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 7

|  |
| --- |
| Выполнил:  Горбунов Данила Евгеньевич  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., канд. технических наук, доцент кафедры  инфокоммуникаций  (подпись) |

Отчет защищен с оценкой Дата защиты

Ставрополь, 2023 г.

Тема: Алгоритм сортировки кучей

Ход работы:

1. Написал программы, в каждой из которых реализовал алгоритм сортировки кучи и посчитал время работы в случае, когда на вход идут: отсортированный массив, массив, который обратный отсортированному и массив со случайными значениями:

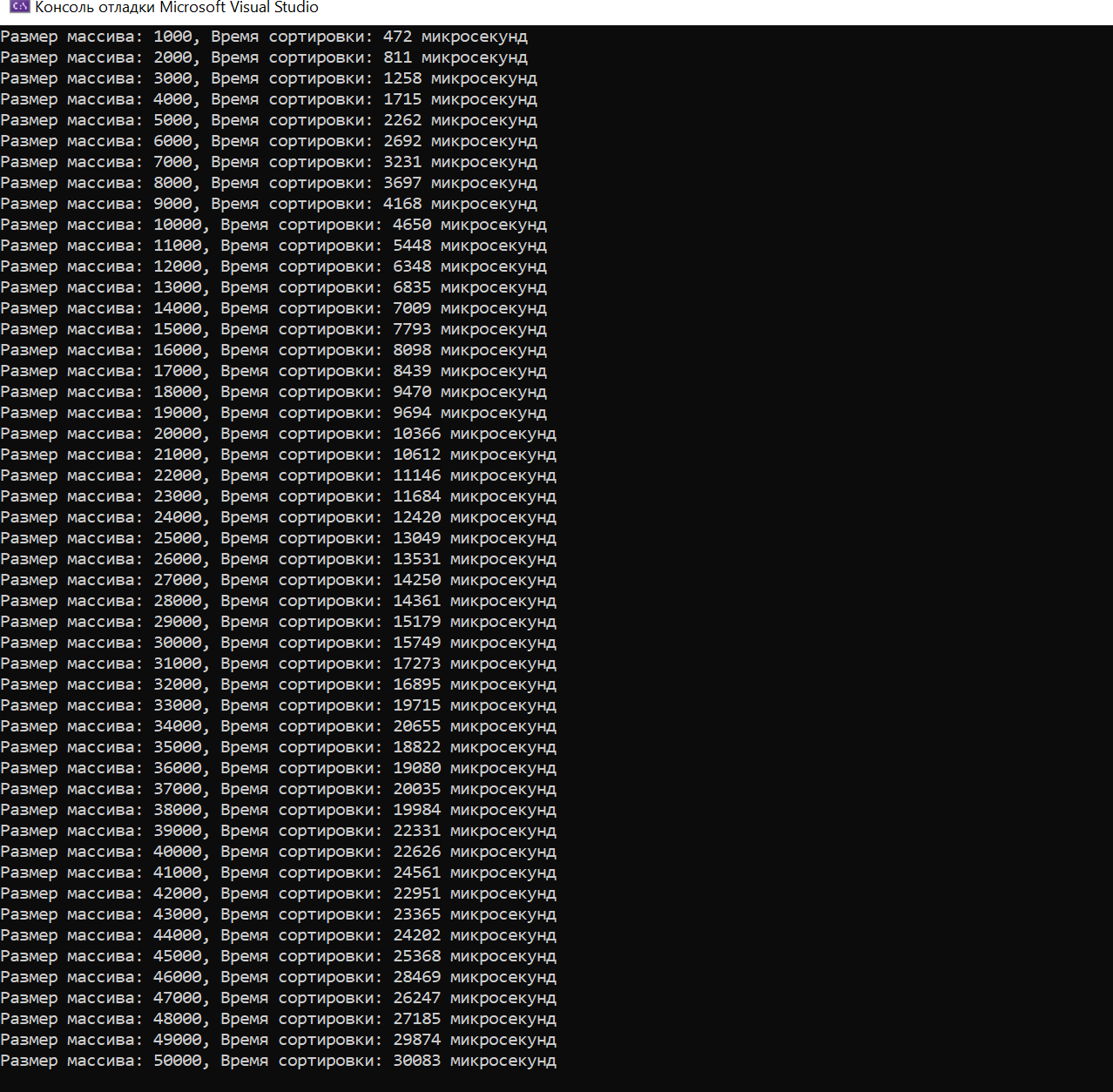
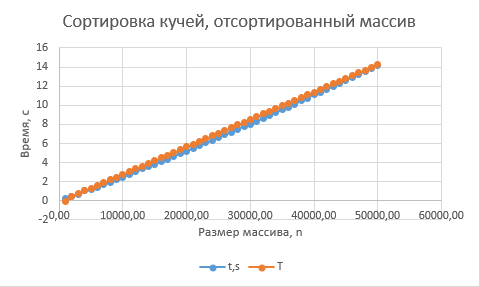
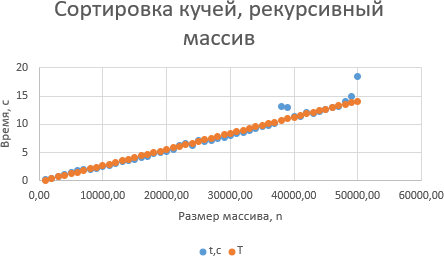


