Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

НАПРАВЛЕНИЕ ПОДГОТОВКИ «09.03.04 Программная инженерия»

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ №3

Тема: Алгоритмы сортировки

Студент: Сироткина П.Ю.

Группа: ИУ7-56Б

Оценка: _____

Преподаватель: Волкова Л.Л.

Москва, 2021 г.

Содержание

Введение

Алгоритм сортировки — это алгоритм для упорядочивания элементов в массиве каких-либо данных. Другими словами, это перегруппировка данных по какому-либо ключу. По некоторым источникам, именно программа сортировки стала первой программой для вычислительных машин.

В основном алгоритмы сортировки являются не конечной целью, а промежуточной - отсортированные данные намного легче обрабатывать. Например, чтобы удалить дупликаты в массиве, будет разумно сначала отсортировать массив. Также намного легче производить поиск какой-либо информации в заранее отсортированном массиве. Ассимптотическая сложность при поиске в отсортированном массиве составляет O(log(n)) против O(n) в произвольном массиве данных.

Каждый алгоритм сортировки имеет свои особенности реализации, однако в конечном итоге эту задачу можно разбить на следующие шаги:

— Сравнение пары элементов массива. Для этого нужно указать способ сравнения (бинарное отношение) для двух элементов, например, учесть природу данных или задать ключ в случае, если записи в массиве имеют несколько полей;

— Перестановка двух элементов, которые могут располагаться как последовательно друг за другом, так и находиться в разных частях массива;

— Алгоритм должен завершать свою работу, когда массив становится полностью упорядоченным.

Алгоритмы сортировки оцениваются по скорости выполнения и эффективности использования памяти. Обычно ассимптотическая сложность алгоритмов сортировок массивов размерностью n лежит в диапозоне от $O(n)$ до $O(n^2)$.

Целью лабораторной работы является изучение и реализация алгоритмов сортировки.

Задачи лабораторной работы:

а) Рассмотреть и изучить следующие 3 сортировки: пузырьком, слиянием и гномья.

б) Реализовать сортировки.

в) Рассчитать трудоемкость сортировок.

г) Сравнить временные характеристики экспериментально.

д) На основании проделанной работы сделать выводы.

1 Аналитический раздел

В данном разделе будут описаны ключевые идеи каждого из выбранных алгоритмов сортировки[1].

1.1 Сортировка пузырьком

Сортировка пузырьком является одним из простейших алгоритмов сортировки данных.

Пусть имеется массив для сортировки размерностью N.

В классической версии массив обходится слева направо. Алгоритм состоит из повторяющихся проходов по этому массиву - за каждый проход элементы последовательно сравниваются попарно и, если порядок в паре неверный, выполняется перестановка элементов так, чтобы пара стала упорядоченной.

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.

В результате каждого прохода по внутреннему циклу определяется очередной максимум подмассива, этот максимум помещается в конец рядом с предыдущим наибольшим элементом, вычисленным на предыдущем проходе, а наименьший элемент перемещается ближе к началу массива («всплывает» до нужной позиции, как пузырёк в воде — отсюда и название алгоритма).

1.2 Сортировка выбором

Шаги алгоритма:

— Находится номер минимального значения в текущем списке;

— Производится обмен этого значения со значением первой неотсортированной позиции (обмен не нужен, если минимальный элемент уже находится на данной позиции);

— Сортируется хвост списка, исключив из рассмотрения уже отсортированные элементы.

1.3 Гномья сортировка

Гномья сортировка очень похожа на сортировку вставками, но ее отличие от этой сортировки заключается в том, что перед вставкой элемента в остортированную часть на нужное место происходит серия обменов аналогично сортировке пузырьком (см. пункт 1.1).

Гномья сортировка основана на технике сортировки линии цветочных горшков, используемой обычным голландским садовым гномом.

В процессе прохода по массиву рассматриваются текущий и предыдущий элементы: если они в правильном порядке, то нужно перейти на один шаг вперед, иначе необходимо поменять их местами и сделать шаг назад. Граничные условия: если нет предыдущего элемента, то нужно перейти на одну позицию вперед; если нет следующего элемента, то алгоритм закончил работу.

1.4 Вывод

В данном разделе были рассмотрены ключевые идеи выбранных алгоритмов сортировки данных: пузырьком, выбором и гномья.

2 Конструкторский раздел

В данном разделе представлены схемы алгоритмов, описанных в аналитическом разделе, а также будет подсчитана их трудоемкость.

2.1 Схемы алгоритмов

На рисунке 2.1 представлена схема алгоритма сортировки пузырьком.

