УДК 004

Реализация сортировки слиянием на Python

Андриенко Иван Сергеевич
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
Целью данного исследования является изучение и реализация алгоритма сортировки слиянием на языке Python. Для достижения данной цели был использован метод программирования, который позволил реализовать алгоритм и провести его тестирование на текстовых данных. В результате исследования было выявлено, что сортировка слиянием является эффективным и универсальным методом сортировки, который позволяет работать с любыми типами данных.
Ключевые слова: Google Colab, Python, сортировка слиянием

Implementation of Merge sort in Python

Andrienko Ivan Sergeevich
Sholom-Aleichem Priamursky State University
Student

Abstract
The purpose of this study is to study and implement a merge sorting algorithm in Python. To achieve this goal, a programming method was used that allowed the algorithm to be implemented and tested on text data. As a result of the research, it was revealed that merge sorting is an effective and universal sorting method that allows you to work with any type of data.
Keywords: Google Colab, Python, Merge sorting

1 Введение

1.1 Актуальность

Сортировка является одним из основных алгоритмов в программировании и применяется во многих областях, таких как анализ данных, машинное обучение, компьютерная графика и т.д. Существует множество алгоритмов сортировки, каждый из которых имеет свои преимущества и недостатки. Среди них особое место занимает алгоритм сортировки слиянием, который обладает высокой эффективностью и универсальностью.

1.2 Обзор исследований

В своей статье Д. М. Курышов и Б. Б. Чумак исследуют применение параллельного алгоритма сортировки слиянием для эффективной сортировки

больших объемов данных. В статье описывается сам алгоритм, его принцип работы и особенности применения в контексте параллельных вычислений. Авторы также проводят эксперименты для оценки производительности алгоритма и сравнивают его с другими алгоритмами сортировки, показывая преимущества и эффективность параллельного алгоритма сортировки слиянием [1].

Д.В. Щукин исследует применение алгоритма сортировки слиянием для эффективной сортировки файлов большого объема. В статье описывается сам алгоритм, его принцип работы и особенности применения при работе с большими файлами. Автор проводит эксперименты для оценки производительности алгоритма и рассматривает различные подходы к оптимизации и параллелизации сортировки слиянием для улучшения времени выполнения [2].

Д.М. Галстян описывает применение параллельных технологий MPI и OpenMP для сортировки слиянием. В статье проводится анализ эффективности параллельной сортировки слиянием с использованием данных технологий и оценивается время выполнения и масштабируемость алгоритма. Авторы рассматривают различные варианты реализации параллельной сортировки слиянием с использованием MPI и OpenMP, а также проводят сравнение с другими алгоритмами сортировки, демонстрируя преимущества и эффективность параллельного подхода [3].

1.3 Цель исследования

Целью исследования – изучение и реализация алгоритма сортировки слиянием на языке Python.

2 Материалы и методы

Для работы используется онлайн среда программирования Google Colab [4].

3 Результаты и обсуждение

Принцип работы сортировки слиянием состоит из нескольких шагов:

1. Разделение на подмассивы: Исходный массив делится на две примерно равные части. Это делается рекурсивно до тех пор, пока каждый подмассив не будет состоять из одного элемента. Если массив содержит нечетное количество элементов, один из подмассивов будет иметь на один элемент больше.

2. Рекурсивная сортировка: Каждый подмассив сортируется отдельно с помощью рекурсивного вызова сортировки слиянием. Этот шаг выполняется до тех пор, пока все подмассивы не будут состоять из одного элемента, то есть будут отсортированы.

3. Слияние подмассивов: Отсортированные подмассивы сливаются в один отсортированный массив. Для этого используется вспомогательный массив или буфер. На каждом шаге выбираются наименьшие элементы из двух подмассивов и помещаются в конечный массив. После этого указатели на

выбранные элементы сдвигаются вперед, и процесс повторяется до тех пор, пока все элементы не будут помещены в конечный массив.

4. Запись оставшихся элементов: Если один из подмассивов закончился раньше другого, оставшиеся элементы из другого подмассива просто копируются в конечный массив.

5. Завершение: В результате слияния подмассивов получается полностью отсортированный массив.

Принцип работы сортировки слиянием основан на идее разделения и слияния. Разделение массива на подмассивы происходит рекурсивно до тех пор, пока каждый подмассив не будет состоять из одного элемента. Затем отсортированные подмассивы сливаются в один отсортированный массив, путем выбора наименьших элементов на каждом шаге и помещения их в конечный массив. В результате получается полностью отсортированный массив. Этот процесс повторяется до тех пор, пока все элементы не будут помещены в конечный массив. Временная сложность сортировки слиянием составляет O(n log n), что делает ее одним из самых эффективных алгоритмов сортировки.

Сортировка слиянием имеет несколько преимуществ и недостатков:

Преимущества:

1. Устойчивость: Сортировка слиянием сохраняет относительный порядок элементов с одинаковыми значениями. Это особенно важно, когда требуется сортировка объектов с несколькими ключами или свойствами.

2. Гарантированная производительность: Временная сложность сортировки слиянием составляет O(n log n), что делает ее одним из самых эффективных алгоритмов сортировки. Она работает с одинаковой скоростью на лучших, средних и худших случаях.

3. Легкость реализации: Сортировка слиянием основана на простой и понятной идее разделения массива на две половины и их последующего слияния. Рекурсивный подход делает алгоритм легким для понимания и реализации.

