УДК 004.414:004.43:519.683:004.9

Діброва І. С., здобувач 2 курсу спеціальності 122 Комп'ютерні науки,

Вєтров О. С., страший викладач кафедри прикладної математики та кібербезпеки

СОРТУВАННЯ МАСИВІВ МЕТОДОМ БУЛЬБАШКИ

Донецький національний університет імені Василя Стуса, м. Вінниця

Сортування масиву означає розташування всіх елементів масиву в певному порядку. Сортування даних робить пошук у масивах ефективнішим як для людини, так і для комп'ютера. Сортування є фундаментальним поняттям в інформатиці та програмуванні. Цей процес організовує елементи в певному порядку, щоб полегшити їх подальше використання та аналіз. Ефективні методи сортування є важливим фактором у вирішенні різноманітних завдань, від оптимізації пошуку до підготовки даних до подальшої обробки.

У мовах програмування важливо мати доступ до ефективних алгоритмів сортування, щоб ефективно обробляти дані. Багато мов програмування надають стандартні бібліотеки сортування, які використовують різні алгоритми, як-то швидке сортування, сортування злиттям, і в нашому випадку, – сортування бульбашками [1].

Бульбашковий метод – це простий алгоритм сортування, який працює шляхом послідовного порівняння та обміну місцями сусідніх елементів, доки весь масив не буде вирівняно. Це алгоритм, який порівнює два сусідні елементи і міняє їх місцями, поки вони не опиняться в потрібному порядку. Подібно до того, як бульбашки у воді піднімаються на поверхню, кожен елемент масиву переміщується в кінець кожної ітерації. Хоча він не є найефективнішим для великих обсягів даних, він відображає основні принципи сортування і може бути використаний для розуміння базових концепцій сортування.

Загалом роботу алгоритму можна поділити на дві основні ітерації. Перша ітерація – порівняння та обмін. Друга – ітерації, що залишилися [2].

Порівняння розпочнеться з першого елементу, який необхідно порівняти з другим елементом. Якщо перший елемент більший за другий, треба поміняти їх місцями. Потім так само другий і третій елементи. Якщо вони розташовані в неправильному порядку, їх також треба поміняти місцями. Цей процес повторюється до останнього елементу.

Другий етап – цей самий процес триває для решти ітерацій. Після кожної ітерації найбільший елемент серед несортованих елементів розміщується в кінці.

У кожній ітерації порівняння відбувається до останнього невідсортованого елементу.

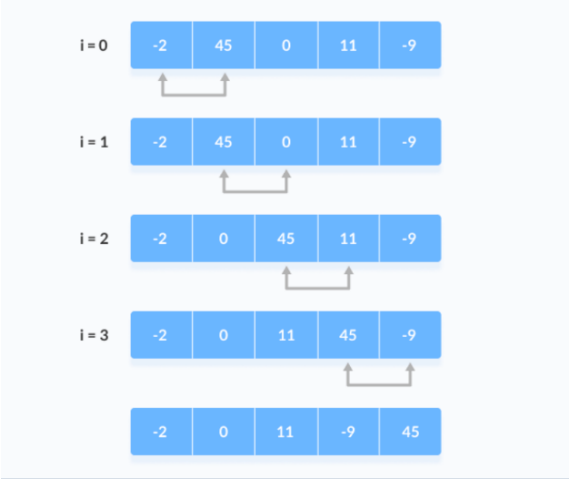

Рисунок 1. Перша ітерація. Порівняння суміжних елементів

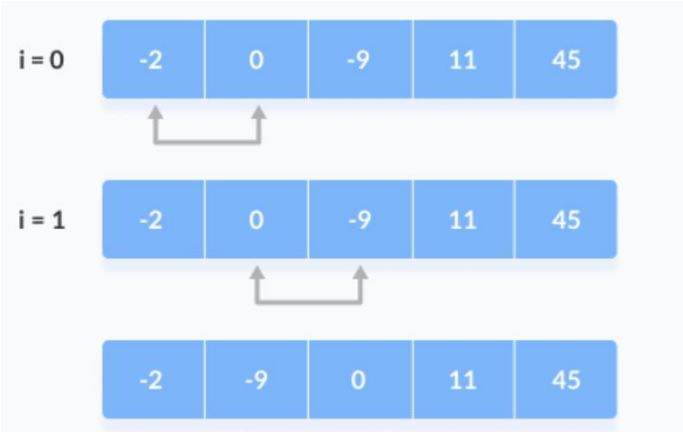

Рисунок 2. Порівняння сусідніх елементів

Масив можна вважати відсортованим, коли всі невідсортовані елементи розміщено на їх правильних позиціях.

