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, искусственный интеллект и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Лабораторная работа № 3
по дисциплине "Анализ алгоритмов"

Тема Трудоемкость алгоритмов сортировки

Студент Царев А.А.

Группа ИУ7-53Б

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

Москва, 2022

Оглавление

Введение

Сортировкой называют процесс упорядочивания данных в коллекции по определенному признаку. Такое преобразование обеспечивает оптимизацию работы с агрегатными типами данных. В частности, алгоритмы сортировки позволяют облегчить поиск элемента в коллекции.

Существует великое множество алгоритмов сортировки, отличающихся друг от друга не только последовательностью операций, но и количеством потребляемых ресурсов. Поэтому при разработке программного обеспечения необходимо обращать внимание на такие параметры алгоритмов сортировки, как количество потребляемой памяти и время, затраченное на процесс упорядочивания данных.

Целью данной лабораторной работы является получение навыка оценки трудоемкости и временной эффективности на материале алгоритмов сортировки.

Задачи лабораторной работы:

- изучить и реализовать три алгоритма сортировки - блинную, перемешиванием и выбором;
- выполнить оценку трудоемкости рассматриваемых алгоритмов сортировки;
- провести замеры процессорного времени работы реализаций алгоритмов для данных, соответствующих лучшему, худшему и произвольному случаям по трудоемкости;
- провести сравнительный анализ временной эффективности алгоритмов сортировки.

1. Аналитическая часть

1.1 Блинная сортировка

Единственной допустимой операцией в алгоритме является переворот элементов последовательности до определенного индекса. Процесс можно визуально представить как стопку блинов, которую изменяют путём взятия нескольких изделий сверху и их переворачивания [1].

1.2 Сортировка перемешиванием

Сортировка перемешиванием является модифицированной версией «пузырька». В отличие от сортировки пузырьком, во время выполнения алгоритма программа проходит по массиву слева-направо и справо-налево, по необходимости меняя два соседних элемента местами [2].

1.3 Сортировка выбором

Основные шаги алгоритма:

- найти индекс минимального элемента в массиве;
- произвести обмен между полученным и первым неотсортированным элементами;
- повторить вышеописанные действия, не затрагивая отсортированные элементы массива.

Заметим, что данный алгоритм требует наличия всех исходных элементов до начала сортировки, а элементы вывода порождает последовательно, один за другим [3].

Вывод

В данном разделе было рассмотрено три алгоритма сортировки: блинная, перемешиванием и выбором.

2. Конструкторская часть

2.1 Схемы алгоритмов сортировки

На рисунках 2.1 - 2.5 представлены схемы исследуемых алгоритмов сортировки.

2.2 Оценка трудоемкости алгоритмов сортировки

Для оценки трудоемкости алгоритмов сортировки используется следующая модель:

- Трудоемкость операций

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

единична;

- Трудоемкость операций

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

равна 2;

- Трудоемкость условного оператора вычисляется по формуле

$$f_{if} = f_{cond} + \begin{cases} min(f_1, f_2), & \text{лучший случай,} \\ max(f_1, f_2), & \text{худший случай,} \end{cases}\tag{2.3}$$

где f_{cond} - трудоемкость проверки условия, f_1 - трудоемкость тела условного оператора, если выполняется условие; f_2 - трудоемкость тела условного оператора, если условие не выполняется.

- Трудоемкость цикла вычисляется по формуле

$$f_c = f_{init} + f_{comp} + N(f_{body} + f_{inc} + f_{comp}),\tag{2.4}$$

где f_{init} - трудоемкость инициализации счетчика цикла, f_{comp} - трудоемкость проверки условия цикла, f_{body} - трудоемкость тела цикла, f_{inc} - трудоемкость инкремента счетчика цикла, N - количество итераций.

2.2.1 Алгоритм блинной сортировки

Оценим трудоемкость алгоритма блинной сортировки.
Трудоемкость условного оператора внутри цикла 2 определяется следующим образом:

$$f_{if1} = \begin{cases} 3, & \text{лучший случай,} \\ 4, & \text{худший случай.} \end{cases}.\tag{2.5}$$

Используя формулы (2.4) и (2.5), определим суммарную трудоемкость цикла 2 для лучшего и худшего случаев:

$$f_{c2_{best}} = (N-1)(1+1) + \frac{N(N-1)}{2}(f_{if1} + 2 + 1) = 3N^2 - N - 2, \tag{2.6}$$

$$f_{c2_{worst}} = (N-1)(1+1) + \frac{N(N-1)}{2}(f_{if1} + 2 + 1) = \frac{7}{2}N^2 - \frac{1}{2}N - 2. \tag{2.7}$$

Трудоемкость цикла 4 вычисляется по формуле (2.4):

$$f_{c4} = 1 + 1 + 1 + \lfloor\frac{N}{2}\rfloor(7 + 1 + 2 + 2) = 6N + 3. \tag{2.8}$$

Вычислим суммарную трудоемкость цикла 3 для лучшего случая. Для него характерна ситуация, при которой значение переменной $mi = 0$. Очевидно, что в лучшем случае суммарная трудоемкость цикла 3 будет равна

$$f_{c3_{best}} = (1+1)(N-1) = 2N - 2. \tag{2.9}$$

Определим суммарную трудоемкость цикла 3 для худшего случая. Для него характерна ситуация, при которой переменная mi равна $right$ после работы цикла 2. Тогда суммарное количество итераций цикла 3 равно

$$k_{c4_{worst}} = \lfloor\frac{N}{2}\rfloor, \lfloor\frac{N-1}{2}\rfloor, ...2, 1. \tag{2.10}$$

