Государственное образовательное учреждение высшего
профессионального образования
“Московский государственный технический университет имени
Н.Э.Баумана”

Дисциплина: Анализ алгоритмов

Лабораторная работа № 3

Алгоритмы сортировок

Гибадулин О.Н.
Студент группы ИУ7-52

2019 г.

Содержание

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:

1. сортировка Шелла;
2. гномья сортировка;
3. сортировка расческой.

Цель работы: изучение алгоритмов сортировки.

Задачи работы:

1. разработка и реализация алгоритмов;
2. исследование временных затрат алгоритмов;
3. описание и обоснование полученных результатов.

1 Аналитический раздел

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

1.1 Описание алгоритмов

Задача сортировки формулируется следующим образом: дана последовательность элементов

$$a_1, a_2, \ldots, a_n \tag{1}$$

Требуется упорядочить элементы по не убыванию или по не возрастанию – найти перестановку (i_1, i_2, ..., i_n) ключей (1), либо по неубыванию:

$$a(i_1) \leq a(i_1) \leq ... \leq a(i_1) \tag{2}$$

либо по не возрастанию:

$$a(i_1) \geq a(i_1) \geq ... \geq a(i_1) \tag{3}$$

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:

$$d_1 = \frac{N}{2}, d_i = \frac{d_{i-1}}{2}, d_k = 1 \tag{4}$$

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:

$$\frac{1}{1 - e^{\phi}} = 1,247..., \tag{5}$$

где е — основание натурального логарифма, а ф — 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 Вывод

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 Разработка Алгоритмов

В данном пункте представлены схемы алгоритмов сортировки Шелла (рис. 2.1.1), гномьей сортировки (рис. 2.1.2) и сортировки расческой (рис. 2.1.3).

