Государственное образовательное учреждение высшего профессионального образования
«Московский государственный технический университет имени Н.Э. Баумана»
(МГТУ им. Н.Э. Баумана)

Лабораторная работа №3

по курсу Анализ алгоритмов

Сортировка массивов

Студент: Тимонин А.С., группа ИУ7-52Б

Преподаватели: Волкова Л.Л., Строганов Ю.В.

2019 г.

Оглавление

Введение

Алгоритмов сортировок существует много, они широко применимы во многих сферах, и поэтому важно знать сложность алгоритмов, их преимущества и недостатки.

Цель работы: изучение и анализ алгоритмов сортировок.

Задачи работы:

- изучить и реализовать алгоритм сортировки пузырьком, вставками и выбором;
- исследовать временные затраты алгоритмов;
- описание и обоснование полученных результатов;

1. Аналитический раздел

В данном разделе вы найдете описание и применение адгоритмов сортировок пузырьком, вставками и выбором.

1.1 Описание алгоритмов

Задача с алгоритмами сортировок формулируетс так: дана последовательность элементов

$$a_1, a_2, \ldots, a_n \tag{1.1}$$

Требуется упорядочить элементы по возрастанию или по неубыванию, чтобы результатом была последовательность их n элементов, в которой каждый следующий элемент больше или равен предыдущему:

$$a_1 \leq a_2 \leq \cdots \leq a_n \tag{1.2}$$

1.2 Сортировка пузырьком

Алгоритм состоит из повторяющихся проходов по сортируемому массиву. За каждый проход элементы последовательно сравниваются попарно и, если порядок в паре неверный, выполняется обмен элементов. Проходы по массиву повторяются N-1 раз или до тех пор, пока на очередном проходе не окажется, что обмены больше не нужны, что означает — массив отсортирован.

При каждом проходе алгоритма по внутреннему циклу, очередной наибольший элемент массива ставится на своё место в конце массива рядом с предыдущим «наибольшим элементом», а наименьший элемент перемещается на одну позицию к началу массива («всплывает» до нужной позиции, как пузырёк в воде — отсюда и название алгоритма).

Время работы алгоритма зависит от длины последовательности N и от состояния этой последовательности(отсортирована она или нет, или элементы расположены в хаотичном порядке)

1.3 Сортировка вставками

В данном алгоритме сортировки элементы входной последовательности просматриваются по одному, и каждый новый поступивший элемент размещается в подходящее место среди ранее упорядоченных элементов.

Время выполнения алгоритма зависит от входных данных: чем большее множество нужно отсортировать, тем большее время потребуется для выполнения сортировки. Также на время выполнения влияет исходная упорядоченность массива.

1.4 Сортировка выбором

В алгоритме сортировки выбором находится номер минимального значение в текущем списке, производится обмен этого значения со значением первой неотсортированной позиции, и заьем сортируется хвост списка.

Особенностью сортировки выбором является независимость скорости от характера сортируемых данных.

2. Конструкторский раздел

В данном разделе приведены блок-схемы трех алгоритмов сортировки. Также приведен расчет сложности алгоритмов.

2.1 Разработка алгоритмов

На рисунках 2.1, 2.2, 2.3 приведены схемы алгоритмов сортировки пузырбком, вставками и выбором соответственно.