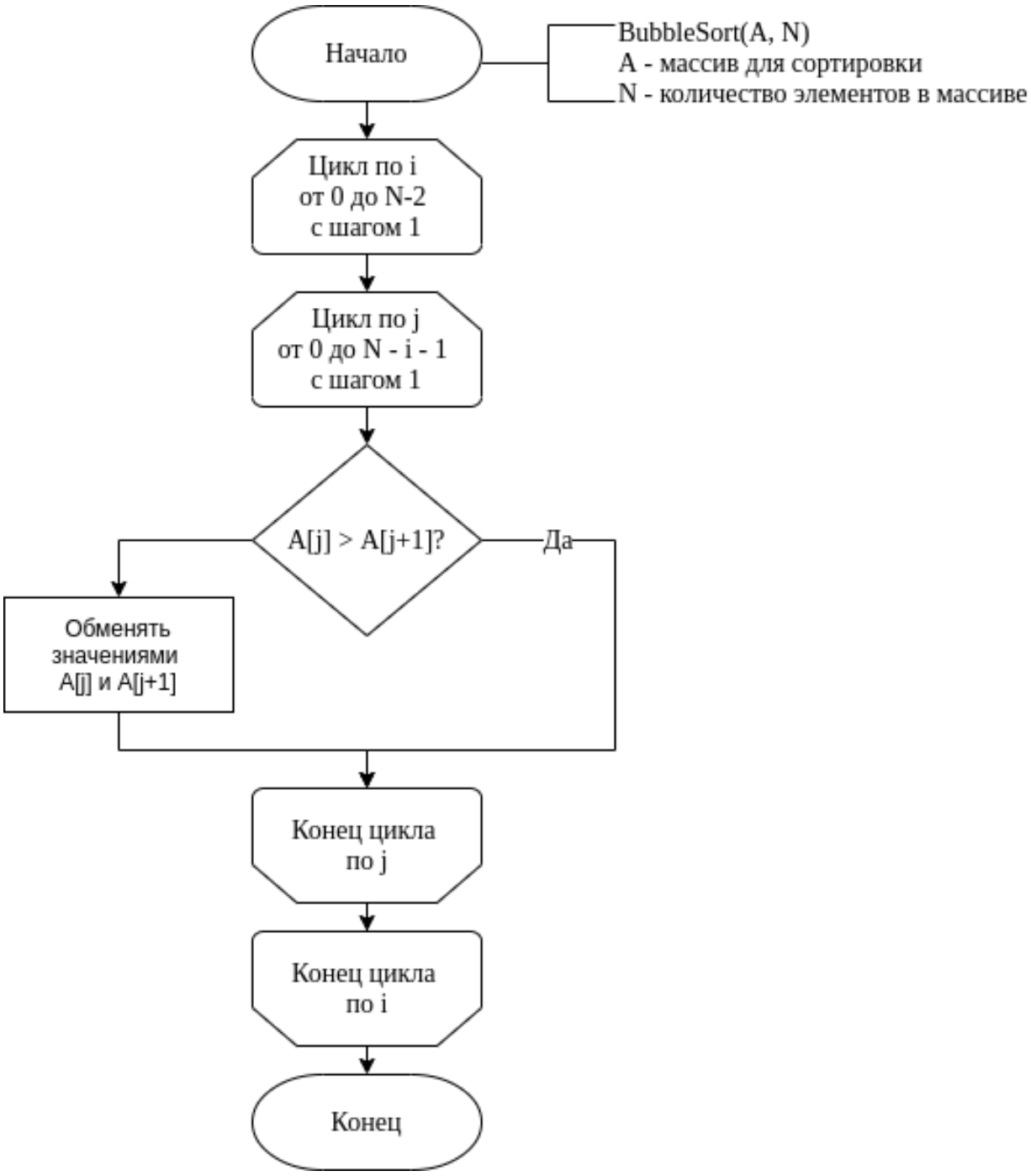

Рисунок 2.1 — Схема алгоритма сортировки пузырьком

На рисунке 2.2 представлена схема алгоритма сортировки выбором.