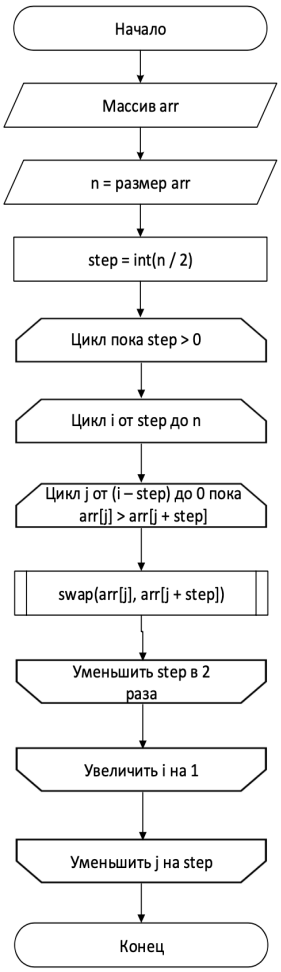

Рисунок 2.1.1. – схема алгоритма сортировки Шелла

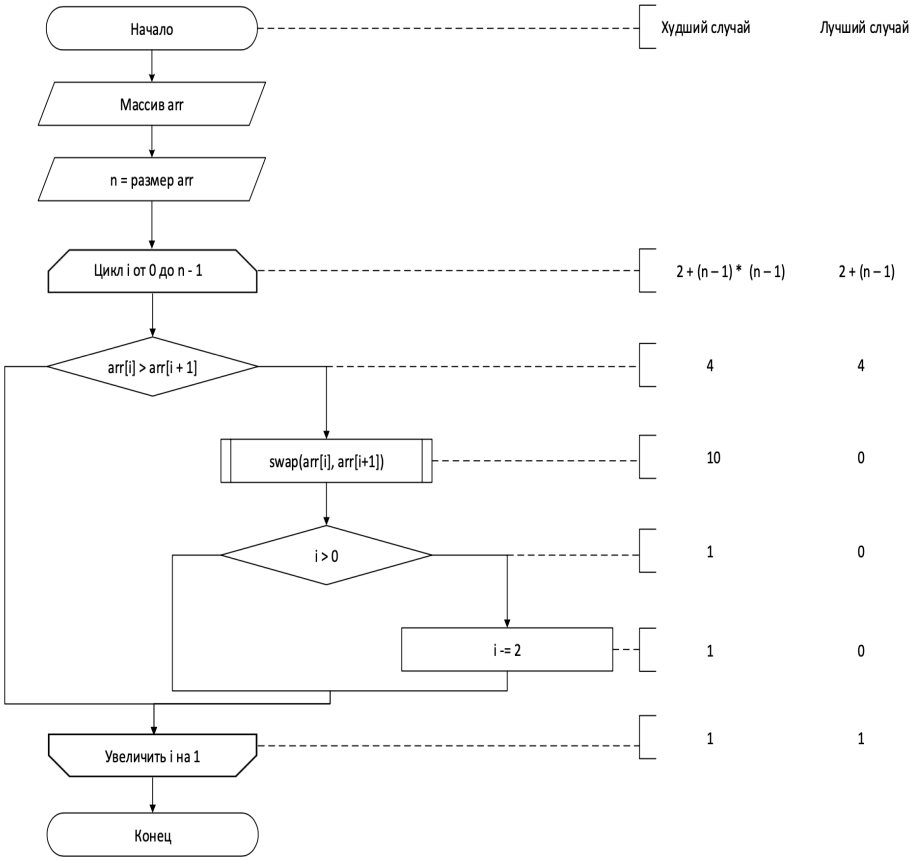

Рисунок 2.1.2. – схема алгоритма гномьей сортировки

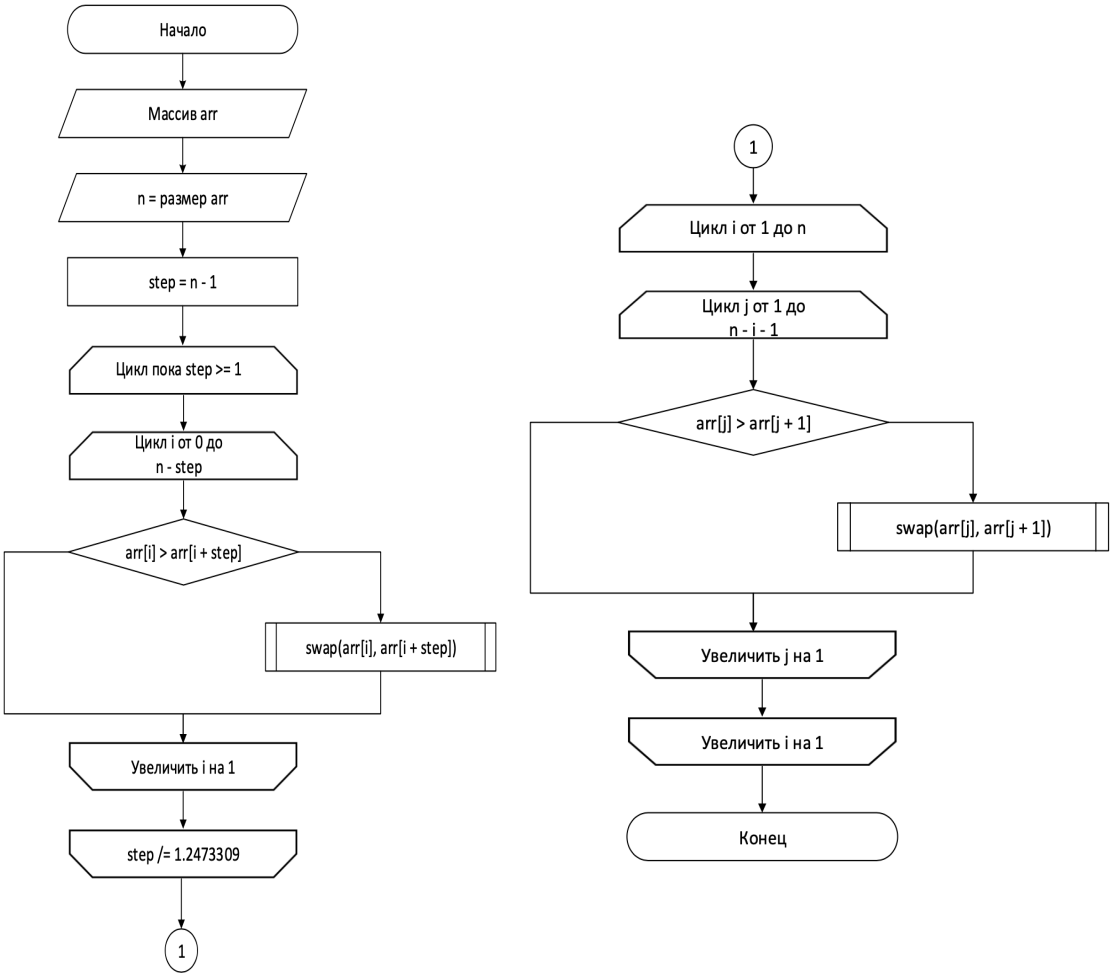

Рисунок 2.1.3. – схема алгоритма сортировки расческой

2.2 Расчет сложности алгоритмов

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.

Сортировка Шелла[1]:

- Лучший случай $\sim$ O$(nlog_2n)$

- Худший случай $\sim$ O(n^2n)

Гномья сортировка:

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

Сортировка расческой[1]:

- Лучший случай $\sim$ O(n^2)
- Худший случай $\sim$ O($nlog_2n$)

2.3 Вывод

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 XCode[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 использованием средств языка C++ было разработано программное обеспечение, содержащее реализации выбранных алгоритмов. В данном пункте приведён листиннг этих реализаций.

Листинг 1. Код сортировки Шелла

```
void ShellSort (vector<int>& arr, const int
    arrSize)
{
for (int step = arrSize / 2; step > 0; step /= 2)
    {
for (int i = step; i < arrSize; i++) {
for (int j = i - step; j >= 0 && arr[j] > arr[j +
    step]; j -= step) {
swap(arr[j], arr[j + step]);
}
}
}
}
```

Листинг 2. Код гномьей сортировки

```
void gnomeSort(vector<int>& arr, const int
    arrSize) {
```

```
        for (std::size_t i = 0; i + 1 < arrSize; ++i)
        if (arr[i] > arr[i + 1]) {
        swap(arr[i], arr[i + 1]);