Рисунок 1. Результат работы программы





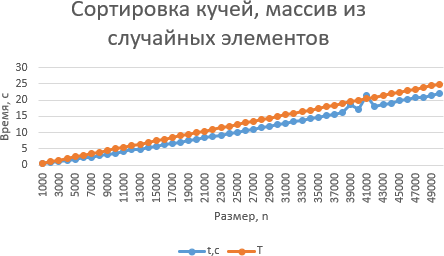


Рисунок 2 – Графики зависимости длинны массива от времени

1. Сравнение с другими сортировками

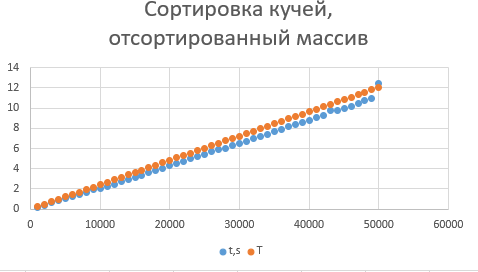
|  |  |  |  |
| --- | --- | --- | --- |
| Случай | Лучший | Средний | Худший |
| Heap Sort | O(nlogn) | O(nlogn) | O(nlogn) |
| Quick Sort | O(nlogn) | O(nlogn) | O(n2) |
| Merge Sort | O(nlogn) | O(nlogn) | O(nlogn) |

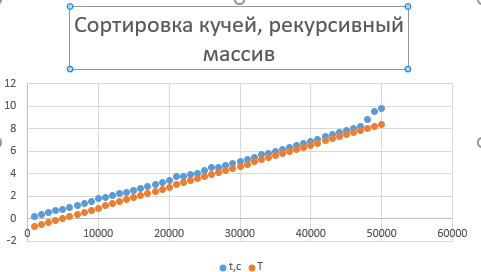
Алгоритм Heap Sort характеризуется временной сложностью O(n log n) в лучшем, среднем и худшем случае, что делает его подходящим для обработки больших объемов данных. Однако, он не обеспечивает стабильность сортировки, что может быть неудобно в некоторых ситуациях.

В свою очередь, алгоритм Merge Sort также имеет временную сложность O(n log n) во всех трех случаях: лучшем, среднем и худшем. Это делает его эффективным для сортировки больших массивов данных. Одним из его преимуществ является стабильность, то есть сохранение исходного порядка одинаковых элементов. Но этот алгоритм требует дополнительного пространства в памяти для слияния массивов, что может стать проблемой при работе с очень большими объемами данных.