Если значение N нечетно, то формула (2.10) примет вид

$$k_{c4_{worst}} = \frac{N-1}{2} + \frac{N-1}{2} + \frac{N-3}{2} + \frac{N-1}{2} + ... + 2 + 2 + 1 + 1 = \frac{N^2-1}{4}. \tag{2.11}$$

Если же значение N четно, то формула (2.10) примет вид

$$k_{c4_{worst}} = \frac{(N-1)^2-1}{4} + \frac{N}{2} = \frac{N^2}{4}. \tag{2.12}$$

Очевидно, что при нечетном значении N формула (2.12) описывает наиболее трудоемкую ситуацию. Поэтому, используя формулы (2.4) и (2.12), рассчитаем суммарную трудоемкость цикла 3 для худшего случая:

$$f_{c3_{worst}} = (N-1)(1+1) + k_{c4_{worst}}(2 + 3 + 2 + 2 + 2 + 1) = 3N^2 + 2N - 2. \tag{2.13}$$

В итоге, используя формулы (2.4), (2.6), (2.7), (**??**) и (2.8), мы можем получить трудоемкость алгоритма блинной сортировки для лучшего и худшего случаев:

$$f_{best} = f_{c1_{best}} = 2 + 1 + (N-1)(1 + f_{c4} + 2) + f_{c2_{best}} + f_{c3_{best}} = 9N^2 + N - 7 = O(N^2), \tag{2.14}$$

$$f_{worst} = f_{c1_{worst}} = 2+1+(N-1)(1+f_{c4}+2)+f_{c2_{worst}}+f_{c3_{worst}} = \frac{25}{2}N^2+\frac{3}{2}N-7 = O(N^2). \tag{2.15}$$

2.2.2 Алгоритм сортировки перемешиванием

Оценим трудоемкость алгоритма сортировки перемешиванием.

Трудоемкости условных операторов во внутренних циклах for определяются формулой (2.3):

$$f_{if1} = f_{if2} = f_{if} = \begin{cases} 4, & \text{лучший случай,} \\ 13, & \text{худший случай.} \end{cases} . \tag{2.16}$$

Используя формулы (2.4) и (2.16), определим суммарную трудоемкость внутренних циклов for для лучшего и худшего случаев:

$$f_{c2_{best}} = (1+1+1+1)(N-1) + \frac{N(N-1)}{2}(2+1+f_{if}) = \frac{7}{2}N^2 + \frac{1}{2}N - 4, \tag{2.17}$$

$$f_{c2_{worst}} = (1+1+1+1)(N-1) + \frac{N(N-1)}{2}(2+1+f_{if}) = 8N^2 - 4N - 4. \tag{2.18}$$

Используя формулы (2.4), (2.17) и (2.18), определим трудоемкость цикла while для лучшего и худшего случаев:

$$f_{c1_{best}} = 1 + (N-1)(2+2) + f_{c2_{best}} = \frac{7}{2}N^2 + \frac{9}{2}N - 7, \tag{2.19}$$

$$f_{c1_{worst}} = 1 + (N-1)(2+2) + f_{c2_{worst}} = 8N^2 - 7. \tag{2.20}$$

В итоге, применяя (2.19) и (2.20), мы можем вычислить трудоемкость алгоритма перемешиванием для лучшего и худшего случаев:

$$f_{best} = 1 + 2 + f_{c1_{best}} = \frac{7}{2}N^2 + \frac{9}{2}N - 4 = O(N^2), \tag{2.21}$$

$$f_{worst} = 1 + 2 + f_{c1_{worst}} = 8N^2 - 4 = O(N^2). \tag{2.22}$$

2.2.3 Алгоритм сортировки выбором

Оценим трудоемкость алгоритма сортировки выбором.

Трудоемкость условного оператора во внешнем цикле определяем по формуле (2.4):

$$f_{if1} = \begin{cases} 3, & \text{лучший случай,} \\ 10, & \text{худший случай.} \end{cases} \tag{2.23}$$

По формуле (2.3) получаем трудоемкость условного оператора во внутреннем цикле:

$$f_{if2} = \begin{cases} 3, & \text{лучший случай,} \\ 4, & \text{худший случай.} \end{cases} \tag{2.24}$$

Используя формулы (2.4) и (2.24), определяем суммарную трудоемкость внутренних циклов для лучшего и худшего случаев:

$$f_{c2_{best}} = (2+1)(N-1) + \frac{N(N-1)}{2}(f_{if2}+2+1) = 3N^2 - 3, \tag{2.25}$$

$$f_{c2_{worst}} = (2+1)(N-1) + \frac{N(N-1)}{2}(f_{if2}+2+1) = \frac{7}{2}N^2 - \frac{1}{2}N - 3. \tag{2.26}$$

По формулам (2.4), (2.23), (2.25) и (2.26) определяем трудоемкость внешнего цикла для лучшего и худшего случаев:

$$f_{c1_{best}} = 3 + (N-1)(2+2+1+f_{if1}) + f_{c2_{best}} = 3N^2 + 8N - 8, \tag{2.27}$$

$$f_{c1_{worst}} = 3 + (N-1)(2+2+1+f_{if1}) + f_{c2_{worst}} = \frac{7}{2}N^2 + \frac{29}{2}N - 15. \tag{2.28}$$

В итоге, из (2.27) и (2.28) получаем трудоемкость алгоритма сортировки выбором. Для лучшего случая:

$$f_{best} = 3N^2 + 8N - 8 = O(N^2). \tag{2.29}$$

Для худшего случая:

$$f_{worst} = \frac{7}{2}N^2 + \frac{29}{2}N - 15 = O(N^2). \tag{2.30}$$

Вывод

В данном разделе были разработаны и оцененены трудоемкости следующих алгоритмов сортировки: блинной, перемешиванием и выбором.

