ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчёт по лабораторной работе № 5

«Коктейль, куча и подсчёт»

Выполнил работу

Фамилия Имя

Семенов Никита

Академическая группа №группы

j3114, 467414

Принято

Должность, звание Фамилия Имя преподавателя

Санкт-Петербург

2024

**Задание 1**

**Коктейльная сортировка (Cocktail Sort)**:

Память: O(1)

Потому что элементы меняются местами в одном и том же массиве (векторе).

**Сортировка кучей (Heap Sort):**

Память: O(1)

Потому что элементы меняются местами в одном и том же массиве (векторе).

**Сортировка подсчётом (Counting Sort)**:

Память: O(k+n)

Создаётся дополнительный массив (вектор), размера k, и второй массив, размера n.

**Задание 2**

**Коктейльная сортировка:**

**сложность:** O(n^2) в худшем и среднем случае, так как каждый элемент может потребовать O(n) сравнений и перестановок.

Если изначальный массив будет уже отсортирован, то сложность сортировки становится O(2n), потому что алгоритм проходит по всем элементам в 2 стороны.

**Сортировка кучей:**

**Сложность сортировки** в худшем, лучшем и среднем случае составляет **O(n log n)**, потому что **высота полного бинарного дерева, содержащего n элементов, равна log(n)** и нам необходимо пройтись по всем элементам.

**Сортировка подсчётом:**

**Сложность сортировки** составляет **O(N + k)**, где N — количество элементов массива, а k — значение максимального элемента в изначальном массиве.

Почему так? Потому что нужно пройтись по всем элементам массива, состоящего из k элементов и для каждого не нулевого элемента нужно совершить ещё h операций, где h это и есть этот элемент.

Пример:

[0, 0, 1, 0, 2, 1] – массив из k элементов.

Проходясь по нему, нули дают 1 итерацию, еденички дают 2 итерации, 2-ки дают 2 итерации и так далее. Таким образом получаем k + n итераций. (сумма элементов и есть n)

**Задание 3, 4**

Код всех сортировок находится на гит хабе (+ тесты):

<https://github.com/user6778899/polygon/tree/lab5>

**Разбор тестов:**

На вход: 0 0 9 86 -6 23 6 0 -7 -48 63 4 90 2

Сортировка подсчётом: 0 0 0 2 4 6 9 23 63 86 90 4 90 2

Правильная сортировка: -48 -7 -6 0 0 0 2 4 6 9 23 63 86 90

Как видно из теста, сортировка подсчётом не работает с отрицательными числами, потому что проход по массиву (вектору) начинается с нуля.

Однако можно модернизировать этот алгоритм и начинать с минимального элемента.

vector<int> countsort(vector<int> arr){

int max1 = arr[0];

for(int i = 0;i<arr.size();i++){

if(arr[i]>max1){max1 = arr[i];}

}

int min1 = arr[0];

for(int i = 0;i<arr.size();i++){

if(arr[i]<min1){min1 = arr[i];}

}

vector<int> count(max1 + 1, 0);

for (int num : arr) {

count[num]++;

}

int index = 0;

for (int i = min1; i <= max1; i++) {

while (count[i] > 0) {

arr[index++] = i;

count[i]--;

}

}

return arr;

}

После добавления минимального элемента сложность возрастает до O(n + k + h), где n – кол-во элементов в изначальном массиве; k, h – максимальный и минимальный элементы.

**Примеры лучших и худших случаев для сортировок в зависимости от входного массива**

1. **Коктейльная сортировка**

Лучший случай при отсортированном массиве.

Худший при отсортированном в обратном порядке, так как для каждого элемента совершается n операций.

1. **Сортировка кучей**

Она имеет одинаковую сложность для любого входного массива, потому что происходит проход по всем элементам и построение бинарного дерева для каждого массива.

1. **Сортировка подсчётом**

Всё зависит от максимального элемента во входном массиве

Лучший случай достигается при массиве, в котором k меньше n. Если элементы начинаются с 0 и идут до максимального элемента (порядок не важен), тогда сложность становится O(2n).

Примеры: [0, 1, 3, 2, 5, 4] O(2n), [2, 2, 2, 2, 2] O(n+1)

Худший случай при k сильно большем n.