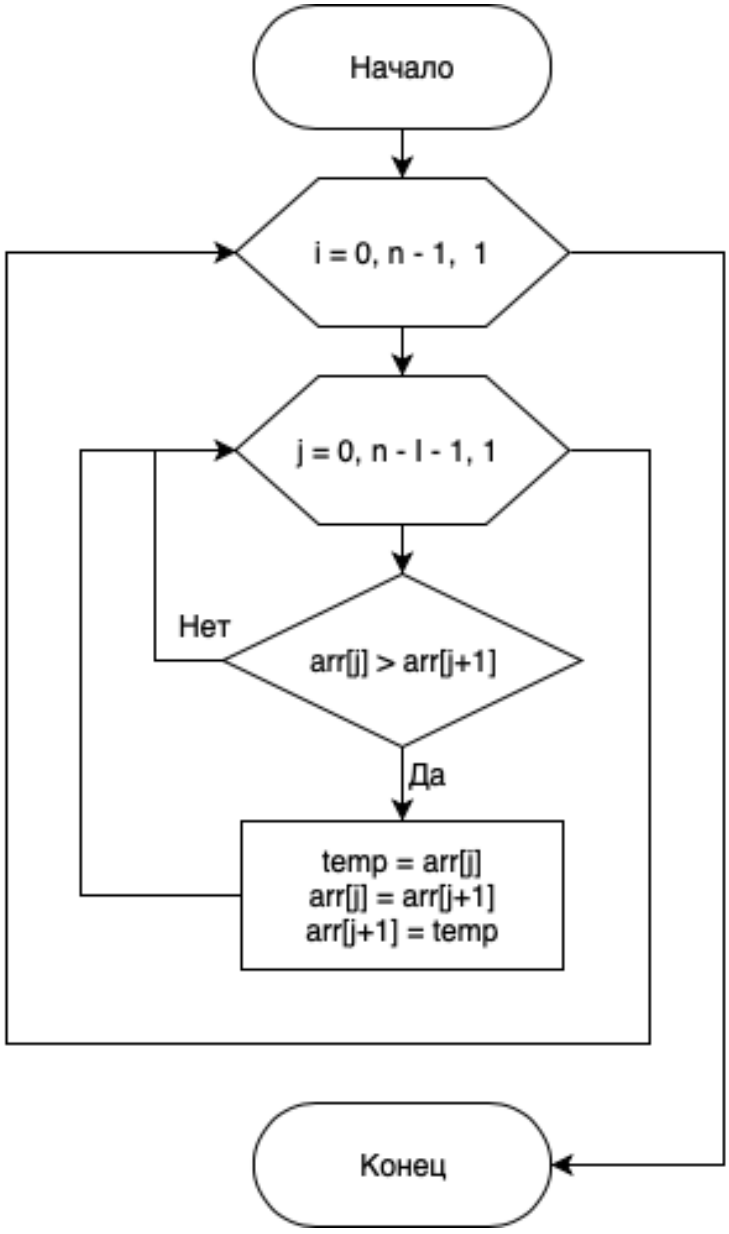

Рис. 2.1: Схема алгоритма сортировки пузырьком

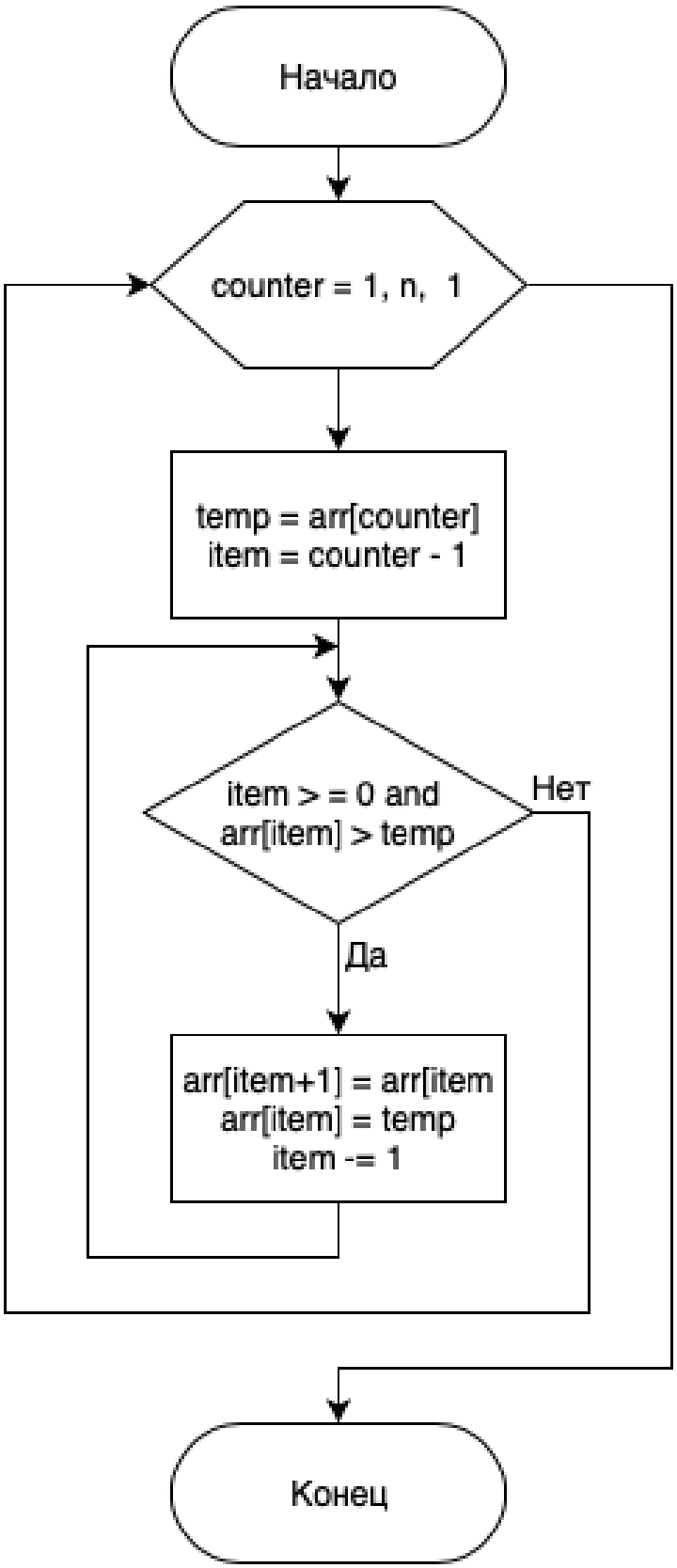

Рис. 2.2: Схема алгоритма сортировки вставками

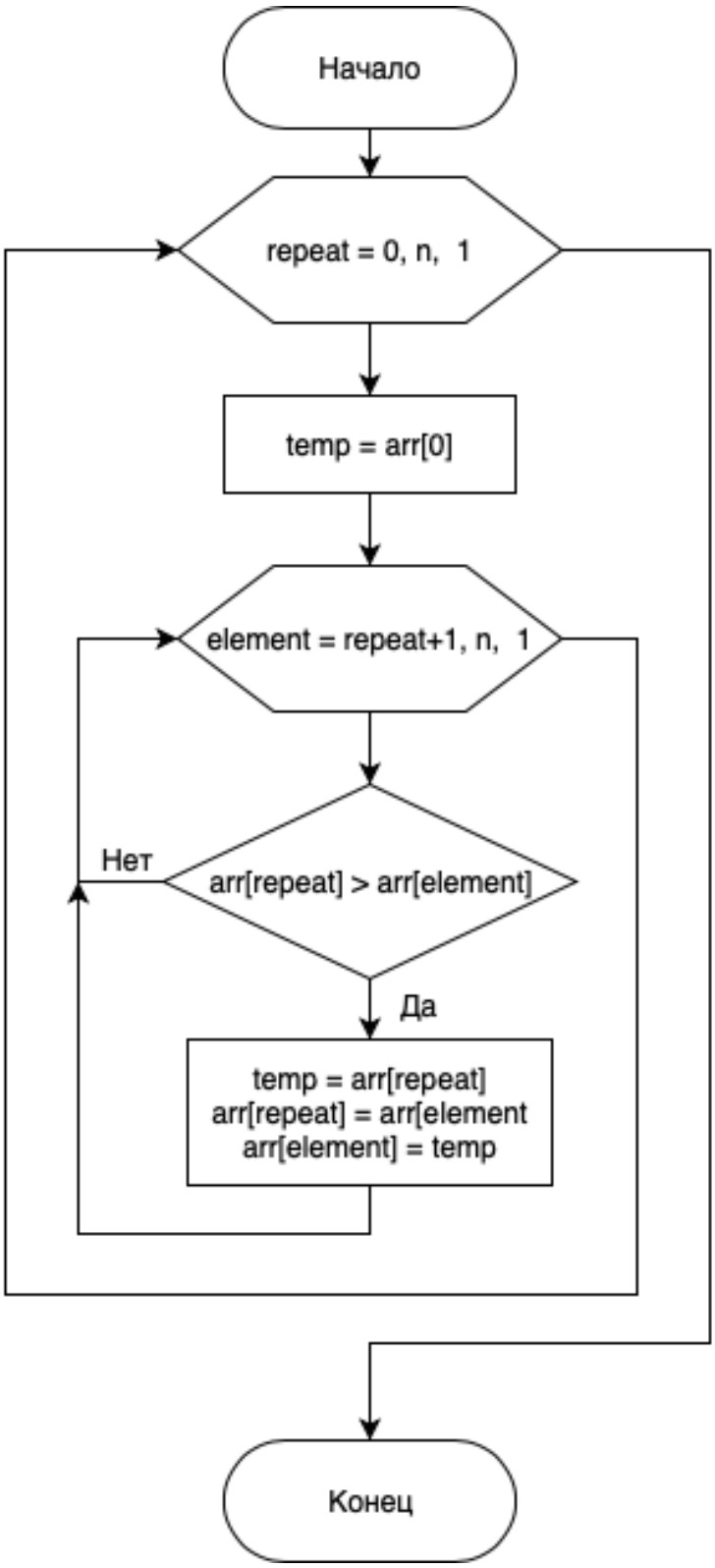

Рис. 2.3: Схема алгоритма сортировки выбором

2.2 Расчет сложности алгоритмов

Модель вычислений

Пусть любые арифметические операции имеют стоимость 1, операции словного перехода if-else оцениваются в 0. Тогда будем учитывать только стоимость вычисления логического выражения

В программе цикл начинается с инициализации и проверки, стоимость которых в сумме равна 2. Каждый раз в цикле происходит проверка на