Результаты оценки:

- блинная сортировка: $f_{best} = O(N^2)$, $f_{worst} = O(N^2)$;
- сортировка перемешиванием: $f_{best} = O(N^2)$, $f_{worst} = O(N^2)$;
- сортировка выбором: $f_{best} = O(N^2)$, $f_{worst} = O(N^2)$.

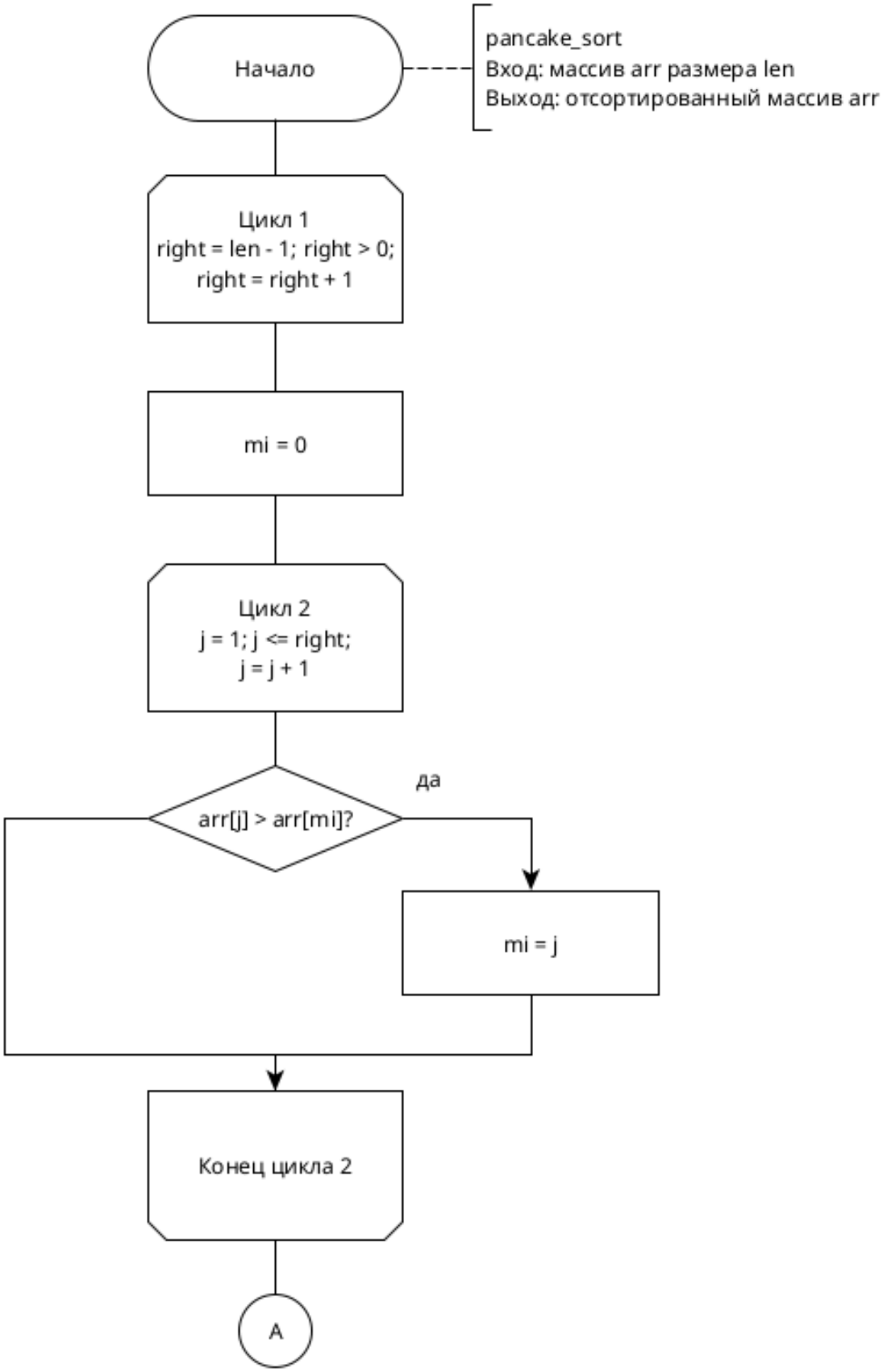

Рис. 2.1. Схема алгоритма блинной сортировки

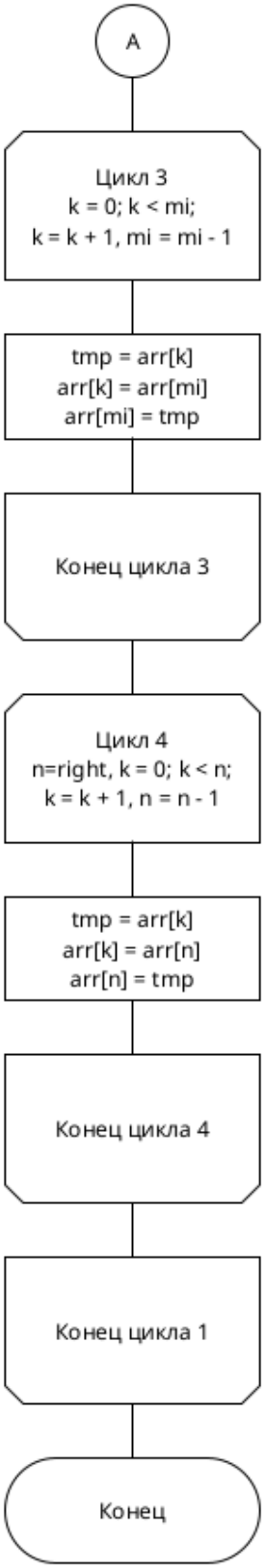

Рис. 2.2. Схема алгоритма блинной сортировки

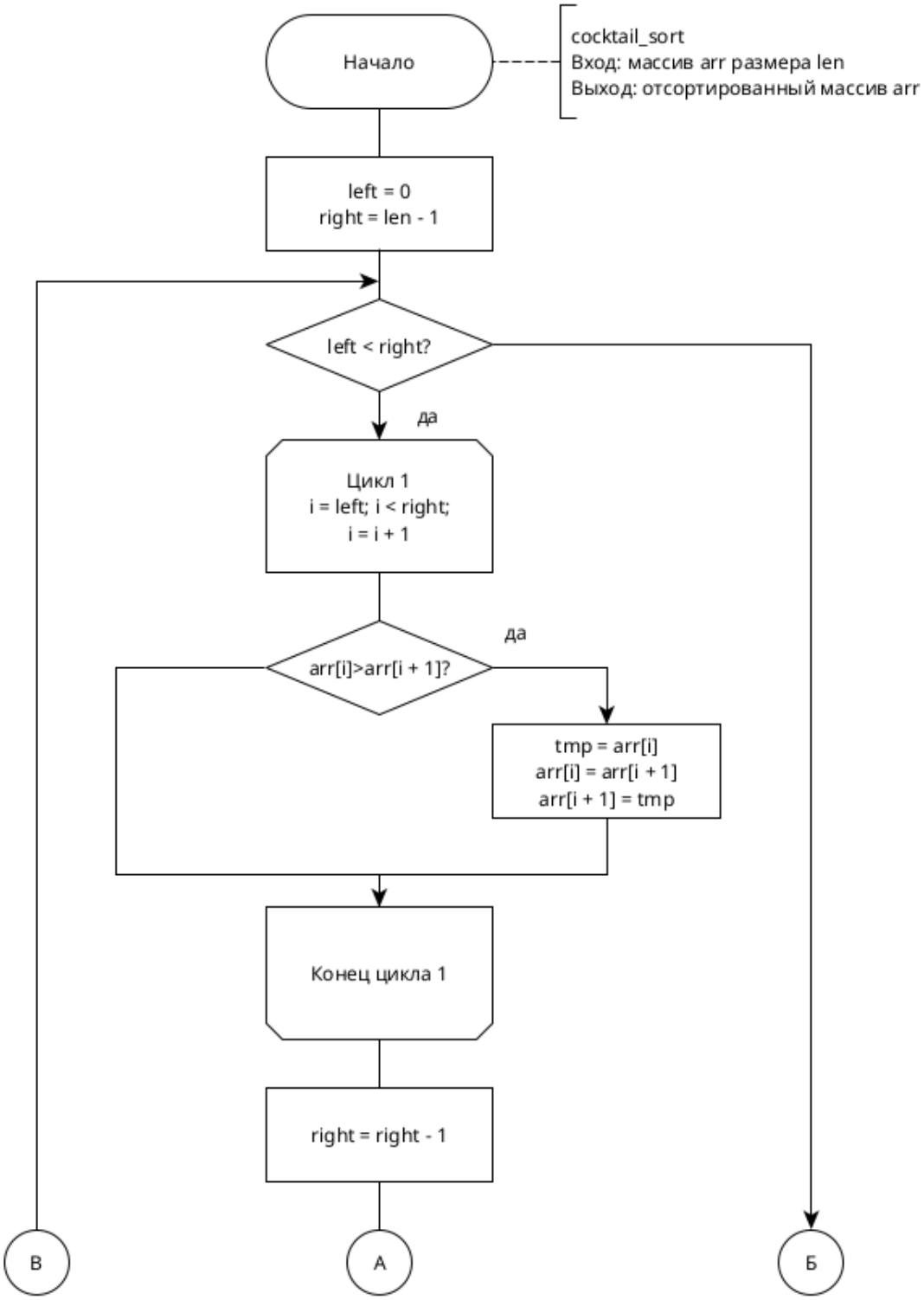

Рис. 2.3. Схема алгоритма сортировки перемешиванием

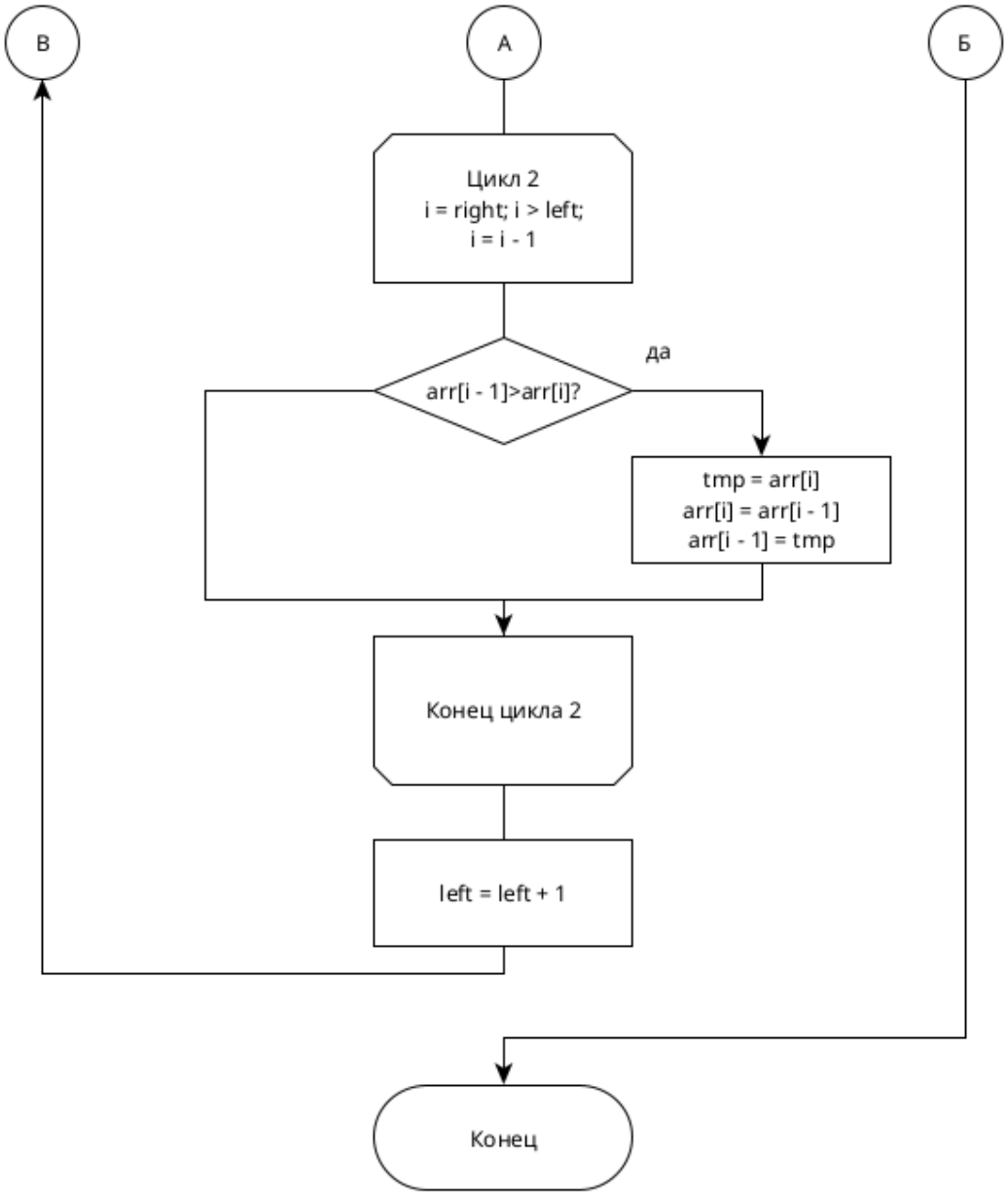

Рис. 2.4. Схема алгоритма сортировки перемешиванием

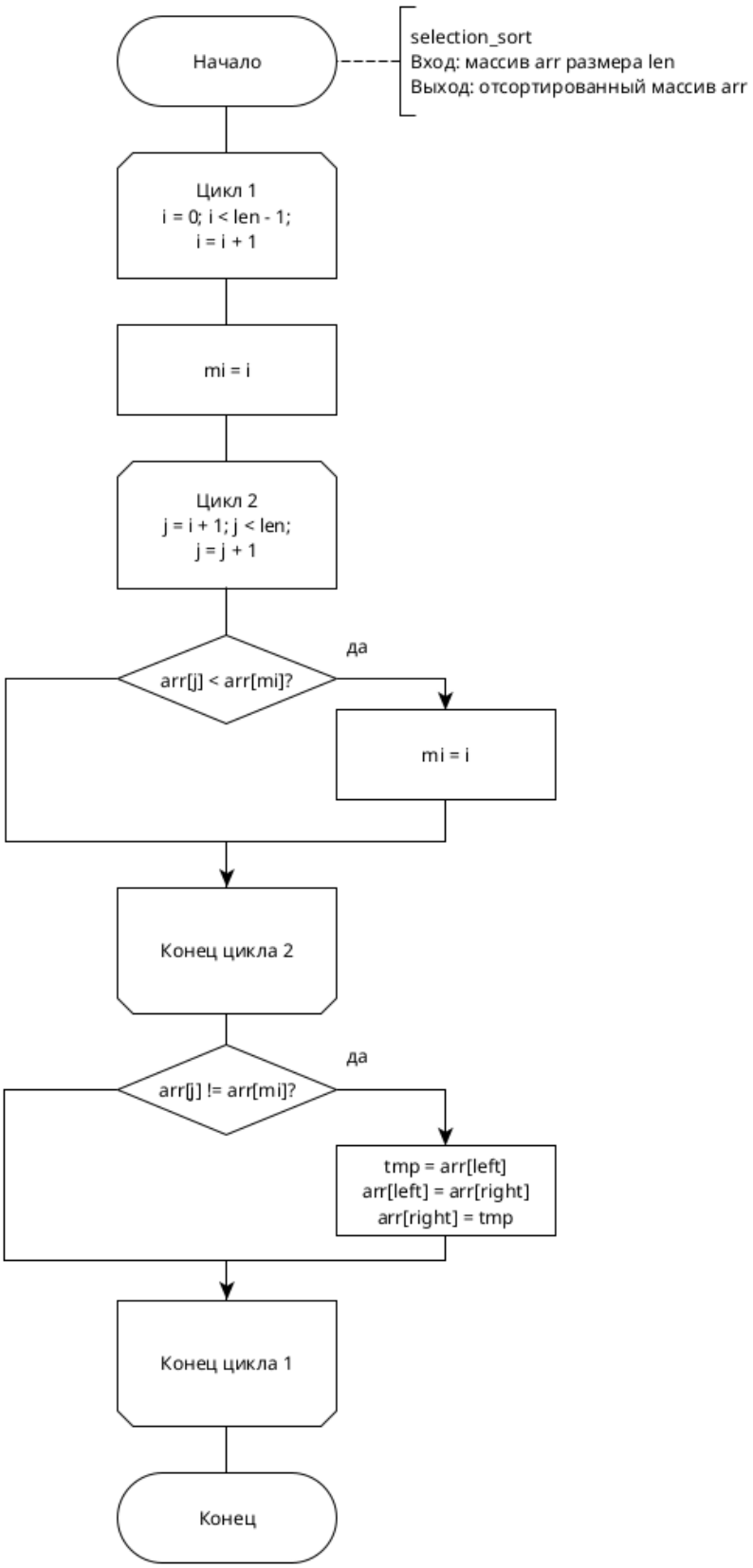

Рис. 2.5. Схема алгоритма сортировки выбором

3. Технологическая часть

3.1 Требования к программе

К программе предъявляется ряд требований:

- программа должна предоставить пользователю возможность выбора алгоритма сортировки;
- после выбора алгоритма программа принимает на вход длину и целочисленные элементы массива;
- на выходе - массивы до и после сортировки и среднее время выполнения алгоритма.

3.2 Средства реализации

