МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. Н.Э.Баумана
КАФЕДРА ПРОЕКТИРОВАНИЕ И ТЕХНОЛОГИЯ ПРОИЗВОДСТВА ЭЛЕКТРОННОЙ АППАРАТУРЫ

Отчет о выполнении практического задания №4 «Сортировка подсчётом. Быстрая сортировка»

Выполнили: студенты группы ИУ4-23Б
Якимов П.Н.
Турчинский Н.А.
Крюкова А.К.
Денисов В.А.
Горбунов З.И.

Проверил: Казаков В.В.

Москва, 2022 г.

Цель работы: изучить сортировку подсчётом и быструю сортировку

Задание:

- Описать алгоритмы сортировок.
- Привести блок-схему сортировок.
- Оценить плюсы и минусы данных сортировок.

Сортировка подсчётом
(Counting sort)

Сортировка подсчётом (Counting sort)

Описание: Сортировка подсчётом - алгоритм сортировки, в котором используется диапазон чисел сортируемого массива для подсчёта совпадающих элементов. Применение сортировки подсчётом целесообразно лишь тогда, когда сортируемые числа имеют диапазон возможных значений, который достаточно мал по сравнению с сортируемым множеством.

Алгоритм сортировки состоит из следующих шагов:

- Просмотр исходного массива и подсчет количества элементов в этом массиве (количество сохраняется во вспомогательном массиве).
- Просмотр вспомогательного массива и запись элементов в отсортированном порядке.

Сортировка подсчётом (Counting sort)

Исходный массив:

0	2	3	5	1	2	3	5

Счётчик:

0	1	2	3	4	5
1	1	2	2	0	2

Результирующий массив:

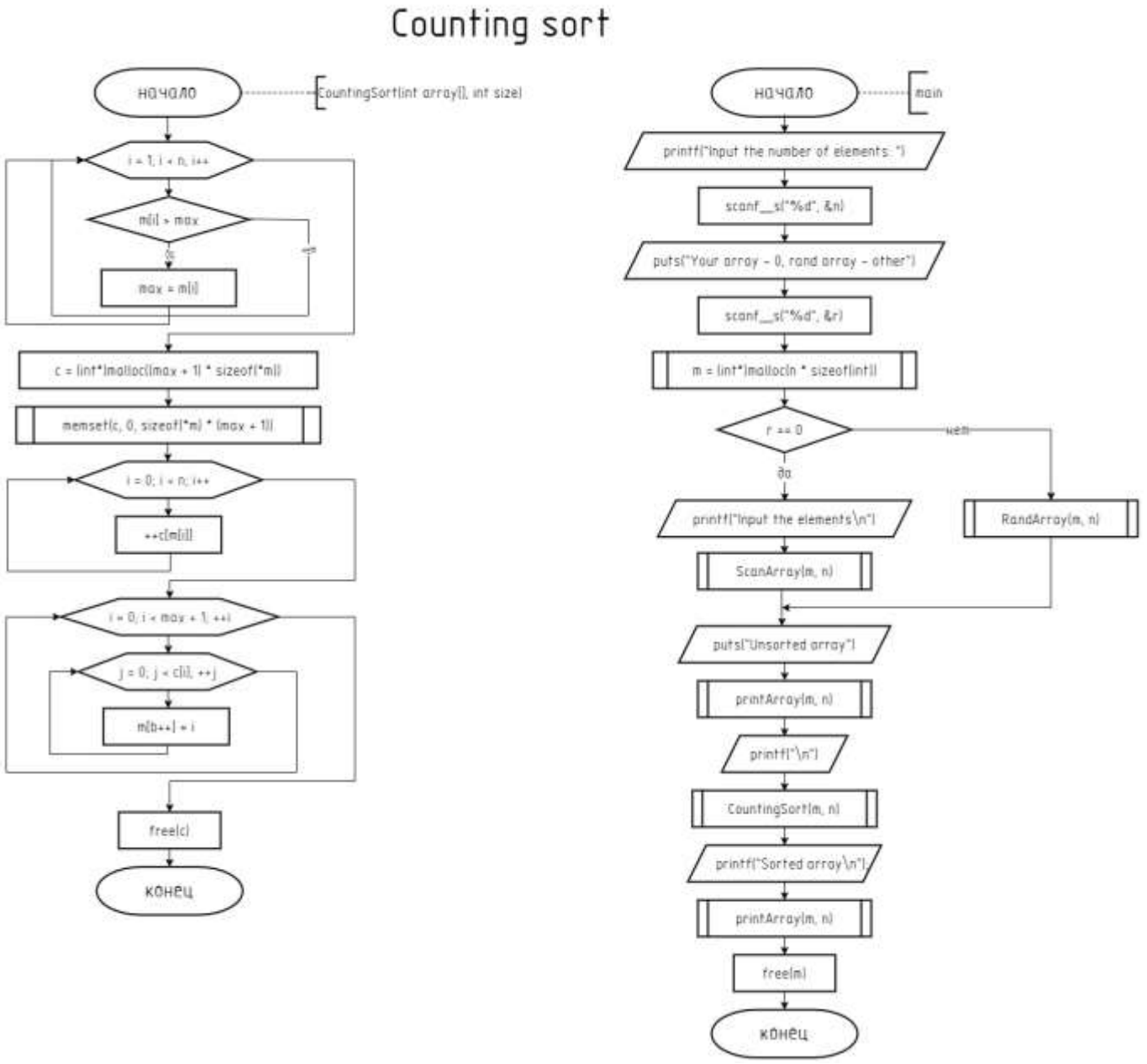
0	1	2	2	3	3	5	5

Код алгоритма сортировки подсчётом:

```
4   void CountingSort(int array[], int size) {
5       int max = array[0];
6       for (int i = 1; i < size; i++)
7           if (array[i] > max)
8               max = array[i];
9       int* count = (int*)malloc(max + 1 * sizeof(int));
10
11      for (int i = 0; i <= max; i++)
12          count[i] = 0;
13
14      for (int i = 0; i < size; i++)
15          count[array[i]]++;
16
17      for (int i = 1; i <= max; i++)
18          count[i] += count[i - 1];
19
20      int* out = (int*)malloc(size * sizeof(int));
21      for (int i = size - 1; i >= 0; i--) {
22          out[count[array[i]] - 1] = array[i];
23          count[array[i]]--;
24      }
25
26      for (int i = 0; i < size; i++)
27          array[i] = out[i];
28      free(out);
29  }
```

Сортировка подсчётом (Counting sort)

Блок-схема алгоритма сортировки подсчётом:



Сортировка подсчётом (Counting sort)

Оценка сложности:

- Временная сложность: Усреднённый случай (при относительно небольшом диапазоне чисел и большом количестве входных данных) - O(n+k)
- Пространственная сложность: O(k)

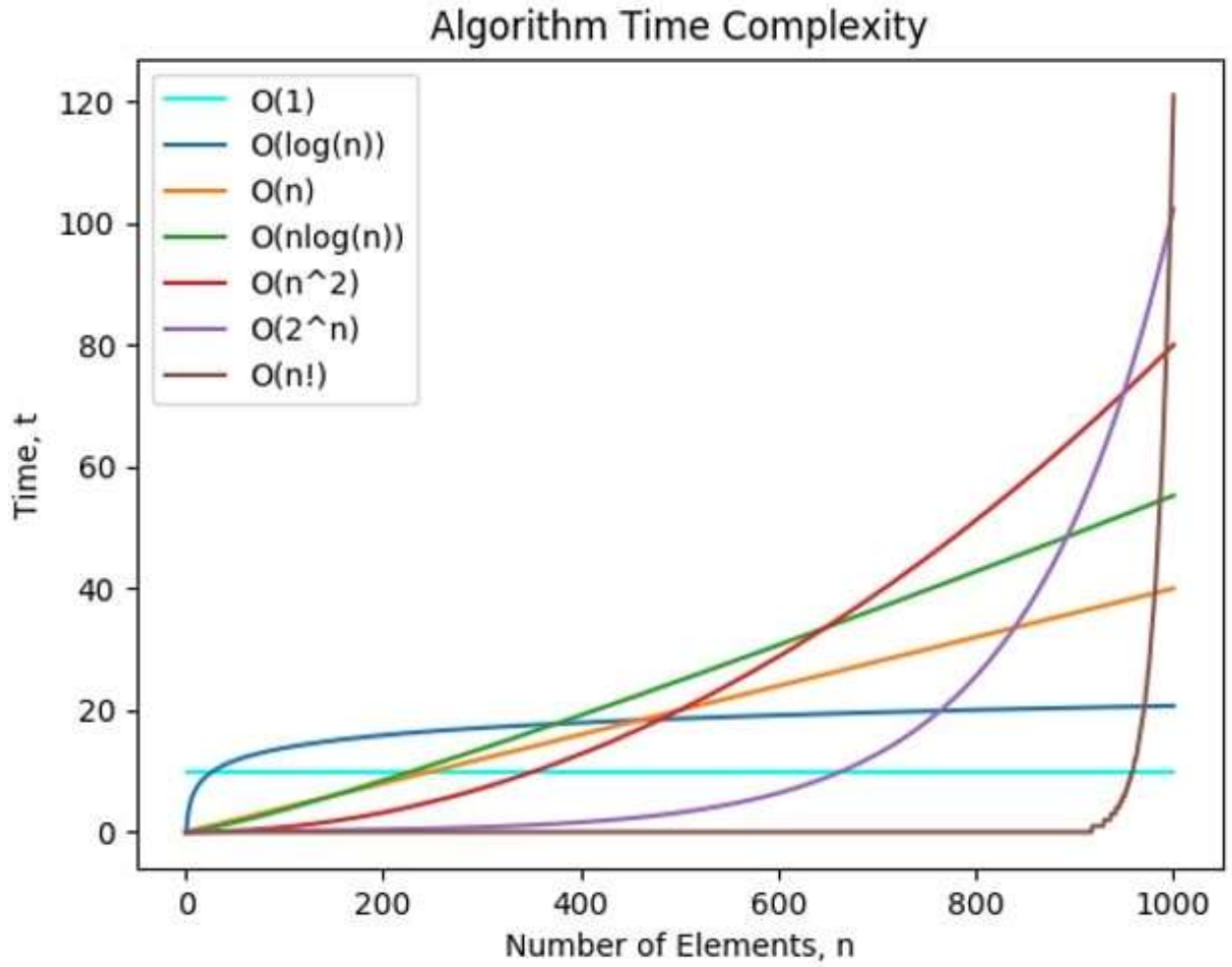
Плюсы:

- Применять данную сортировку целесообразно, когда имеется большое количество элементов массива с заданным небольшим верхним пределом.