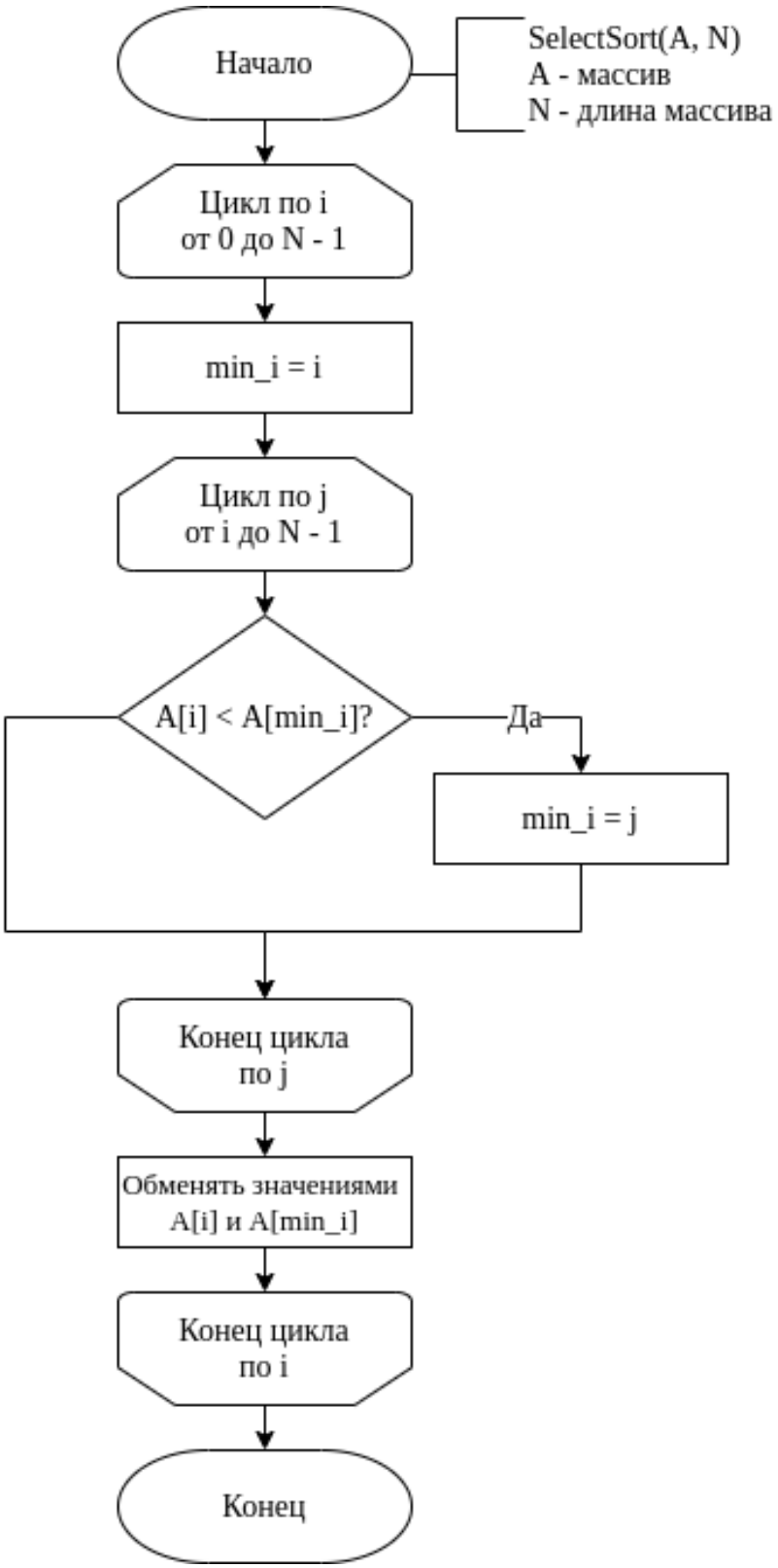

Рисунок 2.2 — Схема алгоритма сортировки выбором

На рисунке 2.3 представлена схема алгоритма гномьей сортировки.