Рисунок 3. Відсортований масив

На мові програмування Python лістинг програми для сортування масиву матиме такий вигляд:

```
# Сортування бульбашкою на мові Python
```

```
def bubbleSort(array):

  # Цикл для доступу до кожного елементу масиву
  for i in range(len(array)):

    # Цикл для порівняння елементів масиву
    for j in range(0, len(array) - i - 1):

      # Порівняння двох сусідніх елементів
      # Змініть > на <, щоб сортувати за спаданням
      if array[j] > array[j + 1]:

        # Обмін елементів, якщо вони не в потрібному порядку
        temp = array[j]
        array[j] = array[j + 1]
        array[j+1] = temp

data = [-2, 45, 0, 11, -9]

bubbleSort(data)

print('Відсортований масив за зростанням:')
print(data)
```

Ще одним важливим завданням є оптимізація алгоритму та пришвидшення роботи програми. У наведеному вище алгоритмі всі порівняння виконуються, навіть якщо масив уже відсортовано. Це збільшує час виконання.

Для вирішення цієї проблеми можна ввести додаткову змінну swapped. Якщо елементи міняються місцями, то значення swapped набуває значення true. В іншому випадку – значення False. Після ітерації значення swapped дорівнює false, якщо перестановки не відбулося. Це означає, що елементи вже відсортовано, і подальші ітерації не потрібні. Це зменшує час виконання і допомагає оптимізувати бульбашкове сортування [3].

```
# Оптимізоване сортування бульбашкою на мові Python

def bubbleSort(array):

  # цикл для проходження крізь кожен елемент масиву
  for i in range(len(array)):

    # відстежуємо, чи відбувся обмін
    swapped = False

    # цикл для порівняння елементів масиву
    for j in range(0, len(array) - i - 1):
```

```
        # порівнюємо два сусідні елементи
        # замініть > на <, щоб сортувати за спаданням
        if array[j] > array[j + 1]:

          # обмін відбувається, якщо елементи
          # не в потрібному порядку
          temp = array[j]
          array[j] = array[j+1]
          array[j+1] = temp

          swapped = True

      # якщо обмін не відбувся, це означає, що масив уже впорядкований
      # і немає потреби у подальших порівняннях
      if not swapped:
        break

data = [-2, 45, 0, 11, -9]

bubbleSort(data)

print('Відсортований масив за зростанням:')
print(data)
```

Підсумовуючи, варто зазначити, що природа сортування масивів є важливим етапом у програмуванні та інформатиці. Спрощуючи пошук та аналіз даних, сортування забезпечує ефективне використання та обробку інформації як для користувачів, так і для комп'ютерів. Алгоритми сортування є важливим аспектом вирішення різноманітних завдань, розширюючи можливості від оптимізації пошуку до підготовки даних до подальшої обробки. Хоча метод бульбашок не завжди є оптимальним для великих обсягів даних, він є гарним вступом до базових понять сортування і може бути використаний для навчання основам алгоритму.

Список використаних джерел

1. Кормен, Т., Лейзерсон, Ч., Ривест, Р., Штайн, К. (2019). Введення в алгоритми. Київ: К.І.С.

2. Sedgewick, R., Wayne, K. (2011). Algorithms, 4th edition. Addison-Wesley Professional.

3. Mühlegg, S., Scherer, S., Floreano, D. (2020). Path Planning and Control of UAVs using Model Predictive Control and Reinforcement Learning. Robotics and Autonomous Systems, 128, 103–628.