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 ЛАБОРАТОРНОЙ РАБОТЕ №10
дисциплины «Алгоритмизация»

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

_____
(подпись)

Руководитель практики:
Воронкин Р.А.-доцент кафедры
инфокоммуникаций

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2023 г.

Порядок выполнения работы:

1. Написала 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:

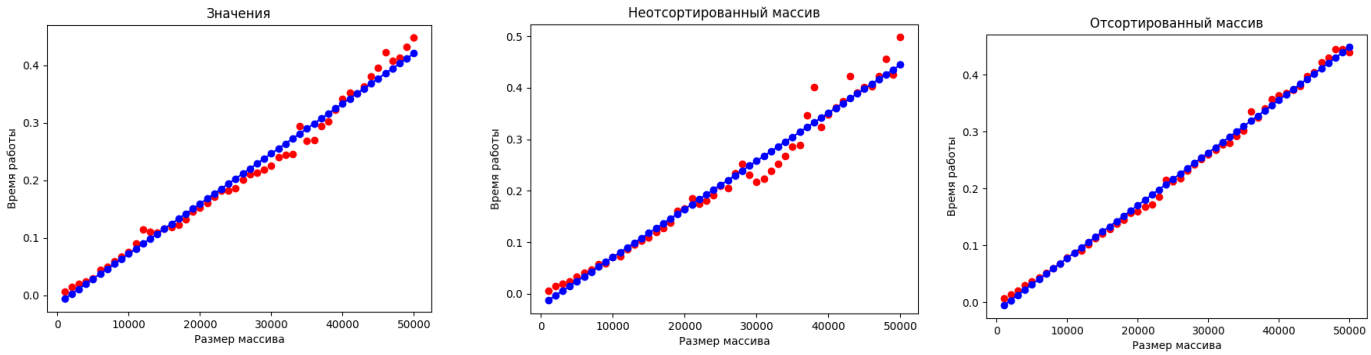

Рисунок 1. Графики трех функций

2. Алгоритм Heap Sort имеет временную сложность O(n log n) во всех случаях, что делает его подходящим для сортировки больших объемов данных. Однако он не является стабильным, в отличие от некоторых других алгоритмов. Алгоритм слияния (Merge Sort) также имеет временную сложность O(n log n) во всех случаях и является эффективным для сортировки больших объемов данных. Его преимуществом является стабильность, но он требует дополнительной памяти для слияния массивов. Алгоритм Quick Sort имеет временную сложность O(n log n) в среднем и лучшем случае, но в худшем случае его производительность может снизиться до O(n^2) из-за неоптимального выбора опорного элемента. Однако на практике Quick Sort обычно эффективен и широко используется из-за высокой производительности в среднем случае.

3. Написала программу, в которой реализовал алгоритм сортировки кучи и посчитала время работы в случае, когда на вход идут: отсортированный список, список, который обратный отсортированному и список с рандомными значениями:

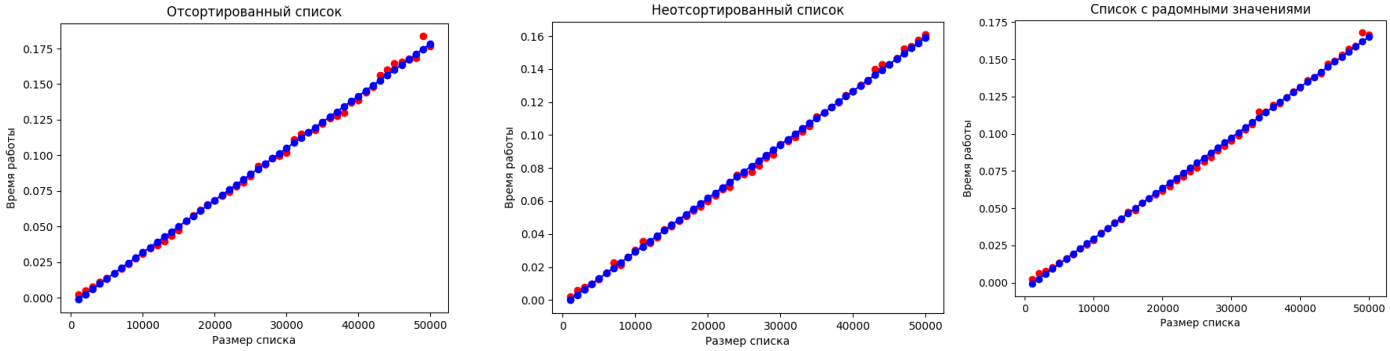

Рисунок 2. Графики трех функций

4. Алгоритм Heap Sort может быть применен в реальных сценариях, таких как оптимизация работы баз данных и событийной обработки. В таких случаях он может быть предпочтительным выбором благодаря гарантированному времени выполнения в худшем случае. Например, в базах данных, где необходим 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 с гарантированным временем выполнения, алгоритм Heap Sort может быть предпочтительным выбором.