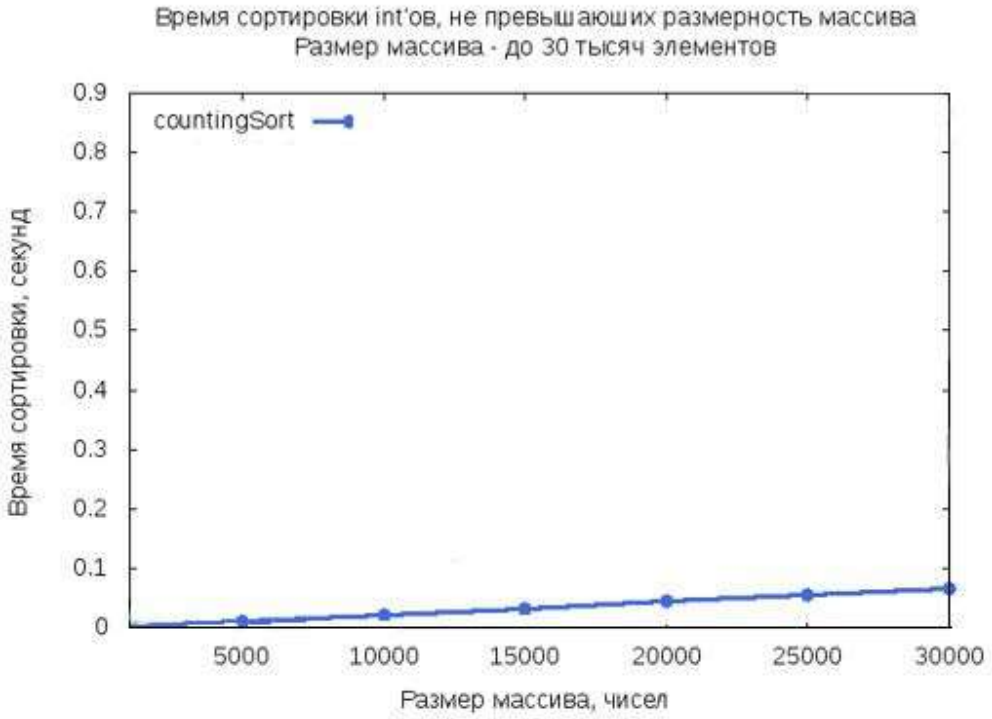
Минусы:

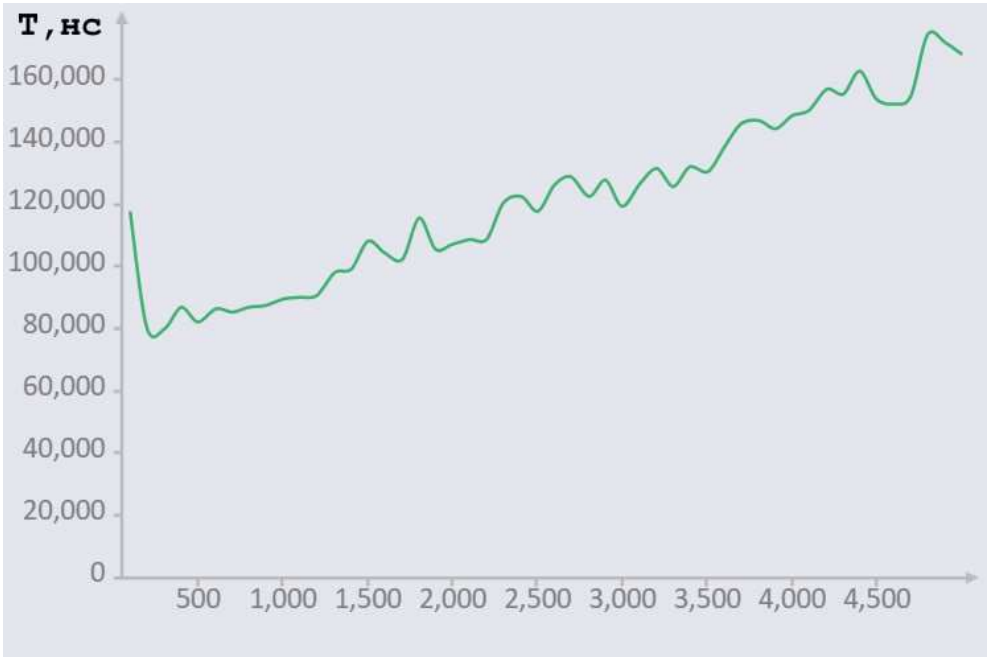
- Будет сортировать только при условии, что верхняя граница элементов не слишком большая.
- Есть менее затратные сортировки.
- Малая распространенность.

Сортировка подсчётом (Counting sort)
Оценка временной сложности:



Сортировка подсчётом (Counting sort)
Зависимость времени обработки от величины массива:





Быстрая сортировка (Quicksort)

Быстрая сортировка (Quicksort)

Описание: Быстрая сортировка - один из самых быстрых известных универсальных алгоритмов сортировки массивов.