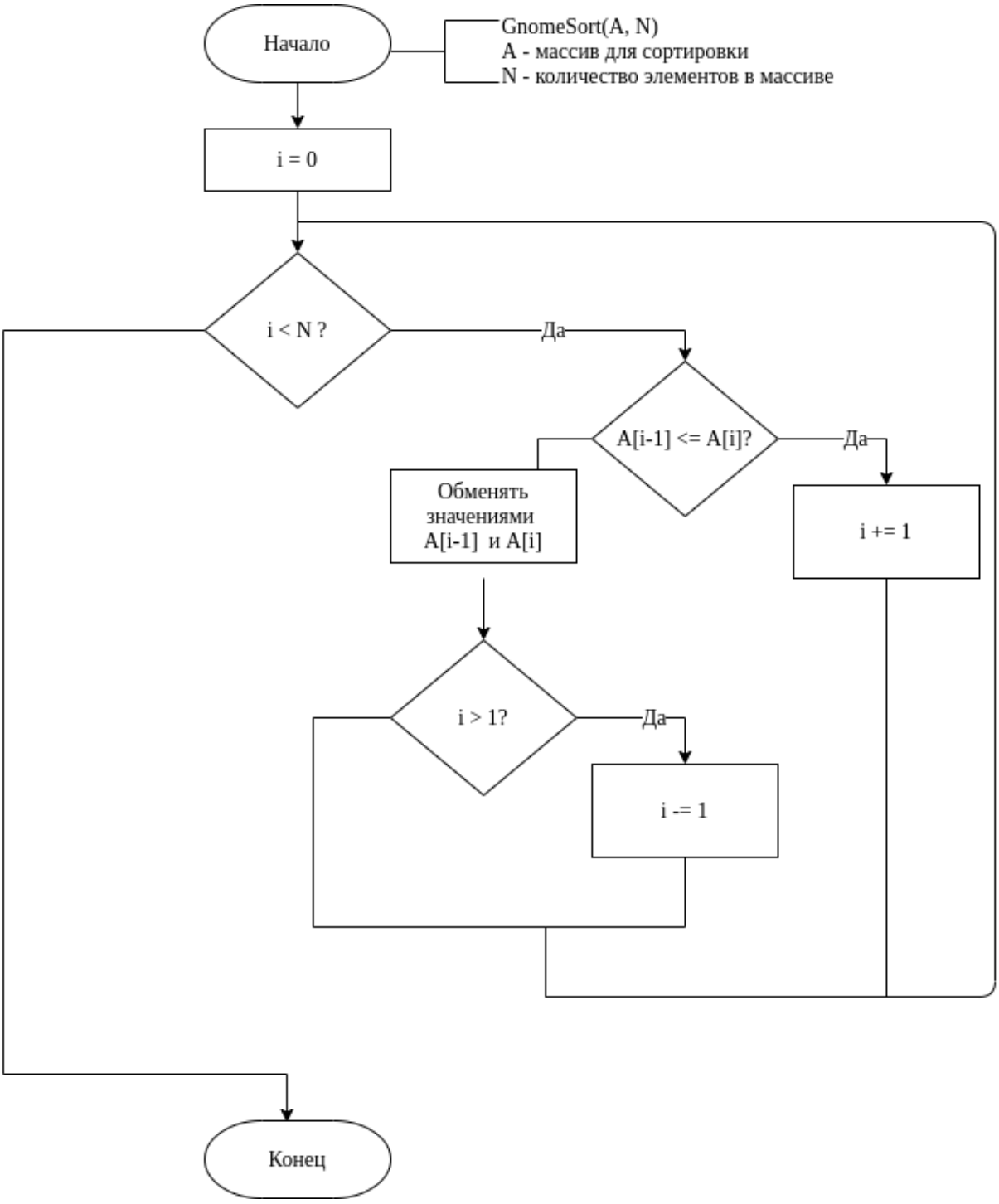

Рисунок 2.3 — Схема алгоритма гномьей сортировки

2.2 Трудоемкость алгоритмов

Модель вычислений

Для последующего вычисления трудоемкости вводится следующая модель вычислений:

а) Операции из списка (2.1) имеют трудоемкость 1.

$$+, -, =, ==, !=, <, >, <=, >=, [], ++, --\tag{2.1}$$

б) Операции из списка (2.2) имеют трудоемкость 2.

$$*, /, //, \%\tag{2.2}$$

в) Трудоемкость оператора выбора if (условие) then (блок кода A) else (блок кода B) рассчитывается, как (2.3).

$$f_{if} = f_{\text{условия}} + \begin{cases} f_A, & \text{если условие выполняется,} \\ f_B, & \text{иначе.} \end{cases}\tag{2.3}$$

г) Трудоемкость цикла рассчитывается по формуле (2.4).

$$f = f_{\text{инициализации}} + f_{\text{сравнения}} + N(f_{\text{тела}} + f_{\text{инкремента}} + f_{\text{сравнения}})\tag{2.4}$$

д) Трудоемкость вызова функции равна 0.

Трудоемкость алгоритма сортировки пузырьком

Пусть имеется массив размерностью N.

Во внутреннем цикле предел переменный, поэтому напрямую вычислить трудоемкость не получится. Воспользуемся следующей формулой:

$$f = f_{\text{служебное}} + Q_{\text{Сравнения}} \cdot f_{\text{на 1 сравнение}}\tag{2.5}$$

$f_{\text{служебное}}$ - затраты на содержание внешнего цикла и инициализацию и стартовое сравнение вложенного цикла.

