**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра систем автоматизированного проектирования**

**ОТЧЕТ**

**по лабораторной работе №1**

**«Алгоритмы сортировки сравнением»**

|  |  |  |
| --- | --- | --- |
| Студент гр. 3353 |  | Шушков .В.А. |
| Преподаватель |  | Пестерев Д.О. |

Санкт-Петербург

2024

#### Тема лабораторной работы: «Алгоритмы сортировки сравнением»

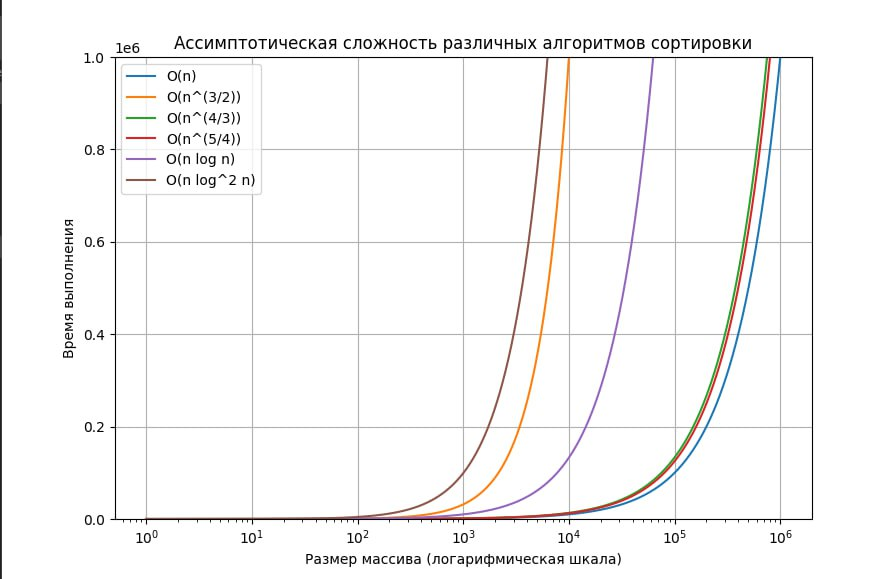
#### Цель лабораторной работы:

Реализовать различные алгоритмы сортировки, основанные на сравнении элементов, и исследовать их временные и пространственные характеристики. На основе экспериментальных данных оценить фактическую эффективность алгоритмов и сопоставить с теоретическими результатами.

#### Задачи лабораторной работы:

1. Реализовать следующие алгоритмы сортировки:
   * Сортировка вставками
   * Пузырьковая сортировка
   * Сортировка слиянием
   * Сортировка Шелла (с использованием различных последовательностей: Шелла, Хиббарда, Пратта)
   * Быстрая сортировка
   * Пирамидальная сортировка (Heapsort)
2. Для каждого алгоритма:
   * Написать описание алгоритма, объяснив его работу своими словами.
   * Определить асимптотическую временную и пространственную сложность для лучшего, среднего и худшего случая (если применимо).
   * Описать алгоритм с точки зрения устойчивости (проверить, сохраняются ли исходные относительные позиции одинаковых элементов).
   * Построить функции временной сложности для лучшего, среднего и худшего случая.
   * Получить асимптотическую оценку временной и пространственной сложности для каждого случая.
3. Исследовать экспериментальные данные:
   * Измерить фактическое время выполнения каждого алгоритма на массивах разного состояния:
     + Отсортированный массив.
     + Почти отсортированный массив (90% элементов отсортированы, 10% случайные).
     + Отсортированный в обратном порядке массив.
     + Полностью случайный массив.
   * Представить данные в виде графиков и таблиц.
   * Провести регрессионный анализ данных для определения фактической асимптотики.
   * Сравнить экспериментальные результаты с теоретическими оценками и выявить возможные расхождения.
4. Сравнительный анализ:
   * Построить графики для алгоритмов с квадратичной асимптотикой (например, пузырьковая и вставками) и с асимптотикой **O(n log n)** (например, быстрая сортировка и сортировка слиянием) на отдельных изображениях.
   * Сделать выводы о наибыстрейшей сортировке для массивов большого размера (более 100 000 элементов).
   * Сравнить полученные экспериментальные результаты с предположениями, сделанными в теоретической части.

# **Асимптотика сортировок на графике**



# **Таблица с асимптотической сложностью сортировок**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Алгоритм | Лучшая сложность | Средняя сложность | Худшая сложность | Пространственная сложность |
| Insertion sort | O(n) | O(n^2) | O(n^2) | O(1) |
| Bubble sort | O(n) | O(n^2) | O(n^2) | O(1) |
| Quick sort | O(n log n) | O(n log n) | O(n^2) | O(log n) |
| Shell sort | O(n log² n) | O(n^(3/2)) | O(n²) | O(1) |
| Shell sort(Hibbard) | O(nlogn) | O(n^(3/2)) | O(n^(3/2)) | O(1) |
| Shell sort(Prap) | O(n log n) | O((n log n )^ 2) | O((n log n )^ 2) | O(1) |
| Heap sort | O(n log n) | O(n log n) | O(n log n) | O(1) |
| Merge sort | O(n log n) | O(n log n) | O(n log n) | O(n) |
| Selection sort | O(n^2) | O(n^2) | O(n^2) | O(1) |

1. **Теоретическая часть**
   1. **Сортировка вставками**
2. Описание алгоритма

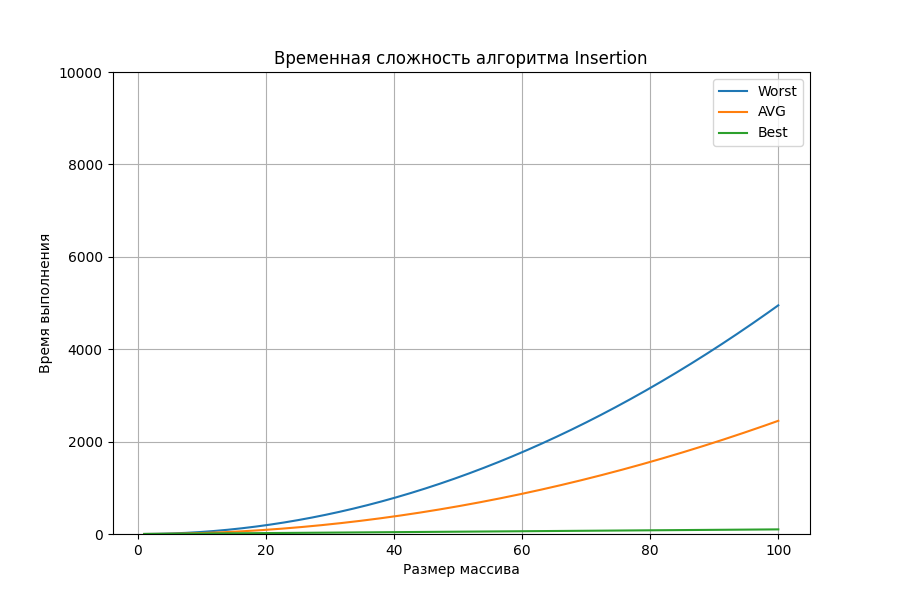
Алгоритм сортировки вставками начинается с прохода по массиву, начиная со второго элемента. На каждой итерации сохраняется текущий элемент (ключ) и начинается проверка его позиции в уже отсортированной части массива, расположенной слева. Если ключ меньше элемента, с которым он сравнивается, то этот элемент сдвигается вправо, чтобы освободить место для ключа. Этот процесс продолжается до тех пор, пока ключ не найдет свое место в отсортированной части. В результате алгоритм постепенно формирует отсортированный массив, перемещая элементы по мере необходимости, пока все элементы не окажутся на своих правильных местах.

1. Устойчивость сортировки :  
   Сортировка вставками является устойчивым алгоритмом. Это означает, что если два элемента равны, их относительный порядок в отсортированном массиве останется таким же, как и в исходном массиве. Устойчивость достигается благодаря тому, что при вставке элемента ключа в уже отсортированную часть массива элементы с одинаковыми значениями не перемещаются относительно друг друга.
2. Расчёт функции временной сложности и ее асимптотическая оценка:  
   **Худший**Сортировка происходит путем вставки элемента в отсортированную часть массива. Так как нам придется сравнивать каждый новый элемент с элементами правой части, то рекуррентная формула будет равна:  
   , где где это число отсортированных элементов. Таким образом временная сложность будет равна:

**Средний**В среднем случае, мы сравниваем вставляемый элемент не со всеми элементами отсортированного массива, поэтому возьмем случайное k равное числу элементов левой части, с которыми мы не сравнивали добавляемый элемент. Тогда рекуррентное выражение примет вид: . Где это число сравнений с отсортированной частью. Тогда временная сложность равна:

**Лучший**В лучшем случай добавляемый элемент сравнивается только с одним из отсортированной части, поэтому внутренний цикл завершает работу за .

Тогда при сортировка ограничивается добавлением элементов в отсортированный массив за . Пространственная сложность О(1)

1. Расчет функции пространственной сложности и ее асимптотическая оценка  
   Для лучшего случая - , поскольку никаких дополнительных рекурсивных вызовов или инициализаций массива не происходит  
   Для худшего случая - , поскольку никаких дополнительных рекурсивных вызовов или инициализаций массива не происходит  
   Для среднего случая - , поскольку никаких дополнительных рекурсивных вызовов или инициализаций массива не происходит
2. Графики функции временной сложности для разных случаев:  
   
   1. **Сортировка пузырем**
3. Описание алгоритма

Алгоритм сортировки пузырем начинает с прохода по массиву, где на каждой итерации сравниваются соседние элементы. Если текущий элемент больше следующего, они меняются местами, что позволяет большему элементу "всплыть" к концу массива. Процесс продолжается до тех пор, пока не будет сделано полное прохождение по массиву без обменов, что указывает на то, что массив отсортирован. Каждый проход по массиву сокращается на одну позицию, поскольку последний элемент уже находится на своем месте. Таким образом, алгоритм постепенно упрощает задачу, пока весь массив не окажется отсортирован.

1. Устойчивость сортировки :  
   Сортировка пузырем не является устойчивым алгоритмом. Это означает, что при наличии элементов с одинаковыми значениями их относительный порядок может измениться в результате сортировки. Например, если два элемента равны и один из них появляется раньше другого в исходном массиве, после сортировки их порядок может поменяться.
2. Расчёт функции временной сложности и ее асимптотическая оценка:  
   **Худший**Рекуррентное выражение для временной сложности в худшем случае будет выглядеть так:   
   Развивая рекуррентное выражение, подставляя значения, получаем:  
    И так далее, пока не дойдем до базового случая :  
   Сумма 1+2+3+…+(n-1) является суммой арифметической прогрессии и может быть вычислена по формуле   
   Таким образом: T(n) = O()

**Средний**Рекуррентное выражение для среднего случая будет выглядеть приблизительно также, как и для худшего. Так как в среднем случае итераций внутреннего цикла меньше n-1 раз. . Таким образом : T(n) = O()

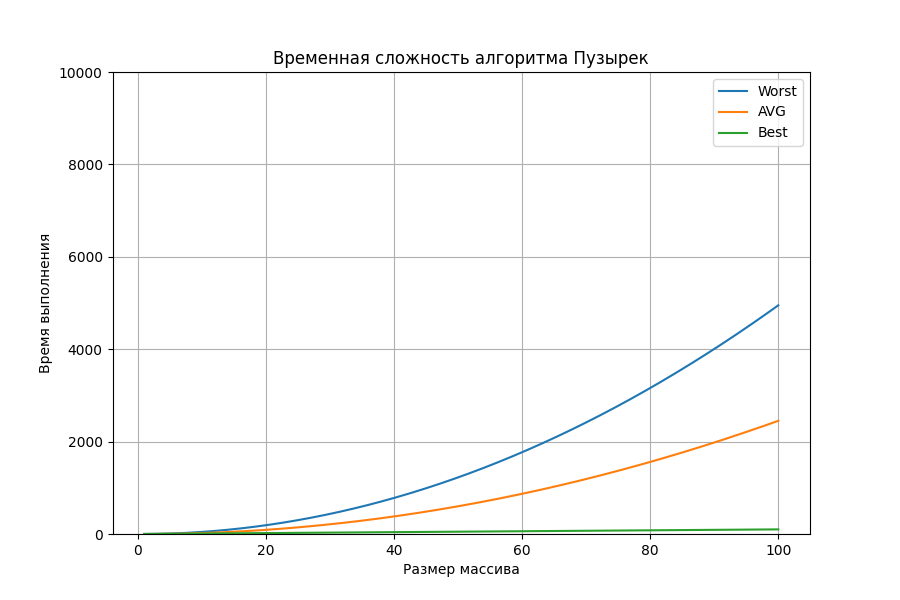
**Лучший**В лучшем случае внутренние итерации происходить не будут, так как элементы уже стоят в верном порядке, поэтому сложность алгоритма в этом случае зависит от числа итераций внешнего цикла и равно n.   
Таким образом

1. Расчет функции пространственной сложности и ее асимптотическая оценка  
   Для лучшего случая - , поскольку никаких дополнительных рекурсивных вызовов или инициализаций массива не происходит

Для худшего случая -, поскольку никаких дополнительных рекурсивных вызовов или инициализаций массива не происходит

Для среднего случая -, поскольку никаких дополнительных рекурсивных вызовов или инициализаций массива не происходит

1. Графики функции временной сложности для разных случаев:

****

* 1. **Сортировка выбором**

1. Описание алгоритма

Алгоритм сортировки выбором начинается с прохода по массиву, где на каждой итерации ищется минимальный элемент среди неотсортированной части. Для этого устанавливается текущий минимальный элемент и его индекс. Внутренний цикл сравнивает этот элемент с остальными элементами, и если находится элемент, меньший текущего минимума, обновляется индекс минимума. После завершения поиска минимального элемента он меняется местами с первым элементом неотсортированной части массива. Этот процесс повторяется для всех элементов, постепенно перемещая минимальные значения в начало массива, пока весь массив не будет отсортирован..

