

# **Звіт**

з практичної роботи № 1

з дисципліни: “Алгоритми і структури даних”

**“Асимптотична складність алгоритмів. О-нотація ”**

Виконав:

Ст. гр. КІ-24-1

Огоновський О.Є.

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

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.

**1. Дано функцію  $f(n) = 5n^2 + 1$ . Знайти асимптотичну складність у  $O$ -нотації.**

Домінантний доданок при великих  $n$  – це  $5n^2$ .

Доданок  $+1$  стає несуттєвим.

Ігноруємо константу 5.

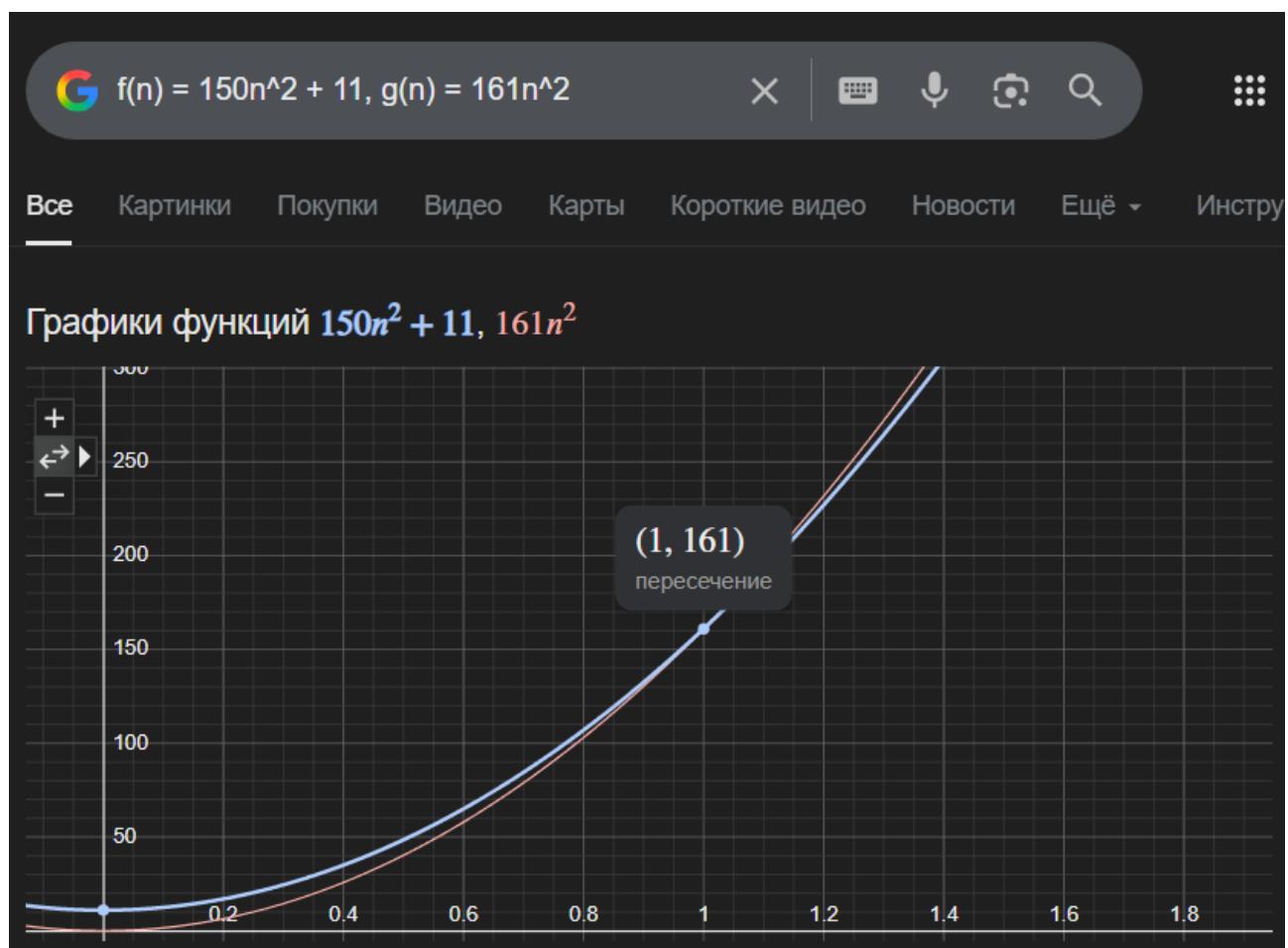
$$f(n) = O(n^2)$$

**6. Довести, що  $f(n) = 150n^2 + 11 = O(n^2)$**

Згідно з визначенням,  $f(n) = 150n^2 + 11 = O(n^2)$  якщо існують дві константи  $c$  і  $n_0$  такі, що

$$f(n) = 150n^2 + 11 \leq c \cdot n^2 \text{ для } \forall n \geq n_0$$

$$161n^2$$



Таким чином,  $c = 161$  та  $n_0 = 1$  задовільняє визначення і, відповідно,  $f(n) = O(n^2)$ .

## Контрольні питання

### **1. Що таке асимптотична складність алгоритму?**

Асимптотична складність — це спосіб оцінки ефективності алгоритму шляхом аналізу того, як час виконання (часова складність) або обсяг пам'яті (просторова складність) змінюються зі збільшенням розміру вхідних даних.

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

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

Сутність: О показує верхню межу зростання алгоритму, тобто найгірший порядок складності.

### **3. Основні правила використання O-нотації**

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 нескінченості, 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  $O(n^3)$ .

### **4. Що означають вирази $O(1)$ , $O(n)$ , $O(n^2)$**

$O(1)$  - константна складність: час не залежить від розміру вхідних даних (наприклад, доступ до елемента масиву).

$O(n)$  - лінійна складність: час зростає пропорційно  $n$  (наприклад, один цикл по масиву).

$O(n^2)$  - квадратична складність: час зростає пропорційно  $n^2$  (наприклад, подвійний вкладений цикл).

### **5. Як визначити асимптотичну складність алгоритму за кодом або математичним виразом?**

За кодом:

Перевірити кількість і вкладеність циклів.

Оцінити виклики функцій.

Скласти вираз для кількості операцій.

Спрощувати до домінантного члена.

За формулою:

Взяти вираз для кількості операцій  $f(n)$ .

Викинути константи й «менші» доданки.

Записати результат у Big-O.