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»

Отчет по лабораторной работе №2 по курсу "Анализ алгоритмов"

Тема Алгоритмы сортировки

Студент Чепиго Д.С.

Группа ИУ7-54Б

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

Москва — 2022 г.

Содержание

Введение **3**

1 Аналитическая часть **4**

1.1 Сортировка слиянием 4

1.2 Сортировка подсчетом 4

1.3 Битонная сортировка 5

2 Конструкторская часть **6**

2.1 Разработка алгоритма сортировки слиянием 6

2.2 Модель вычислений 8

2.3 Трудоемкость алгоритмов 9

2.3.1 Алгоритм сортировки слиянием 9

2.3.2 Алгоритм сортировки подсчетом 9

2.3.3 Алгоритм битонной сортировки 10

Литература **11**

Введение

Одной из важнейших процедур обработки структурированной информации является сортировка [1]. Сортировкой называют процесс перегруппировки заданной последовательности (кортежа) объектов в некотором определенном порядке. Определенный порядок (например, упорядочение в алфавитном порядке, по возрастанию или убыванию количественных характеристик, по классам, типам и.т.п.) в последовательности объектов необходимо для удобства работы с этим объектом. В частности, одной из целей сортировки является облегчение последующего поиска элементов в отсортированном множестве.

Любой алгоритм сортировки можно разбить на три основные части:

- сравнение элементов для определения их упорядоченности;
- перестановка элементов;
- сортирующий алгоритм, который осуществляет сравнение и перестановку элементов до тех пор, пока все элементы не будут упорядочены.

Важнейшей характеристикой любого алгоритма сортировки является скорость его работы, которая определяется функциональной зависимостью среднего времени сортировки последовательностей элементов данных, заданной длины, от этой длины. Время сортировки будет пропорционально количеству сравнений и перестановки элементов данных в процессе их сортировки.

Цель – изучить и исследовать трудоемкость алгоритмов сортировки.

Задачи лабораторной работы:

- изучить и реализовать 3 алгоритма сортировки: слиянием, подсчетом, битонная;
- выбрать инструменты для процессорного времени выполнения реализаций алгоритмов;
- провести анализ затрат работы программы по времени и по памяти;
- подготовить отчет по лабораторной работе.

1 Аналитическая часть

1.1 Сортировка слиянием

Сортировка слиянием применяется для структур данных, доступ к элементам которых можно получать только последовательно, например, списки или массивы. Алгоритм сортировки слиянием основан на парадигме «разделяй и властвуй». Приведем алгоритм сортировки для массива:

1. сортируемый массив разбивается на две части примерно одинакового размера;

2. каждая из получившихся частей сортируется отдельно тем же самым алгоритмом;

3. два упорядоченных массива половинного размера соединяются в один.

Рекурсивное разбиение задачи на меньшие происходит до тех пор, пока размер массива не достигнет единицы.

Слияние работает за $O(n)$, уровней всего $\log n$, поэтому асимптотика всего алгоритма – $O(n \log n)$. Худшее, лучшее и среднее время одинаково. Эффективно заранее создать временный массив и передать его в качестве аргумента функции. Эта сортировка рекурсивна, поэтому возможен переход на квадратичную при небольшом числе элементов.

1.2 Сортировка подсчетом

Сортировка подсчетом применяется для структур данных, доступ к элементам которых только последовательно, например, списки или массивы. Используется диапазон чисел сортируемого массива (списка) для подсчёта совпадающих элементов. Приведем алгоритм сортировки для массива:

1. в сортируемом массиве находится максимальный и минимальный элемент;

2. создается дополнительный массив, размер которого – разница между максимальным и минимальным элементом;

3. считается колличество каждого числа сортируемого массива и результат записывается в дополнительный массив;

4. с помощью дополнительного массива формируется отсортированный массив.

Если в массиве используются только натуральные числа, то минимальный элемент находить не требуется, он будет равен 0.

Асимптотика алгоритма – $O(n+k)$, где n – количество элементов во входном массиве, k – диапазон ввода. Применение сортировки подсчётом целесообразно когда сортируемые числа имеют диапазон возможных значений, который достаточно мал по сравнению с сортируемым множеством, например, миллион натуральных чисел меньших 1000.

1.3 Битонная сортировка

Алгоритм применяется для массивов, размер которых степень двойки, так как он рекурсивно делит массив пополам. Из-за этого может понадобиться добавлять фиктивные элементы в сортируемый массив, что не влияет на асимптотику. Алгоритм основан на сортировке битонных последовательностей. Такой последовательностью называется последовательность, которая сначала монотонно не убывает, а затем монотонно не возрастает. Приведем алгоритм сортировки для массива:

1. сортируемый массив преобразуется в битонную последовательность;

2. сортируемый массив разбивается на две части одинакового размера и также преобразовывается в битонную последовательность;

3. два упорядоченных массива половинного размера соединяются в один.