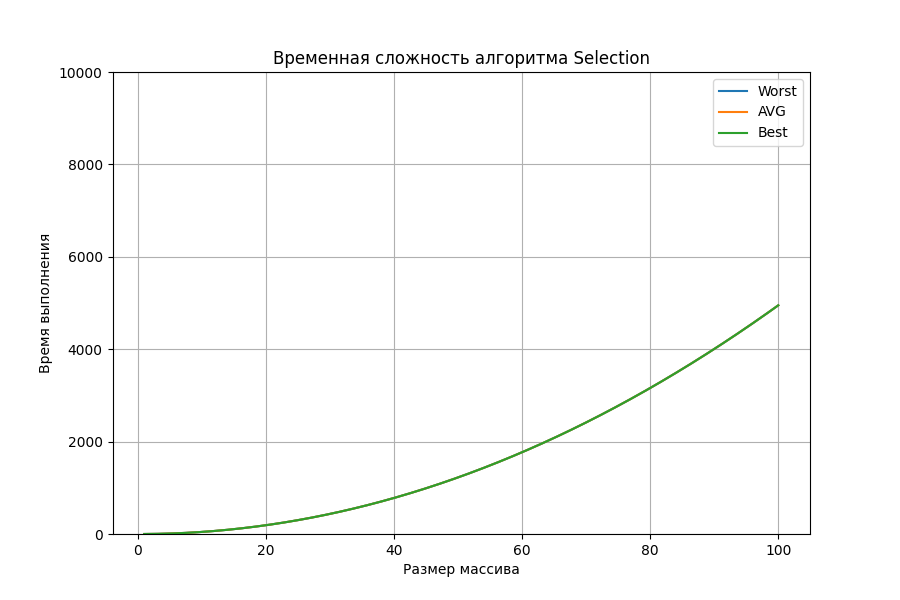
1. Устойчивость сортировки :  
   Сортировка выбором не является устойчивым алгоритмом. Это означает, что при наличии элементов с одинаковыми значениями их относительный порядок может измениться в результате сортировки. Например, если два одинаковых элемента находятся в разных местах в исходном массиве, после сортировки их порядок может оказаться другим.
2. Расчёт функции временной сложности и ее асимптотическая оценка:  
   Алгоритм данной сортировки подразумевает нахождения минимального значения из неотсортированной части и вставке в конец отсортированной, поэтому в любом случае(худшем, среднем и лучшем) рекуррентная формула будет равна: и соответственно временная сложность во всех случаях будет

Пространственная сложность O(1)

1. Расчет функции пространственной сложности и ее асимптотическая оценка  
   Для лучшего случая - , поскольку никаких дополнительных рекурсивных вызовов или инициализаций массива не происходит

Для худшего случая -, поскольку никаких дополнительных рекурсивных вызовов или инициализаций массива не происходит

Для среднего случая -, поскольку никаких дополнительных рекурсивных вызовов или инициализаций массива не происходит

1. Графики функции временной сложности для разных случаев:  
   
   1. **Сортировка быстрая**
2. Описание алгоритма

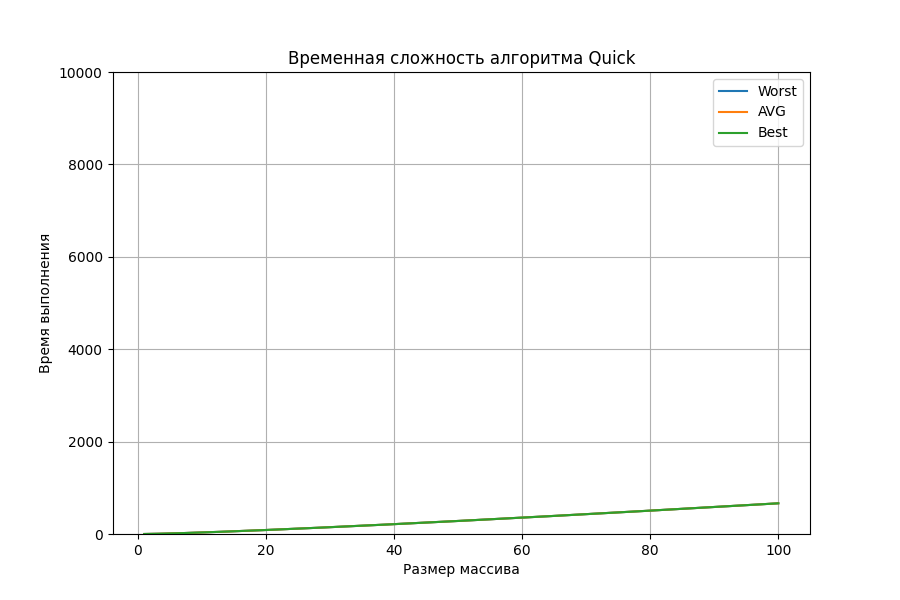
Алгоритм начинается с выбора опорного элемента, который обычно берется как средний элемент массива. Затем массив делится на две части: элементы, меньшие опорного, и элементы, большие опорного. Это достигается с помощью двух указателей, которые перемещаются по массиву, пока не встретятся. После разделения массив перестраивается так, чтобы все элементы слева от опорного были меньше него, а все элементы справа — больше. Затем алгоритм рекурсивно применяется к обеим частям массива, продолжается до тех пор, пока не останется частей размером один или ноль, что завершает сортировку. Таким образом, быстрая сортировка эффективно разбивает проблему на подзадачи, обеспечивая высокую производительность в среднем случае.

1. Устойчивость сортировки :  
   Быстрая сортировка не является устойчивым алгоритмом.
2. Расчёт функции временной сложности и ее асимптотическая оценка:  
   Суть метода заключается в поэтапном перемещении элементов массива относительно опорного. При его работе рекурсивно вызывается функция сортировки для половин исходного массива. Таким образом продолжается до момента когда длина обрабатываемой части массива на итерации не достигнет 2 или 3, тогда на каждой итерации необходимо будет проверять правильность положения n элементов, откуда значение рекурсивного выражения равно: Соответственно временная сложность данного алгоритма равна:

1. Расчет функции пространственной сложности и ее асимптотическая оценка  
   Для лучшего случая - , поскольку происходит хранение рекурсии , постоянного деления массива

Для худшего случая -, поскольку никаких дополнительных рекурсивных вызовов или инициализаций массива не происходит

Для среднего случая -, поскольку никаких дополнительных рекурсивных вызовов или инициализаций массива не происходит

1. Графики функции временной сложности для разных случаев  
   
   1. **Сортировка Шелла**
2. Описание алгоритма

Делим список на несколько групп, выбирая некоторый **шаг** (расстояние между элементами для сравнения)(последовательность Шелла, Прапа и Хиббарда). Сортируем каждую группу с помощью сортировки вставками. Уменьшаем шаг и повторяем процесс сортировки для новых групп. Постепенно уменьшаем шаг до 1, когда элементы сравниваются и сортируются уже по соседству, как в обычной сортировке вставками.

Когда шаг становится 1, список будет почти отсортирован, и финальная сортировка завершит процесс.

1. Устойчивость сортировки :  
   Сортировка Шелла является нестабильной. Это означает, что при сортировке элементов с одинаковыми ключами их относительный порядок может измениться. Например, если в массиве есть два одинаковых элемента, алгоритм не гарантирует, что они останутся в исходном порядке после сортировки. Нестабильность возникает из-за того, что элементы могут перемещаться на значительные расстояния друг от друга в процессе сортировки.
2. Расчёт функции временной сложности и ее асимптотическая оценка:  
   Сортировка Шелла является доработкой сортировки вставками. Используя сравнение элементов с некоторым шагом, добиваемся перераспределения элементов таким образом, чтобы большинство из них оказалось близко к позициям в отсортированном массиве. Уменьшив шаг до 1, завершаем сортировку. Так как последний шаг алгоритма происходит в почти отсортированном массиве, то можно сказать, что его временная сложность равна .При любых входных данных данный алгоритм производит не более чем сравнений при прохождении с любым шагом.

Shell Hibbard:

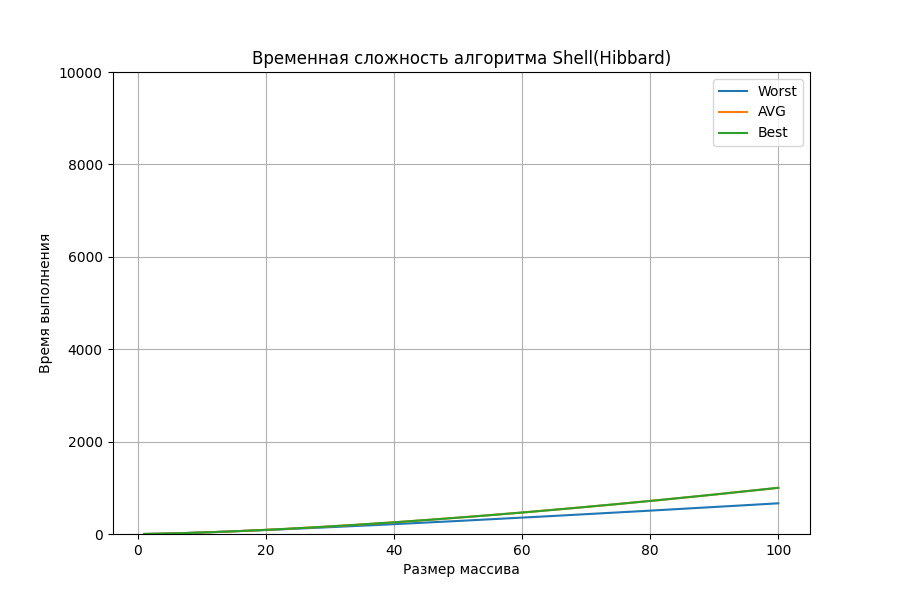
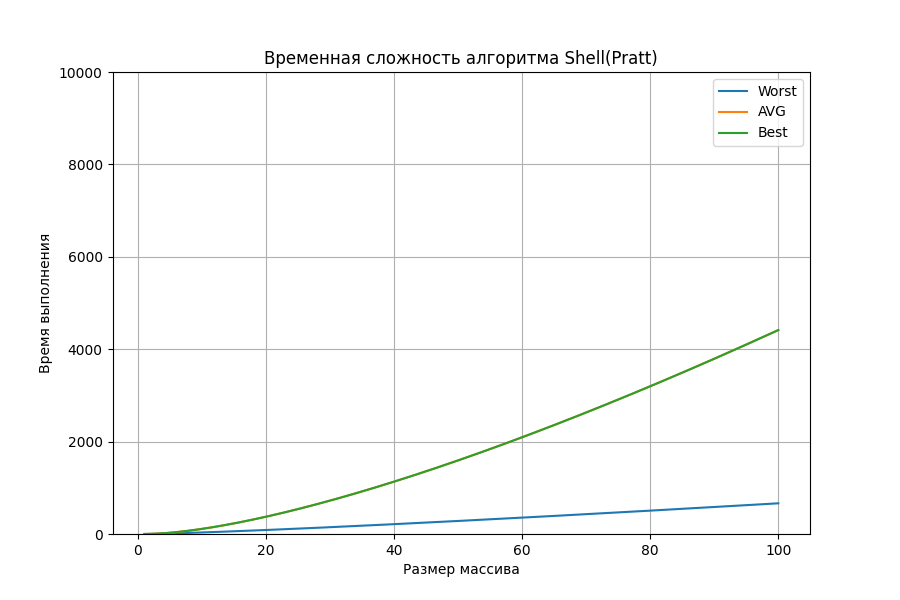
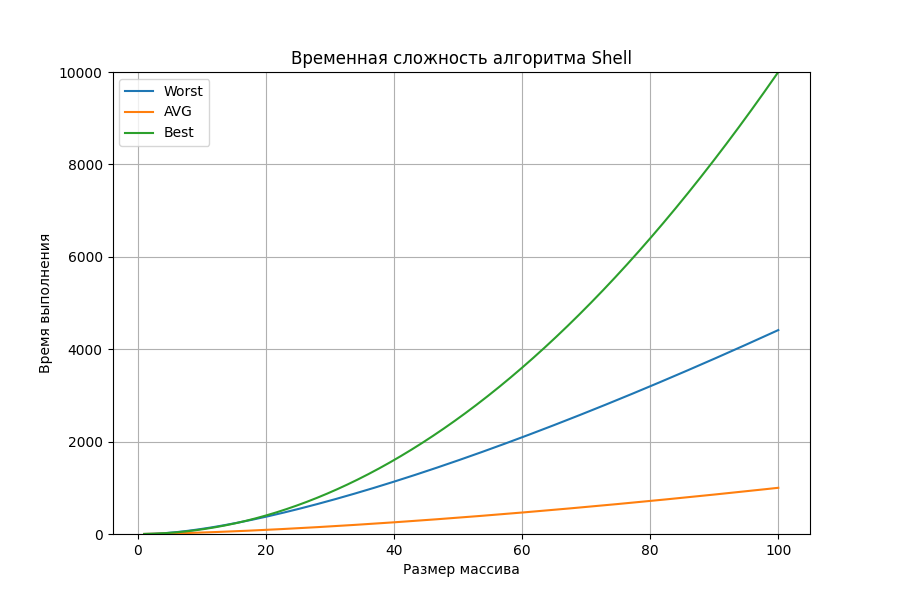
Shell Pratt:

1. Расчет функции пространственной сложности и ее асимптотическая оценка  
   Для лучшего случая - - ,поскольку никаких дополнительных рекурсивных вызовов или инициализаций массива не происходит

Для худшего случая -- поскольку никаких дополнительных рекурсивных вызовов или инициализаций массива не происходит

Для среднего случая -- , поскольку никаких дополнительных рекурсивных вызовов или инициализаций массива не происходит

1. Графики функции временной сложности для разных случаев

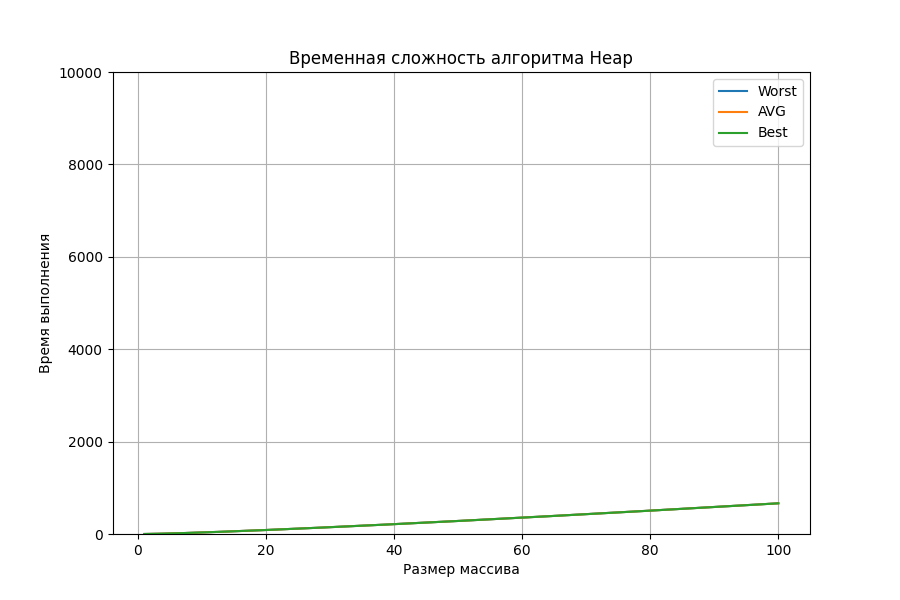


* 1. **Сортировка пирамидальная**

1. Описание алгоритма

Алгоритм сортировки кучей организует массив в структуру, называемую кучей, где каждый элемент родительского узла больше своих дочерних. Сначала массив преобразуется в такую кучу, начиная с нижних уровней и поднимаясь к корню. Затем, чтобы отсортировать элементы, наибольший элемент (корень кучи) меняется местами с последним элементом массива, и процесс восстановления кучи продолжается для уменьшенного массива. Этот процесс повторяется, пока все элементы не окажутся в правильном порядке. В результате массив сортируется по возрастанию, и алгоритм работает достаточно эффективно, обеспечивая хорошую производительность.

1. Устойчивость сортировки :  
   Сортировка кучей является неустойчивым алгоритмом.
2. Расчёт функции временной сложности и ее асимптотическая оценка:  
     
   При данной сортировке формируется бинарная куча, в которой корень является самым большим или самым маленьким числом. При этом если добавляемый элемент больше текущего корня поддерева, то он заменяет его и освободившийся элемент сравнивается с дочерними. Таким образом при вставке элемента будет произведено число сравнений равное высоте поддерева, то есть . Для вставки элемента на нижний уровень производиться меньше операций сравнения, поэтому временная сложность построения кучи равна . Отсортированный массив получается при удалении корня кучи и перестройке структуры кучи. Для удаления элемента требуется операций. И значит временная сложность . Так как построение кучи и извлечение элементов из нее не зависят от порядка элементов во входном массиве, то временная сложность постоянна
3. Расчет функции пространственной сложности и ее асимптотическая оценка  
   Пространственная сложность O(1) так как метод итерационный и работает сразу с массивом. Если строить кучу, то пространственная сложность O(n)
4. Графики функции временной сложности для разных случаев:

****

* 1. **Сортировка слиянием**

1. Описание алгоритма

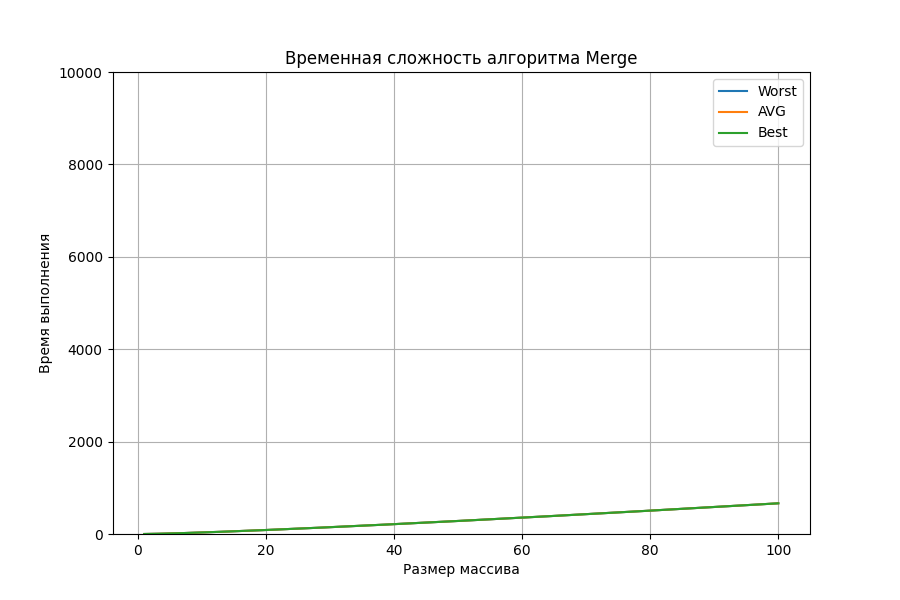
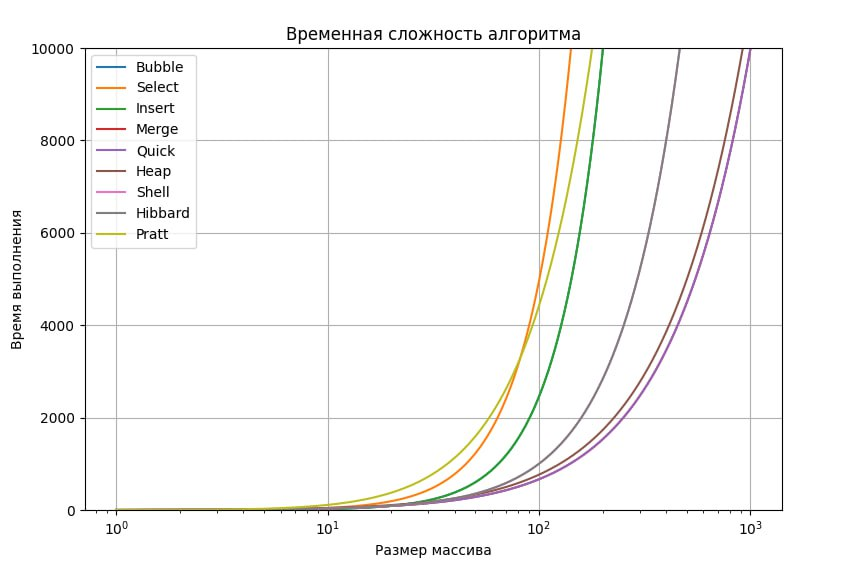
Суть алгоритма в сортировке половинок массива и потом их слияния. Так как после разделения массива на 2 надо отсортировать обе части то в рекуррентном выражении есть множитель 2. После сортировок частей нужно их объединить, для этого требуется n операций, . Так как массив каждый раз делиться на 2, до тех пор, пока в каждой части не останется по 2 элемента, то всего разделений будет . Таким образом сортировка имеет временную сложность

Пространственная сложность .

1. Устойчивость сортировки :  
   Сортировка слиянием является устойчивым алгоритмом.
2. Расчет функции пространственной сложности и ее асимптотическая оценка

Пространственная сложность .

1. Графики функции временной сложности для разных случаев:

**** Графики функции временной сложности всех сортировок в среднем случае

Исходя из теоретических данных, можно предположить что самой быстрой сортировой в среднем случае при размере массива > 100 000 будет сортировка Быстрая и Слияния (quick , merge)

**Практическая часть.**

Для выполнения практической части использовался язык Python, где были написаны все сортировки и функции заполнения, отрисовки графиков. Для каждой сортировки и для каждого заполнения считалось время работы функции на определенном интервале элементов, результаты были представлены в таблице и на графиках.

**Результаты эксперимента:**

**Сортировка выбором.**

Отсортированный массив

|  |  |
| --- | --- |
| Количество элементов | Время сортировки(сек) |
| 0 | 0,000000 |
| 2000 | 0,055668 |
| 5000 | 0,335159 |
| 8000 | 0,861265 |
| 11000 | 1,685833 |
| 13000 | 2,210437 |
| 16000 | 3,350304 |
| 18000 | 4,297839 |
| 20000 | 5,275854 |
| 23000 | 7,002310 |
| 27000 | 9,845957 |
| 32000 | 14,055262 |

Таблица 1 – Время работы сортировки выбором (отсортированный массив)

Почти отсортированный массив

|  |  |
| --- | --- |
| Количество элементов | Время сортировки(сек) |
| 0 | 0,000000 |
| 2000 | 0,059248 |
| 5000 | 0,347481 |
| 8000 | 0,884697 |
| 11000 | 1,709292 |
| 13000 | 2,368438 |
| 16000 | 3,618246 |
| 18000 | 4,479177 |
| 20000 | 5,728683 |
| 23000 | 7,294148 |
| 27000 | 10,062837 |
| 32000 | 14,326499 |

Таблица 2 – Время работы сортировки выбором (почти отсортированный массив)

Отсортированный в сторону убывания массив

|  |  |
| --- | --- |
| Количество элементов | Время сортировки(сек) |
| 0 | 0,000000 |
| 2000 | 0,063847 |
| 5000 | 0,376495 |
| 8000 | 0,956267 |
| 11000 | 1,829663 |
| 13000 | 2,549348 |
| 16000 | 3,834219 |
| 18000 | 4,860902 |
| 20000 | 5,996847 |
| 23000 | 7,941182 |
| 27000 | 10,881551 |
| 32000 | 15,706555 |

Таблица 3 – Время работы сортировки выбором (массив отсортированный в сторону убывания)

Случайный массив

|  |  |
| --- | --- |
| Количество элементов | Время сортировки(сек) |
| 0 | 0,000000 |
| 2000 | 0,058166 |
| 5000 | 0,334454 |
| 8000 | 0,864836 |
| 11000 | 1,608900 |
| 13000 | 2,246584 |
| 16000 | 3,412827 |
| 18000 | 4,335800 |
| 20000 | 5,501855 |
| 23000 | 7,111799 |
| 27000 | 10,128178 |
| 32000 | 14,193541 |

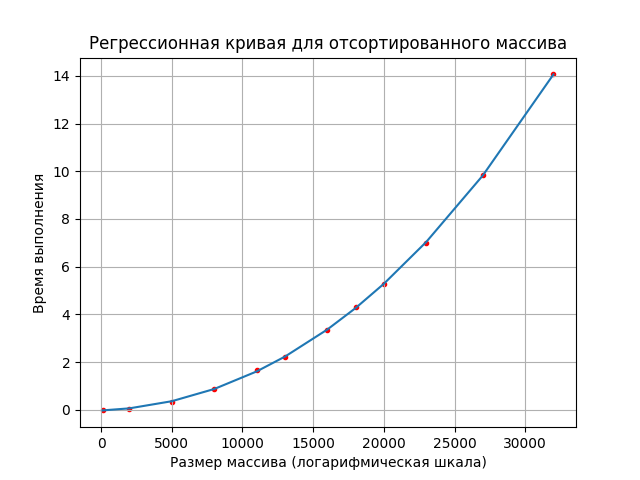
****Таблица 4 – Время работы сортировки выбором (случайный массив)