граничное значение и увеличение счетчика, каждая операция имеет стоимость 1, каждая операция цикла имеет добавочную стоимость 2.

Табл. 2.1 Построчная оценка веса для сортировки пузырьком

Код	Цена
//иниициализация и проверка	3
for (int i = 0; i < n - 1; i++) {	3
//иниициализация и проверка	4
for (int j = 0; j < n - i - 1; j++) {	4
if (arr[j] > arr[j + 1]) {	4
int temp = arr[j];	2
arr[j] = arr[j + 1];	4
arr[j + 1] = temp;	3
}	0
}	0

Сортировка пузырьком:

- Лучший случай: $3 + 7 * n + 8 * \frac{n-1}{2} = 3 + 7 * n + 8 * \frac{n*(n-1)}{2} = O(n^2)$ (отсортированный массив)
- Худший случай: $3 + 7 * n + \frac{n-1}{2} * (4 + 13) = 3 + 7 * n + 17 * \frac{n*(n-1)}{2} = O(n^2)$ (массив отсортирован наоборот)

Сортировка вставками:

- Лучший случай: $P(n) = 3*n+(n-1)*(2+2+6) = 13*n-10 = O(n)$ (отсортированный массив)
- Худший случай: $P(n) = 3 * n + 4 * (n - 1) + 4(\frac{n(n-1)}{2} - 1) + 5\frac{n(n-1)}{2} + 3(n - 1) = 4.5n^2 + 9.5n - 11 = O(n^2)$ (массив отсортирован наоборот)

Сортировка выбором:

- Лучший случай: $O(n^2)$
- Худший случай: $O(n^2)$

Основывась на статье из источника [3]

2.3 Вывод

В данном разделе были рассмотрены схемы алгоритмов сортировок пузырьком, вставками и выбором, а также проведен расчет сложности для худшего и лучшего случая.