Для написания программы был выбран язык C. Данный выбор обуславливается высокой скоростью выполнения кода и наличием богатой стандартной библиотеки.

3.3 Реализация алгоритмов

На листингах 3.1 - 3.3 представлены реализации алгоритмов сортировки.

Листинг 3.1. Функция блинной сортировки

```
1  void pancake_sort(int *arr, const size_t len)
2  {
3    for (size_t right = len - 1; right > 0; --right)
4    {
5      size_t max_elem_ind = 0;
6      for (size_t j = 1; j <= right; ++j)
7        if (arr[j] > arr[max_elem_ind])
8          max_elem_ind = j;
9      if (max_elem_ind > 0)
10     {
11       for (size_t k = 0; k < max_elem_ind; ++k, --max_elem_ind)
12       {
13         int tmp = arr[k];
14         arr[k] = arr[max_elem_ind];
15         arr[max_elem_ind] = tmp;
16       }
17     }
18     for (size_t k = 0, n = right; k < n; ++k, --n)
19     {
```

```
      int tmp = arr[k];
      arr[k] = arr[n];
      arr[n] = tmp;
    }
  }
}
```

Листинг 3.2. Функция сортировки перемешиванием

```
void cocktail_sort(int *arr, const size_t len)
{
  size_t left = 0, right = len - 1;
  while (left < right)
  {
    for (size_t i = left; i < right; ++i)