        if (i != 0) {
        i -= 2;
        }
        }
        }
```

Листинг 3. Код сортировки расчёской

```
        void combSort(vector<int>& arr, const int arrSize
            ) {
        const double fakt = 1.2473309;
        double step = arrSize - 1;

        while (step >= 1) {
        for (size_t i = 0; i + step < arrSize; ++i) {
        if (arr[i] > arr[i + step]) {
        swap(arr[i], arr[i + step]);
        }
        }
        step /= fakt;
        }

        for (int i = 0; i < arrSize - 1; ++i) {
        bool swapped = false;
        for (size_t j = 0; j < arrSize - i - 1; ++j) {
        if (arr[j] > arr[j + 1]) {
        swap(arr[j], arr[j + 1]);
        swapped = true;
        }
        }

        if (!swapped)
        break;
        }
        }
```

3.4 Вывод

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 1000 до 10000 с шагом 1000 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. Все замеры были произведены на процессоре 2,7 GHz Intel Core i5 с паматью 8 ГБ 1867 MHz DDR3.

Таблица 4.1.1. Сравнение времени работы алгоритма сортировки Шелла в тактах процессора

Размер массива	Лучший сл.	Средний сл.	Худший сл.
1000	268019	409070	239524
2000	446201	464129	457456
3000	635237	691899	646896
4000	826583	1034587	846536
5000	1148264	1269852	1169357
6000	1415645	1504515	1603466
7000	1671576	1755663	1713003
8000	1889102	2086757	2045428
9000	2176738	2174689	2260487
10000	2549698	2707274	2739824

