МГТУ им. Баумана

Лабораторная работа №3

По курсу: "Анализ алгоритмов"

Алгоритмы сортировок

Работу выполнил: Левушкин Илья, ИУ7-52Б

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

Москва, 2019

Оглавление

Введение

Алгоритмы сортировки находят широкое применение во многих сферах, поэтому сортировки являются одной из наиболее обширных и проработанных областей информатики.

В данной работе требуется реализовать и оценить трудоемкость трех алгоритмов сортировок:

- сортировка Шелла;
- гномья сортировка;
- сортировка расческой.

Цель работы: изучение алгоритмов сортировки.
Задачи работы:

- разработка и реализация алгоритмов;
- исследование временных затрат алгоритмов;
- описание и обоснование полученных результатов.

1 | Аналитический раздел

В данном разделе будут описаны алгоритмы сортировки Шелла, гномьей сортировки и сортировки расческой.

1.1 Описание алгоритмов

Задача сортировки формулируется следующим образом: дана последовательность элементов

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

Требуется упорядочить элементы по не убыванию или по не возрастанию - найти перестановку $(i_1, i_2, \cdots, i_n)$ ключей 1.1, либо по неубыванию:

$$a(i_1) \leq a(i_2) \leq \cdots \leq a(i_n) \tag{1.2}$$

либо по не возрастанию:

$$a(i_1) \geq a(i_2) \geq \cdots \geq a(i_n) \tag{1.3}$$

1.2 Сортировка Шелла

Сортировка Шелла — это алгоритм сортировки, являющийся усовершенствованным вариантом сортировки вставками. Идея метода Шелла состоит в сравнении элементов, стоящих не только рядом, но и на определённом расстоянии друг от друга. При сортировке Шелла сначала сравниваются и сортируются между собой значения, стоящие один от

другого на некотором расстоянии d. После этого процедура повторяется для некоторых меньших значений d, а завершается сортировка Шелла упорядочивающем элементов при d = 1, то есть обычной сортировкой вставками.

Среднее время работы алгоритма зависит от длин промежутков — d, на которых будут находиться сортируемые элементы исходного массива емкостью N на каждом шаге алгоритма. Первоначально используемая Шеллом последовательность длин промежутков:

$$d_1 = \frac{N}{2}, d_i = \frac{d_{i-1}}{2}, d_k = 1 \tag{1.4}$$

1.3 Гномья сортировка

Гномья сортировка — это алгоритм сортировки, похожий на сортировку вставками, но в отличие от последней перед вставкой на нужное место происходит серия обменов, как в сортировке пузырьком. Алгоритм находит первое место, где два соседних элемента стоят в неправильном порядке и меняет их местами. Он пользуется тем фактом, что обмен может породить новую пару, стоящую в неправильном порядке, только до или после переставленных элементов. Он не допускает, что элементы после текущей позиции отсортированы, таким образом, нужно только проверить позицию до переставленных элементов.

1.4 Сортировка расческой

Сортировка расческой является улучшенной модификацией сортировки пузырьком. В сортировке пузырьком, когда сравниваются два элемента, промежуток между элементами равен единице. Основная идея алгоритма сортировки расческой заключается в том, чтобы первоначально брать достаточно большое расстояние между сравниваемыми элементами и по мере упорядочивания массива сужать это расстояние вплоть до минимального. Первоначальный разрыв между сравниваемыми элементами лучше брать с учётом специальной величины, называемой фактором уменьшения, оптимальное значение которой равно примерно:

$$\frac{1}{1 - e^{\phi}} = 1,247 \cdots , \tag{1.5}$$

Где

e : основание натурального логарифма
ϕ : золотое сечение

Начальное расстояние между элементами равно размеру массива, разделённого на фактор уменьшения. Массив проходится с этим шагом, после чего шаг делится на фактор уменьшения и проход по списку повторяется вновь. Так продолжается до тех пор, пока разность индексов не достигает единицы. После этого массив продолжает упорядочивание обычным пузырьком.

