Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЕТ
ПО ПРАКТИЧЕСКОЙ РАБОТЕ №10
дисциплины «Алгоритмизация»
Вариант

Выполнил:

Говоров Егор Юрьевич

2 курс, группа ИВТ-б-о-22-1,

09.03.02 «Информатика и вычислительная техника», направленность (профиль) «Программное обеспечение средств вычислительной техники и автоматизированных систем», очная форма обучения

_____

(подпись)

Руководитель практики:

Воронкин Р А., канд. технических наук, доцент кафедры инфокоммуникаций

_____

(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Порядок выполнения работы:

1. Написал программу (heap_sort.py), в рамках проведенного исследования был реализован алгоритм сортировки кучи. Для оценки времени работы данного алгоритма были использованы различные типы входных данных, включая отсортированный массив, массив, обратный отсортированному, и массив с произвольными значениями. Результаты измерений времени работы алгоритма позволяют сделать выводы о его эффективности и применимости в различных сценариях использования.

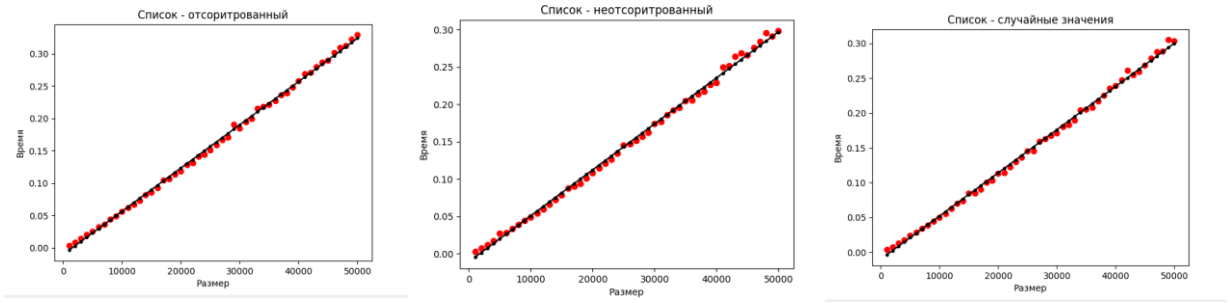

Рисунок 1 – Графики зависимости длинны массива от времени

2. Сравнение с другими сортировками

Таблица 1 – О-большое сортировок

Случай	Лучший	Средний	Худший
Heap Sort	O(nlogn)	O(nlogn)	O(nlogn)
Quick Sort	O(nlogn)	O(nlogn)	O(n2)
Merge Sort	O(nlogn)	O(nlogn)	O(nlogn)

Алгоритм сортировки кучей (Heap Sort) обладает временной сложностью O(nlogn) во всех случаях - лучшем, среднем и худшем. Это делает его эффективным для обработки больших наборов данных. Однако, стоит отметить, что он не является стабильным, что может быть критично в некоторых ситуациях.

Алгоритм сортировки слиянием (Merge Sort) также имеет временную сложность O(nlogn) во всех трех случаях, что делает его эффективным для обработки больших массивов данных. Одним из его преимуществ является стабильность, то есть сохранение относительного порядка одинаковых элементов. Однако, этот алгоритм требует дополнительного пространства

для слияния массивов, что может быть проблематично при работе с большими объемами данных.

Алгоритм быстрой сортировки (Quick Sort) имеет временную сложность O(nlogn) в среднем и лучшем случае, но в худшем случае его сложность может достигать $O(n^2)$. Несмотря на это, Quick Sort часто используется на практике из-за своей эффективности в среднем случае. Однако, в худшем случае, когда выбор опорного элемента не оптимален, производительность алгоритма может снизиться до квадратичной сложности.

3. Написал программу (heap_sort_upgrade.py), исследование включало реализацию алгоритма сортировки кучи на основе модуля heapq:

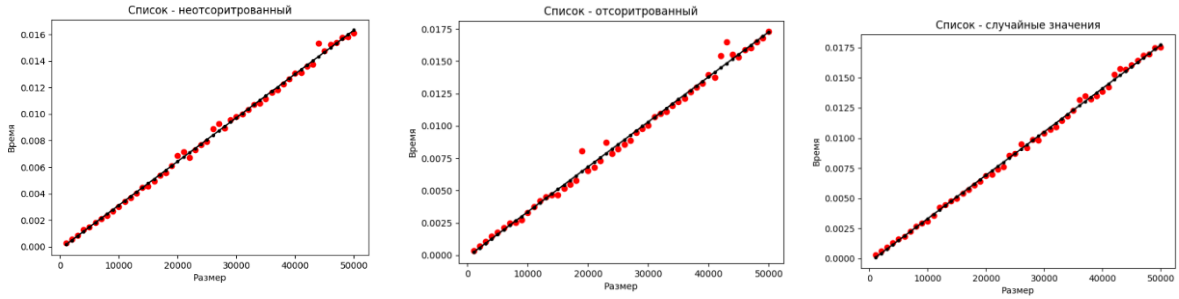

Рисунок 2 – Графики зависимости длинны списка от времени

4. Применение в реальной жизни

Алгоритм сортировки кучей может быть применен в реальных ситуациях, включая оптимизацию баз данных, обработку событий и решение других вычислительных проблем. В некоторых контекстах он может быть наиболее подходящим решением, благодаря его гарантированному времени выполнения даже в наихудших условиях. Например, в ситуациях, когда нужен быстрый доступ к упорядоченной информации в базе данных, сортировка кучей может оказаться полезной за счет своей эффективности и стабильности времени выполнения. Кроме того, в условиях, когда необходимо отсортировать большое количество данных с критически важным гарантированным временем выполнения, алгоритм сортировки кучей может быть оптимальным выбором.

