функції з найвищим темпом зростання визначає порядок всієї функції.

Константні множники можна опускати.

Набув практичних навичок у розв'язанні задач на оцінку асимптотичної складності:

Навчився знаходити асимптотичну складність функцій у О-нотації.

Навчився доводити, що функція належить до певного класу складності

за визначенням О-нотації.

Зрозумів практичне значення О-нотації для аналізу ефективності алгоритмів, що дозволяє оцінювати, як швидко зростає час або простір, необхідний для роботи алгоритму при збільшенні розміру вхідних даних.

Отримані знання та навички будуть корисними для оцінки та порівняння ефективності різних алгоритмів при вирішенні практичних задач програмування та розробки програмного забезпечення.