1.5 Выводы по аналитической части

В данном разделе были описаны алгоритмы сортировки Шелла, быстрой сортировки и сортировки расческой.

2 | Конструкторский раздел

В данном разделе в соответствии с описанием алгоритмов, приведенными в аналитической части работы, будут рассмотрены схемы алгоритмов сортировки двух массивов данных, а также будет произведен расчёт сложности алгоритмов.

2.1 Схемы алгоритмов

В данном пункте представлены схемы алгоритмов сортировки Шелла (2.1), гномьей сортировки (2.2) и сортировки расческой (2.3)

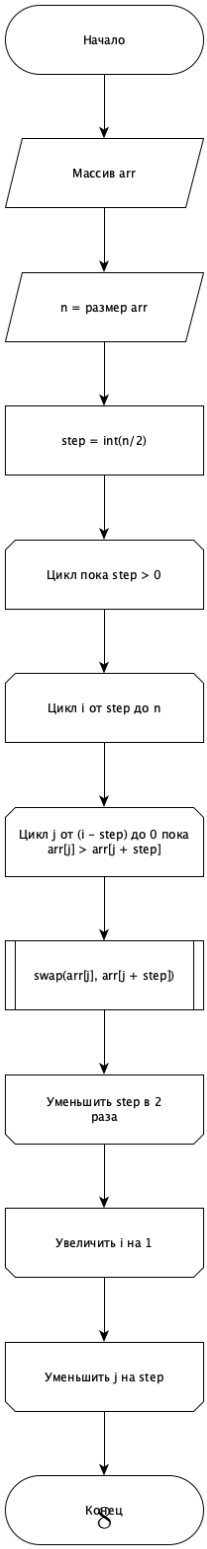


Рис. 2.1: Алгоритм сортировки Шелла.