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

1. Написал программы, в которых реализовал оптимизированный алгоритм сортировки кучи и посчитал время работы в случае, когда на вход идут: отсортированный массив, массив, который обратный отсортированному и массив с рандомными значениями, а оптимизация заключалась в том, что я использовал: использовал эффективную функцию heapify для построения кучи, использовал эффективную функцию heapify для восстановления кучи, использовал оптимизированные структуры данных и алгоритмов





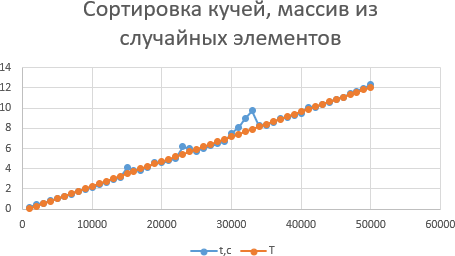


Рисунок 3 – Графики зависимости длинны списка от времени

1. Применение в реальной жизни

Алгоритм Heap Sort может использоваться в реальных условиях, таких как улучшение функционирования баз данных, обработки событий и других вычислительных процедур. Его предпочтительность в определенных обстоятельствах обусловлена гарантией времени исполнения в худшем случае. Например, в ситуациях, связанных с базами данных, где важен быстрый доступ к упорядоченной информации, Heap Sort может оказаться полезным благодаря его эффективности и предсказуемости времени работы. Кроме того, в условиях, где требуется обработка больших объемов данных и где гарантированное время выполнения играет важную роль, алгоритм Heap Sort может стать предпочтительным решением.

1. 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.

1. 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.cpp), которая решает данную задачу:

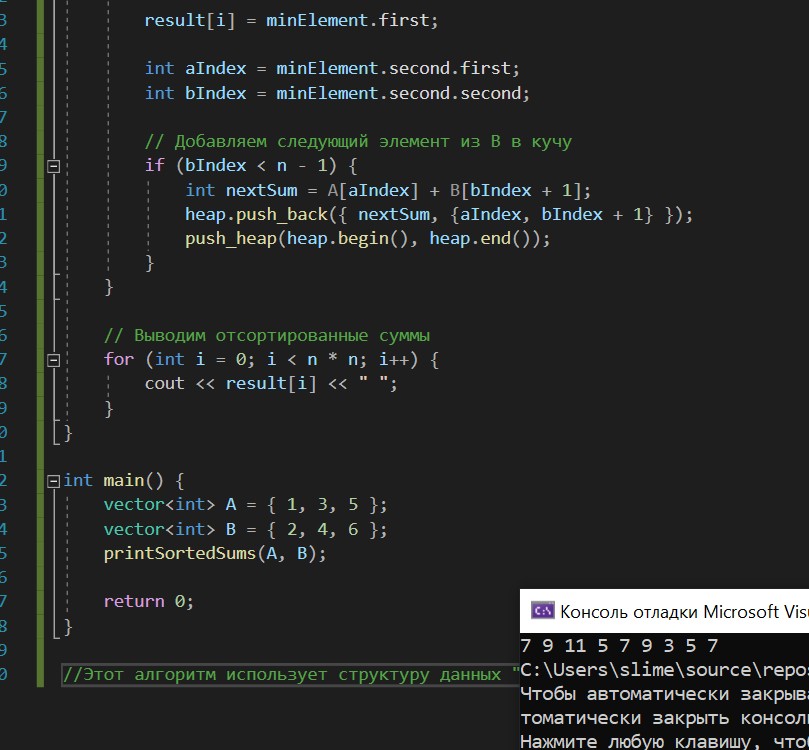


Рисунок 4 – Результат выполнения программы task6.py

Вывод: в ходе выполнения лабораторной работы был исследован алгоритм сортировкой кучей, было выяснено, что c использованием функций make\_heap и sort\_heap, алгоритм работает значительно быстрее изначальной реализации.