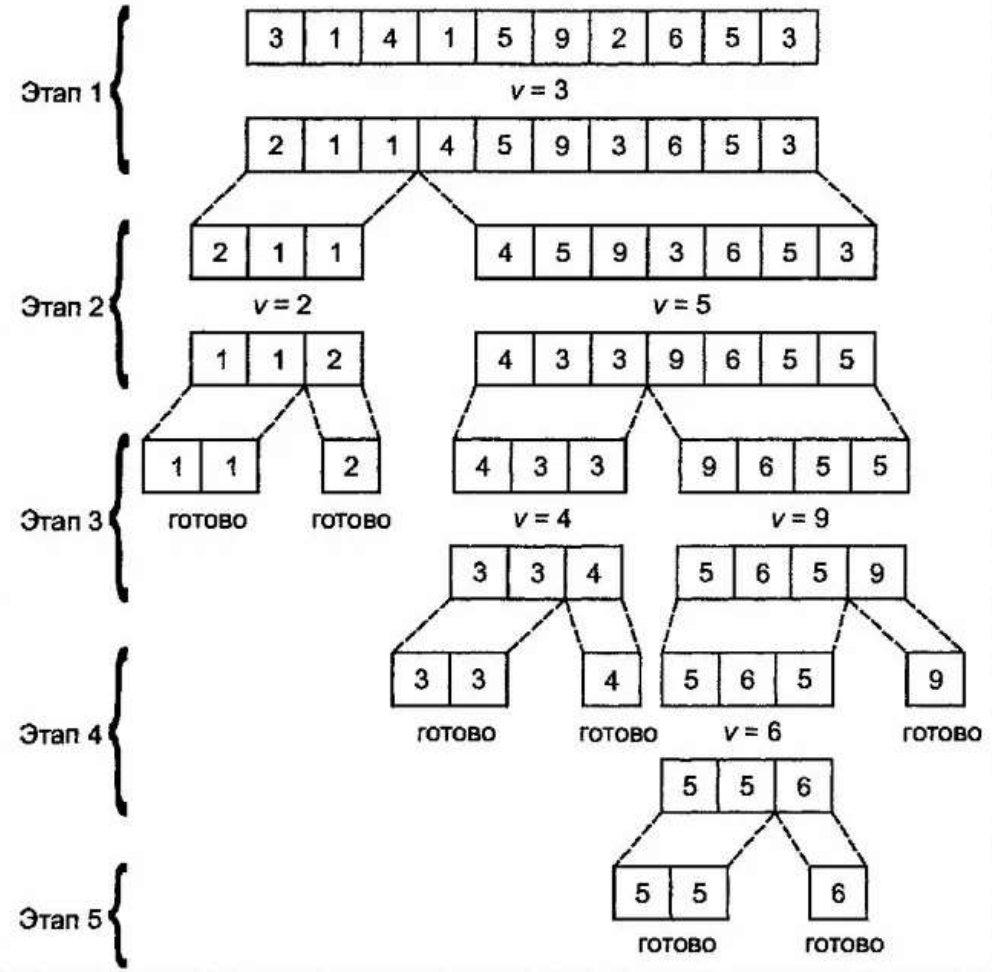
Алгоритм состоит из трёх шагов:

- Выбрать из массива любой элемент, называемый опорным. От выбора опорного элемента не зависит корректность алгоритма, но в отдельных случаях может сильно зависеть его эффективность.
- Сравнить все остальные элементы с опорным и переставить их в массиве так, чтобы разбить массив на три непрерывных отрезка, следующих друг за другом: «элементы меньшие опорного», «равные» и «большие».
- Для отрезков «меньших» и «больших» значений выполнить рекурсивно ту же последовательность операций, если длина отрезка больше единицы.

Быстрая сортировка (Quicksort)