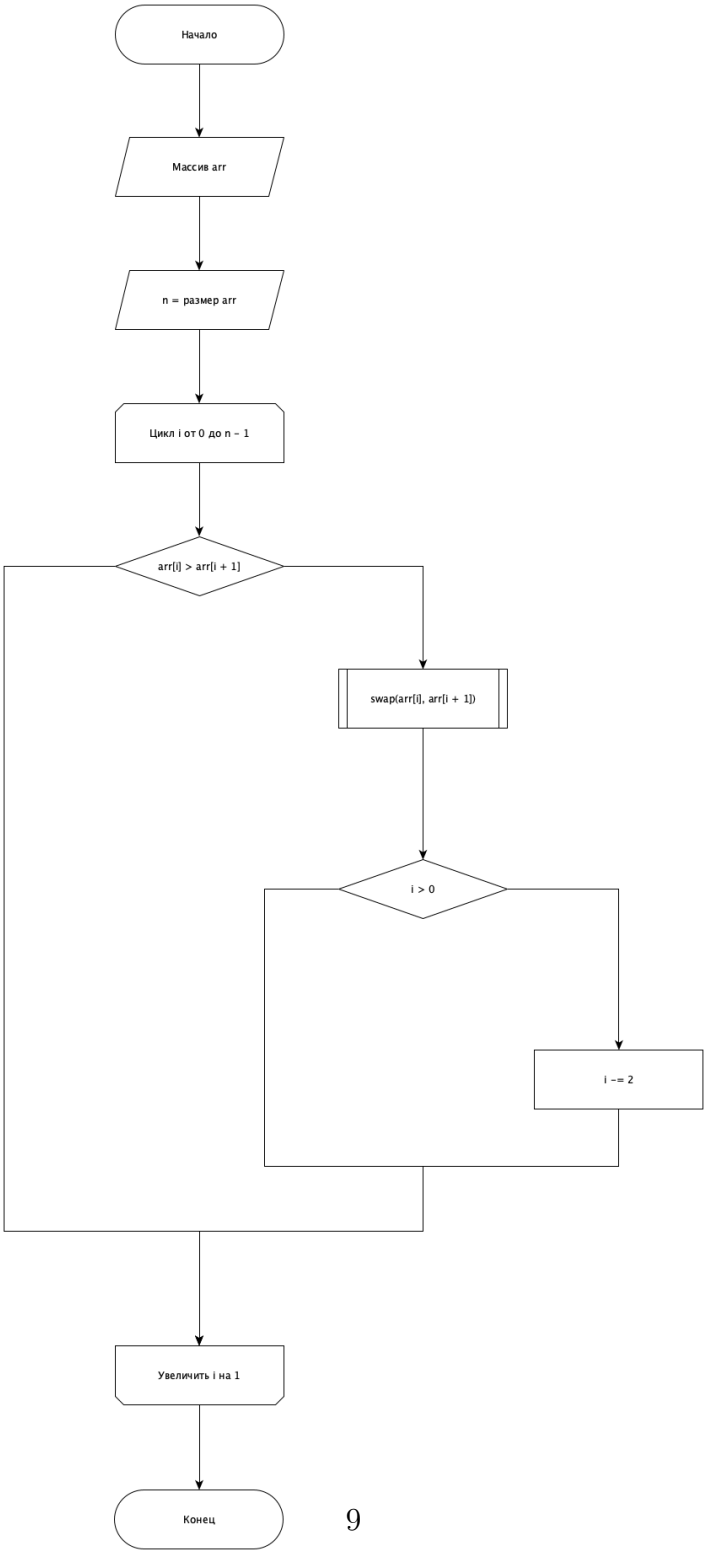


Рис. 2.2: Алгоритм гномьей сортировки.