4. Масштабируемость: Сортировка слиянием хорошо масштабируется для больших объемов данных. Она может быть эффективно распараллелена для ускорения сортировки массивов большого размера.

5. Сортировка внешних данных: Сортировка слиянием может быть использована для сортировки данных, которые не помещаются в оперативную память компьютера. Это делает ее полезной для сортировки больших файлов или баз данных.

Недостатки:

1. Дополнительное пространство: Сортировка слиянием требует дополнительного пространства для хранения временных массивов при слиянии. Это может быть проблемой, особенно при работе с огромными объемами данных, когда доступная память ограничена.

2. Некоторая неэффективность: В сортировке слиянием требуется выполнить несколько операций сравнения и слияния массивов, что может

привести к некоторой неэффективности по сравнению с другими алгоритмами сортировки, такими как быстрая сортировка.

3. Рекурсивный подход: Хотя рекурсивный подход делает алгоритм понятным, он также требует дополнительных вызовов функций и использования стека вызовов, что может привести к переполнению стека при работе с очень большими массивами.

4. Не подходит для сортировки связанных списков: Сортировка слиянием требует доступа к элементам массива по индексу, что делает ее неэффективной для сортировки связанных списков, где доступ к элементам происходит последовательно.

Для реализации алгоритма сортировки создаём функцию merge_sort, которая реализует алгоритм сортировки слиянием. В начале функция проверяет, если длина массива arr меньше или равна 1, то возвращается сам массив. Это базовый случай для рекурсии, когда массив уже отсортирован. Если массив состоит из более чем одного элемента, то функция находит средний индекс массива mid и разделяет исходный массив на две равные части left_arr и right_arr. Затем функция рекурсивно вызывает merge_sort для каждой части, чтобы отсортировать их. Наконец, функция объединяет отсортированные массивы с помощью функции merge (рис.1).

```
def merge_sort(arr):
    if len(arr) <= 1:
        return arr

    mid = len(arr) // 2
    left_arr = arr[:mid]
    right_arr = arr[mid:]

    left_arr = merge_sort(left_arr)
    right_arr = merge_sort(right_arr)

    return merge(left_arr, right_arr)
```

Рисунок 1 - Функция merge_sort

Функция merge сливает два отсортированных массива left_arr и right_arr в один отсортированный массив sorted_arr. Для этого используется два указателя left_index и right_index, которые указывают на текущие индексы элементов в левом и правом массивах соответственно. В цикле while сравниваются элементы на текущих индексах, и добавляется меньший из них в sorted_arr. Затем увеличиваем соответствующий указатель на 1. Этот процесс продолжается до тех пор, пока один из массивов не закончится. В конце добавляем оставшиеся элементы в sorted_arr и возвращаем его (рис.2).

```python
def merge(left_arr, right_arr):
    sorted_arr = []
    left_index = right_index = 0

    while left_index < len(left_arr) and right_index < len(right_arr):
        if left_arr[left_index] < right_arr[right_index]:
            sorted_arr.append(left_arr[left_index])
            left_index += 1
        else:
            sorted_arr.append(right_arr[right_index])
            right_index += 1

    sorted_arr += left_arr[left_index:]
    sorted_arr += right_arr[right_index:]

    return sorted_arr
```

Рисунок 2 - Функция merge

Чтобы проверить алгоритм, создаём массив arr с неотсортированными элементами и сортирует его с помощью функции merge_sort. Затем выводится исходный массив arr и отсортированный массив sorted_arr (рис.3).

```python
arr = [5, 2, 8, 6, 1, 9, 3, 7, 56, 58, 23, 2, 4, 890, 432, -89, 568, 34]
sorted_arr = merge_sort(arr)
print("Исходный массив:", arr)
print("Отсортированный массив:", sorted_arr)
```

Рисунок 3 - Проверка алгоритма сортировки

После выполнения алгоритма, получаем отсортированный массив данных(рис.4).

```
Исходный массив: [5, 2, 8, 6, 1, 9, 3, 7, 56, 58, 23, 2, 4, 890, 432, -89, 568, 34]
Отсортированный массив: [-89, 1, 2, 2, 3, 4, 5, 6, 7, 8, 9, 23, 34, 56, 58, 432, 568, 890]
```

Рисунок 4 - Результат выполнения программы

Выводы

Исходя из результатов исследования, можно сделать вывод, что сортировка слиянием является одним из наиболее эффективных и универсальных алгоритмов сортировки. Она может быть использована во многих областях программирования, где требуется обработка больших объемов данных. Реализация данного алгоритма на языке Python позволяет быстро и удобно сортировать массивы любой длины и типов данных.

Библиографический список

1. Курышов Д. М., Чумак Б. Б. Параллельный алгоритм сортировки слиянием // Научный альманах. 2016. № 12-2 (26). С. 71-73.
2. Щукин, Д. В. Сортировка слиянием больших файлов // Дни науки студентов Владимирского государственного университета имени Александра Григорьевича и Николая Григорьевича Столетовых. Владимир: Владимирский государственный университет им. Александра Григорьевича и Николая Григорьевича Столетовых, 2018. С. 361-368.
3. Галстян Д. М. Анализ сортировки слиянием с использованием mpi и openm // Наука: следующее поколение. Саратов: Общество с ограниченной ответственностью "Центр профессионального менеджмента "Академия Бизнеса", 2020. С. 17-20.
4. Google Colab URL: https://colab.research.google.com