$Q_{\text{Сравнения}}$ - количество сравнений.

$f_{\text{на 1 сравнение}}$ - затраты на содержание 1 сравнения, т.е. условия, инкремента и сравнения вложенного цикла.

$$f_{\text{на 1 сравнение}} = 7 + \begin{cases} 0, & \text{в лучшем случае} \\ 9, & \text{в худшем случае} \end{cases} \tag{2.6}$$

$$Q_{\text{Сравнения}} = \frac{N \cdot (N-1)}{2} = \frac{N^2}{2} - \frac{N}{2} \tag{2.7}$$

$$f_{\text{Служебное}} = 3 + (N-1) \cdot (3+3) \tag{2.8}$$

Трудоёмкость в **лучшем** случае (когда массив уже отсортирован) описывается формулой(2.14):

$$f_{best} = 3 + (N-1) \cdot 6 + (\frac{N^2}{2} - \frac{N}{2}) \cdot 7 = \frac{7 \cdot N^2}{2} + \frac{5 \cdot N}{2} - 3 \approx \frac{7 \cdot N^2}{2} = O(N^2) \tag{2.9}$$

Трудоёмкость в **худшем** случае (когда массив отсортирован в обратном порядке) описывается формулой (2.15):

$$f_{worst} = 3 + (N-1) \cdot 6 + (\frac{N^2}{2} - \frac{N}{2}) \cdot 16 = 8 \cdot N^2 - 2 \cdot N - 3 \approx 8 \cdot N^2 = O(N^2) \tag{2.10}$$

Трудоемкость алгоритма сортировки выбором

Пусть имеется массив размерностью N.

Аналогично анализу трудоемкости сортировки пузырьком:

$$f_{\text{на 1 сравнение}} = 5 + \begin{cases} 0, & \text{в лучшем случае} \\ 1, & \text{в худшем случае} \end{cases} \tag{2.11}$$

$$Q_{\text{Сравнения}} = \frac{N \cdot (N-1)}{2} = \frac{N^2}{2} - \frac{N}{2} \tag{2.12}$$

$$f_{\text{Служебное}} = 2 + N \cdot (2 + 3) \tag{2.13}$$

Трудоёмкость в **лучшем** случае (когда массив уже отсортирован) описывается формулой(2.14):

$$f_{best} = 5 + \frac{N^2}{2} - \frac{N}{2} + 2 + 5 \cdot N \approx \frac{N^2}{2} = O(N^2) \tag{2.14}$$

Трудоёмкость в **худшем** случае (когда массив отсортирован в обратном порядке) описывается формулой (2.15):

$$f_{worst} = 6 + \frac{N^2}{2} - \frac{N}{2} + 2 + 5 \cdot N \approx \frac{N^2}{2} = O(N^2) \tag{2.15}$$

Трудомекость алгоритма гномьей сортировки

Гномья сортировка практически эквивалентна сортировке вставками, соответственно:

Трудоёмкость в **лучшем** случае (когда массив уже отсортирован) составляет $O(N^2)$.

Трудоёмкость в **худшем** случае (когда массив отсортирован в обратном порядке) составляет $O(N^2)$.

2.3 Вывод

На основе теоретических данных, полученных из аналитического раздела, были построены схемы трех алгоритмов сортировки. На основе введенной модели вычислений оценена трудоемкость алгоритмов в лучшем и худшем случаях.

3 Технологический раздел

В данном разделе приведны требования к программному обеспечению, средства реализации и листинги кода.

3.1 Требования к ПО

К программе предъявляются следующие требования:

— На вход подается размер массива (целое неотрицательное число), а затем вводятся элементы массива (целые числа);

— Результат работы программы - массив, отсортированный 3 разными сортировками: пузырьком, слиянием и гномья;

— Программа должна корректно реагировать на любые действия пользователя, в том числе и корректно обрабатывать массив длины 0.

3.2 Средства реализации

В качестве языка программирования для реализации лабораторной работы был выбран язык C. Выбор этого языка обусловен его быстродействием и эффективностью, это легкочитаемый, лаконичный и гибкий язык. Также выбор обусловлен моим личным желанием получить больше практики написания программ на этом языке.

3.3 Листинг кода

В листингах 3.1-3.3 представлены реализации алгоритмов сортировки.