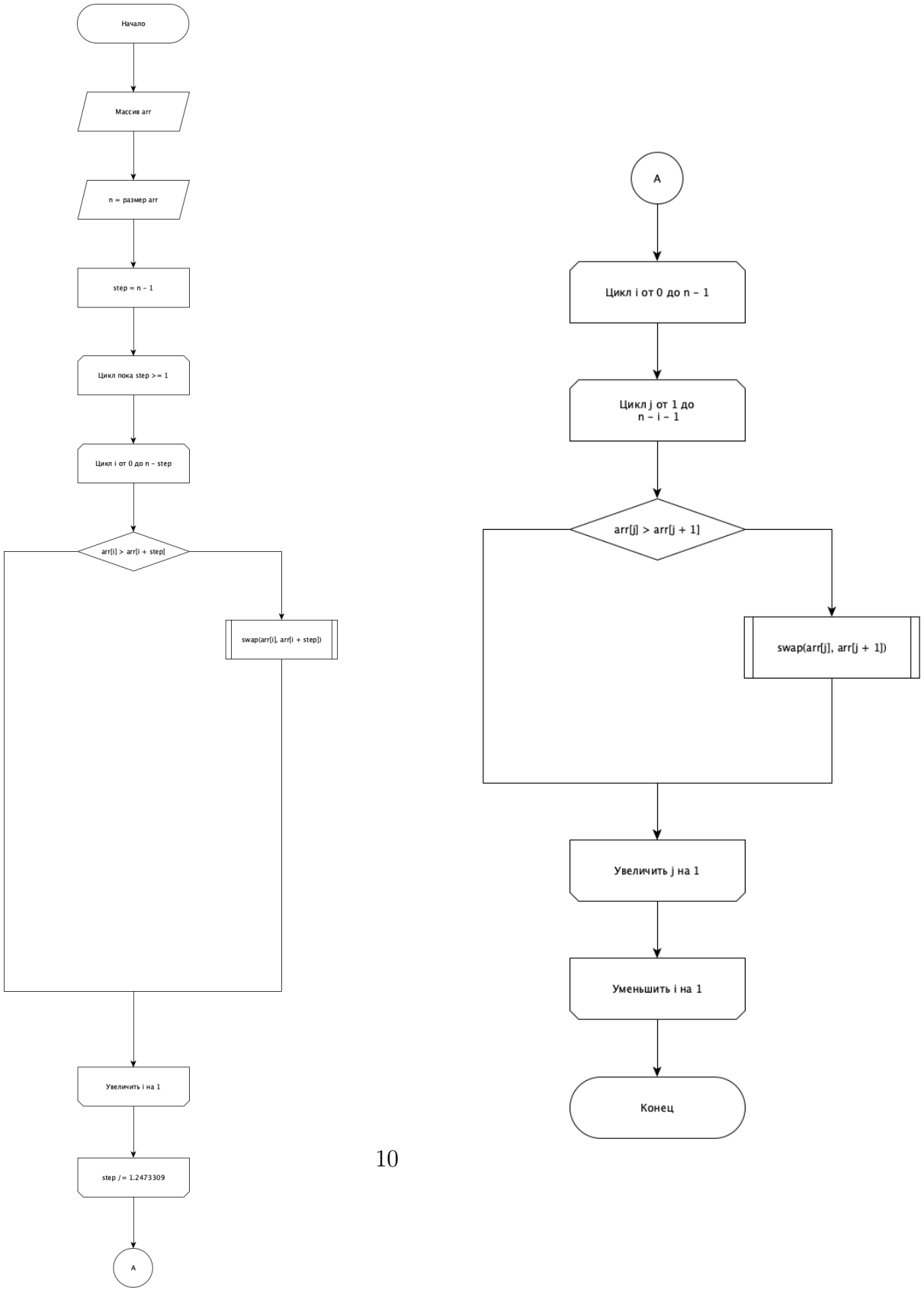


Рис. 2.3: Алгоритм сортировки расческой.

2.2 Расчет сложности алгоритмов

2.2.1 Модель вычислений

Для корректного расчёта сложности алгоритмов следует описать модель вычислений. Пусть любые арифметические операции имеют стоимость 1. Стоимость условного перехода if-else возьмем за 0 и будем учитывать лишь стоимость вычисления логического выражения. Цикл начинается с инициализации и проверки, стоимость которых в сумме составляет 2. В каждой итерации цикла происходит проверка условия и увеличение счётчика, каждый из которых имеет стоимость 1, соответственно, каждая итерация цикла имеет добавочную стоимость 2.

2.2.2 Сортировка Шелла [1]

- Лучший случай $\sim \Theta(n \log^2 n)$
- Худший случай $\sim \Theta(n^2)$

2.2.3 Гномья сортировка

- Лучший случай - $2 + (n-1) * 3 = 3n - 1 \sim \Theta(n)$
- Худший случай - $2+(n-1)(n-1)(3+10+1+1) = 15n^2-30n+17 \sim \Theta(n^2)$

2.2.4 Сортировка расческой [1]

- Лучший случай $\sim \Theta(n^2)$
- Худший случай $\sim \Theta(n \log n)$

2.3 Вывод из конструкторской части

В данном разделе были рассмотрены схемы алгоритмов сортировки Шелла, гномьей сортировки и сортировки расческой, а также был произведен расчёт сложности алгоритмов.

3 | Технологический раздел

В данном разделе будут рассмотрены требования к разрабатываемому программному обеспечению, средства, использованные в процессе разработки для реализации поставленных задач, а также представлен листинг кода программы.

3.1 Требования к программному обеспечению

Программное обеспечение должно реализовывать 3 алгоритма сортировки - сортировка Шелла, гномья сортировка и сортировка расческой. Пользователь должен иметь возможность произвести вычисления для массивов, размер которых он вводит, а также иметь возможность сравнить время работы этих алгоритмов.

