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»

Выполнил:
Степанов Леонид Викторович
2 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и вычислительная техника»,
направленность (профиль) «Программное обеспечение средств вычислительной техники и автоматизирование систем», очная форма обучения

_____
(подпись)

Руководитель практики:
Воронкин Р.А., канд. техн. наук, доцент, доцент кафедры инфокоммуникаций

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2023 г.

Тема: Алгоритм сортировки кучей

Порядок выполнения работы:

1. Написал программу (heap_sort.py), в которой реализовал алгоритм сортировки кучи и посчитал время работы в случае, когда на вход идут: отсортированный массив, массив, который обратный отсортированному и массив с рандомными значениями:

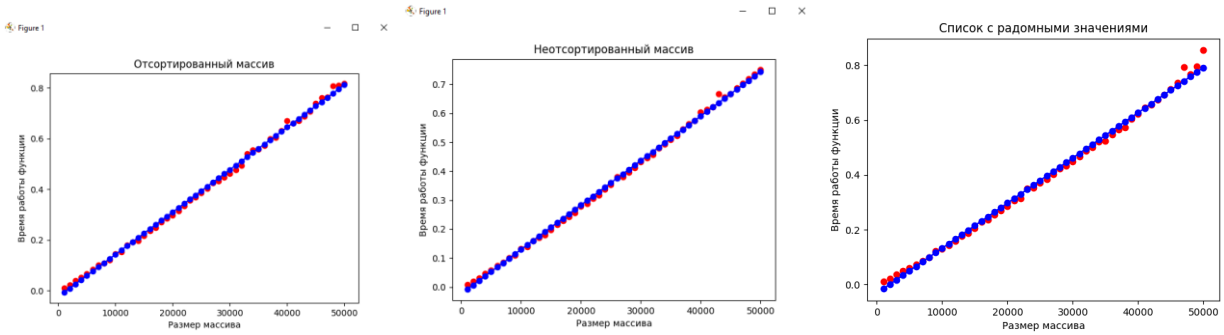

Рисунок 1 – Графики зависимости длинны массива от времени

2. Сравнение с другими сортировками

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 Heap Sort обладает временной сложностью O(n log n) в лучшем, среднем и худшем случае. Это делает его эффективным для сортировки больших наборов данных. Однако, в отличие от некоторых других алгоритмов, он не является стабильным, что может быть недостатком в определенных сценариях.

Алгоритм сортировки слиянием (Merge Sort) обладает временной сложностью O(n log n) в лучшем, среднем и худшем случае. Это делает его эффективным для сортировки больших наборов данных. Одним из его преимуществ является стабильность, что означает, что одинаковые элементы не меняют свой порядок относительно друг друга. Однако, алгоритм сортировки слиянием требует дополнительной памяти для слияния массивов, что может быть недостатком при работе с огромными данными.

Алгоритм Quick Sort обладает временной сложностью O(n log n) в среднем и в лучшем случае. В худшем случае его временная сложность составляет O(n^2). Quick Sort обычно эффективен на практике и широко используется из-за своей высокой производительности 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), в которой реализовал алгоритм сортировки кучи и посчитал время работы в случае, когда на вход идут: отсортированный список, список, который обратный отсортированному и список с рандомными значениями:

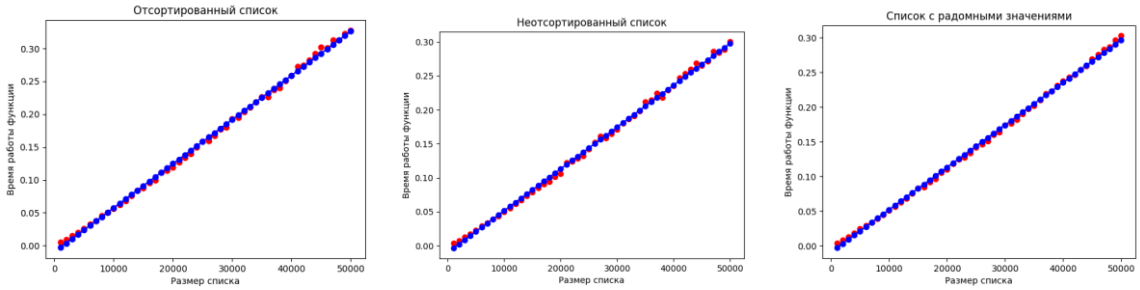

Рисунок 2 – Графики зависимости длинны списка от времени

4. Применение в реальной жизни

Алгоритм Heap Sort может быть применен в реальных сценариях, таких как оптимизация работы баз данных, событийной обработки и других вычислительных задач. Он может быть предпочтительным выбором в некоторых ситуациях благодаря гарантированному времени выполнения в худшем случае. Например, в базах данных, где необходимо обеспечить быстрый доступ к отсортированным данным, Heap Sort может быть полезен из-за своей эффективности и предсказуемости времени выполнения. Также, в сценариях, где требуется сортировка больших объемов данных, где гарантированное время выполнения важно, алгоритм Heap Sort может быть предпочтительным выбором.