Рисунок 13 – График сортировки выбором для отсортированного массива

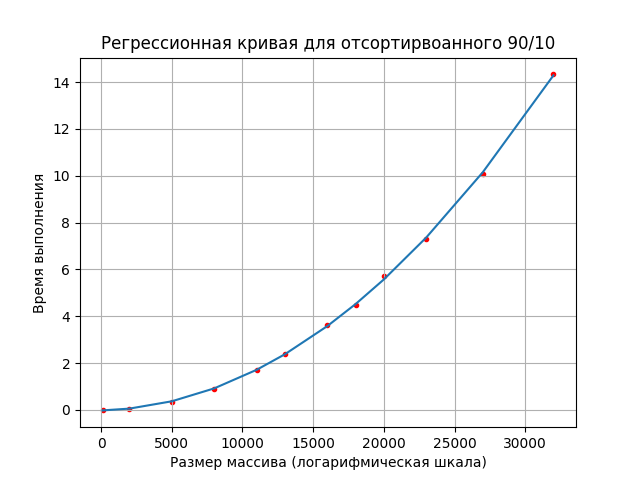
****

Рисунок 12 – График сортировки выбором для почти отсортированного массива

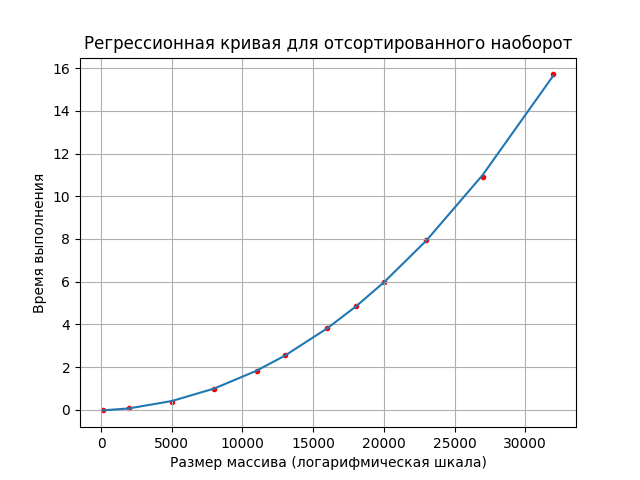
****

Рисунок 13 – График сортировки выбором для обратно отсортированного массива

****

Рисунок 14 – График сортировки выбором для случайного порядка

**Сортировка вставками.**

Случайный массив

|  |  |
| --- | --- |
| Количество элементов | Время сортировки(сек) |
| 0 | 0,000000 |
| 2000 | 0,061781 |
| 5000 | 0,356690 |
| 8000 | 0,916287 |
| 11000 | 1,825581 |
| 13000 | 2,349799 |
| 16000 | 3,607297 |
| 18000 | 4,607056 |
| 20000 | 5,900614 |
| 23000 | 7,601359 |
| 27000 | 10,518216 |
| 32000 | 15,205527 |

Таблица 5 – Время работы сортировки вставками (случайный порядок)

Отсортированный массив

|  |  |
| --- | --- |
| Количество элементов | Время сортировки(сек) |
| 0 | 0,000000 |
| 2000 | 0,000140 |
| 5000 | 0,000383 |
| 8000 | 0,000680 |
| 11000 | 0,000863 |
| 13000 | 0,001212 |
| 16000 | 0,001286 |
| 18000 | 0,001343 |
| 20000 | 0,001484 |
| 23000 | 0,001771 |
| 27000 | 0,002342 |
| 32000 | 0,002325 |

Таблица 6 – Время работы сортировки вставками (отсортированный массив)

Почти отсортированный массив

|  |  |
| --- | --- |
| Количество элементов | Время сортировки(сек) |
| 0 | 0,000000 |
| 2000 | 0,015125 |
| 5000 | 0,086947 |
| 8000 | 0,232284 |
| 11000 | 0,404753 |
| 13000 | 0,578803 |
| 16000 | 0,853899 |
| 18000 | 1,077078 |
| 20000 | 1,337994 |
| 23000 | 1,747283 |
| 27000 | 2,499707 |
| 32000 | 3,411598 |

Таблица 7 – Время работы сортировки вставками (почти отсортированный массив)

Случайный массив

|  |  |
| --- | --- |
| Количество элементов | Время сортировки(сек) |
| 0 | 0,000000 |
| 2000 | 0,107570 |
| 5000 | 0,699104 |
| 8000 | 1,832356 |
| 11000 | 3,383450 |
| 13000 | 4,637608 |
| 16000 | 7,087008 |
| 18000 | 8,912993 |
| 20000 | 11,464651 |
| 23000 | 14,565425 |
| 27000 | 19,969513 |
| 32000 | 28,501170 |

Таблица 8 – Время работы сортировки вставками (обратный массив)

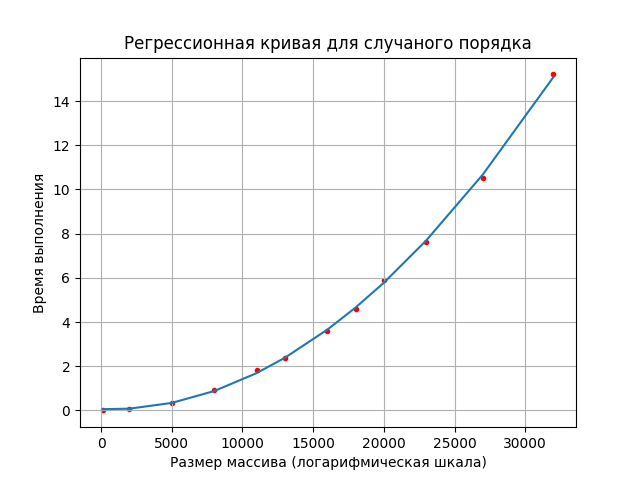
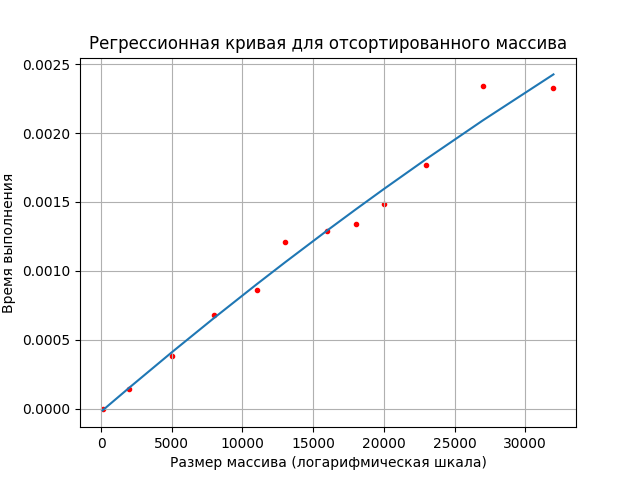


Рисунок 15 – График сортировки вставками для случайного порядка



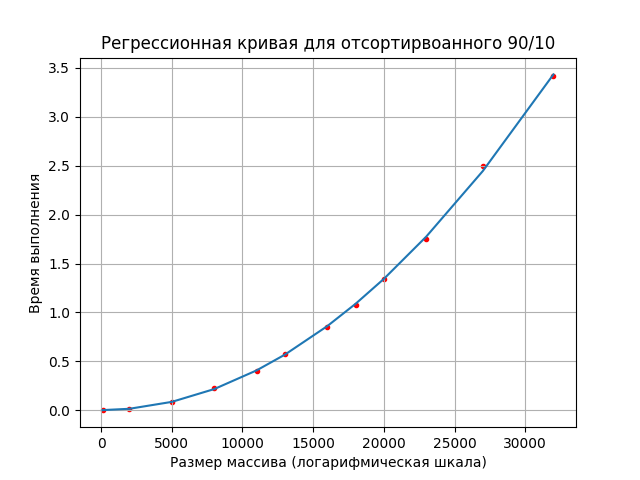
Рисунок 16 – График сортировки вставками для отсортированного массива

Рисунок 17 – График сортировки вставками почти отсортированного массива

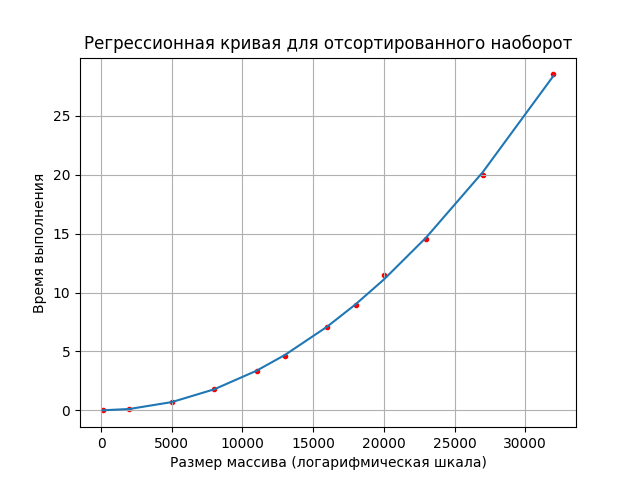


Рисунок 18 – График сортировки вставками для отсортированного массива

**Сортировка пузырьком.**

Произвольный массив

|  |  |
| --- | --- |
| Количество элементов | Время сортировки(сек) |
| 0 | 0,000000 |
| 2000 | 0,131545 |
| 5000 | 0,815465 |
| 8000 | 2,125261 |
| 11000 | 4,018430 |
| 13000 | 5,615871 |
| 16000 | 8,680333 |
| 18000 | 13,636754 |
| 20000 | 13,636754 |
| 23000 | 18,443403 |
| 27000 | 24,839962 |
| 32000 | 35,337024 |

Таблица 9 – Время работы сортировки пузырьком (произвольный массив)

Отсортированный массив

|  |  |
| --- | --- |
| Количество элементов | Время сортировки(сек) |
| 0 | 0,000000 |
| 2000 | 0,000078 |
| 5000 | 0,000207 |
| 8000 | 0,000337 |
| 11000 | 0,000471 |
| 13000 | 0,000532 |
| 16000 | 0,000677 |
| 18000 | 0,000743 |
| 20000 | 0,001243 |
| 23000 | 0,001374 |
| 27000 | 0,001324 |
| 32000 | 0,001453 |

Таблица 10 – Время работы сортировки пузырьком (отсортированный массив)

Почти отсортированный массив

|  |  |
| --- | --- |
| Количество элементов | Время сортировки(сек) |
| 0 | 0,000000 |
| 2000 | 0,089244 |
| 5000 | 0,558720 |
| 8000 | 1,377035 |
| 11000 | 2,629283 |
| 13000 | 3,892609 |
| 16000 | 5,647541 |
| 18000 | 7,099976 |
| 20000 | 9,149449 |
| 23000 | 11,564336 |
| 27000 | 16,207333 |
| 32000 | 22,852360 |

Таблица 11 – Время работы сортировки пузырьком (почти отсортированный убывания)

Обратный массив

|  |  |
| --- | --- |
| Количество элементов | Время сортировки(сек) |
| 0 | 0,000000 |
| 2000 | 0,167989 |
| 5000 | 1,063803 |
| 8000 | 2,696286 |
| 11000 | 5,103778 |
| 13000 | 7,294161 |
| 16000 | 11,150679 |
| 18000 | 13,946140 |
| 20000 | 17,168230 |
| 23000 | 22,258460 |
| 27000 | 31,898123 |
| 32000 | 43,757604 |

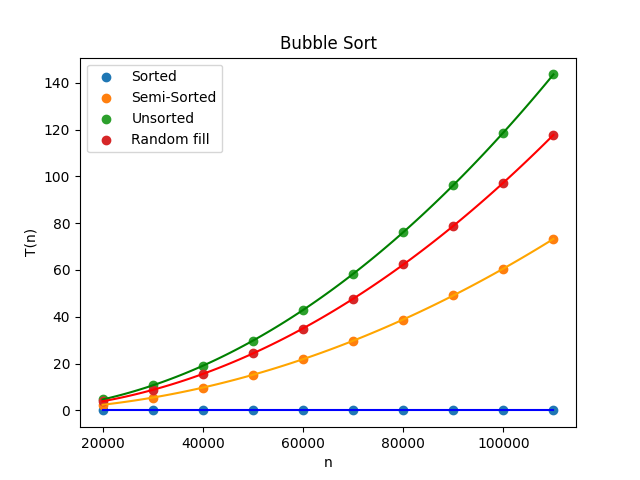
Таблица 12 – Время работы сортировки пузырьком (обратный массив) 

Рисунок 19 – График сортировки пузырьком для произвольного массива

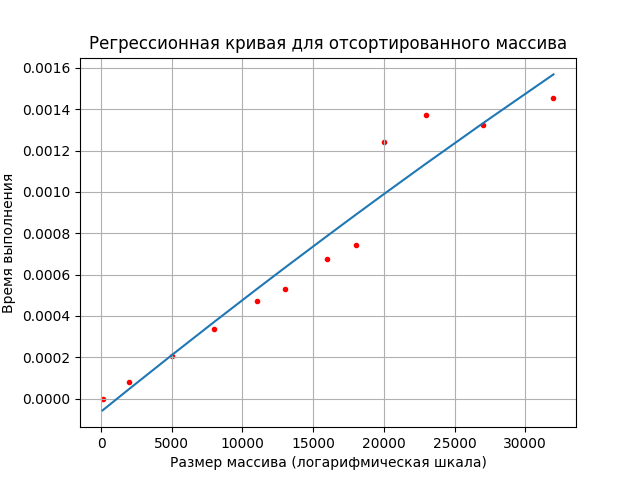
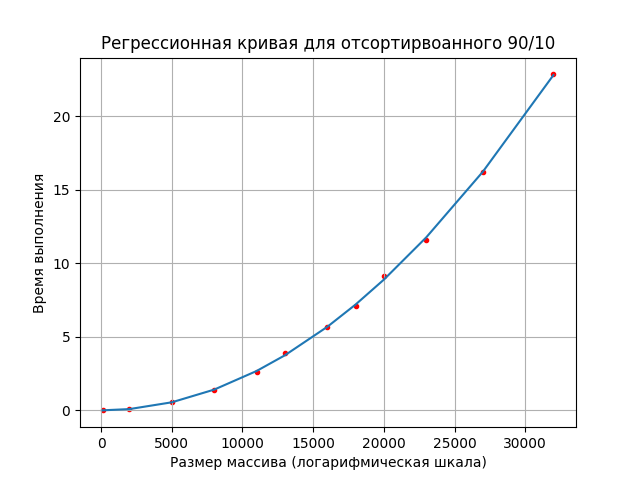
Рисунок 20 – График сортировки пузырьком для отсортированного массива