5. Анализ сложности

Алгоритм сортировки кучей имеет временную сложность O(nlogn) в худшем, лучшем и среднем случае. Пространственная сложность составляет O(1), что означает, что дополнительная память, используемая для сортировки, не зависит от размера входных данных.

С увеличением размера входных данных время выполнения алгоритма сортировки кучей увеличивается логарифмически, что делает его эффективным для обработки больших объемов данных. Однако, по сравнению с другими алгоритмами сортировки, такими как Quick Sort или Merge Sort, в некоторых случаях алгоритм сортировки кучей может быть менее эффективным из-за постоянных операций с памятью и более сложной реализации.

Таким образом, алгоритм сортировки кучей может быть более эффективным в случаях, когда требуется гарантированное время выполнения в худшем случае и при работе с большими объемами данных. Однако, в некоторых сценариях, где важна простота реализации и минимальное использование памяти, другие алгоритмы сортировки, такие как Quick Sort или Merge Sort, могут быть предпочтительными.

6. Даны массивы A[1…n] и B[1…n]. Мы хотим вывести все n2 сумм вида A[i]+B[j] в возрастающем порядке. Наивный способ — создать массив, содержащий все такие суммы, и отсортировать его. Соответствующий алгоритм имеет время работы O(n2logn) и использует O(n2) памяти. Приведите алгоритм с таким же временем работы, который использует линейную память.

Написал программу (task6.py), которая решает данную задачу:

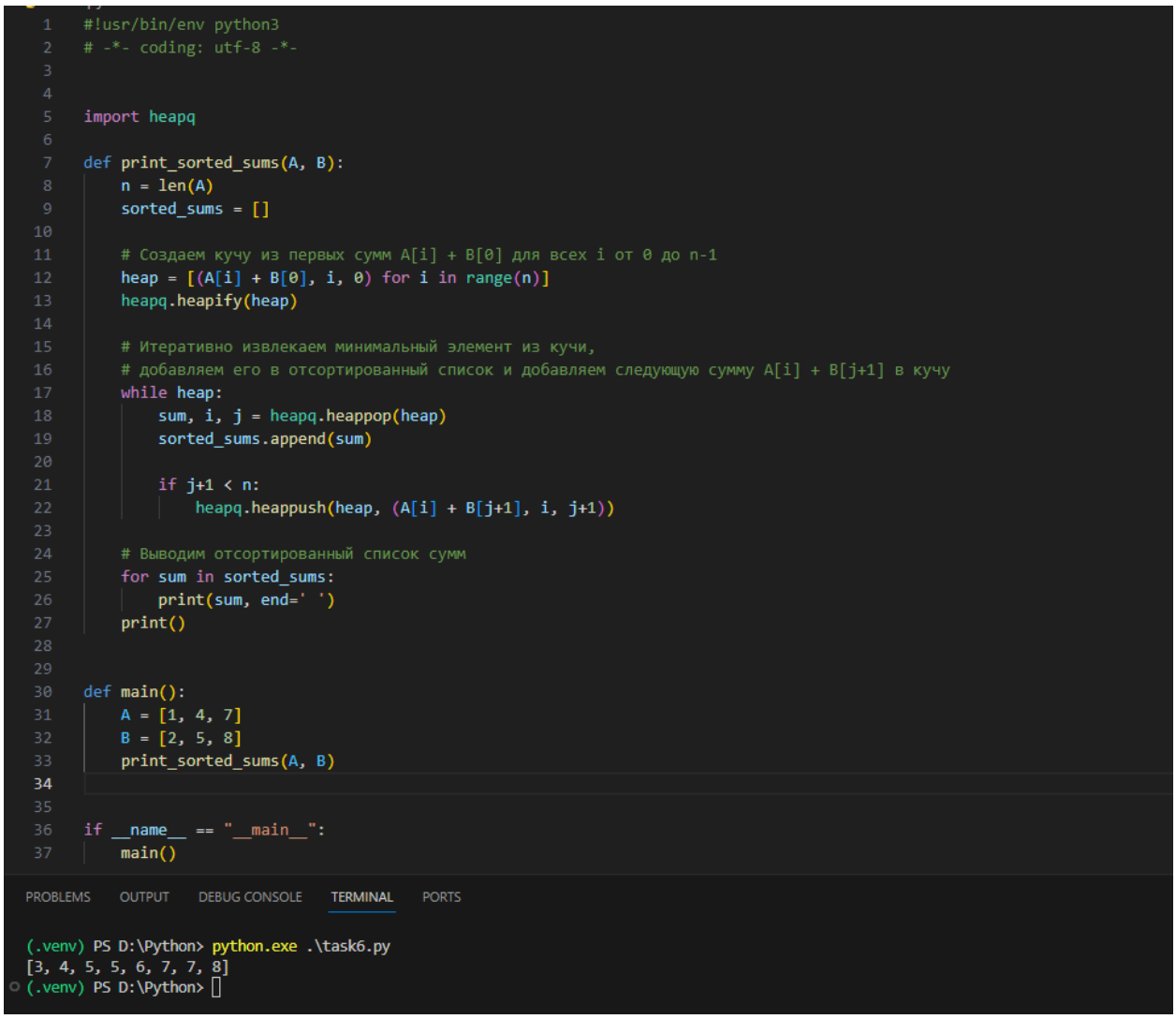

Рисунок 3 – Результат выполнения программы task6.py

Вывод: в результате лабораторной работы было изучено использование алгоритма сортировки кучей, и было установлено, что алгоритм реализованный на основе модуля heap работает в разы быстрее.