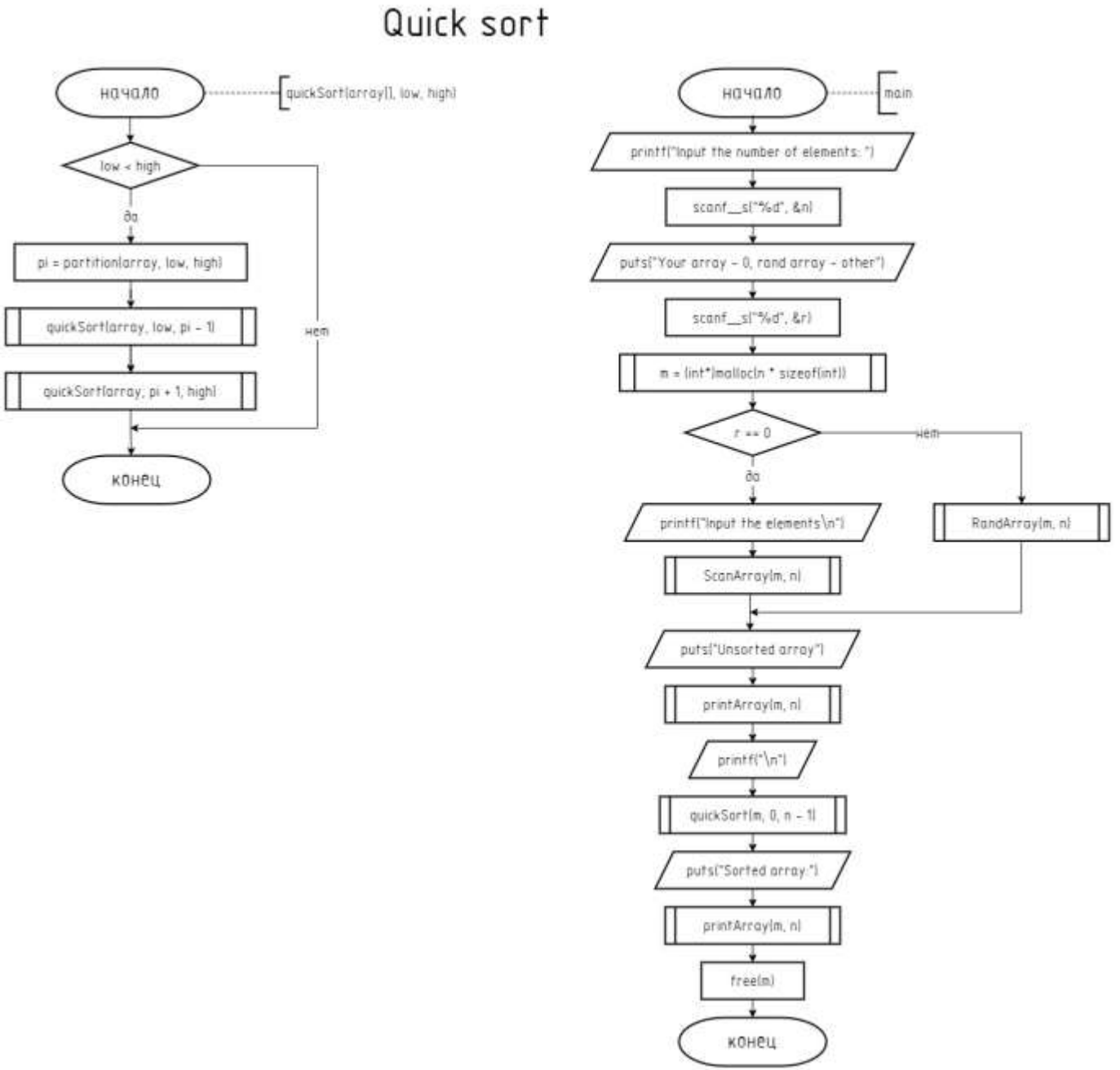
Код алгоритма быстрой сортировки:

```
void quickSort(int array[], int low, int high) {
    if (low < high) {
        int pi = partition(array, low, high);
        quickSort(array, low, pi - 1);
        quickSort(array, pi + 1, high);
    }
}
```



Быстрая сортировка (Quicksort)

Блок-схема алгоритма быстрой сортировки:



Быстрая сортировка (Quicksort)

Оценка сложности:

- Временная сложность: Лучший случай - O(n*log n)
  Худший случай - O(n^2)
  Усреднённый случай - O(n*log n)
- Пространственная сложность: O(log n)

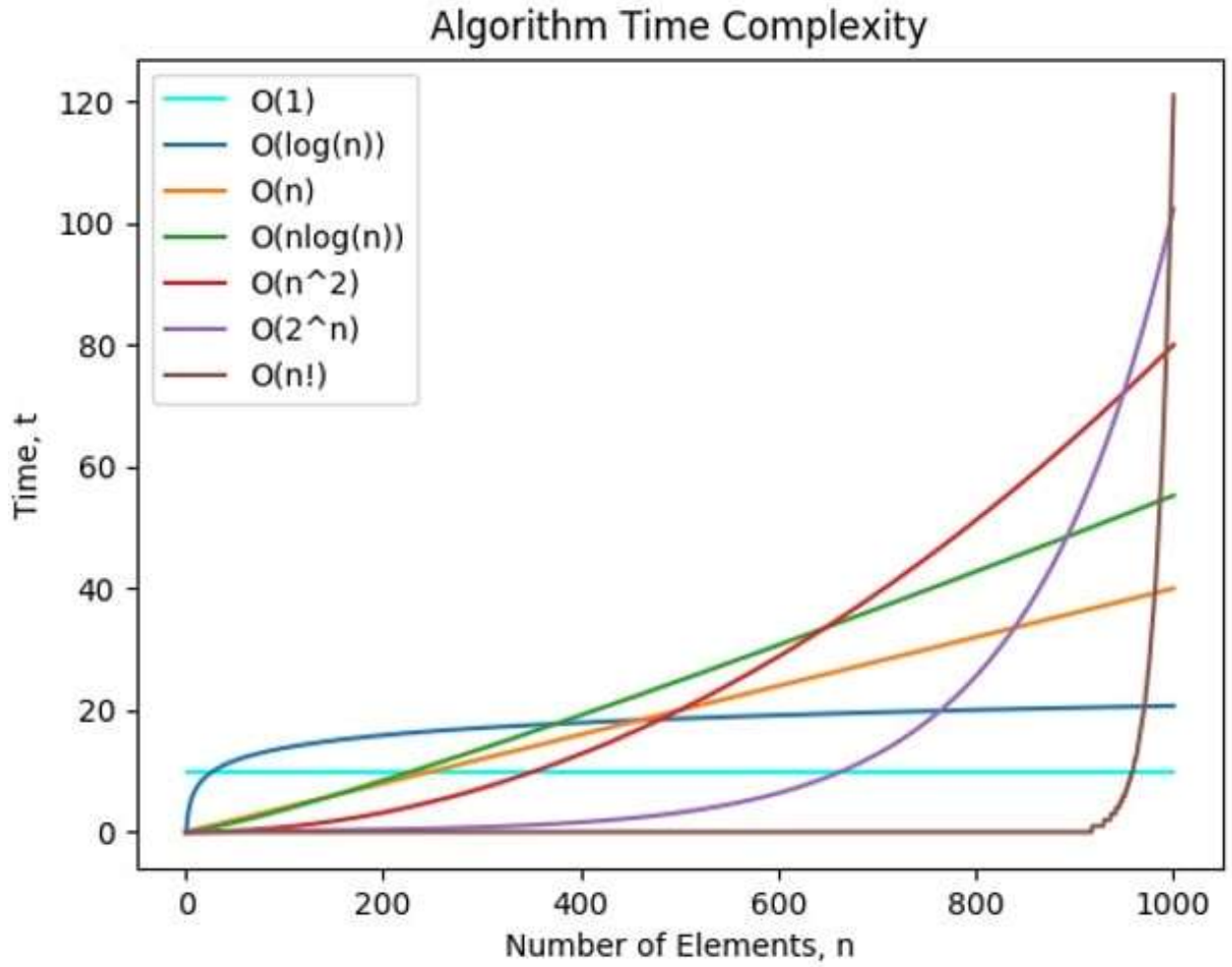
Плюсы:

- Один из самых быстродействующих (на практике) из алгоритмов внутренней сортировки общего назначения.
- Допускает эффективную модификацию.
- Работает на связных списках и других структурах с последовательным доступом, допускающих эффективный проход как от начала к концу, так и от конца к началу.

Минусы:

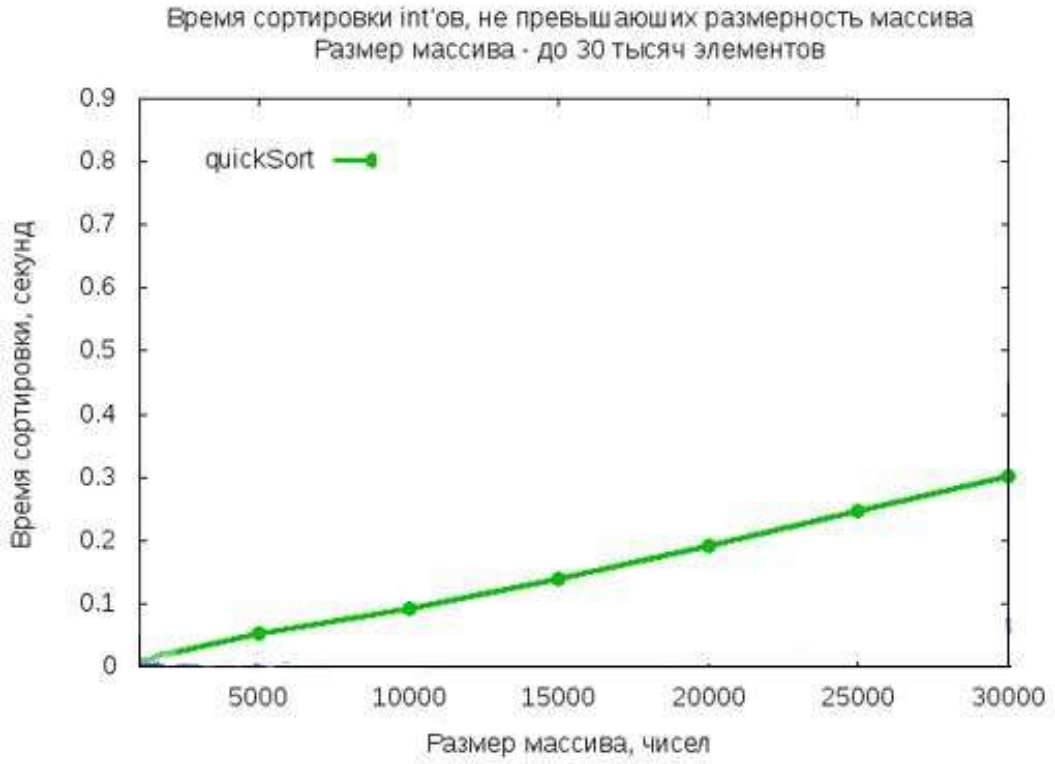
- Сильно деградирует по скорости [до O(n^2)] и может привести к исчерпанию памяти в худшем случае при неудачных входных данных.