Рисунок 21 – График сортировки пузырьком для почти отсортированного массива

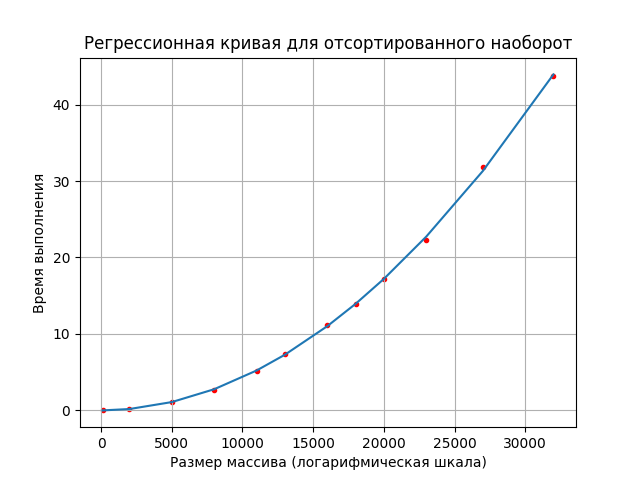


Рисунок 22 – График сортировки пузырьком для отсортированного наоборот

**Сортировка слиянием.**

Произвольный массив

|  |  |
| --- | --- |
| Количество элементов | Время сортировки(сек) |
| 0 | 0,000000 |
| 50000 | 0,078979 |
| 100000 | 0,162885 |
| 150000 | 0,246588 |
| 200000 | 0,350348 |
| 250000 | 0,434269 |
| 300000 | 0,530345 |
| 350000 | 0,633283 |
| 400000 | 0,724604 |
| 450000 | 0,806822 |
| 500000 | 0,916867 |
| 550000 | 1,015946 |
| 600000 | 1,129379 |
| 650000 | 1,223267  1,474543 |
| 750000 | 1,502233 |
| 800000 | 1,506771 |
| 850000 | 1,614401 |
| 900000 | 1,696352 |
| 950000 | 2,256161 |
| 1000000 | 2,715006 |

Таблица 13 – Время работы сортировки слиянием (произвольный массив)

Отсортированный массив

|  |  |
| --- | --- |
| Количество элементов | Время сортировки(сек) |
| 0 | 0,000000 |
| 50000 | 0,071452 |
| 100000 | 0,154513 |
| 150000 | 0,250257 |
| 200000 | 0,313085 |
| 250000 | 0,391682 |
| 300000 | 0,475574 |
| 350000 | 0,666118 |
| 400000 | 0,682522 |
| 450000 | 0,772061 |
| 500000 | 0,900672 |
| 550000 | 0,953385 |
| 600000 | 1,219520 |
| 650000  700000 | 1,212640 1,309860 |
| 750000 | 1,337493 |
| 800000 | 1,379848 |
| 850000 | 1,548380 |
| 900000 | 1,644419 |
| 950000 | 1,700745 |
| 1000000 | 1,841999 |

Таблица 14 – Время работы сортировки слиянием (отсортированный массив)

Почти отсортированный в сторону убывания массив

|  |  |
| --- | --- |
| Количество элементов | Время сортировки(сек) |
| 0 | 0,000000 |
| 50000 | 0,075912 |
| 100000 | 0,164239 |
| 150000 | 0,255805 |
| 200000 | 0,351919 |
| 250000 | 0,478263 |
| 300000 | 0,607075 |
| 350000 | 0,740298 |
| 400000 | 1,072933 |
| 450000 | 1,237851 |
| 500000 | 1,271442 |
| 550000 | 1,366473 |
| 600000 | 1,787323 |
| 650000  700000 | 1,941406 2,097948 |
| 750000 | 2,207943 |
| 800000 | 1,940445 |
| 850000 | 2,256539 |
| 900000 | 2,438372 |
| 950000 | 2,427404 |
| 1000000 | 2,423642 |

Таблица 15 – Время работы сортировки слиянием (массив почти отсортированный)

Случайный массив

|  |  |
| --- | --- |
| Количество элементов | Время сортировки(сек) |
| 0 | 0,000000 |
| 50000 | 0,074425 |
| 100000 | 0,146248 |
| 150000 | 0,230458 |
| 200000 | 0,305833 |
| 250000 | 0,391134 |
| 300000 | 0,623638 |
| 350000 | 0,660766 |
| 400000 | 0,747947 |
| 450000 | 0,739175 |
| 500000 | 0,826752 |
| 550000 | 1,103458 |
| 600000 | 1,292859 |
| 650000  700000 | 1,095150 1,315258 |
| 750000 | 1,540532 |
| 800000 | 1,369669 |
| 850000 | 1,444050 |
| 900000 | 1,552594 |
| 950000 | 1,630768 |
| 1000000 | 1,833318 |

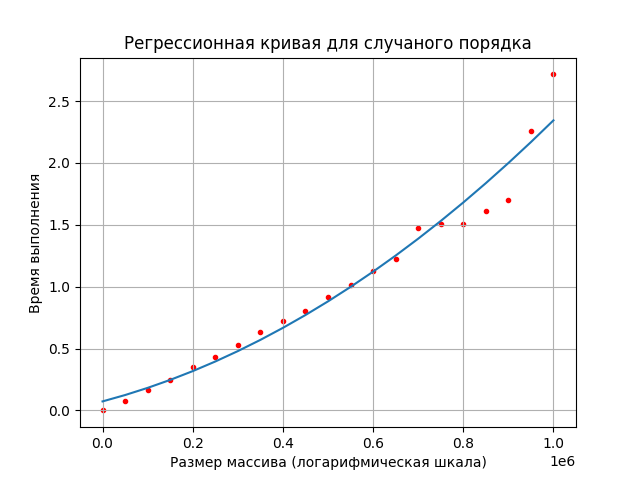
Таблица 16 – Время работы сортировки слиянием (случайный массив)

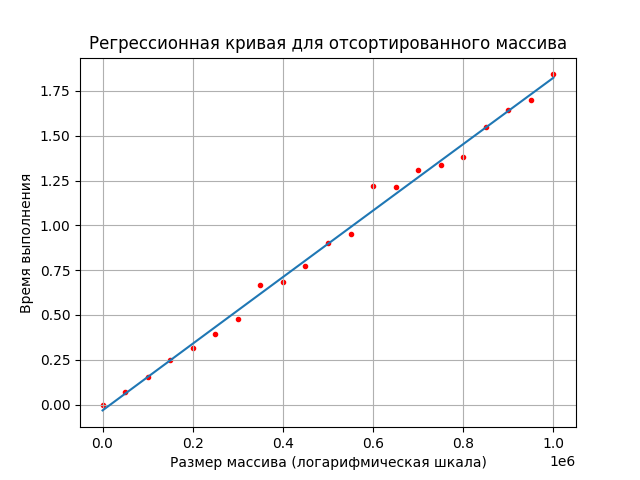
Рисунок 23– График сортировки слиянием для случайного порядка

Рисунок 24– График сортировки слиянием для отсортированного массива

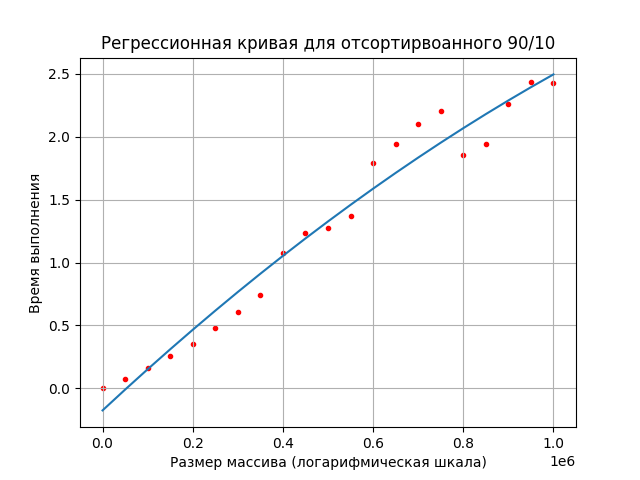


Рисунок 25– График сортировки слиянием для почти отсортированного

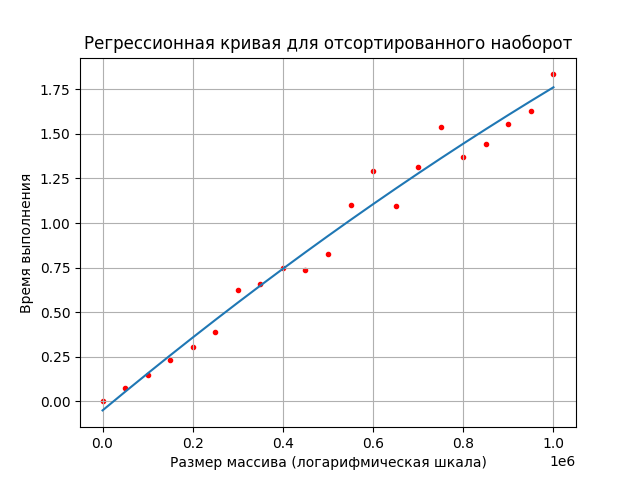


Рисунок 26– График сортировки слиянием для отсортированного наоборот

**Сортировка Шелла.**

Случайный массив

|  |  |
| --- | --- |
| Количество элементов | Время сортировки(сек) |
| 0 | 0,000077 |
| 50000 | 0,186144 |
| 100000 | 0,439323 |
| 150000 | 0,654000 |
| 200000 | 1,015476 |
| 250000 | 1,330783 |
| 300000 | 1,663865 |
| 350000 | 2,578834 |
| 400000 | 3,803804 |
| 450000 | 3,301714 |
| 500000 | 3,493172 |
| 550000 | 3,730123 |
| 600000 | 4,200051 |
| 650000  700000 | 4,606421 4,856339 |
| 750000 | 5,381111 |
| 800000 | 6,420922 |
| 850000 | 6,238230 |
| 900000 | 6,867711 |
| 950000 | 7,387278 |
| 1000000 | 8,074786 |

Таблица 17 – Время работы сортировки Шелла (случайный массив)

Отсортированный массив

|  |  |
| --- | --- |
| Количество элементов | Время сортировки(сек) |
| 0 | 0,000036 |
| 50000 | 0,058746 |
| 100000 | 0,123217 |
| 150000 | 0,201253 |
| 200000 | 0,264781 |
| 250000 | 0,329010 |
| 300000 | 0,420277 |
| 350000 | 0,492368 |
| 400000 | 0,560478 |
| 450000 | 0,628717 |
| 500000 | 0,718345 |
| 550000 | 0,850931 |
| 600000 | 0,990092 |
| 650000  700000 | 0,995633 1,194934 |
| 750000 | 1,351316 |
| 800000 | 1,185491 |
| 850000 | 1,254340 |
| 900000 | 1,386354 |
| 950000 | 1,577590 |
| 1000000 | 1,560049 |

Таблица 18 – Время работы сортировки Шелла (Отсортированный массив)

Почти Отсортированный массив

|  |  |
| --- | --- |
| Количество элементов | Время сортировки(сек) |
| 0 | 0,000059 |
| 50000 | 0,152297 |
| 100000 | 0,342117 |
| 150000 | 0,529883 |
| 200000 | 0,823248 |
| 250000 | 1,142145 |
| 300000 | 1,332878 |
| 350000 | 1,566535 |
| 400000 | 1,982152 |
| 450000 | 2,137640 |
| 500000 | 2,449235 |
| 550000 | 2,689428 |
| 600000 | 2,867935 |
| 650000  700000 | 3,355466 3,651039 |
| 750000 | 3,936741 |
| 800000 | 4,410255 |
| 850000 | 4,854069 |
| 900000 | 5,497771 |
| 950000 | 6,614773 |
| 1000000 | 6,488420 |

Таблица 19 – Время работы сортировки Шелла (почти отсортированный массив)

Массив в обратном порядке

|  |  |
| --- | --- |
| Количество элементов | Время сортировки(сек) |
| 0 | 0,000054 |
| 50000 | 0,102579 |
| 100000 | 0,211320 |
| 150000 | 0,321323 |
| 200000 | 0,433504 |
| 250000 | 0,522385 |
| 300000 | 0,694746 |
| 350000 | 0,777846 |
| 400000 | 0,910319 |
| 450000 | 0,987128 |
| 500000 | 1,103065 |
| 550000 | 1,306200 |
| 600000 | 1,387799 |
| 650000  700000 | 1,521134 1,654675 |
| 750000 | 1,794404 |
| 800000 | 1,932014 |
| 850000 | 2,601143 |
| 900000 | 2,099815 |
| 950000 | 2,269828 |
| 1000000 | 2,345772 |

Таблица 20 – Время работы сортировки Шелла (Массив в обратном порядке)