    {
      if (arr[i] > arr[i + 1])
      {
        int tmp = arr[i];
        arr[i] = arr[i + 1];
        arr[i + 1] = tmp;
      }
    }
    --right;
    for (size_t i = right; i > left; --i)
    {
      if (arr[i - 1] > arr[i])
      {
        int tmp = arr[i];
        arr[i] = arr[i - 1];
        arr[i - 1] = tmp;
      }
    }
    ++left;
  }
}
```

Листинг 3.3. Функция сортировки выбором

```
void selection_sort(int *arr, const size_t len)
{
  for (size_t i = 0; i < len - 1; ++i)
  {
    size_t min_elem_ind = i;
    for (size_t j = i + 1; j < len; ++j)
      if (arr[j] < arr[min_elem_ind])
        min_elem_ind = j;
    if (arr[i] != arr[min_elem_ind])
    {
      int tmp = arr[i];
      arr[i] = arr[min_elem_ind];
      arr[min_elem_ind] = tmp;
    }
  }
```

```
16 }
```

3.4 Тестовые данные

В таблице 3.1 приведены тестовые данные для функций сортировки. Все тесты пройдены успешно.

Таблица 3.1. Тестовые данные для функций сортировки

Входной массив	**Результат**	**Ожидаемый результат**
17, 23, 1, 0, -5, 4	-5, 0, 1, 4, 17, 23	-5, 0, 1, 4, 17, 23
-5, 0, 1, 4, 17, 23	-5, 0, 1, 4, 17, 23	-5, 0, 1, 4, 17, 23