3.2 Средства реализации

Для реализации поставленной задачи был использован язык программирования C++ [2]. Проет был выполнен в среде QT Creator [3]. Для измерения процессорного времени была использована ассемблерная инструкция rdtsc [4].

3.3 Листинг кода

На основе схем алгоритмов, представленных в конструкторском разделе, в соответствии с указанными требованиями к реализации с использова-

нием средств языка C++ было разработано программное обеспечение, содержащее реализации выбранных алгоритмов. В данном пункте приведён листинг этих реализаций: листинги 3.1, 3.2, 3.3.

Листинг 3.1: Алгоритм сортировки Шелла

```
1  void ShellSort (vector<int>& array)
2  {
3  int tmp;
4  int size = array.size();
5  for (int step = size / 2; step > 0; step /= 2)
6  {
7  for (int i = step; i < size; i++)
8  {
9  for (int j = i - step; j >= 0 && array[size_t(j)] > array[
       size_t(j + step)]; j -= step)
10 {
11 tmp = array[size_t(j)];
12 array[size_t(j)] = array[size_t(j + step)];
13 array[size_t(j + step)] = tmp;
14 }
15 }
16 }
17 }
```

Листинг 3.2: Алгоритм сортировки расческой

```
1  void comb(vector<int>& sort)
2  {
3  int size = sort.size();
4  double fakt = 1.2473309;
5  int step = size - 1;
6
7  while (step >= 1)
8  {
9  step /= fakt;
10 for (int i = 0; i < size - step; i++)
11 {
12 if (sort[i] > sort[i + step])
13 {
14 swap(sort[i], sort[i + step]);
15 }
```

```
16  }
17  }
18
19  for (int i = 0; i < size - 1; i++)
20  {
21  bool swapped = false;
22  for (int j = 0; j < size - i - 1; j++)
23  {
24  if (sort[j] > sort[j + 1])
25  {
26  swap(sort[j], sort[j + 1]);
27  swapped = true;
28  }
29  }
30  if (!swapped)
31  break;
32  }
33  }
```

Листинг 3.3: Алгоритмгномьей сортировки

```
1   void gnome_sort(vector<int>& sort)
2   {
3   int size = sort.size();
4   for (size_t i = 0; i < size - 1; i++)
5   {
6   if (sort[i] > sort[i + 1])
7   {
8   swap(sort[i], sort[i + 1]);
9   if (i != 0)
10  {
11  i -= 2;
12  }
13  }
14  }
15  }
```

3.4 Выводы по технологической части

В данном разделе были рассмотрены требования к разрабатываемому программному обеспечению, средства, использованные в процессе разработки, а также был представлен листинг реализаций выбранных алгоритмов.

4 | Экспериментальный раздел

В данном разделе будет проведено исследование временных затрат разработанного программного обеспечения, вместе с подробным сравнительным анализом реализованных алгоритмов на основе экспериментальных данных.

4.1 Сравнительное исследование

Замеры времени выполнялись на массивах размера от 100 до 1000 с шагом 100 для сортировки Шелла (табл. 4.1.1, рис. 4.1.1), гномьей сортировки (табл. 4.1.2, рис. 4.1.2) и сортировки расческой (табл. 4.1.3, рис. 4.1.3). Числа в матрицах генерировались случайным образом.

Таблица 4.1: Сравнение времени работы алгоритма сортировки Шелла в тактах процессора.