Листинг 3.1 — Сортировка пузырьком

```
1 void bubble_sort(int a[], int n)
2 {
3     for (int i = 0; i < n - 1; i++)
4         for (int j = 0; j < n - i - 1; j++)
5             if (a[j] > a[j + 1])
6                 swap_int(&a[j], &a[j + 1]);
7 }
```

Листинг 3.2 — Сортировка выбором

```
8  void select_sort(int a[], int n)
9  {
10     int min_i;
11     for (int i = 0; i < n; i++)
12     {
13         min_i = i;
14         for (int j = i; j < n; j++)
15         {
16             if (a[j] < a[min_i])
17             {
18                 min_i = j;
19             }
20         }
21         swap_int(&a[i], &a[min_i]);
22     }
23 }
24 }
```

Листинг 3.3 — Гномья сортировка

```
25 void gnome_sort(int a[], int n)
26 {
27     int i = 1;
28     while (i < n)
29     {
30         if (a[i - 1] <= a[i])
31         {
32             i += 1;
33         }
34         else
35         {
36             swap_int(&a[i - 1], &a[i]);
37             if (i > 1)
38             {
39                 i--;
40             }
41         }
42
43     }
44 }
```

Листинг 3.4 — Вспомогательная функция

```
45 void swap_int(int *a, int *b)
46 {
47     int temp = *a;
48     *a = *b;
49     *b = temp;
50 }
```

3.4 Вывод

В данном разделе были реализованы выбранные алгоритмы сортировок: пузырьком, выбором и гномья.

4 Экспериментальный раздел

В данном разделе представлен пользовательский интерфейс, а также проведена оценка эффективности алгоритмов.

4.1 Пример работы

На рисунке 4.1 приведен пример работы программы.

```
polina@polina-IdeaPad-5-14ARE05:~/aa/lab_03$ ./app.exe
Выберите действие:

1) Ввести массив и отсортировать его тремя разными сортировками;
2) Показать сравнительный анализ эффективности алгоритмов сортировки;
3) Выход.

Ответ: 1

Введите размерность массива: 8

Введите элементы массива:

A[0] = -56

A[1] = 0

A[2] = 56

A[3] = 2

A[4] = 7

A[5] = 4

A[6] = 56

A[7] = 55

Результат сортировки пузырьком:

A = [-56, 0, 2, 4, 7, 55, 56, 56]


Результат сортировки выбором:

A = [-56, 0, 2, 4, 7, 55, 56, 56]


Результат гномьей сортировки:

A = [-56, 0, 2, 4, 7, 55, 56, 56]
```

Рисунок 4.1 — Пример работы программы

4.2 Тестовые данные

В таблице 4.1 приведены тестовые данные, на которых было протестировано разработанное ПО. Все тесты были успешно пройдены.

Таблица 4.1 — Таблица тестовых данных

Входные данные	Ожидаемый результат	Реальный результат
[7, 8, 9, 10, 11]	[7, 8, 9, 10, 11]	[7, 8, 9, 10, 11]
[11, 10, 9, 8, 7]	[11, 10, 9, 8, 7]	[11, 10, 9, 8, 7]
[1, 5, -20, 4]	[-20, 1, 4, 5]	[-20, 1, 4, 5]
[1, 5, 0, -20, 4]	[-20, 0, 1, 4, 5]	[-20, 0, 1, 4, 5]
[1]	[1]	[1]
пустой массив	пустой массив	пустой массив

4.3 Технические характеристики

Технические характеритиски машины, на которой выполнялось тестирование:

— Операционная система: Ubuntu[2] Linux[3] 20.04 64-bit.
— Оперативная память: 16 Gb.
— Процессор: AMD(R) Ryzen(TM)[4] 5 4500U CPU @ 2.3 CHz

4.4 Механизм замера времени выполнения алгоритмов

Время выполнения агоритмов (процессорное) замерялось с помощью ассемблерной вставки, которая ведет посчет тиков процессора[5]:

Листинг 4.1 — "Ассемблерная вставка для замера тиков процессора"

```
1 uint64_t tick(void)
2 {
3     uint32_t high, low;
4     __asm__ __volatile__(
5         "rdtsc\n"
```

```
6           "movl %%edx, %0\n"
7           "movl %%eax, %1\n"
8           : "=r"(high), "=r"(low)::"%rax", "%rbx", "%rcx", "%rdx");
9
10      uint64_t ticks = ((uint64_t)high << 32) | low;
11
12      return ticks;
13  }
```

4.5 Время выполнения алгоритмов

В таблице 4.2 приведены замеры процессорного времени для каждого из алгоритмов сортировки для *отсортированных в обратном порядке* массивов.