23, 17, 4, 1, 0, -5	-5, 0, 1, 4, 17, 23	-5, 0, 1, 4, 17, 23
3, 3, 3, 3, 3, 3	3, 3, 3, 3, 3, 3	3, 3, 3, 3, 3, 3
-3, -3, -3, -3, -3, -3	-3, -3, -3, -3, -3, -3	-3, -3, -3, -3, -3, -3
1	1	1

Вывод

В данном разделе были разработаны исходные коды следующих алгоритмов сортировки: блинной, перемешиванием и выбором.

4. Исследовательская часть

4.1 Технические характеристики ЭВМ

Ниже приведены технические характеристики электронной вычислительной машины, на которой было произведено исследование работы программы:

- ОС Manjaro Linux x86_64;
- ЦП Intel i7-10510U (8) @ 4.900 ГГц;
- ОЗУ 8 ГБ.

4.2 Время выполнения алгоритмов

Для измерения процессорного времени выполнения программы был использован заголовочный файл time.h стандартной библиотеки языка C [4]. Для построения графиков зависимости времени работы реализаций алгоритмов от длины массива был использован пакет PGFPlots [5].

В таблицах 4.1 - 4.3 приведены результаты исследований алгоритмов сортировки. На рисунках 4.1 - 4.3 приведены графики зависимости времени работы реализаций алгоритмов сортировки от длины массива.

