Водночас бінарний пошук ефективний у відсортованих наборах, проте його ефективність залежить від ступеня відсортованості даних. Під час порівняння з хеш-таблицями, які забезпечують швидкий доступ за ключем, виявлено, що вони ефективні для асоціативних масивів та операцій з великими обсягами даних. Такий аналіз підкреслює важливість вибору конкретного алгоритму у поєднанні зі структурою даних, зокрема враховуючи обсяг і характеристики даних для оптимального використання в конкретних умовах [4].

Отже, вибір алгоритму пошуку має критичне значення для оптимального використання ресурсів та швидкості обробки інформації. Аналіз лінійного пошуку, лінійного пошуку з бар'єром та бінарного пошуку в контексті різних структур даних підкреслює їх переваги та обмеження. Для великих обсягів даних бінарний пошук впритул до відсортованих масивів є ефективним, тоді як хеш-таблиці виявляються корисними для швидкого доступу за ключем.

**Список використаних джерел**

1. Tudor Octavian Pocola. The evolution of search algorithms over time. Delft University of Technology. URL: https://filelist.tudelft.nl/Websections/Honours%20Exhibition/Scientific%20 Writing/The_evolution_of_search_algorithms_over_time.pdf

2. Пошукові алгоритми. Алгоритми і структури даних: вебсайт. URL: http://elcat.pnpu.edu.ua/docs/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC%D0%B8%20%D1%96%20%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B8%20%D0%B4%D0%B0%D0%BD%D0%B8%D1%85/lab2_search.html

3. Поняття структур даних, масив, список, словник, стек, черга, хеш-таблиця. Структури даних: вебсайт. URL: https://ua5.org/struktury-danyh/1621-ponyattya-struktur-danyh-masyv-spysok-slovnyk-stek-cherga-hesh-tablyczya.html

4. Introduction to Algorithms. 3rd Edition – H. Thomas, E. Charles, L. Ronald, C. Stein. *MIT PRESS*, 2009. 1292 p.

**УДК 004.07**

*Диркач Х. М., здобувачка 2 курсу спеціальності 122 Комп'ютерні науки,*

*Потапова Н. А., канд. екон. наук, доцент, доцент кафедри інформаційних технологій*

## СОРТУВАННЯ ТА ЙОГО ВПЛИВ НА АРХІТЕКТУРУ ПАМ'ЯТІ

*Донецький національний університет імені Василя Стуса, м. Вінниця*

Алгоритм сортування визначається як алгоритм, що розміщує елементи якоїсь однотипної послідовності даних (масив, список, файл) у певному по-

рядку, який може бути або числовим порядком, або лексикографічним порядком, або будь-яким іншим заданим користувачем порядком.

Мета сортування – полегшити подальший пошук, оновлення, виключення, включення елементів у структуру даних. На відсортованих даних легше визначити, чи є пропущені елементи, чи всі елементи перевірені, легше знайти загальні елементи двох однотипних структур, злити їх воєдино. Сортування є важливим засобом для прискорення роботи практично будь-якого алгоритму, в якому потрібно часте звертання до певних елементів структури даних.

Ефективність алгоритму сортування може залежати від низки факторів:

- кількості сортованих елементів;
- діапазону і розподілу значень сортованих елементів;
- ступеня початкової відсортованості елементів;
- характеристик алгоритму (складність, вимоги до пам'яті тощо);
- місць розміщення елементів (в оперативній пам'яті (масив) або на диску (файл)).

Основними вимогами до алгоритмів сортування, як і до будь-яких алгоритмів, є вимоги до пам'яті і часу виконання. Існує безліч класичних алгоритмів сортування. Деякі з них не надто швидкі, проте мають невеликі вимоги до пам'яті, інші, навпаки, – дуже швидкі, проте потребують значних ресурсів пам'яті. Розглянемо кілька основних класичних алгоритмів сортування, що використовуються для сортування послідовностей чисел.

Бульбашкове сортування (також використовується термін сортування обміном, англ. *Bubble sort*) є найпростішим з алгоритмічного погляду, алгоритмом сортування. Його ідея полягає у тому, що здійснюється кілька проходів по списку, під час кожного з яких порівнюють пари сусідніх елементів. Якщо елементи стоять неправильно, вони міняються місцями. Кожен прохід по списку ставить наступне найбільше значення на його правильну позицію.

Алгоритм сортування вибором дуже подібний до бульбашкового сортування, проте є дещо оптимальнішим, оскільки за кожен прохід по списку відбувається лише одна операція перестановки елементів. Його ідея полягає у тому, що на кожному кроці відбувається лінійний пошук найбільшого елементу серед невідсортованої частини списку, який переставляється на відповідну позицію (крайню праву / ліву позицію невідсортованої частини списку).

Під час сортування вставкою, кожен наступний елемент вставляється у потрібну позицію так, щоб підсписок лишався відсортованим. Спочатку вважаємо, що підсписок з одного елементу (що знаходиться на нульовій позиції) відсортований. Далі кожен наступний елемент на кожному проході вставляється у відповідну позицію. Під час цього, може виникнути ситуація, коли для вставки необхідно зсунути частину елементів списку.

Сортування злиттям працює за таким принципом: невідсортовану вхідну множину довільно розбивають на підмножини. Потім ці підмножини сортують окремо цим самим методом і об'єднують в одну множину. У цьому методі діє відомий принцип «розділяй і володарюй».

Алгоритм швидкого сортування використовується для того, щоб отримати ті ж переваги у швидкості, що і сортування злиттям, не використовуючи під час цього додатку пам'ять. Однією з особливостей алгоритму є те, що він використовує значно меншу кількість порівнянь і перестановок елементів. Основна ідея полягає у виборі «опорного» елементу з масиву та розбитті масиву на два підмасиви: один, що містить елементи, менші за опорний, інший – більші.

Сортування впливає на архітектуру пам'яті через вимоги до часу та простору. Бульбашкове сортування та сортування вибором хоча й прості в реалізації, є менш ефективними на великих обсягах даних. Сортування вставкою підходить для відсортованих або малих масивів. Сортування злиттям та швидке сортування дають швидші результати, але вимагають більше ресурсів. Швидке сортування, з використанням стратегії розбиття на підмасиви з допомогою опорного елементу, характеризується швидкістю та меншими вимогами до пам'яті, тому є оптимальним у багатьох випадках.

Під час вибору алгоритму важливо брати до уваги конкретні умови використання та характеристики даних, щоб оптимізувати роботу програми і забезпечити оптимальне використання архітектури пам'яті.

## Список використаних джерел

1. Коротєєва Т. О. Алгоритми та структури даних: навч. посіб., Львів: Видавництво Львівської політехніки, 2014. 280 с.

2. Креневич А. П. Алгоритми і структури даних: підручник. Київ: ВПЦ Київський Університет, 2021. 200 с.

3. Грудзинський Ю. Є. Алгоритми та структури даних. Київ: КПІ ім. Ігоря Сікорського. 215 с. URL: https://ela.kpi.ua/bitstream/123456789/56538/1/Alhorytmy_ta_struktury%20danykh_Navch_posib.pdf