3. Технологический раздел

В данном разделе будут приведены требования к разрабатываемому ПО, средства , используемые для реализации поставленных задач.

3.1 Требования к программному обеспечению

Программный комплекс должен реализовывать 3 алгоритма сортировки: пузырьком, вставками, выбором. Пользователь должен иметь возможность создать массив любой длины, а также иметь возможность наблюдать время за которое сортируется созданный им массив.

3.2 Средства реализации

Для выполнения поставленной задачи был использован язык программирования C++. Среда для разработки XCode. Для измерения процессорного времени была взята функция rdtsc из библиотеки ctime.

Данный язык обусловлен тем, что функции замеры времени могут считывать не только абсолютное время, но и процессорное.

3.3 Листинг кода

В данном подразделе приведены листинги кодов для алгоритмов сортировок пузырьком, вставками, выбором.

Листинг 3.1: Алгоритм сортировки пузырьком

```
1  void bubble_sort(int *arr, unsigned n) {
2    for (int i = 0; i < n - 1; i++)
3      for (int j = 0; j < n - i - 1; j++) {
4        if (arr[j] > arr[j + 1]) {
5          int temp = arr[j];
6          arr[j] = arr[j + 1];
7          arr[j + 1] = temp;
8        }
9      }
10 }
```

Листинг 3.2: Алгоритм сортировки вставками

```
1  void insertion_sort(int *arr, unsigned n) {
2    int temp, item;
3    for (int counter = 1; counter < n; counter++)
4    {
5      temp = arr[counter];
6      item = counter-1;
7      while(item >= 0 && arr[item] > temp)
8      {
9        arr[item + 1] = arr[item];
10       arr[item] = temp;
11       item--;
12     }
13   }
14 }
```

Листинг 3.3: Алгоритм сортировки выбором

```
1  void choices_sort(int *arr, unsigned n) {
2    for (int repeat_counter = 0; repeat_counter < n; repeat_counter++)
3    {
4      int temp = arr[0];
5      for (int element_counter = repeat_counter + 1; element_counter < n;  element_counter
           ++)
6      {
7        if (arr[repeat_counter] > arr[element_counter])
8        {
9          temp = arr[repeat_counter];
10         arr[repeat_counter] = arr[element_counter];
11         arr[element_counter] = temp;
12       }
13     }
14   }
15 }
```

3.4 Вывод

В данном разделе были рассмотрены требования к программному обеспечению, обозначен язык программирования, в котором проводилась данная работа, а также причина почему взят именн этот язык. В разделе приведен листинг кода в соответствии с требованиями к ПО.

4. Экспериментальный раздел

В данном разделе будет проведено исследование временных затрат разработанного ПО, будет проведен сравнительный анализ трех алгоритмов сортировок.

4.1 Сравнительный анализ

Замеры времени выполнялись для сортировки массивами длиной от 1000 до 10000 с шагом 1000. Массивы были трех видов: отсортированные, отсортированные в обратном порядке, случайные(числа в массиве генерируются случайным образом). Все замеры проводились на процессоре 1,4 GHz Intel Core i5 с памятью 8 ГБ 2133 MHz LPDDR3.

В таблицах 4.1, 4.2, 4.3 и рисунках 4.1, 4.2, 4.3 приведены результаты экспериментов с замером времени в тиках.

