масиву. До того ж перестановки використовуються в алгоритмах, які оптимізують розташування даних у пам'яті комп'ютера.

У математичному моделюванні перестановки використовуються для аналізу різних систем і процесів. Наприклад, їх можна використовувати для вивчення маршрутизації, аналізу графів та інших математичних структур у комп'ютерних мережах.

До того ж аранжування можуть знайти застосування в теорії музики, для аналізу та створення музичних творів. Наприклад, порядок нот у мелодії або розташування музичних фраз можна відобразити аранжуваннями.

Усі ці приклади демонструють важливість і універсальність механізмів у різних галузях науки і техніки, що робить їх незамінними інструментами для аналізу, обробки та захисту даних, а також для вирішення різноманітних математичних і практичних задач. Отже, перестановки виявляються важливим та універсальним інструментом у багатьох галузях науки та технологій. Розуміння їх концепції та застосування дає змогу розв'язувати різноманітні математичні та практичні задачі, що робить їх незамінним елементом у сучасному світі.

Список використаних джерел

1. Васильків І. М. Основи теорії ймовірностей і математичної статистики: навч. посібник. Львів: ЛНУ імені Івана Франка, 2020.

2. Дороговцев А. Я. Математичний аналіз. Частина 1. Київ: Либідь, 1993.

3. Ferreiros J. Labyrinth of Thought: A History of Set Theory and Its Role in Modern Mathematics. Birkhäuser Basel, 2007.

УДК 519.642 004.424.5.021

Левченко М. Р., *здобувачка 2 курсу спеціальності 122 Комп'ютерні науки,*
Вєтров О. С., *старший викладач кафедри інформаційних технологій*

ІСТОРІЯ ТА РОЗВИТОК АЛГОРИТМУ ШВИДКОГО СОРТУВАННЯ

Донецький національний університет імені Василя Стуса, м. Вінниця

Швидке сортування (QuickSort) – це один із найпотужніших та найпоширеніших алгоритмів сортування, який має широке застосування у комп'ютерних науках та програмуванні. Саме його унікальна комбінація простоти реалізації та високої ефективності робить QuickSort об'єктом численних досліджень і вдосконалень з моменту його створення [1]. Алгоритм QuickSort, або ж qsort, був розроблений понад 40 років тому британським комп'ютерним вченим Тоні Хоаром у 1960 році під час його роботи в московському університеті. На той час основною задачею було створення алгоритму сортування, який був би ефективнішим за наявні методи. Хоар спочатку розробив QuickSort для сортування слів у словнику англійської мови, що стало основою для подальших досліджень та вдосконалень алгоритму [2].

Алгоритм працює за принципом «розділяй і володарюй» – потрібно розбивати масив і застосовувати той самий алгоритм до кожної частини, які поступово зменшуватимуться. Основна ідея алгоритму така:

- З масиву вибирається елемент, який називається опорним (pivot).
- Потім масив ділиться на дві частини: одна частина містить усі елементи, менші або рівні опорному елементу, а друга – всі елементи, більші за опорний елемент.
- Для кожного підмасиву, якщо в ньому більше двох елементів, рекурсивно виконується та сама процедура. Якщо в підмасиві два елементи, вони порівнюються між собою і, якщо потрібно, міняються місцями.
- Після виконання цих кроків ми отримаємо повністю відсортований масив [3].

Протягом років алгоритм зазнав численних вдосконалень:

- Вибір середнього елемента, медіани трьох або випадкового елемента для уникнення найгірших випадків.
- Комбінація QuickSort з іншими алгоритмами, як-от вставка (Insertion Sort), для покращення продуктивності на малих підмасивах.

Розробка багатопоточних реалізацій QuickSort застосовується для ефективного використання багатоядерних процесорів [1]. У цього методу, як і в будь-якого іншого методу сортування, є свої переваги та недоліки.

Перевагами QuickSort є:

- висока швидкодія на практиці. QuickSort є одним із найшвидших алгоритмів внутрішнього сортування: потребує лише O(n) пам’яті, для покращеної версії – O(1), у найкращому випадку складність становить $\Omega(n \log n)$;
- дає змогу розпаралелювання для сортування підмасивів;
- працює на пов’язаних списках.