Вывод

В данном разделе было измерено время работы реализаций следующих алгоритмов сортировки: блинной, перемешиванием и выбором.

По результатам измерений видно, что сортировка выбором работает быстрее других рассматриваемых алгоритмов. Кроме того, графики зависимости времени работы реализаций алгоритмов сортировки от длины массива показывают, что с увеличением числа элементов время работы всех трех реализаций возрастает по параболе. Следовательно, теоретические и практические результаты измерений совпадают.

Таблица 4.1. Результаты измерений для неотсортированного массива

Количество элементов	Блинная, сек	Перемешиванием, сек	Выбором, сек
100	0.036689	0.025143	0.016479
200	0.111341	0.071797	0.068694
300	0.219065	0.130772	0.118504
500	0.542456	0.305166	0.285259
700	1.020449	0.621595	0.582602
1000	1.987858	1.174300	1.147254

Таблица 4.2. Результаты измерений для отсортированного массива

Количество элементов	Блинная, сек	Перемешиванием, сек	Выбором, сек
100	0.026217	0.014462	0.014656
200	0.096718	0.050436	0.050770
300	0.236083	0.113531	0.109582
500	0.536780	0.296318	0.299467
700	1.007199	0.588859	0.580525
1000	1.930346	1.140053	1.145935

Таблица 4.3. Результаты измерений для обратно отсортированного массива

Количество элементов	Блинная, сек	Перемешиванием, сек	Выбором, сек
100	0.045580	0.032039	0.013493
200	0.098482	0.058368	0.053884
300	0.194850	0.134700	0.124417
500	0.546437	0.318207	0.316290
700	1.001914	0.602354	0.581368
1000	1.968974	1.196855	1.154253