Таблица 4.1: Время в алгоритме сортировки пузырьком

Размер массива	**Лучший сл.**	**Случайн. посл.**	**Худший сл.**
1000	1394902	2100514	2410690
2000	5317552	8851960	9870330
3000	11599342	21510914	22701846
4000	213518842	42208300	38537536
5000	33353502	68555472	60887562
6000	46214164	96965718	85945234
7000	61765714	133482894	117172790
8000	85548032	175500448	158163078
9000	108727938	242365392	189925138
10000	133380770	302778376	238093546

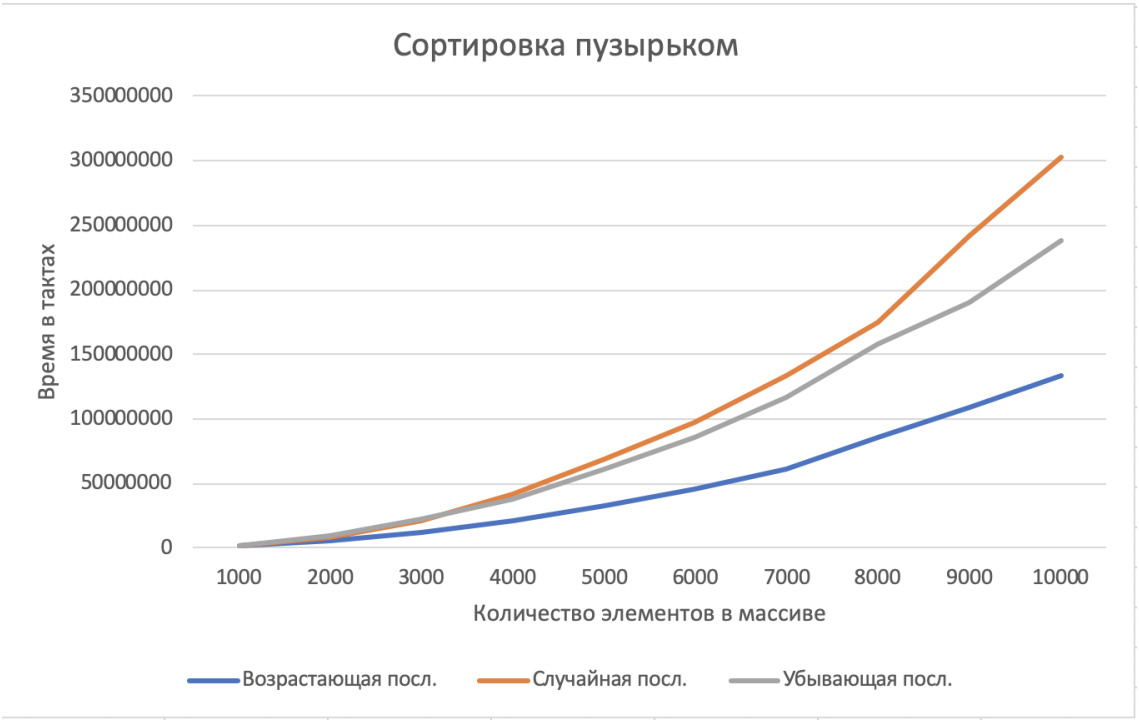

Рис. 4.1: График зависимости времени работы от размера массива в алгоритме сортировки пузырьком

Таблица 4.2: Время в алгоритме сортировки вставками

Размер массива	**Лучший сл.**	**Случайн. посл.**	**Худший сл.**
1000	3884	945430	1859214
2000	7554	4131256	7721438
3000	11638	8642752	16920316
4000	16570	15014674	31343220
5000	18762	23559030	51553560
6000	35108	34122758	68011852
7000	26892	47572958	92381688
8000	29986	60860702	118904484
9000	33746	77162658	149305702
10000	37576	93215776	191729656

Рис. 4.2: График зависимости времени работы от размера массива в алгоритме сортировки вставками

Таблица 4.3: Время в алгоритме сортировки выбором

Размер массива	**Лучший сл.**	**Случайн. посл.**	**Худший сл.**
1000	1419936	3497892	1873894
2000	5581406	12877722	7487530
3000	12634844	26650100	21068640
4000	22169658	43230950	37621488
5000	35096760	62053106	59106270
6000	49627130	84698072	80561108
7000	67094992	107109266	105201620
8000	88686984	138134540	135526774
9000	112261826	170070930	173078504
10000	137168498	205658698	199069930