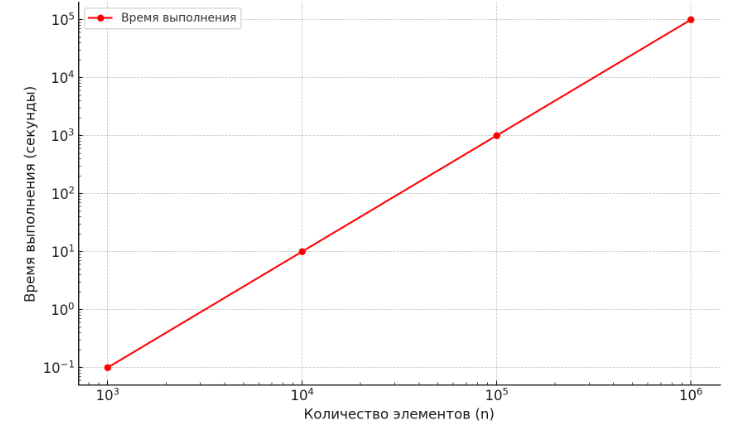
Пример: [1, 4, 100000000] O(n+k), k = 100000000

Потому что создаётся массив из k элементов и алгоритм проходится по нему.

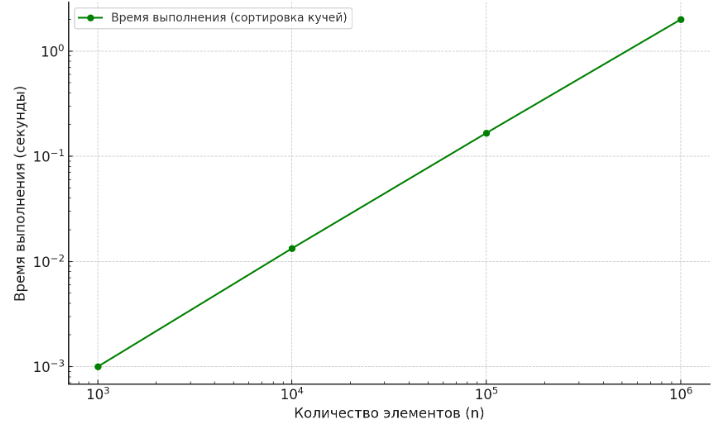
n – кол-во элементов, k – максимальный элемент.