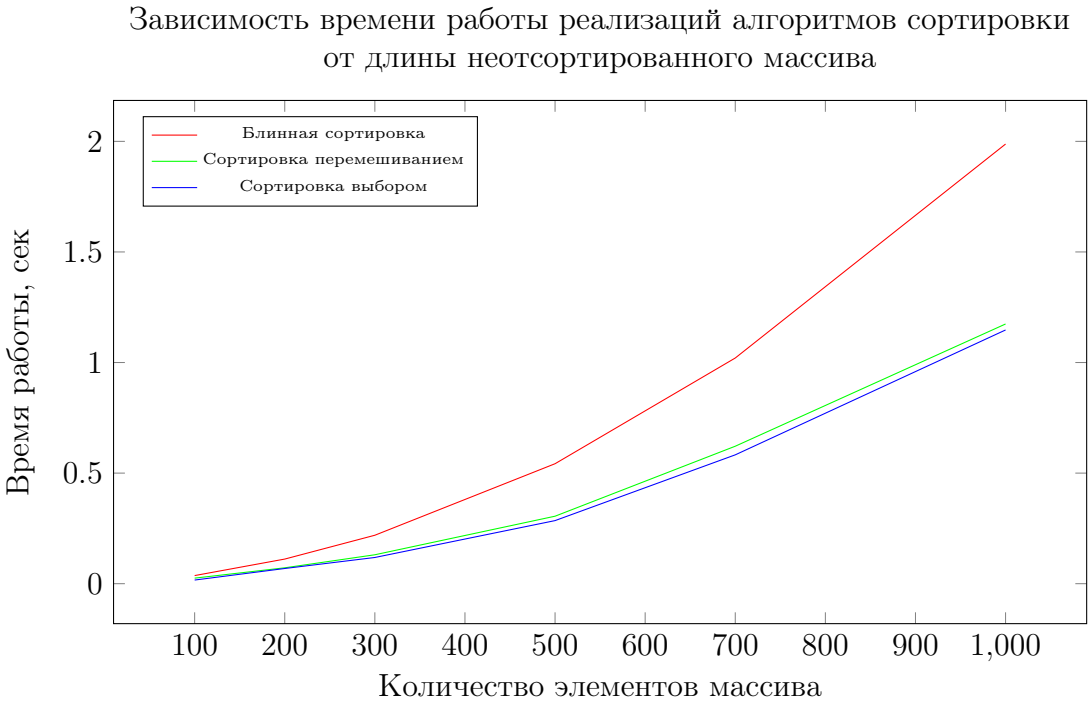

Рис. 4.1. График для неотсортированного массива

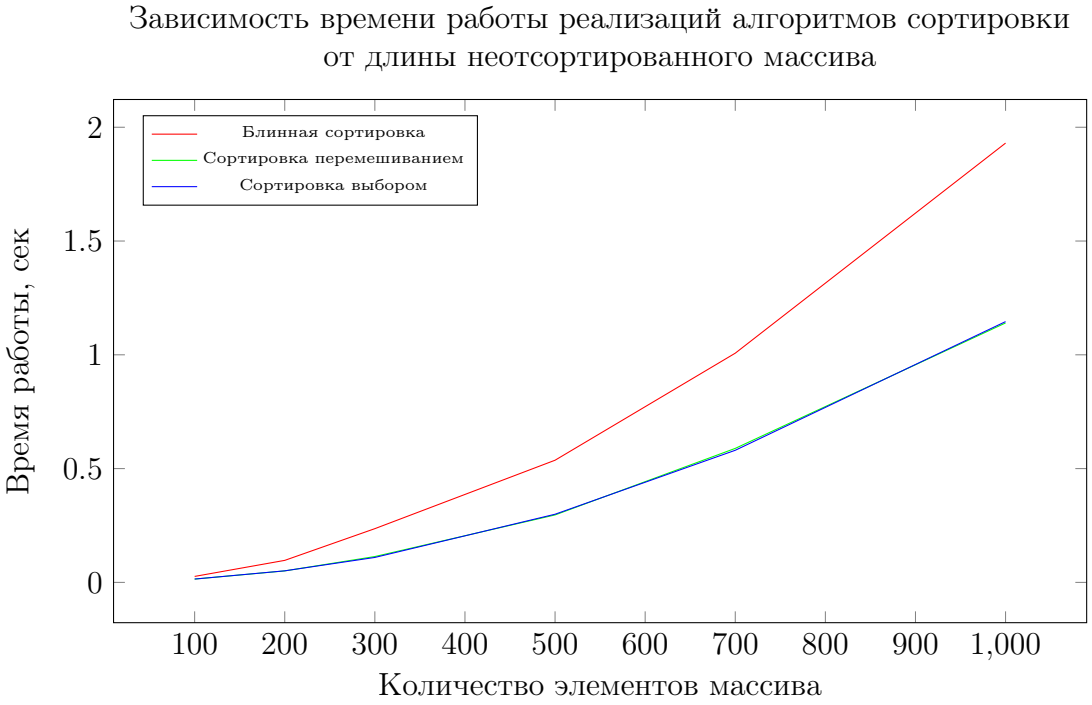

Рис. 4.2. График для отсортированного массива

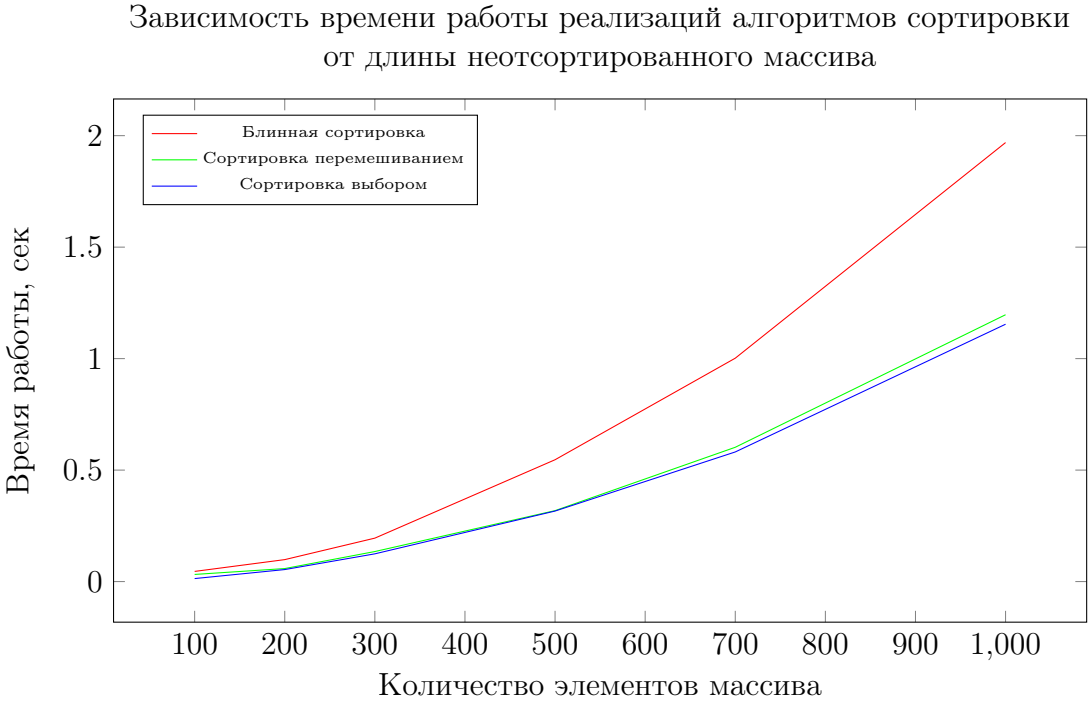

Рис. 4.3. График для обратно отсортированного массива

Заключение

В данной лабораторной работе были получены навыки оценки трудоемкости и временной эффективности на материале алгоритмов сортировки. Также, были выполнены следующие задачи:

- изучены и реализованы три алгоритма сортировки - блинная, перемешиванием и выбором;
- выполнена оценка трудоемкости рассматриваемых алгоритмов сортировки;
- проведены замеры процессорного времени работы реализаций алгоритмов для данных, соответствующих лучшему, худшему и произвольному случаям по трудоемкости;
- проведен сравнительный анализ временной эффективности алгоритмов сортировки.

Литература

1. Weisstein, Eric W. Pancake Sorting [Электронный ресурс] - Режим доступа: https://faculty.math.illinois.edu/ west/openp/pancake.html (дата обращения 29.10.22)

2. Knuth, Donald E. (1973). "Sorting by Exchanging". Art of Computer Programming. Vol. 3. Sorting and Searching (1st ed.). Addison-Wesley. pp. 110–111. ISBN 0-201-03803-X.

3. Кнут Д.Э. Искусство программирования, том 3. Сортировка и поиск, 2-е изд. : Пер. с англ. - М.: ООО "И.Д. Вильямс 2007. - 832 с. : ил. - Парал. тит. англ.

4. man time.h (1): time types [Электронный ресурс] - Режим доступа: https://manpages.org/timeh (дата обращения 29.10.22)

5. PGFPlots - A LaTeX Package to create normal/logarithmic plots in two and three dimensions [Электронный ресурс] - Режим доступа: https://pgfplots.sourceforge.net/ (дата обращения 29.10.22)