5. В отношении анализа сложности, временная сложность алгоритма Heap Sort составляет O(n log n) в худшем, лучшем и среднем случае, а пространственная сложность - O(1), что означает, что дополнительная память, используемая для сортировки, не зависит от размера входных данных. С увеличением размера входных данных время выполнения алгоритма увеличивается логарифмически, что делает его эффективным для больших объемов данных. Однако в некоторых случаях он может быть менее эффективным по сравнению с другими алгоритмами сортировки 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. Написала программу (task6.py), которая решает данную задачу:

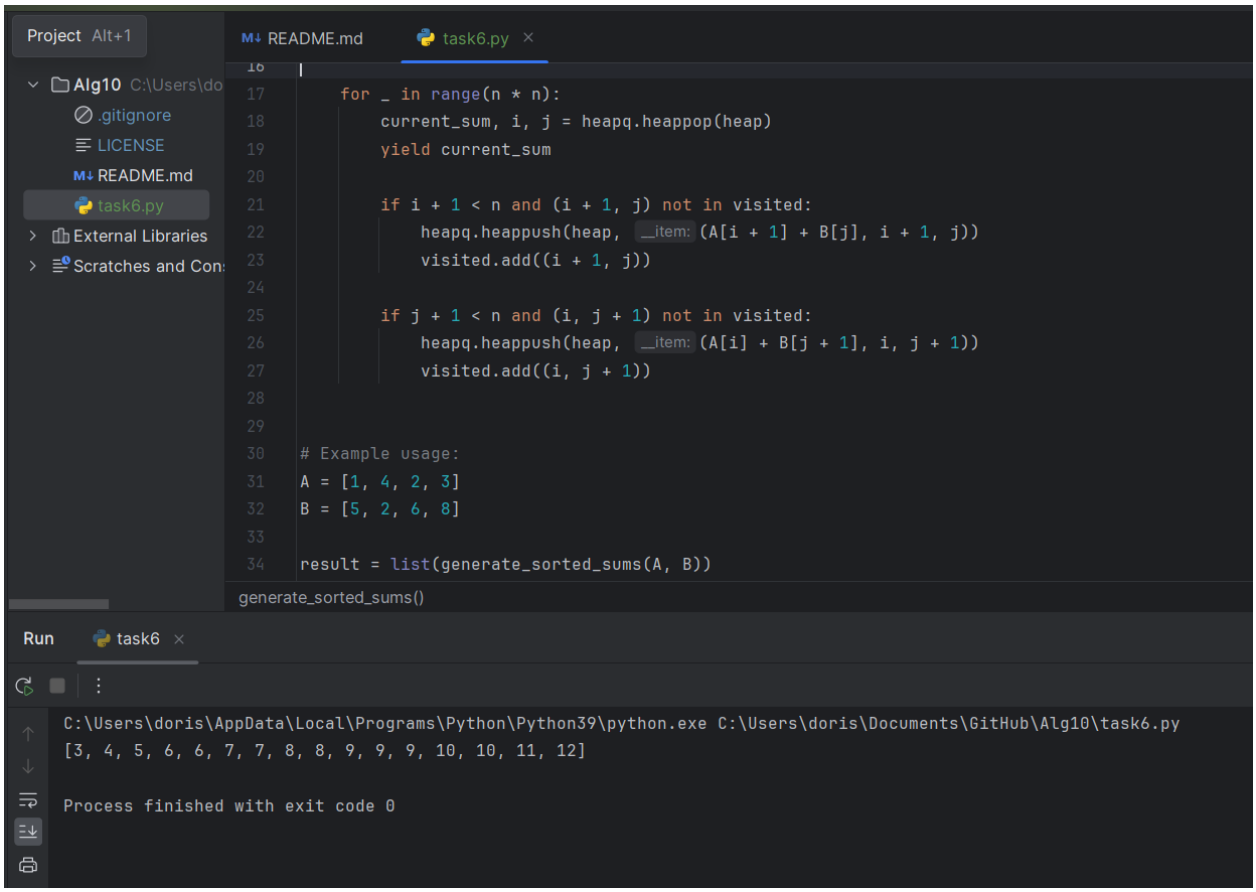

Рисунок 3. Результат выполнения программы

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