Размер массива	Лучший сл.	Средний сл.	Худший сл.
100	13577	59134	32601
200	34494	139994	76995
300	60990	215931	129692
400	80575	311772	164619
500	99125	361400	230048
600	133820	445444	265233
700	154214	514780	334282
800	173332	642633	345377
900	197332	765237	448265
1000	157747	779791	457217

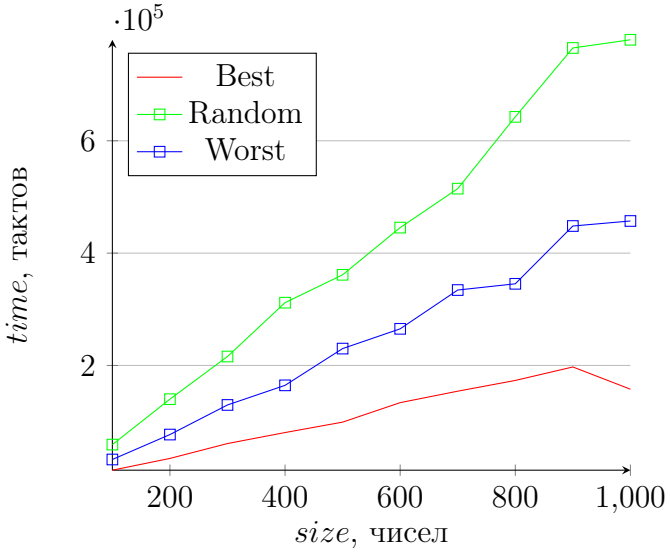
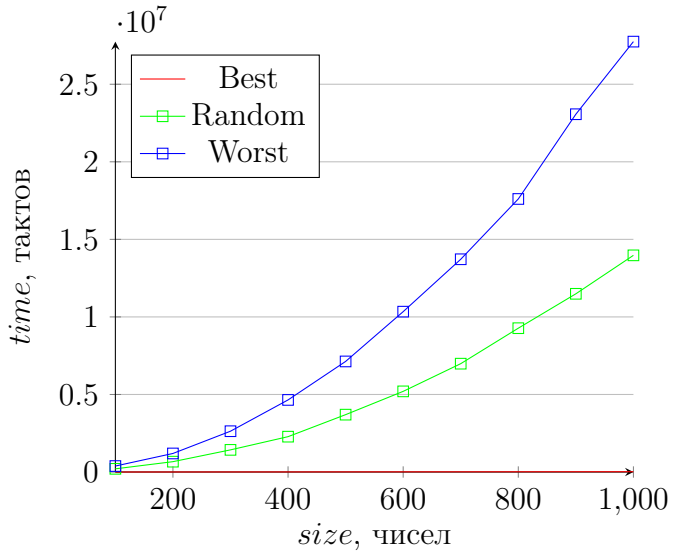


Таблица 4.2: Сравнение времени работы алгоритма гномьей сортировки в тактах процессора

Размер массива	Лучший сл.	Средний сл.	Худший сл.
100	2156	207040	386612
200	4333	668476	1197562
300	6462	1429117	2632503
400	7954	2280314	4648589
500	10781	3699399	7129986
600	12787	5200664	10339203
700	15001	6985759	13717497
800	17273	9273739	17604188
900	19385	11490163	23067462
1000	21410	13970190	27748485


$\cdot 10^7$
Best
Random
Worst
2.5
2
1.5
1
0.5
0
200
400
600
800
1,000
time, тактов
size, чисел