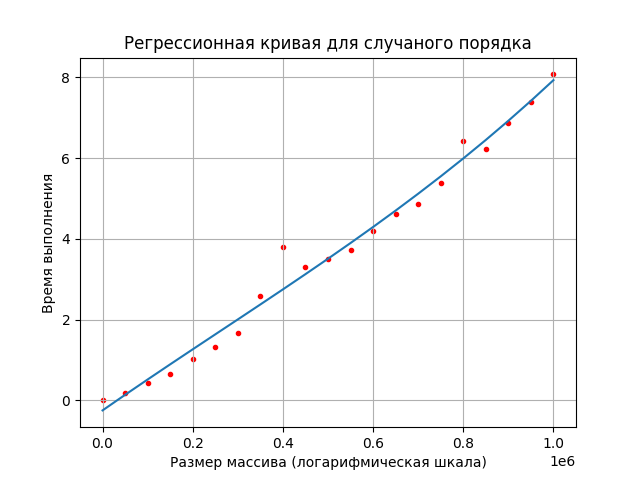


Рисунок 27 – График сортировки Шелла для случайного массива

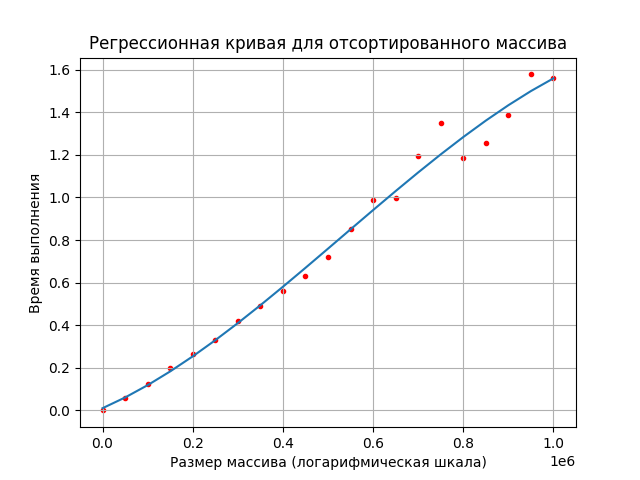


Рисунок 28 – График сортировки Шелла для отсортированного массива

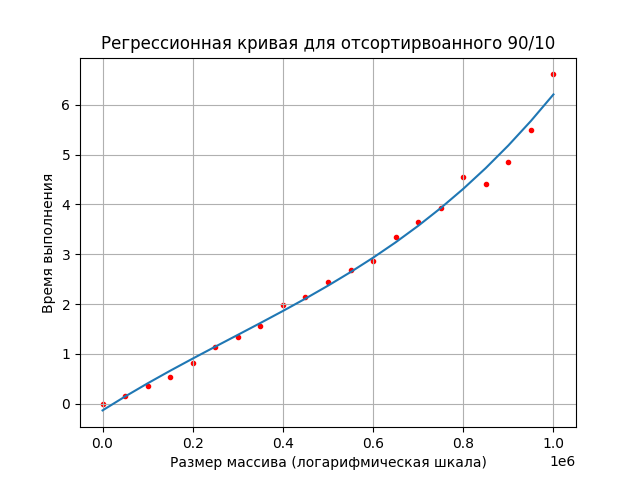


Рисунок 29 – График сортировки Шелла для почти отсортированного массива

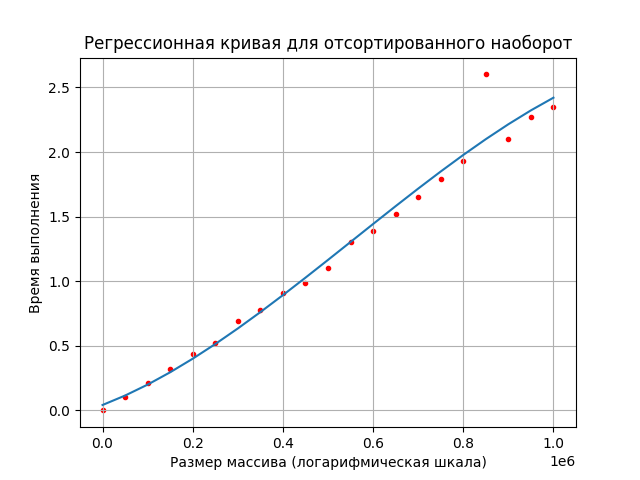


Рисунок 30 – График сортировки Шелла для отсортированного наоборот

**Сортировка Шелла (последовательность Хиббарда).**

Произвольный массив

|  |  |
| --- | --- |
| Количество элементов | Время сортировки(сек) |
| 0 | 0,000041 |
| 50000 | 0,063239 |
| 100000 | 0,132676 |
| 150000 | 0,212144 |
| 200000 | 0,300592 |
| 250000 | 0,412552 |
| 300000 | 0,531452 |
| 350000 | 0,871057 |
| 400000 | 1,152255 |
| 450000 | 1,232681 |
| 500000 | 1,225530 |
| 550000 | 1,295948 |
| 600000 | 1,550012 |
| 650000  700000 | 1,580437 1,799177 |
| 750000 | 1,960069 |
| 800000 | 2,160648 |
| 850000 | 2,158902 |
| 900000 | 2,557498 |
| 950000 | 2,786286 |
| 1000000 | 2,958907 |

Таблица 21 – Время работы сортировки Шелла с последовательностью Хиббарда (произвольный массив)

отсортированный массив

|  |  |
| --- | --- |
| Количество элементов | Время сортировки(сек) |
| 0 | 0,000034 |
| 50000 | 0,058335 |
| 100000 | 0,121593 |
| 150000 | 0,191912 |
| 200000 | 0,256804 |
| 250000 | 0,327753 |
| 300000 | 0,400263 |
| 350000 | 0,475855 |
| 400000 | 0,547217 |
| 450000 | 0,626826 |
| 500000 | 0,720184 |
| 550000 | 0,816438 |
| 600000 | 0,913564 |
| 650000  700000 | 1,001647 1,168248 |
| 750000 | 1,239640 |
| 800000 | 1,180481 |
| 850000 | 1,231964 |
| 900000 | 1,462362 |
| 950000 | 1,689717 |
| 1000000 | 1,586900 |

Таблица 22 – Время работы сортировки Шелла с последовательностью Хиббарда (отсортированный массив)

Почти отсортированный массив

|  |  |
| --- | --- |
| Количество элементов | Время сортировки(сек) |
| 0 | 0,000035 |
| 50000 | 0,057675 |
| 100000 | 0,121735 |
| 150000 | 0,193187 |
| 200000 | 0,273419 |
| 250000 | 0,420367 |
| 300000 | 0,514983 |
| 350000 | 0,562331 |
| 400000 | 0,670103 |
| 450000 | 0,764409 |
| 500000 | 0,831801 |
| 550000 | 0,898213 |
| 600000 | 0,974999 |
| 650000  700000 | 1,078333 1,190432 |
| 750000 | 1,250703 |
| 800000 | 1,326608 |
| 850000 | 1,421486 |
| 900000 | 1,678237 |
| 950000 | 2,000397 |
| 1000000 | 2,458951 |

Таблица 23 – Время работы сортировки Шелла с последовательностью Хиббарда (массив отсортированный в сторону убывания)

Массив обратный

|  |  |
| --- | --- |
| Количество элементов | Время сортировки(сек) |
| 0 | 0,000041 |
| 50000 | 0,059419 |
| 100000 | 0,133025 |
| 150000 | 0,193264 |
| 200000 | 0,264433 |
| 250000 | 0,327733 |
| 300000 | 0,421829 |
| 350000 | 0,479947 |
| 400000 | 0,554633 |
| 450000 | 0,622569 |
| 500000 | 0,695256 |
| 550000 | 0,784172 |
| 600000 | 0,854459 |
| 650000  700000 | 0,927647 1,072604 |
| 750000 | 1,150404 |
| 800000 | 1,172481 |
| 850000 | 1,634115 |
| 900000 | 1,333344 |
| 950000 | 1,417841 |
| 1000000 | 1,499050 |

Таблица 24 – Время работы сортировки Шелла с последовательностью Хиббарда (массив обратный)

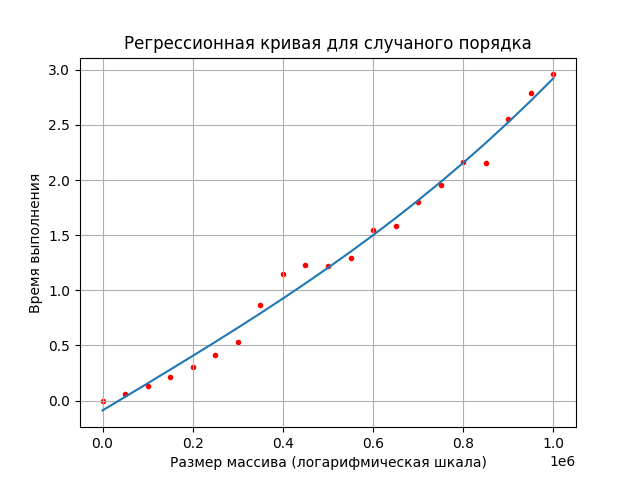


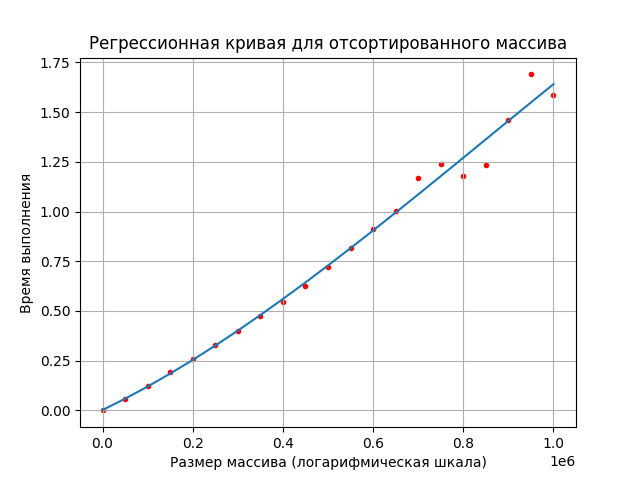
Рисунок 31 – График сортировки Шелла с последовательностью Хиббарда для случайного порядка

Рисунок 32 – График сортировки Шелла с последовательностью Хиббарда для отсортированного массива

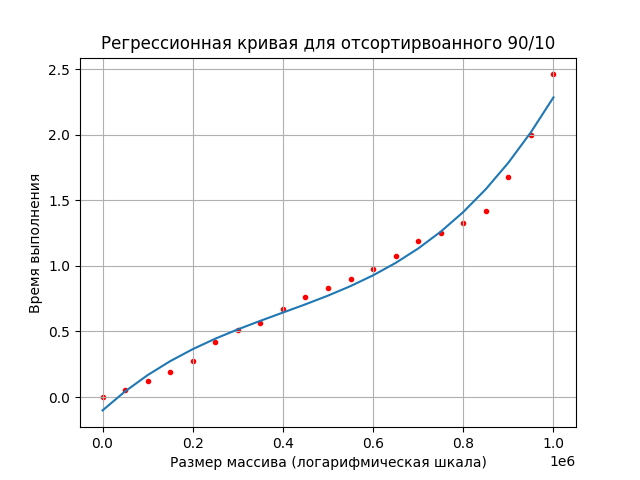


Рисунок 33 – График сортировки Шелла с последовательностью Хиббарда для почти отсортированного массива

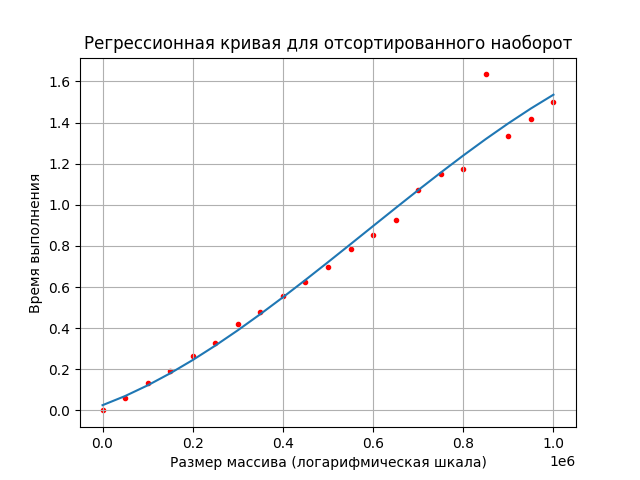


Рисунок 34 – График сортировки Шелла с последовательностью Хиббарда для обратно отсортированного массива

**Сортировка Шелла (последовательность Пратта).**

Произвольный массив

|  |  |
| --- | --- |
| Количество элементов | Время сортировки(сек) |
| 0 | 0,000115 |
| 50000 | 0,209065 |
| 100000 | 0,453378 |
| 150000 | 0,664129 |
| 200000 | 0,917016 |
| 250000 | 1,221624 |
| 300000 | 1,549116 |
| 350000 | 2,641677 |
| 400000 | 3,431701 |
| 450000 | 3,390201 |
| 500000 | 3,556581 |
| 550000 | 3,836848 |
| 600000 | 4,297003 |
| 650000  700000 | 4,480559 5,064272 |
| 750000 | 5,447973 |
| 800000 | 5,976048 |
| 850000 | 5,835555 |
| 900000 | 7,015138 |
| 950000 | 7,550728 |
| 1000000 | 7,840592 |

Таблица 25 – Время работы сортировки Шелла с последовательностью Пратта (произвольный массив)

отсортированный массив

|  |  |
| --- | --- |
| Количество элементов | Время сортировки(сек) |
| 0 | 0,000103 |
| 50000 | 0,200662 |
| 100000 | 0,406953 |
| 150000 | 0,623316 |
| 200000 | 0,819612 |
| 250000 | 1,033404 |
| 300000 | 1,256048 |
| 350000 | 1,454002 |
| 400000 | 1,668987 |
| 450000 | 1,869447 |
| 500000 | 2,162396 |
| 550000 | 2,373934 |
| 600000 | 2,697549 |
| 650000  700000 | 2,881595 3,435165 |
| 750000 | 3,592126 |
| 800000 | 3,369465 |
| 850000 | 3,529170 |
| 900000 | 4,048170 |
| 950000 | 4,409199 |
| 1000000 | 4,427340 |