Таблица 4.2 — Таблица замеров процессорного времени (в тиках) для отсортированных в обратном порядке массивов.

№	Сортировка пузырьком	Сортировка выбором	Гномья сортировка
10	378	504	83
50	11 772	11 980	477
100	45 128	45 518	970
250	275 977	260 495	2 278
500	1 096 777	1 018 433	4 426
750	2 456 522	2 272 930	6 609
1000	4 190 835	3 872 559	8 554

В таблице 4.3 приведены замеры процессорного времени для каждого из алгоритмов сортировки для *заранее отсортированных* массивов.

Таблица 4.3 — Таблица замеров процессорного времени (в тиках) для отсортированных массивов.

№	Сортировка пузырьком	Сортировка выбором	Гномья сортировка
10	296	393	64
50	9 139	9 278	369
100	35 306	34 622	754
250	210 781	198 270	1 735
500	763 750	709 144	3 105
750	1 510 706	1 396 510	4 082
1000	2 657 210	2 453 563	5 327

В таблице 4.4 приведены замеры процессорного времени для каждого из алгоритмов сортировки для *случайных* массивов.

Таблица 4.4 — Таблица замеров процессорного времени (в тиках) для случайных массивов.

№	Сортировка пузырьком	Сортировка выбором	Гномья сортировка
10	1006	918	999
50	22 119	15 517	19 464
100	64 514	48 933	61 080
250	423 969	289 282	386 181
500	1 696 245	1 099 455	1 574 266
750	3 769 024	2 412 172	3 501 929
1000	6 656 798	4 229 664	6 270 019

На рисунке 4.2 показана зависимость процессорного времени от длины массива на остортированных в обратном порядке данных.

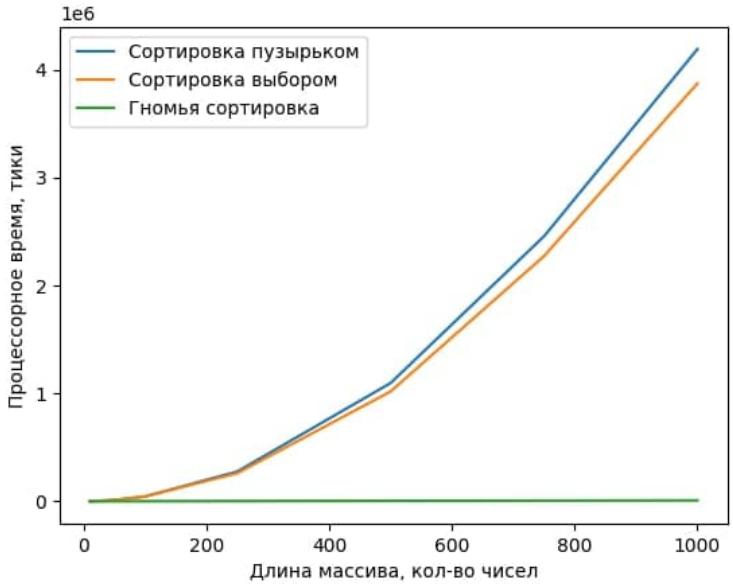

Рисунок 4.2 — Зависимость процессорного времени от длины массива на остортированных в обратном порядке данных

На рисунке 4.3 показана зависимость процессорного времени от длины массива на остортированных данных.

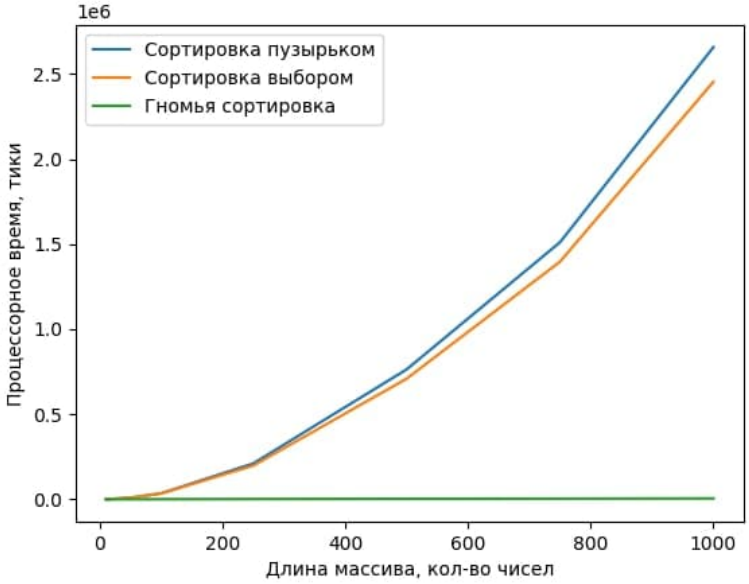

Рисунок 4.3 — Зависимость процессорного времени от длины массива на остортированных данных

На рисунке 4.4 показана зависимость процессорного времени от длины массива на основе случайных данных.