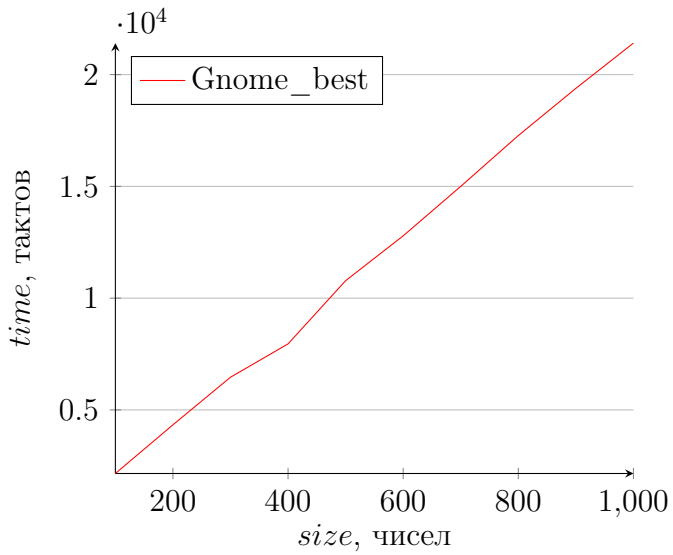
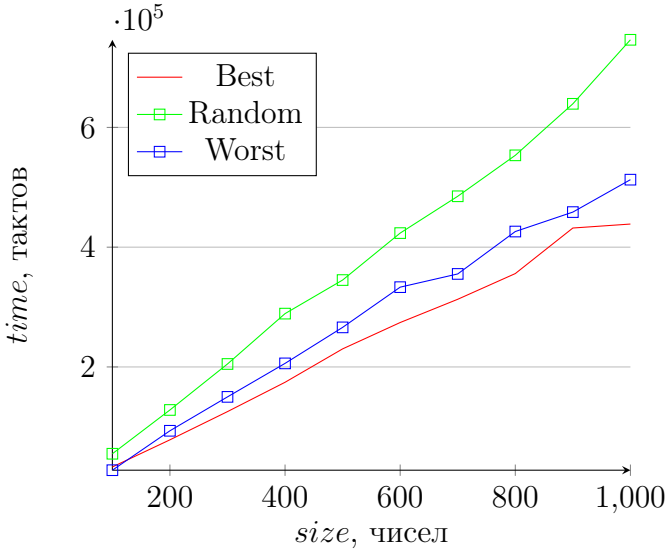

$\cdot 10^4$
Gnome_best
2
1.5
1
0.5
200
400
600
800
1,000
time, тактов
size, чисел

Таблица 4.3: Сравнение времени работы алгоритма сортировки расчески в 12 тактах процессора

Размер массива	Лучший сл.	Средний сл.	Худший сл.
100	32989	55036	27574
200	78287	128151	93381
300	125411	204941	150004
400	174387	289040	205934
500	230146	345098	266063
600	274164	423519	333236
700	312998	484999	355299
800	355905	553373	426127
900	431985	639390	458591
1000	438616	746300	512719



4.2 Выводы по экспериментальному разделу

В данном разделе было проведено исследование временных затрат разработанного программного обеспечения, вместе с подробным срав-

нительным анализом реализованных алгоритмов на основе экспериментальных данных.

Заключение

В ходе выполнения данной лабораторной работы были изучены и реализованы алгоритмы сортировки. Алгоритмы были разработаны и реализацованы, было проведено исследование временных затрат алгоритмов, а также дано описание и обоснование полученных результатов.

В аналитическом разделе было дано описание стандартного алгоритма и алгоритма Винограда. В конструкторском разделе был формализован и описан процесс вычисления пороизвдения двух матриц, разработаны алгоритмы. В технологическом разделе были рассмотрены требования к разрабатываемому программному обеспечению, средства, использованные в процессе разработки для реализации поставленных задач, а также представлен листинг кода программы. В экспериментальном разеде было проведено исследование временных затрат.

Литература

[1] Дж. Макконелл. Анализ алгоритмов. Вводный курс. — Москва: Техносфера, 2004. — ISBN 5-94836-005-9.

[2] ISO/IEC JTC1 SC22 WG21 N 3690 «Programming Languages — C++» [Электронный ресурс]. – Режим доступа: https://devdocs.io/cpp/, свободный. (Дата обращения: 29.09.2019 г.)

[3] QT Creator Manual [Электронный ресурс]. – Режим доступа: https://doc.qt.io/qtcreator/index.html, свободный. (Дата обращения: 29.09.2019 г.)

[4] Microsoft «rdtsc» [Электронный ресурс]. – Режим доступа: https://docs.microsoft.com/ru-ru/cpp/intrinsics/rdtsc?view=vs-2019, свободный. (Дата обращения: 29.09.2019 г.)