Таблица 26 – Время работы сортировки Шелла с последовательностью Пратта (отсортированный массив)

Почти отсортированный массив

|  |  |
| --- | --- |
| Количество элементов | Время сортировки(сек) |
| 0 | 0,000109 |
| 50000 | 0,199546 |
| 100000 | 0,409159 |
| 150000 | 0,621761 |
| 200000 | 0,864515 |
| 250000 | 1,293008 |
| 300000 | 1,534411 |
| 350000 | 1,694894 |
| 400000 | 1,965135 |
| 450000 | 2,185248 |
| 500000 | 2,409194 |
| 550000 | 2,622092 |
| 600000 | 2,800993 |
| 650000  700000 | 3,063574 3,355043 |
| 750000 | 3,527030 |
| 800000 | 3,724428 |
| 850000 | 3,912337 |
| 900000 | 4,598822 |
| 950000 | 5,408736 |
| 1000000 | 6,488420 |

Таблица 27 – Время работы сортировки Шелла с последовательностью Пратта (массив почти отсортированный)

Массив сортированный обратно

|  |  |
| --- | --- |
| Количество элементов | Время сортировки(сек) |
| 0 | 0,000100 |
| 50000 | 0,218451 |
| 100000 | 0,629422 |
| 150000 | 0,819510 |
| 200000 | 1,027479 |
| 250000 | 1,324677 |
| 300000 | 1,450806 |
| 350000 | 1,651785 |
| 400000 | 1,858193 |
| 450000 | 2,173204 |
| 500000 | 2,289615 |
| 550000 | 2,476725 |
| 600000 | 2,685327 |
| 650000  700000 | 2,942175 3,160346 |
| 750000 | 3,333539 |
| 800000 | 4,502984 |
| 850000 | 3,756191 |
| 900000 | 3,990692 |
| 950000 | 4,251247 |

Таблица 28 – Время работы сортировки Шелла с последовательностью Пратта (массив сортированный обратно)

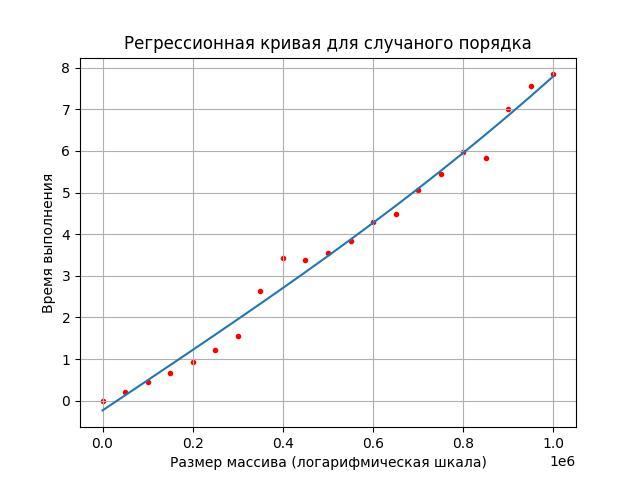


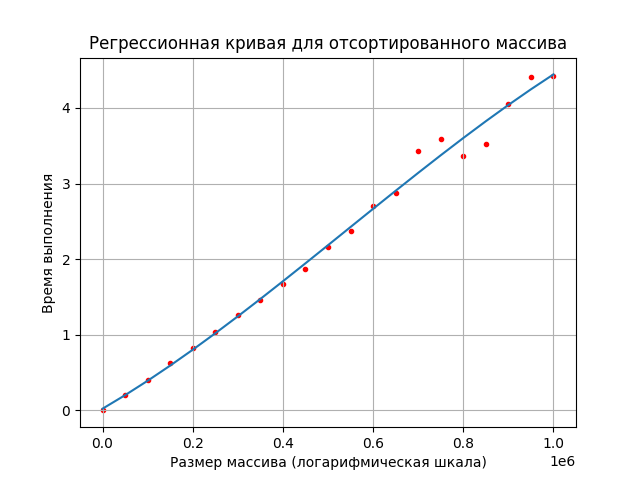
Рисунок 35 – График сортировки Шелла с последовательностью Пратта для случайного порядка

Рисунок 36 – График сортировки Шелла с последовательностью Пратта для отсортированного массива

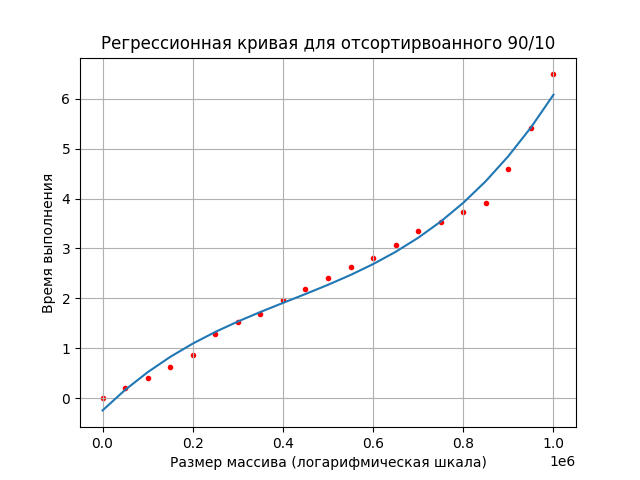


Рисунок 37 – График сортировки Шелла с последовательностью Пратта для почти отсортированного массива

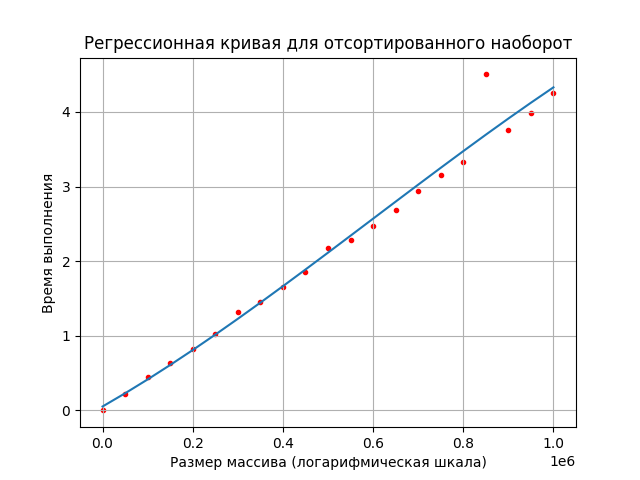


Рисунок 38 – График сортировки Шелла с последовательностью Пратта для наоборот отсортированного

**Быстрая сортировка.**

Произвольный массив

|  |  |
| --- | --- |
| Количество элементов | Время сортировки(сек) |
| 0 | 0,000000 |
| 50000 | 0,055246 |
| 100000 | 0,115813 |
| 150000 | 0,184742 |
| 200000 | 0,250093 |
| 300000 | 0,436423 |
| 400000 | 0,649158 |
| 500000 | 0,774102 |
| 600000 | 0,957984 |
| 700000 | 1,112151 |
| 800000 | 1,335848 |
| 900000 | 1,606593 |
| 1000000 | 1,739834 |
| 1200000 | 2,097799 |
| 1300000 | 2,292078 |
| 1500000 | 2,707116 |
| 1700000 | 3,153495 |
| 1900000 | 3,763204 |
| 2000000 | 3,771462 |
| 2250000 | 4,270667 |
| 2500000 | 4,961658 |
| 2750000 | 5,504497 |
| 3000000 | 5,980028 |
| 3250000 | 6,682974 |
| 3500000 | 6,502504 |
| 3750000 | 7,379328 |
| 4000000 | 7,845639 |
|  |  |

Таблица 29 – Время работы быстрой сортировки (произвольный массив)

Отсортированный массив

|  |  |
| --- | --- |
| Количество элементов | Время сортировки(сек) |
| 0 | 0,000000 |
| 50000 | 0,033912 |
| 100000 | 0,065248 |
| 150000 | 0,097620 |
| 200000 | 0,132175 |
| 300000 | 0,202070 |
| 400000 | 0,272635 |
| 500000 | 0,344210 |
| 600000 | 0,418078 |
| 700000 | 0,490169 |
| 800000 | 0,567485 |
| 900000 | 0,638314 |
| 1000000 | 0,713264 |
| 1200000 | 0,906821 |
| 1300000 | 0,948827 |
| 1500000 | 1,106460 |
| 1700000 | 1,250639 |
| 1900000 | 1,425564 |
| 2000000 | 1,535688 |
| 2250000 | 1,738320 |
| 2500000 | 1,928019 |
| 2750000 | 2,119225 |
| 3000000 | 2,308432 |
| 3250000 | 2,506374 |
| 3500000 | 2,697852 |
| 3750000 | 2,902211 |
| 4000000 | 3,102024 |
|  |  |

Таблица 30 – Время работы быстрой сортировки (отсортированный массив)

Почти тсортированный массив

|  |  |
| --- | --- |
| Количество элементов | Время сортировки(сек) |
| 0 | 0,000000 |
| 50000 | 0,036692 |
| 100000 | 0,074428 |
| 150000 | 0,114219 |
| 200000 | 0,152148 |
| 300000 | 0,237237 |
| 400000 | 0,336345 |
| 500000 | 0,429783 |
| 600000 | 0,523525 |
| 700000 | 0,626994 |
| 800000 | 0,730733 |
| 900000 | 0,893739 |
| 1000000 | 1,065358 |
| 1200000 | 1,255479 |
| 1300000 | 1,313899 |
| 1500000 | 1,472363 |
| 1700000 | 1,675941 |
| 1900000 | 1,863024 |
| 2000000 | 1,986867 |
| 2250000 | 2,367908 |
| 2500000 | 2,584904 |
| 2750000 | 2,900662 |
| 3000000 | 3,189404 |
| 3250000 | 3,382012 |
| 3500000 | 3,895790 |
| 3750000 | 4,461988 |
| 4000000 | 5,059827 |
|  |  |

Таблица 31 – Время работы быстрой сортировки (Почти тсортированный массив)

Обратно отсортированный массив

|  |  |
| --- | --- |
| Количество элементов | Время сортировки(сек) |
| 0 | 0,000000 |
| 50000 | 0,034640 |
| 100000 | 0,065613 |
| 150000 | 0,102142 |
| 200000 | 0,136354 |
| 300000 | 0,210657 |
| 400000 | 0,282355 |
| 500000 | 0,355398 |
| 600000 | 0,437589 |
| 700000 | 0,509667 |
| 800000 | 0,580294 |
| 900000 | 0,655811 |
| 1000000 | 0,751277 |
| 1200000 | 0,958640 |
| 1300000 | 0,964904 |
| 1500000 | 1,121054 |
| 1700000 | 1,286415 |
| 1900000 | 1,453925 |
| 2000000 | 1,520520 |
| 2250000 | 1,722676 |
| 2500000 | 1,920110 |
| 2750000 | 2,115698 |
| 3000000 | 2,418372 |
| 3250000 | 2,553066 |
| 3500000 | 2,783248 |
| 3750000 | 2,983720 |
| 4000000 | 3,148418 |
|  |  |

Таблица 32 – Время работы быстрой сортировки (обратно отсортированный массив)

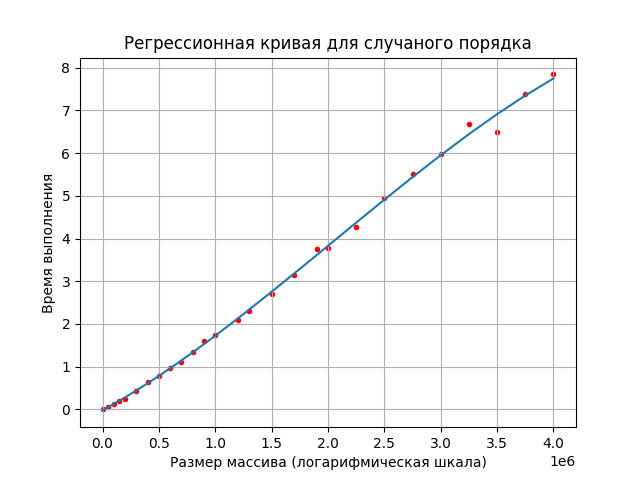


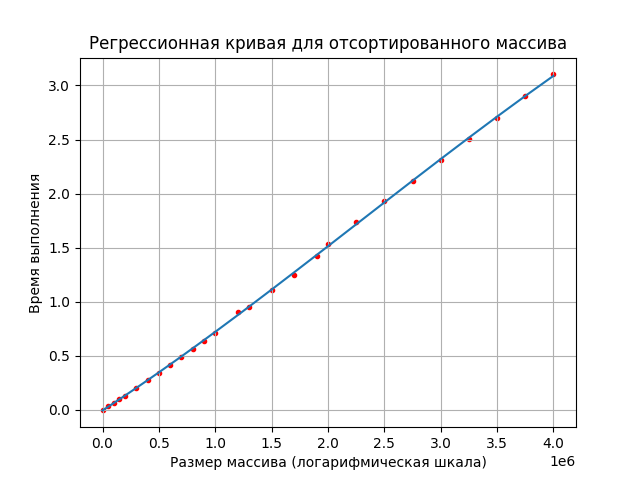
Рисунок 39– График быстрой сортировки для случайного порядка