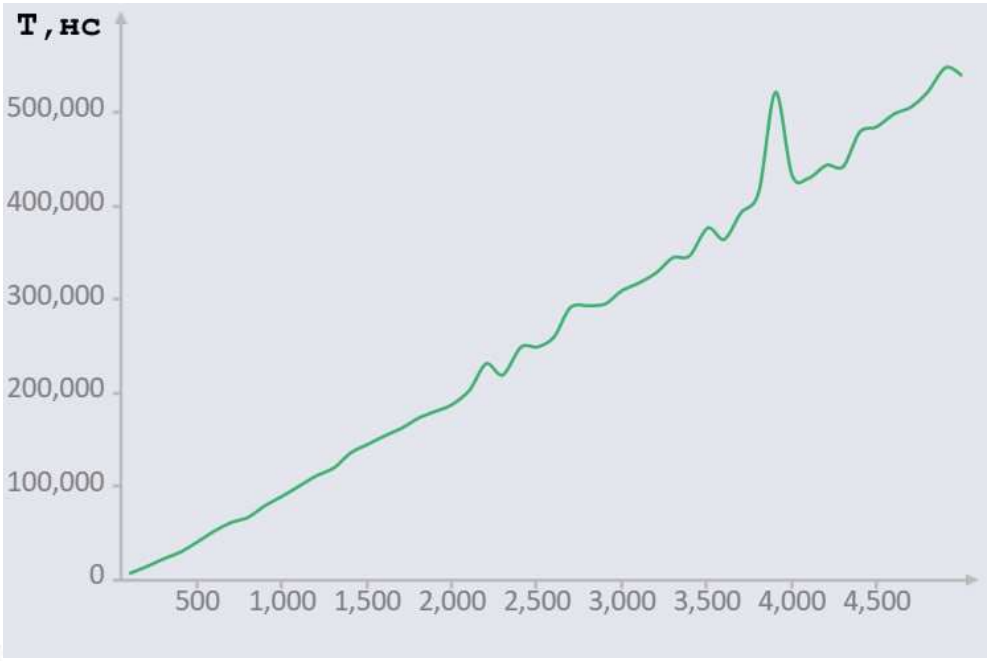
Сортировка подсчётом (Counting sort)
Оценка временной сложности:



Быстрая сортировка (QuickSort)

Зависимость времени обработки от величины массива:



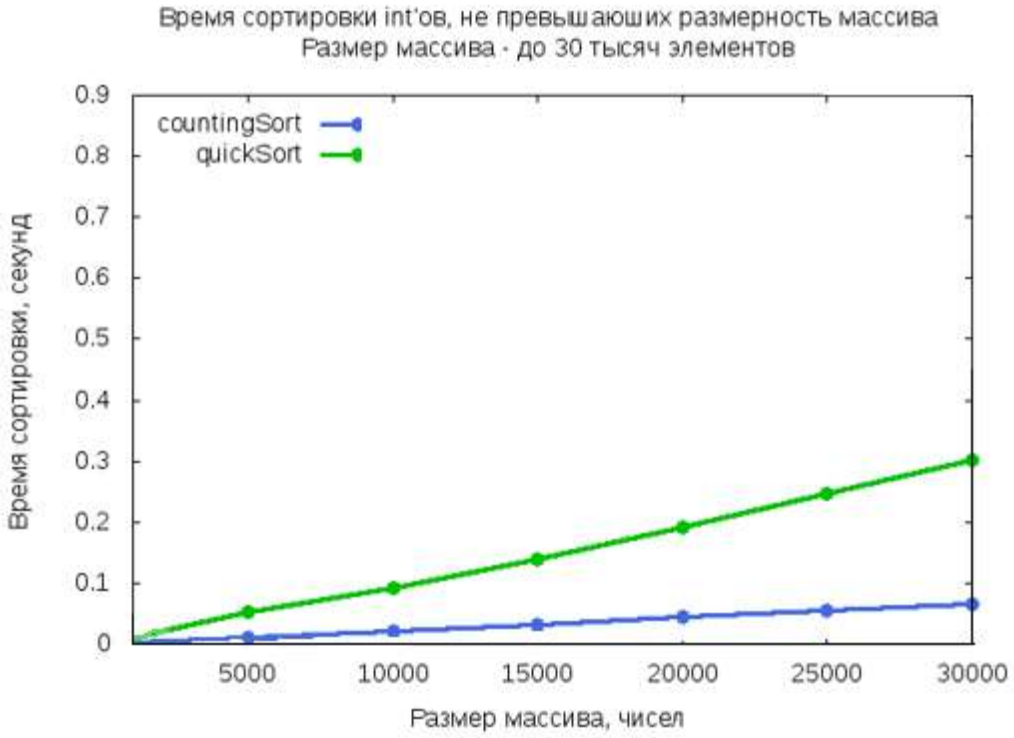


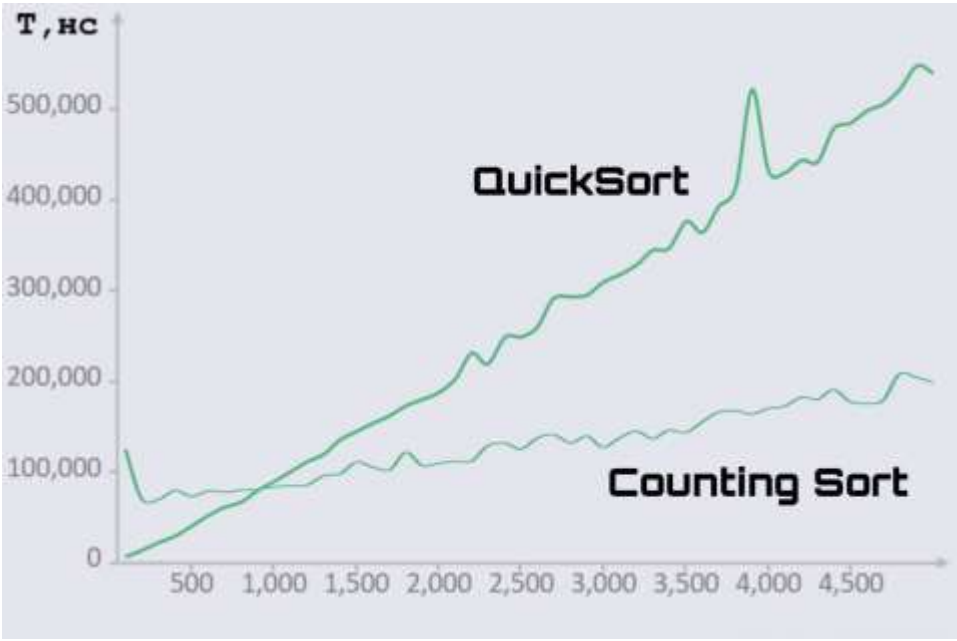
Выводы:

Сортировка подсчётом представляет собой простейший способ сортировки массива за линейное время и в особенности эффективен для упорядочивания больших массивов с малым разбросом значения, устойчив, используется как подпрограмма поразрядной сортировки. Однако данный алгоритм неэффективен на большом диапазоне значений элементов и малом размере массива.

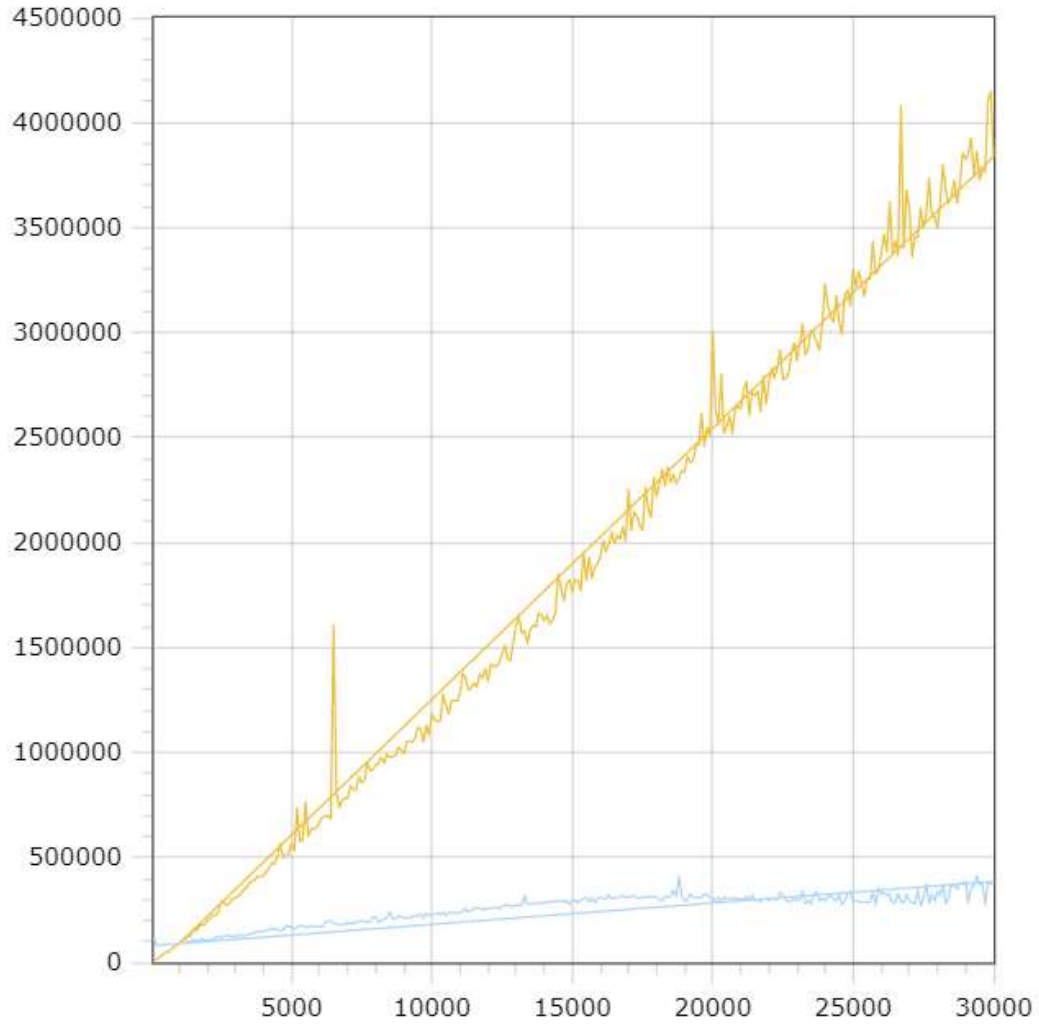
Быстрая сортировка представляет собой универсальный, лёгкий для написания, модифицируемый алгоритм упорядочивания массивов. Однако он неустойчив, при неудачных входных данных значительно замедляется и может привести к исчерпанию памяти.

Сравнение зависимостей времени обработки от величины массива в двух случаях:





Сравнение зависимостей времени обработки от величины массива в двух случаях:



Спасибо за внимание!