

## Практична робота №1

**Тема:** Ефективність алгоритмів.

**Мета:** навчитися оцінювати час виконання завдання різними алгоритмами при роботі з великими обсягами даних та визначати ефективний.

### Теоретичні відомості

Припустимо, швидкодія комп'ютеру та об'єм його пам'яті можна збільшувати до нескінченності. Чи була би тоді необхідність у вивченні алгоритмів? Так, але тільки для того, щоб продемонструвати, що метод розв'язку має скінченний час роботи і що він дає правильну відповідь. Якщо б комп'ютери були необмежено швидкими, підійшов би довільний коректний метод рішення задачі. Звісно, тоді найчастіше обирався би метод, який найлегше реалізувати.

Сьогодні є дуже потужні комп'ютери, але їх швидкодія не є нескінченно великою, як і пам'ять. Таким чином, час обчислення – це такий само обмежений ресурс, як і об'єм необхідної пам'яті. Цими ресурсами слід користуватись розумно, чому й сприяє застосування алгоритмів, які ефективні в плані використання ресурсів часу та пам'яті.

Алгоритми, які розроблені для розв'язання однієї та тієї самої задачі, часто можуть дуже сильно відрізнятись за ефективністю. Ці відмінності можуть бути набагато більше помітними, чим ті, які викликані застосуванням різного апаратного та програмного забезпечення.

Перший алгоритм (сортування включенням) для своєї роботи вимагає часу, кількість якого оцінюється як  $c_1 n^2$ , де  $n$  – розмір вхідних даних (кількість елементів у послідовності для сортування),  $c_1$  – деяка стала. Цей вираз вказує на те, як залежить час роботи алгоритму від об'єму вхідних даних. У випадку сортування включенням ця залежність є квадратичною. Другий алгоритм (сортування злиттям) потребує часу, кількість якого оцінюється як  $c_2 n \log 2^n$ . Зазвичай константа сортування включенням менше константи сортування злиттям, тобто  $c_1 < c_2$ , але ці константи не відіграють ролі у порівнянні швидкодії різних алгоритмів. Адже зрозуміло, що функція  $n^2$  зростає швидше зі збільшенням  $n$ , аніж функція  $n \log 2^n$ . І для деякого значення  $n = n_0$  буде досягнуто такий момент, коли вплив різниці констант перестане мати значення і надалі функція  $c_2 n \log 2^n$  буде менша за  $c_1 n^2$  для будь-яких  $n > n_0$ .

Для демонстрації цього розглянемо два комп'ютери – А та Б.

Комп'ютер А більш швидкий і на ньому працює алгоритм сортування, а комп'ютер Б більш повільний і на ньому працює алгоритм сортування методом злиття. Обидва комп'ютери повинні виконати сортування множини, яка складається з мільйона чисел. Припустимо, що комп'ютер А виконує мільярд операцій в секунду, а комп'ютер Б – лише десять мільйонів, тобто А працює в 100 разів швидше за Б. Щоб різниця стала більш відчутною, припустимо, що код методу включення написаний найкращим програмістом в світі із використанням команд процесору, і для сортування  $n$  чисел за цим алгоритмом потрібно виконати  $2n^2$  операцій (тобто  $c_1=2$ ). Сортування методом злиття на комп'ютері Б написано програмістом початківцем із використанням мови високого рівня і отриманий код потребує  $50n\log_2 n$  операцій (тобто  $c_2=50$ ). Таким чином для сортування мільйона чисел комп'ютеру А буде потрібно

$$\frac{2 \cdot (10^6)^2 \text{ команд}}{10^9 \text{ команд / с}} = 2000 \text{ с},$$

а комп'ютеру Б –

$$\frac{50 \cdot 10^6 \cdot \log_2 10^6 \text{ команд}}{10^7 \text{ команд / с}} \approx 100 \text{ с}.$$

Тож, використання коду, час роботи якого зростає повільніше, навіть при поганому комп'ютері та поганому компіляторі потребує на порядок менше процесорного часу! Для сортування 10 мільйонів чисел перевага сортування злиттям стає ще більш відчутною: якщо сортування включенням потребує для такої задачі приблизно 2,3 дня, то для сортування злиттям – менше 20 хвилин. Загальне правило таке: чим більша кількість елементів для сортування, тим помітніше перевага сортування злиттям.

Наведений вище приклад демонструє, що алгоритми, як і програмне забезпечення комп'ютеру, являють собою **технологію**. Загальна продуктивність системи настільки ж залежить від ефективності алгоритму, як і від потужності апаратних засобів.

### Практичне завдання

1. У таблиці 1 рядки відповідають різним функціям  $f(n)$ , а стовпці – значенням часу  $t$ . Заповніть таблицю максимальними значеннями  $n$ , для яких задача може бути вирішена за час якщо передбачається, що час роботи алгоритму, необхідний для вирішення завдання, дорівнює  $f(n)$  мікросекунд.

2. Порівняти час роботи алгоритмів та зробити висновок, яка функція часу

виконання алгоритму є самою ефективною.

Таблиця1 – Порівняння часу роботу алгоритму

	Секунда	Хвилина	Година	День	Місяць	Рік	Століття
$\lg n$							
$\sqrt{n}$							
$n$							
$n \cdot \lg n$							
$n^2$							
$n^3$							
$2^n$							
$n!$							

3. Оформити звіт за виконанням практичної роботи.

#### Література

1. Кормен, Томас Х. и др. Алгоритмы: построение и анализ: Пер. с англ. М. : ООО "И. Д. Вильямс" 2013. 1328 с.
2. Спрингер Вильям. Гид по Computer Science для каждого программиста. СПб.: Питер, 2020. 192 с.
3. Рафгарден Тим. Совершенный алгоритм. Основы. СПб.: Питер, 2019. 256 с.
4. Седжвик Р. Алгоритмы на C++. Фундаментальные алгоритмы и структуры данных. М. : ИД "Вильямс", 2011. 1056 с.
5. Солтис М. Введение в анализ алгоритмов; пер. с англ. А. В. Логунова. М.: ДМК Пресс, 2019. 278 с.
6. Стивенс, Род. Алгоритмы. Теория и практическое применение. М.: Издательство «Э», 2016. 544 с.
7. Феррейра, Фило В. Теоретический минимум по Computer Science. Все, что нужно программисту и разработчику. СПб.: Питер, 2018. 224 с.