Рисунок 40– График быстрой сортировки для отсортированного массива

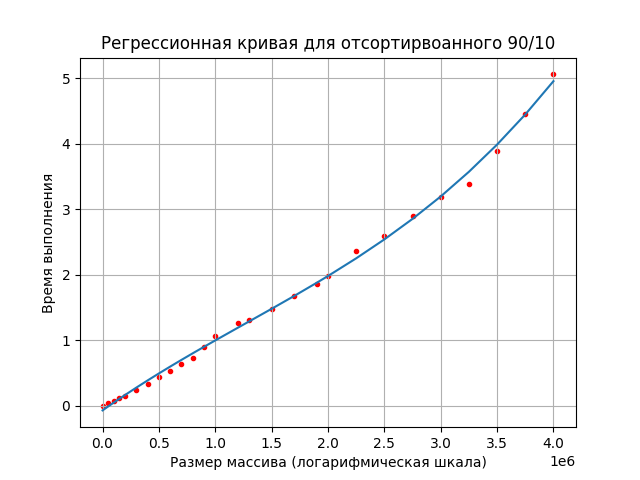


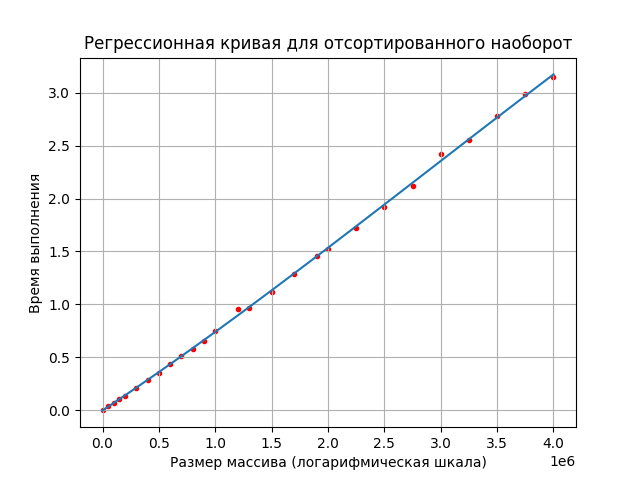
Рисунок 41– График быстрой сортировки для почти отсортированного

Рисунок 42– График быстрой сортировки для наоборот отсортированного

**Пирамидальная Сортировка.**

Произвольный массив

|  |  |
| --- | --- |
| Количество элементов | Время сортировки(сек) |
| 0 | 0,000000 |
| 50000 | 0,129667 |
| 100000 | 0,280867 |
| 150000 | 0,462111 |
| 200000 | 0,663731 |
| 250000 | 0,808458 |
| 300000 | 1,037174 |
| 350000 | 1,335124 |
| 400000 | 1,425213 |
| 450000 | 1,674394 |
| 500000 | 1,885533 |
| 550000 | 2,154315 |
| 600000 | 2,350669 |
| 650000 | 2,592572 |
| 700000 | 2,804033 |
| 750000 | 3,150136 |
| 800000 | 3,360104 |
| 8500000 | 3,631974 |
| 900000 | 3,934488 |
| 950000 | 4,179506 |
| 1000000 | 4,337145 |

Таблица 33 – Время работы пирамидальной сортировки (произвольный массив)

Отсортированный массив

|  |  |
| --- | --- |
| Количество элементов | Время сортировки(сек) |
| 0 | 0,000000 |
| 50000 | 0,130018 |
| 100000 | 0,262490 |
| 150000 | 0,406196 |
| 200000 | 0,565104 |
| 250000 | 0,724303 |
| 300000 | 0,868512 |
| 350000 | 1,038915 |
| 400000 | 1,231637 |
| 450000 | 1,356981 |
| 500000 | 1,536158 |
| 550000 | 1,685930 |
| 600000 | 1,852881 |
| 650000 | 2,002445 |
| 700000 | 2,169517 |
| 750000 | 2,339328 |
| 800000 | 2,522835 |
| 8500000 | 2,733188 |
| 900000 | 2,852736 |
| 950000 | 3,020330 |
| 1000000 | 3,164320 |

Таблица 34 – Время работы пирамидальной сортировки (отсортированный массив)  
Почти отсортированный массив

|  |  |
| --- | --- |
| Количество элементов | Время сортировки(сек) |
| 0 | 0,000000 |
| 50000 | 0,131548 |
| 100000 | 0,278694 |
| 150000 | 0,457654 |
| 200000 | 0,604311 |
| 250000 | 0,789729 |
| 300000 | 0,954139 |
| 350000 | 1,109708 |
| 400000 | 1,285452 |
| 450000 | 1,497600 |
| 500000 | 1,678724 |
| 550000 | 1,959738 |
| 600000 | 2,118375 |
| 650000 | 2,286226 |
| 700000 | 2,628437 |
| 750000 | 2,707575 |
| 800000 | 2,905662 |
| 8500000 | 3,150794 |
| 900000 | 3,329319 |
| 950000 | 3,631511 |
| 1000000 | 3,834171 |

Отсортированный в сторону убывания массив

Таблица 35 – Время работы пирамидальной сортировки (массив отсортированный в сторону убывания)

Обратно отсортированный массив

|  |  |
| --- | --- |
| Количество элементов | Время сортировки(сек) |
| 0 | 0,000000 |
| 50000 | 0,119598 |
| 100000 | 0,245965 |
| 150000 | 0,390604 |
| 200000 | 0,534471 |
| 250000 | 0,677895 |
| 300000 | 0,857434 |
| 350000 | 0,991660 |
| 400000 | 1,148369 |
| 450000 | 1,309329 |
| 500000 | 1,475904 |
| 550000 | 1,652864 |
| 600000 | 1,816235 |
| 650000 | 1,964338 |
| 700000 | 2,131944 |
| 750000 | 2,297818 |
| 800000 | 2,450641 |
| 8500000 | 2,637198 |
| 900000 | 2,816548 |
| 950000 | 3,027291 |
| 1000000 | 3,152385 |

Таблица 36 – Время работы пирамидальной сортировки (Обратно отсортированный)

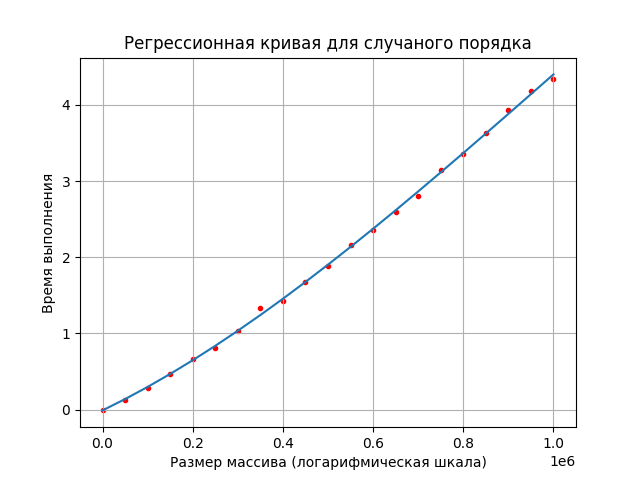


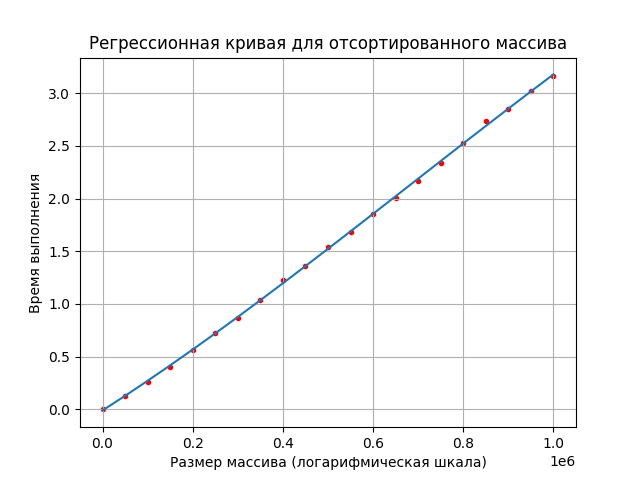
Рисунок 42 – График пирамидальной сортировки для случайного массива  


Рисунок 43 – График пирамидальной сортировки для отсортированного массива

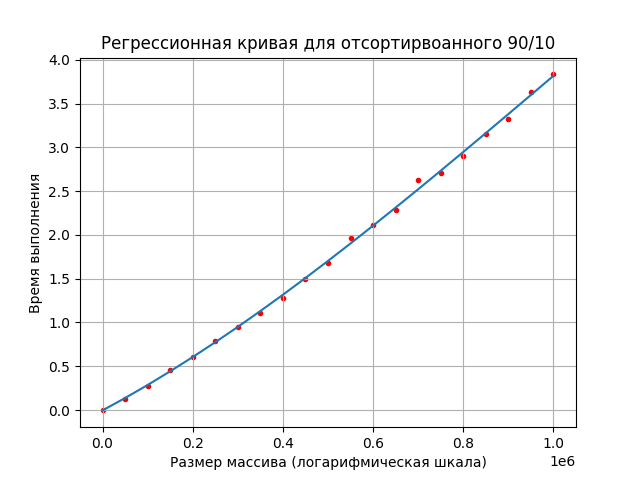
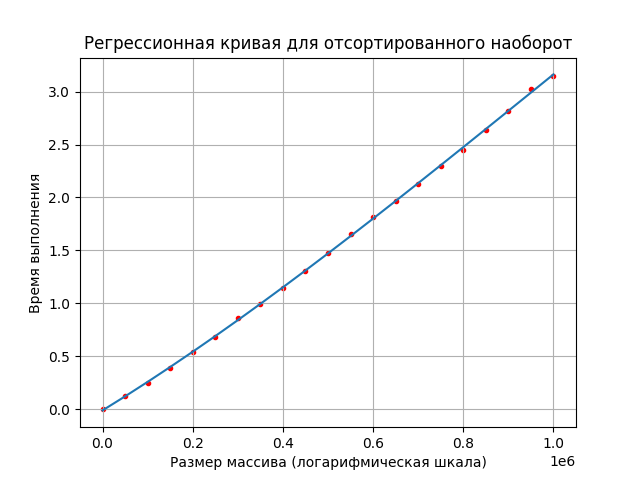
Рисунок 44 – График пирамидальной сортировки для почти отсортированного массива

Рисунок 45 – График пирамидальной сортировки для отсоритрованного в обратном порядке массива

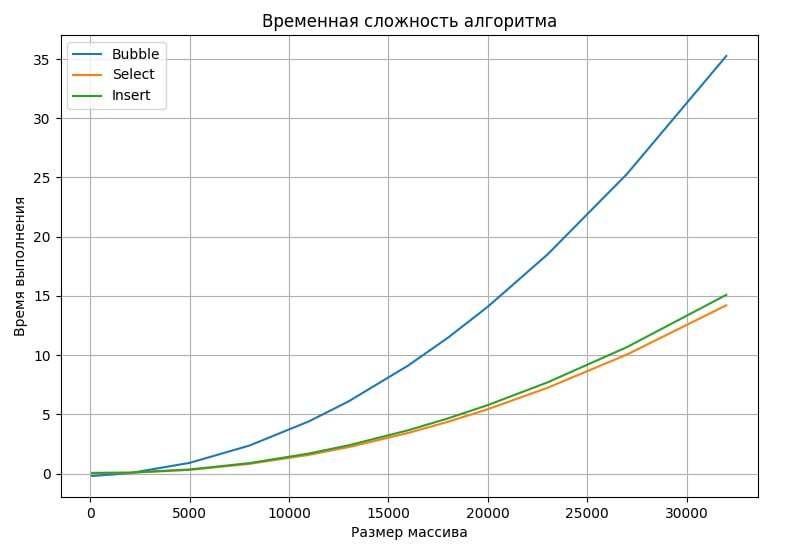
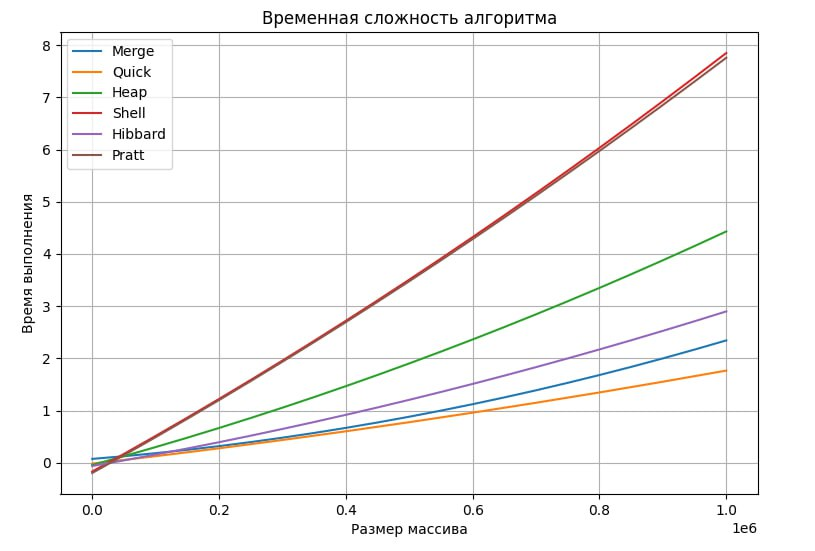


Рисунок 46 – График всех сортировок при случайном заполнении массива с квадратичной ассимптотикой.  
 Рисунок 47 – График всех сортировок при случайном заполнении массива с не квадратичной ассимптотикой.

**Вывод:**

Сравнивая теоретические и практические графики можно сделать вывод, что в большинстве случаев они согласовны и демонстрируют ожидаемую асимптотику, за исключением некоторых отклонений. Также, самой быстрой сортировкой в среднем случае при размере массива > 100 000 оказалась быстрая сортировка, что совпадает с предположением из теоретической части.

**Ссылки :**

**Репозиторий GitHub:** **https://github.com/K4taLizator/-**