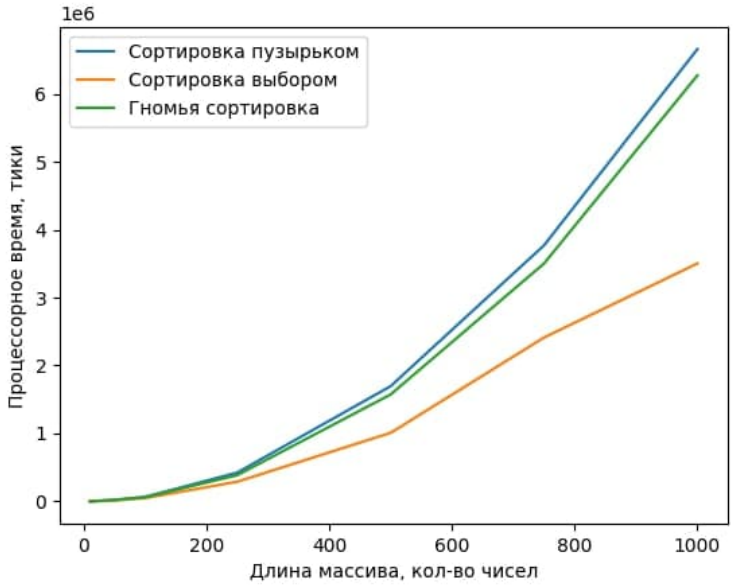

Рисунок 4.4 — Зависимость процессорного времени от длины массива на основе случайных даныхх

4.6 Вывод

Алгоритмы сортировки пузырьком и выбором на отсортированных и обратно отсортированных массивах работают медленнее, чем гномья: уже на 10 элементах эта сортировка работает в 5 раз быстрее остальных, а на тысяче элементов разница составляет порядка 500 раз, что является интересным фактом.

Однако на практике сортируют случайные массивы, и по этому пункту сортировка выбором работает быстрее: при размере массива в тысячу элементов сортировка выбором работает в полтора раза быстрее по сравнению с сортировкой пузырьком и гномьей.

Заключение

В ходе выполнения лабораторной работы были выполнены поставленные задачи, а именно:

а) Были рассмотрены и изучены следующие 3 сортировки: пузырьком, слиянием и гномья.

б) Были реализованы выбранные сортировки.

в) Была рассчитана трудоемкость сортировок.

г) Были сравнены временные характеристики экспериментально.

д) На основании проделанной работы были сделаны выводы.

Экспериментально были установлены различия в производительности сортировок пузырьком, выбором и гномьей:

Алгоритмы сортировки пузырьком и выбором на отсортированных и обратно отсортированных массивах работают медленнее, чем гномья: уже на 10 элементах эта сортировка работает в 5 раз быстрее остальных, а на тысяче элементов разница составляет порядка 500 раз, что является интересным фактом.

Однако на практике сортируют случайные массивы, и по этому пункту сортировка выбором работает быстрее: при размере массива в тысячу элементов сортировка выбором работает в полтора раза быстрее по сравнению с сортировкой пузырьком и гномьей.

Таким образом, выбирая сортировку, нужно учитывать природу данных, степень упорядоченности и размер массива данных.

Список использованных источников

1. Описание алгоритмов сортировки и сравнение их производительности. — [Электронный ресурс]. Режим доступа: https://habr.com/ru/post/335920/., 26.09.2021.

2. Ununtu. — [Электронный ресурс]. Режим доступа: https://ru.wikipedia.org/wiki/Ubuntu., 16.09.2021.

3. Linux. — [Электронный ресурс]. Режим доступа: https://ru.wikipedia.org/wiki/Linux., 16.09.2021.

4. Процессор AMD Ryzen(TM) 5. — [Электронный ресурс]. Режим доступа: https://shop.lenovo.ru/product/81YM007FRU/., 21.09.2021.

5. C/C++: как измерять процессорное время. — [Электронный ресурс]. Режим доступа: https://habr.com/ru/post/282301/., 16.09.2021.