О. П. Кривцова
Полтавський національний педагогічний університет імені В. Г. Короленка
м. Полтава
op-k@ukr.net

АСПЕКТИ ПІДГОТОВКИ МАЙБУТНІХ ФАХІВЦІВ У ГАЛУЗІ КОМП'ЮТЕРНИХ НАУК

На сучасному етапі підготовки майбутніх фахівців у галузі комп'ютерних наук важливим стає формування загальних та спеціальних компетентностей. Серед основних є знання та розуміння основ алгоритмізації та програмування, методів розробки програмного забезпечення з використанням сучасних технологій.

Важливим фактором підготовки майбутніх фахівців у галузі комп'ютерних наук є готовність до співбесіди та можливість майбутнього працевлаштування за спеціальністю.

Вивчаючи дисципліну «Алгоритми та структури даних» одним із важливих питань стає оцінка складності алгоритму. Тому необхідно звернути увагу на достатнє оволодіння базовими навичками розробки ефективних алгоритмів.

Як оцінити роботи програму з точки зору часу? Який алгоритм швидший та ефективніший?

Проблема оцінки складності алгоритму була пов'язана з тим, що ми не можемо оцінити швидкість його роботи з точки зору часу. Точний час виконання програми залежить від процесора, типу даних, мови програмування таі інших параметрів. Одним з перших цю проблему вирішив Дональд Кнут. Кожен алгоритм складається з визначеного числа кроків (*тактів процесора за які виконується алгоритм*) і кількість кроків не залежить від апаратних характеристик. Це і може стати характеристикою оцінки складності алгоритму.

Складність алгоритмів зазвичай оцінюють:

- за часом виконання;
- по використовуванні пам'яті.

Оцінка за часом виконання передбачає, яку кількість операцій необхідно виконати алгоритму, щоб отримати шуканий результат. Оцінка по використаній пам'яті передбачає, яку кількість додаткової пам'яті необхідно алгоритму, щоб отримати шуканий результат [1].

Для визначення складності алгоритмів використовують концепцію **big O**, яка описує відношення кількості операцій від кількості вхідних даних. Використання великої літери «**O**» (*або так звана О-нотація*) прийшло з математики, де її застосовують для порівняння асимптотичної поведінки функцій (*при прагненні розміру вхідних даних до нескінченності*). У рамках комп'ютерної науки **big O** показує верхню межу залежності між вхідними параметрами функції та кількістю операцій, які виконує процесор [3].

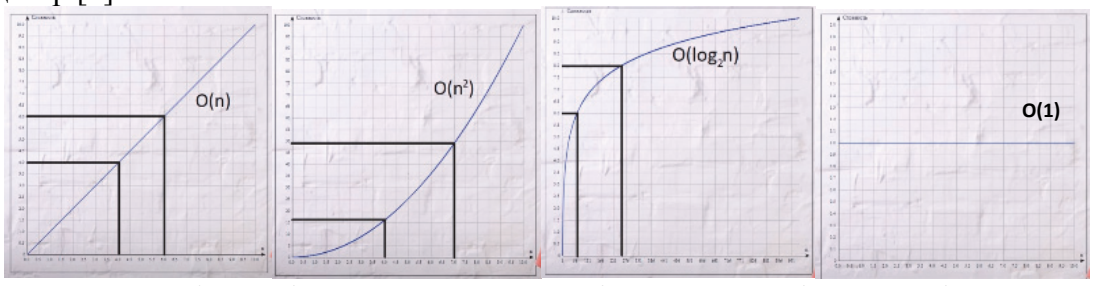

Рис. 1. Графіки залежностей для оцінки складності алгоритмів

Розглянемо графіки залежностей для оцінки складності алгоритмів (рис.1):

Лінійна залежність O(n). Пропорційний ріст кількості елементів до складності. Якщо кількість елементів збільшилось на 100 то і кількість операцій збільшилася на 100. Прикладом такого алгоритму є пошук суми елементі масиву або найбільшого елемента у масиві довільних чисел.

Квадратична залежність $O(n^2)$. Графік у вигляді гілки параболи. Значний ріст складності при незначному збільшенні кількості елементів. Якщо кількість елементів збільшилось на 1000 то і кількість операцій збільшилася на 1млн. Прикладом такого алгоритму є знаходження максимального елементу двовимірного масиву (**n*n= n^2 операцій**).

Логарифмічна залежність $O(\log_2 n)$. Найкраща залежність яку бажано досягти. Повільний ріст складності, коли навіть при значному збільшенні кількості елементів, складність збільшується не суттєво [2]. Прикладом такого алгоритму є пошук елемента у впорядкованому масиві коли ми використовуємо бінарний пошук. Тобто кожний раз будемо зменшувати масив, що переглядаємо вдвічі. Це зменшить кількість операцій, які необхідно виконати.

Залежність O(1). Складність не залежить від кількості вхідних даних. Прикладом такого алгоритму є будь яка арифметична операція, яка не залежить від кількості елементів.

Розглядаючи складність алгоритму по часу важливо бачити тенденцію росту складності алгоритму від збільшення кількості вхідних даних. Якщо кількість вхідних даних прямує до нескінченності то використання констант та цілих коефіцієнтів значення не має. Значення має лише максимальний степінь для **n**.

Аналогічно проводять оцінку і по пам'яті, коли це важливо. Однак алгоритми можуть використовувати значно більше пам'яті при збільшенні розміру вхідних даних, ніж інші, але працюватимуть швидше і навпаки. Це допомагає вибирати оптимальні шляхи вирішення завдань виходячи з поточних умов і вимог.

Отже, можемо відзначити важливість розуміння даної теми для розробки ефективних алгоритмів. Концепцію **big O** необхідно розуміти, щоб вміти бачити та виправляти неоптимальний код. Жоден важливий проект, або співбесіда не може обійтись без питань про дану концепцію. Нерозуміння даної теми може привести до значної втрати продуктивності алгоритмів.

Література

1. Ахо А. Структуры данных и алгоритмы / А. Ахо, Д. Хопкрофт, Д.Ульман. – М.: Издательский дом «Вильямс», 2000. – 384 с.
2. Оцінка складності алгоритмів, або Що таке O (log n) [Електронний ресурс] – Режим доступу: http://echo.lviv.ua/dev/53
3. Часова складність алгоритму [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Часова_складність_алгоритму

Анотація. Кривцова О.П. Аспекти підготовки майбутніх фахівців у галузі комп'ютерних наук. *Підготовка майбутніх фахівців у галузі комп'ютерних наук до визначення складності алгоритмів використовуючи концепцію big O.*

Ключові слова: *комп'ютерні науки, оцінка складності алгоритму, концепція big O.*

Summary. Kryvtsova Olena. Aspects of training future specialists in computer science. *Training future IT professionals to determine the complexity of algorithms using the big O concept.*

Key words: *computer science, algorithm complexity estimation, big O concept.*

Аннотация. Кривцова Е.П. Аспекты подготовки будущих специалистов в области компьютерных наук. *Подготовка будущих специалистов в области компьютерных наук к определению сложности алгоритмов используя концепцию big O.*

Ключевые слова: *компьютерные науки, оценка сложности алгоритма, концепция big O.*