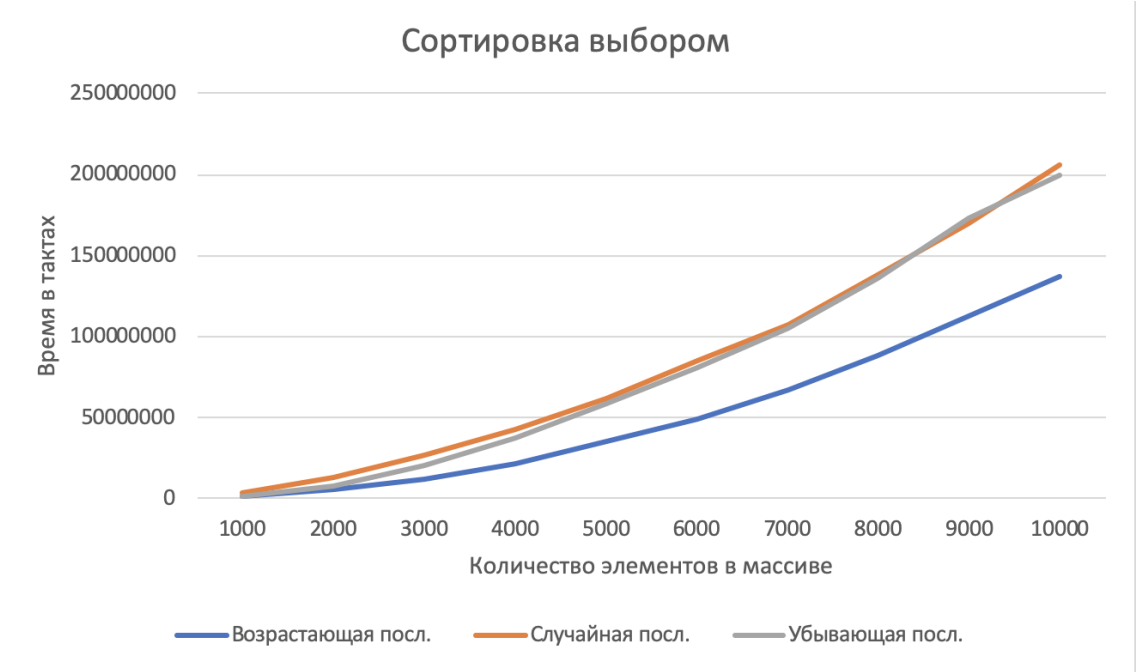

Рис. 4.3: График зависимости времени работы от размера массива в алгоритме сортировки выбором

4.2 Вывод

В данном разделе было проведено исследование алгоритмов сортировок на лучшие, средние и худшие случаи. Приведены графики зависимостей времени работы алгоритма от размеров массива.

Среди трех алгоритмов сортировки самым лучшим оказался алгоритм сортировки вставками. Данный алгоритм показал хороший результат в сортировке всех видов последовательностей.

Заключение

В ходе выполнения данной лабораторной работы были изучены и реализованы алгоритмы сортировки. Алгоритмы были исследованы на временные затраты, а также было приведено обоснование полученных результатов. В ходе исследования был найден оптимальный алгоритм.

Список литературы

1. Дж. Макконелл "Анализ алгоритмов. Вводный курс 2-е дополненное издание (2004) — ISBN 5-94836-005-9

2. Microsoft «rdstc» [Электронный ресурс]. — Режим доступа: https://docs.microsoft.com/ru-ru/cpp/intrinsics/rdtsc?view=vs-2019, свободный (Дата обращения 20.10.2019)

3. Сортировка выбором [Электронный ресурс]. — Режим доступа: https://neerc.ifmo.ru/wiki/index.php?title=Сортировка_выбором, свободный (Дата обращения 21.10.2019)