5. Анализ сложности

Алгоритм Heap Sort имеет временную сложность O(n log n) 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.

Характеристики алгоритма зависят от размера входных данных следующим образом: с увеличением размера входных данных время выполнения алгоритма Heap Sort увеличивается логарифмически, что делает его эффективным для больших объемов данных. Однако, в сравнении с другими алгоритмами сортировки, такими как Quick Sort или Merge Sort, в некоторых случаях Heap Sort может быть менее эффективным из-за постоянных операций с памятью и более сложной реализации.

Таким образом, Heap Sort может быть более эффективным в случаях, когда требуется гарантированное время выполнения в худшем случае и при работе с большими объемами данных, но может быть менее эффективным по сравнению с другими алгоритмами сортировки в некоторых сценариях, где важна простота реализации и минимальное использование памяти.

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:

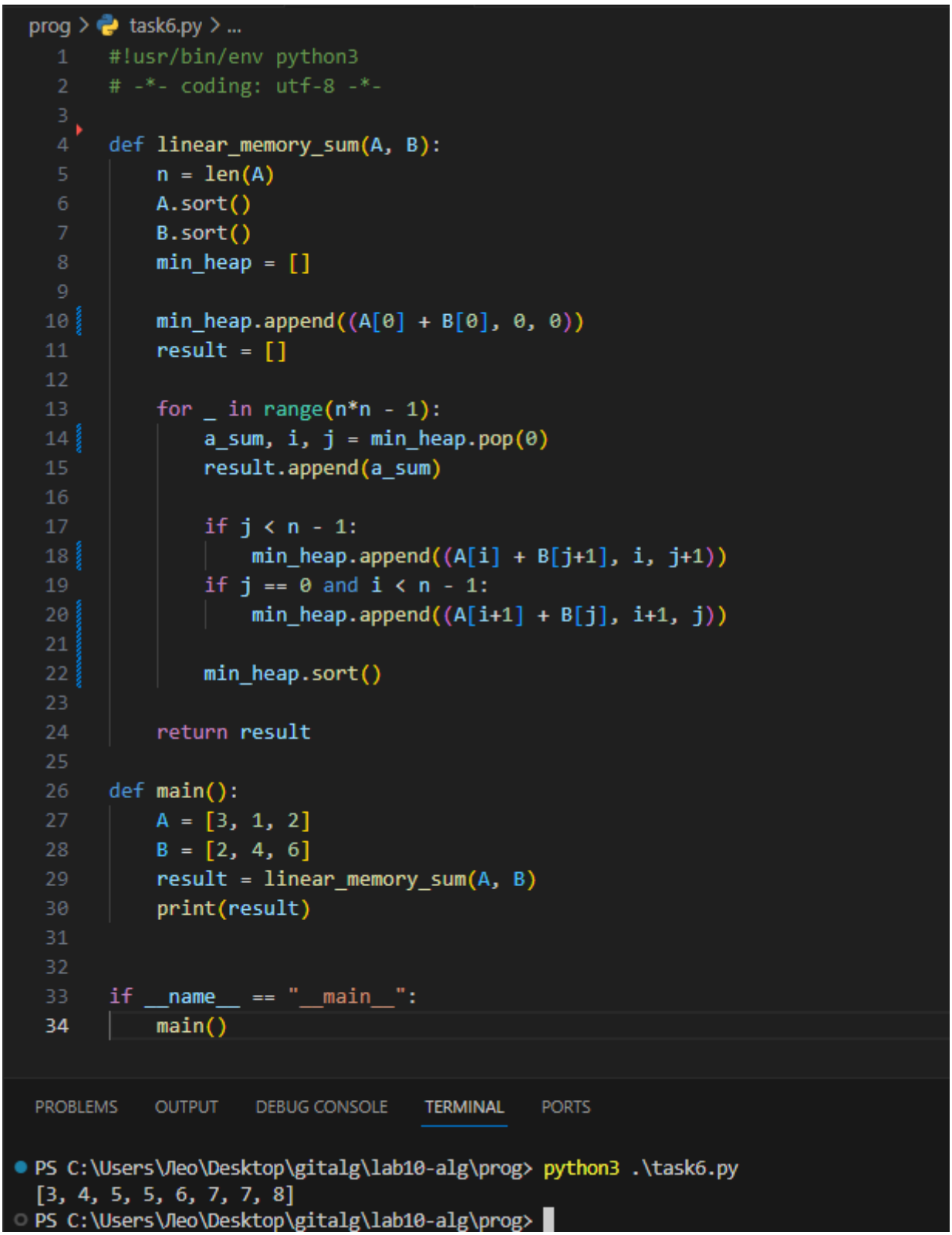

Рисунок 3 – Результат выполнения программы task6.py

Вывод: в результате выполнения лабораторной работы был исследован алгоритм сортировкой кучей, было выяснено, что со списками данных алгоритм работает быстрее, чем с массивом.