УДК 004.42

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

ПОРІВНЯЛЬНИЙ АНАЛІЗ ЕФЕКТИВНОСТІ АЛГОРИТМІВ СОРТУВАННЯ QUICK SORT, HEAP SORT, SHELL SORT

Донецький національний університет імені Василя Стуса, м. Вінниця

Сучасний світ інформаційних технологій невіддільний від обробки та аналізу даних. Алгоритми сортування є одними з основних інструментів у цьому процесі, які забезпечують ефективне управління даними у різних застосуваннях, від програмування до наукових досліджень. Серед численних алгоритмів сортування вирізняються Quick sort, heap sort та shell sort, які пропонують різні підходи до вирішення завдань сортування.

Метою цього дослідження є проведення порівняльного аналізу ефективності алгоритмів сортування Quick sort, heap sort та shell sort. Цей аналіз дасть змогу з'ясувати, який із цих алгоритмів краще справляється зі сортуванням різних типів даних за визначеними критеріями. Об'єктом дослідження є визначення особливостей кожного алгоритму та їх вплив на швидкодію й ефективність сортування.

Алгоритми сортування відіграють ключову роль у різних областях комп'ютерних наук і програмування, забезпечуючи ефективне управління даними.

Короткий опис та принцип дії трьох алгоритмів сортування: Quick sort, heap sort та shell sort.

1. Quick sort (Швидке сортування) є одним з найпопулярніших алгоритмів сортування. Він базується на принципі «розбиття та коригування». Принцип дії: алгоритм обирає певний елемент масиву, який називається опорним (pivot), та розбиває масив на два підмасиви: один з елементами менше опорного, а інший – з елементами більше опорного. Потім рекурсивно сортується кожен із цих підмасивів.

2. Heap sort (Сортування купою) базується на використанні структури даних, відомої як купа (heap). Алгоритм будує максимальну купу (для сортування у порядку зростання) або мінімальну купу (для сортування у порядку спадання), а потім послідовно видаляє корінь купи (найбільший або найменший елемент) і перебудовує купу [1].

3. Shell sort (Сортування Шелла) є розширенням алгоритму вставки (insertion sort) і належить до категорії алгоритмів інкрементного сортування. Алгоритм використовує послідовність «кроків», що визначають, як відбувається процес перегляду та перестановки елементів. Спочатку він застосовує звичайне сортування вставками для елементів, що знаходяться на відстані одного кроку один від одного, а потім зменшує цей крок і повторює процес доти, поки весь масив не буде відсортований [2].

Порівняємо три алгоритми сортування – Quick sort, Heap sort та Shell sort – у одній ситуації. Розглянемо ситуацію, коли потрібно відсортувати великий обсяг даних, наприклад, інформацію про транзакції великого банку.

Приклад порівняння трьох алгоритмів сортування на мові Python:

```
import random
import time
def quick_sort(arr):
    if len(arr) <= 1:
        return arr
    pivot = arr[len(arr) // 2]
    left = [x for x in arr if x < pivot]
    middle = [x for x in arr if x == pivot]
    right = [x for x in arr if x > pivot]
    return quick_sort(left) + middle + quick_sort(right)
def heap_sort(arr):
    import heapq
    heapq.heapify(arr)
    return [heapq.heappop(arr) for _ in range(len(arr))]
def shell_sort(arr):
    n = len(arr)
    gap = n // 2
    while gap > 0:
        for i in range(gap, n):
            temp = arr[i]
            j = i
            while j >= gap and arr[j - gap] > temp:
                arr[j] = arr[j - gap]
                j -= gap
            arr[j] = temp
        gap //= 2
    return arr
# Генеруємо великий список транзакцій (100 000)
transactions = [random.randint(1000, 10000) for _ in range(100000)]
# Тестуємо та виводимо результати
for algorithm in [quick_sort, heap_sort, shell_sort]:
    start_time = time.time()
    sorted_transactions = algorithm(transactions.copy())
    elapsed_time = time.time() - start_time
    print(f"{algorithm.__name__} зайняв {elapsed_time:.6f} секунд")
```

У цьому коді генерується великий список транзакцій (випадкові цілі числа) та порівнюється швидкодія кожного алгоритму для сортування цих транзакцій. Потім виводиться час виконання кожного алгоритму [3].

Внаслідок порівняльного аналізу ефективності алгоритмів сортування для великого списку транзакцій у банківській системі отримані такі результати:

Quick sort зайняв 0.127742 секунд;

Heap sort зайняв 0.042595 секунд;

Shell sort зайняв 0.398416 секунд.

У межах дослідження було проведено порівняльний аналіз трьох алгоритмів сортування – Quick sort, Heap sort та Shell sort – у контексті їх ефективності для великих обсягів даних, як-от транзакції в банківській системі. Heap sort виявився найшвидшим серед трьох алгоритмів, зайнявши найменше часу на сортування великого списку транзакцій. Quick sort також продемонстрував гарні результати, ефективно працюючи з великими обсягами даних. У той час як Shell sort показав

найгірші результати серед трьох алгоритмів для великих списків транзакцій. З цих результатів можна зробити висновок, що для оптимальної швидкодії сортування великих обсягів даних у банківській системі рекомендується використовувати Heap sort.

Список використаних джерел

1. Introduction to Algorithms / T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein. MIT Press, 2009.
2. Sedgewick R., Wayne K. Algorithms, Addison-Wesley Professional, 2011.
3. Weiss M. A. Data Structures and Algorithm Analysis in Java, Pearson, 2011.

УДК 004.94

Войтенко М. О., *здобувач 1 курсу ОС «Магістр» спеціальності 122 Комп'ютерні науки,*
Бабаков Р. М., *д-р техн. наук, доцент, професор кафедри інформаційних технологій*

ПРЕДСТАВЛЕННЯ РОЗВ'ЯЗКУ ЗАДАЧІ АЛГЕБРАЇЧНОГО СИНТЕЗУ МІКРОПРОГРАМНОГО АВТОМАТА У ВИГЛЯДІ ГРАФА

Донецький національний університет імені Василя Стуса, м. Вінниця

Сучасні технології виробництва цифрових систем досягли рівня, за якого проєктування обчислювальних пристроїв здійснюється в автоматичному або автоматизованому (напівавтоматичному) режимі на основі певних вхідних даних. Автоматизація процесу розробки цифрових систем та їх складників дає змогу не тільки зменшити час виготовлення пристроїв, а й провести оптимізацію характеристик пристроїв, як-от вартість, енергоспоживання, надійність тощо [1].

Одним із центральних складників сучасних цифрових систем є пристрій керування, який координує роботу усіх блоків системи і припускає різні пособи структурної організації [2]. Одним із таких способів є мікропрограмний автомат з операційним автоматом переходів (МПА з ОАП), у якому перетворення кодів станів під час автоматних переходів здійснюється за допомогою певної множини арифметико-логічних операцій [1].

Одним із етапів синтезу МПА з ОАП є так званий алгебраїчний синтез автомата [3]. В процесі алгебраїчного синтезу відбуваються такі дії:

1. Станам автомата ставляться у відповідність унікальні коди із певної множини кодів станів. Коди станів можуть розглядатись як алфавіти різних алгебр та інтерпретуватись як цілі числа, бітові вектори або в інший спосіб. Оскільки схемна реалізація кодів станів потребує їх представлення у двійковій формі, інтерпретація у вигляді бітових векторів є обов'язковою.

2. Автоматним переходам ставляться у відповідність певні операції переходів (ОП) із певної множини ОП. Використання однієї ОП для реалізації декількох