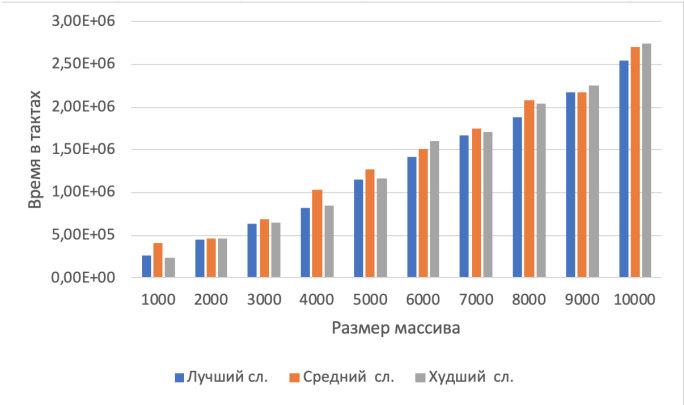

Рисунок 4.1.1. – график зависимости времени работы алгоритма сортировки Шелла

Таблица 4.1.2. Сравнение времени работы алгоритма гномьей сортровки в тактах процессора

Размер массива	Лучший сл.	Средний сл.	Худший сл.
1000	22208	1816344	3135731
2000	41042	5548871	11238385
3000	60411	12333532	24723436
4000	76779	22700661	43849979
5000	92363	35109631	68791066
6000	110856	51423543	100791780
7000	118068	70591770	141610909
8000	135977	95504194	178342099
9000	155488	117782072	226799242
10000	171335	136008021	277913166

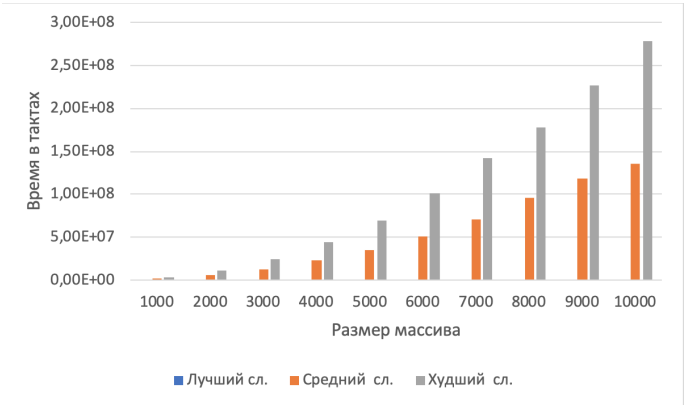

Рисунок 4.1.2. – график зависимости времени работы алгоритмов гномьей сортровки

Таблица 4.1.3. Сравнение времени работы алгоритма сортировки расчески в

тактах процессора

Размер массива	Лучший сл.	Средний сл.	Худший сл.
1000	772173	960189	973066
2000	1431159	1392252	1373447
3000	2728420	2247682	2420525
4000	3281531	3413904	3197173
5000	3855464	3947328	4529103
6000	4913651	5158414	5437713
7000	6524127	6585447	5854714
8000	6486076	7400221	6878903
9000	7791677	8109298	7863042
10000	8330758	9420135	8823914

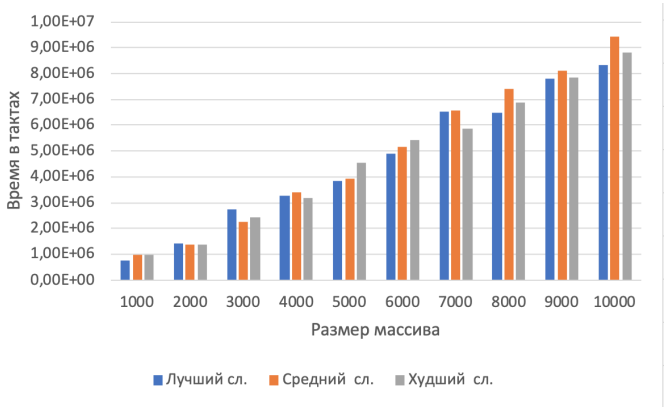

Рисунок 4.1.3. – график зависимости времени работы алгоритма сортировки расчески

4.2 Вывод

В данном разделе было проведено исследование временных затрат разработанного программного обеспечения, вместе с подробным сравнительным 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.

Список литературы

[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] Apple «Apple Developer Documentation» [Электронный ресурс]. – Режим доступа: https://developer.apple.com/documentation/, 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.)