**Задание 5**

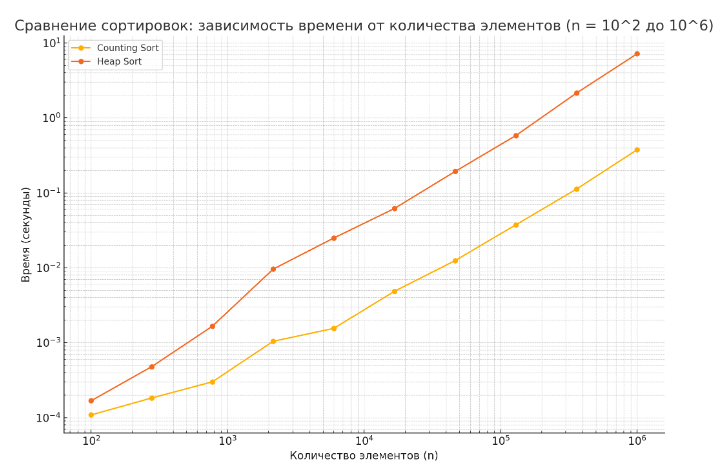
**1) Коктейль (для телефона)**

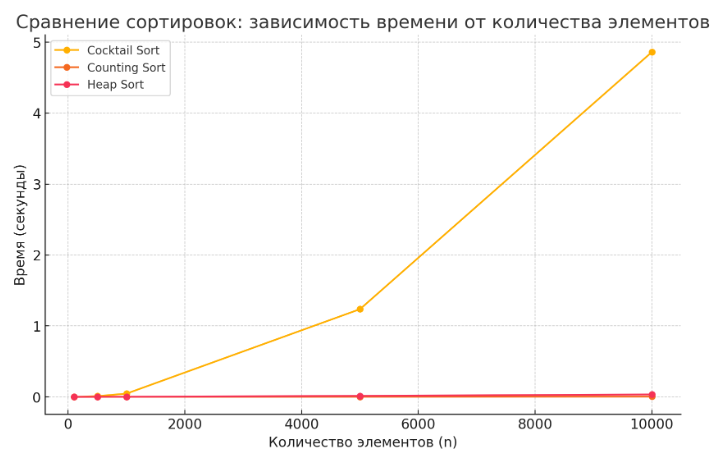


**2) Куча (для телефона)**



**3) Сравнение**

****

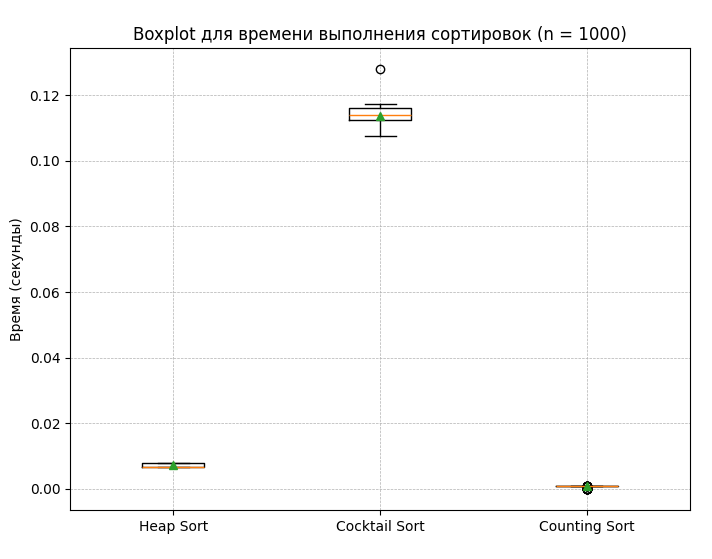


Таким образом, по графикам видно, что коктейльная сортировка не подходит для большого кол-ва элементов.

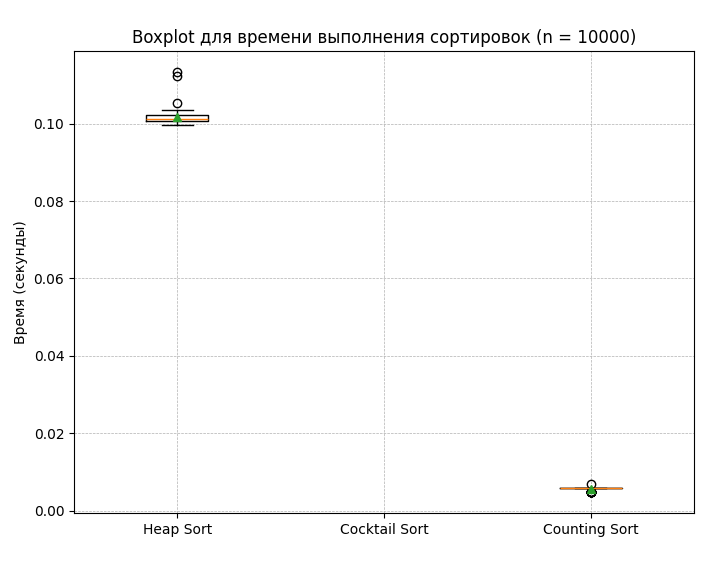
Сортировка подсчётом показала лучший результат, но всё зависит от максимального элемента в массиве, чем он больше, тем дольше будет работать алгоритм.

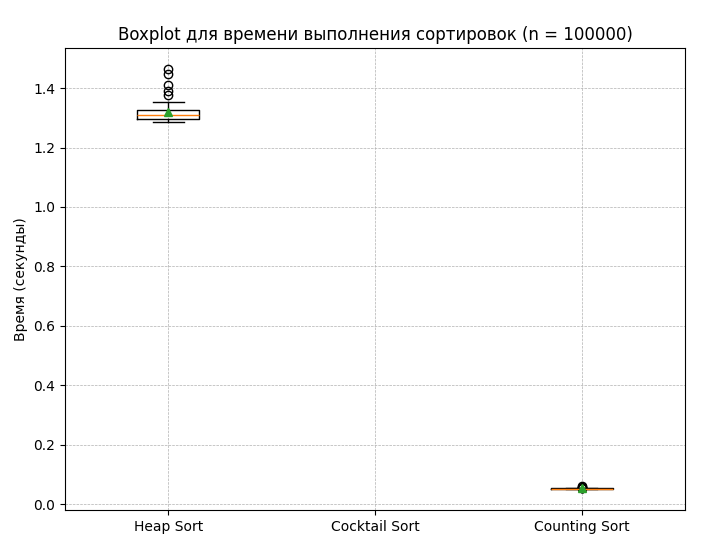
Сортировка кучей показала хороший результат, но она сложна в реализации.

**Задание 6**



Далее требуется слишком много времени для сортировки коктеёлем, поэтому сравниваются только подсчёт и куча.





**Можно сделать вывод по данным графикам**

Сортировка коктейлем не подходит для больших n.

Сортировка подсчётом показала наилучший результат по времени выполнения и у неё минимальный разброс, то есть постоянное время выполнения при разных данных.

Сортировка кучей показала средний результат.

**Вывод**

Сортировка подсчётом показала лучшие результаты, однако если элементов для сортировки не так много, но они очень большие, то алгоритм подсчёта станет самым долгим и затратным относительно кучи и коктейля.

Сортировка кучей – универсальная сортировка с хорошими показателями времени и памяти.

Сортировка коктейлем годится только для малых входных данных. Она подойдёт для обучения, как пример модификации сортировки пузырьком.