Серед характерних недоліків можна зазначити нестійкість (не зберігає відносний порядок однакових елементів) та сильну деградацію обчислювальної складності за умови невдалих вхідних даних, $O(n^2)$.

Розглянемо приклад використання алгоритму QuickSort у мові програмування Python [1]. Цей код реалізує алгоритм швидкого сортування (QuickSort) для сортування списку чисел. Спочатку визначається опорний елемент (pivot), і масив розділяється так, що всі елементи, менші або рівні опорному, переміщаються ліворуч, а всі більші – праворуч. Потім функція QuickSort рекурсивно сортує підмасиви ліворуч і праворуч від опорного елемента. Основна програма ініціалізує список чисел, виводить його, виконує сортування за допомогою QuickSort і виводить відсортований список.

Підсумовуючи, можемо зазначити, що історія QuickSort демонструє, як інноваційні ідеї та постійне вдосконалення можуть зробити алгоритм ключовим інструментом у комп’ютерних науках. Завдяки своїй ефективності та гнучкості QuickSort залишається актуальним і незамінним для вирішення завдань сортування в різних галузях.

Список використаних джерел

1. Розбираємо швидке сортування. *EPAM. Campus.* 27.09.2021. URL: https://training.epam.ua/ua/blog/483 (дата звернення: 20.05.2024).

2. Python Program for QuickSort. *geeksforgeeks.org.* 28.08.2024. URL: https://www.geeksforgeeks.org/python-program-for-quicksort/ (дата звернення: 20.05.2024).

3. Швидке сортування. *Wikiwand.* URL: https://www.wikiwand.com/uk/Швидке_сортування (дата звернення: 20.05.2024).

УДК 004.41

Токар С. С., *здобувач 2 курсу спеціальності 122 Комп'ютерні науки,*
Вєтров О. С., *старший викладач кафедри інформаційних технологій*

РЕАЛІЗАЦІЯ МЕТОДУ ШУЛЬЦЕ МОВОЮ ПРОГРАМУВАННЯ JAVASCRIPT

Донецький національний університет імені Василя Стуса, м. Вінниця

Метод Шульце – це система голосування, розроблена Маркусом Шульце у 1997 році, яка обирає одного переможця за допомогою голосів, які виражають переваги. Метод також можна використовувати для створення відсортованого списку переможців [1, 2]. У цій роботі ми розглянемо імплементацію методу Шульце з використанням мови програмування JavaScript.

Цей метод базується на ідеї попарного порівняння кандидатів. У ньому визначається «сила» взаємного переважання кожного кандидата над іншими. З цієї інформації формується граф переваг, який використовується для визначення переможця.

Кожен бюлетень містить повний список усіх кандидатів. Кожен виборець ранжує цих кандидатів у порядку переваги. Окремий виборець може віддати однакову перевагу кільком кандидатам і залишити кандидатів без рейтингу. Якщо даний виборець не ранжує всіх кандидатів, то вважається, що цей виборець віддає перевагу всім кандидатам із рейтингом перед усіма кандидатами без рейтингу, і що цей виборець байдужий до всіх кандидатів без рейтингу.

Розглянемо кроки методу Шульце:

1. Створення матриці парних порівнянь. Кожен виборець виражає свої переваги щодо кандидатів. Ця інформація збирається і утворює матрицю, де кожен елемент [i] [j] відповідає кількості виборців, які віддали перевагу кандидату i над кандидатом j.

2. Побудова графу переваг. На основі матриці парних порівнянь будується граф, де ребра вказують на те, які кандидати мають більшість переваг над іншими.

3. Визначення найсильніших шляхів. З використанням алгоритму Флойда–Уоршелла знаходяться найсильніші шляхи між усіма парами кандидатів у графі.

4. Визначення переможця. Найкращий кандидат визначається як той, у якого немає конкурента, що має сильніший шлях відносно всіх інших кандидатів.

У мові програмування JavaScript ми можемо реалізувати цей метод так. Спочатку створюємо функцію для обчислення методу Шульце. Потім створюємо початкову структуру даних (матрицю) для зберігання найсильніших шляхів між кандидатами в методі Шульце на основі вхідних даних.