Асимптотика алгоритма – $O(n(\log n)^2)$, поскольку при построении битонной последовательности мы использовали сортировку, работающую за $O(n \log n)$, а всего уровней было $\log n$. Параллельная реализация имеет сложность $O((\log n)^2)$.

2 Конструкторская часть

2.1 Разработка алгоритма сортировки слиянием

На рисунке 2.1 приведена схема алгоритма сортировки слиянием.

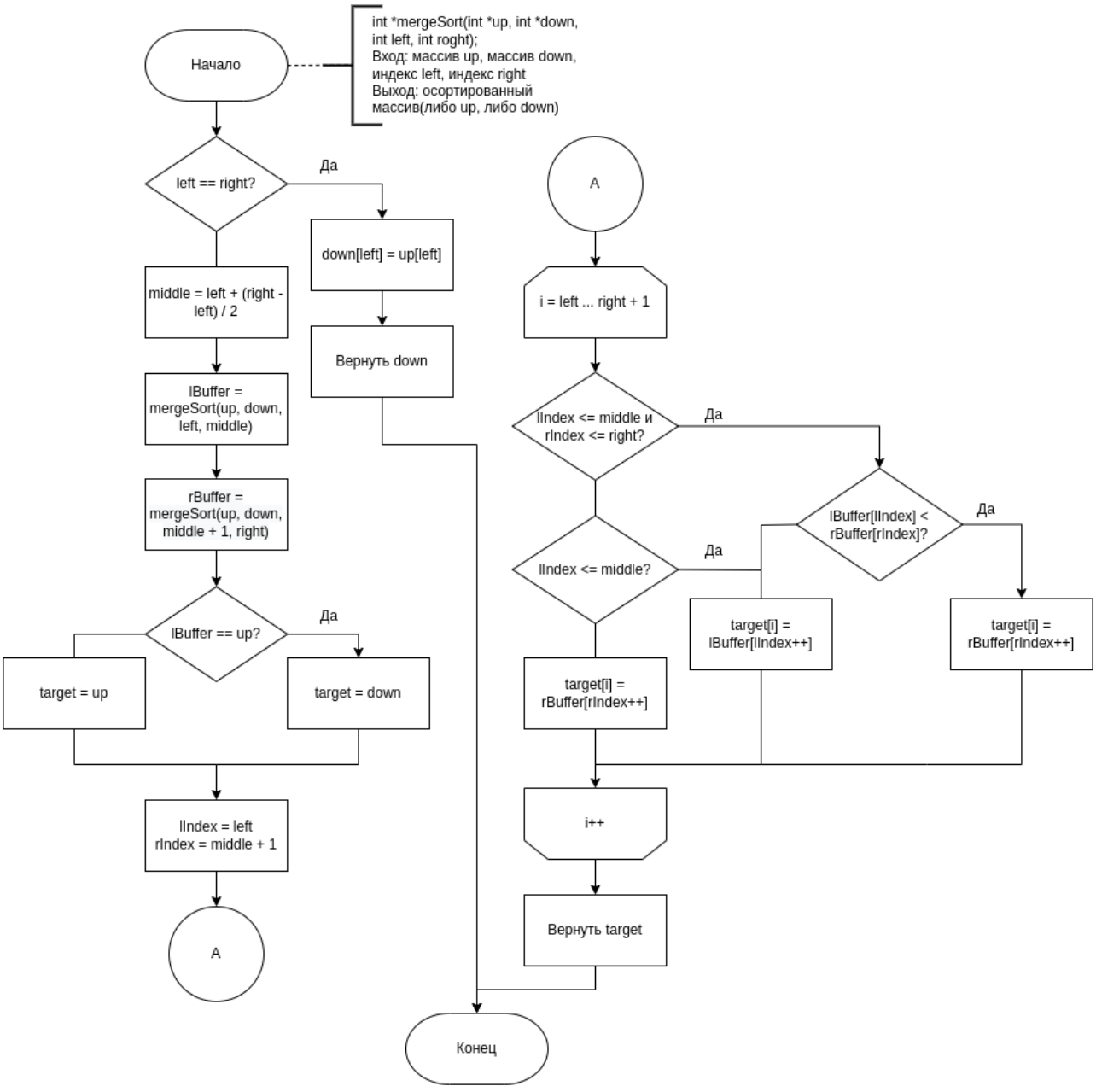

Рисунок 2.1 – Схема алгоритма сортировки слиянием.

На рисунке 2.2 приведена схема алгоритма сортировки подсчетом.

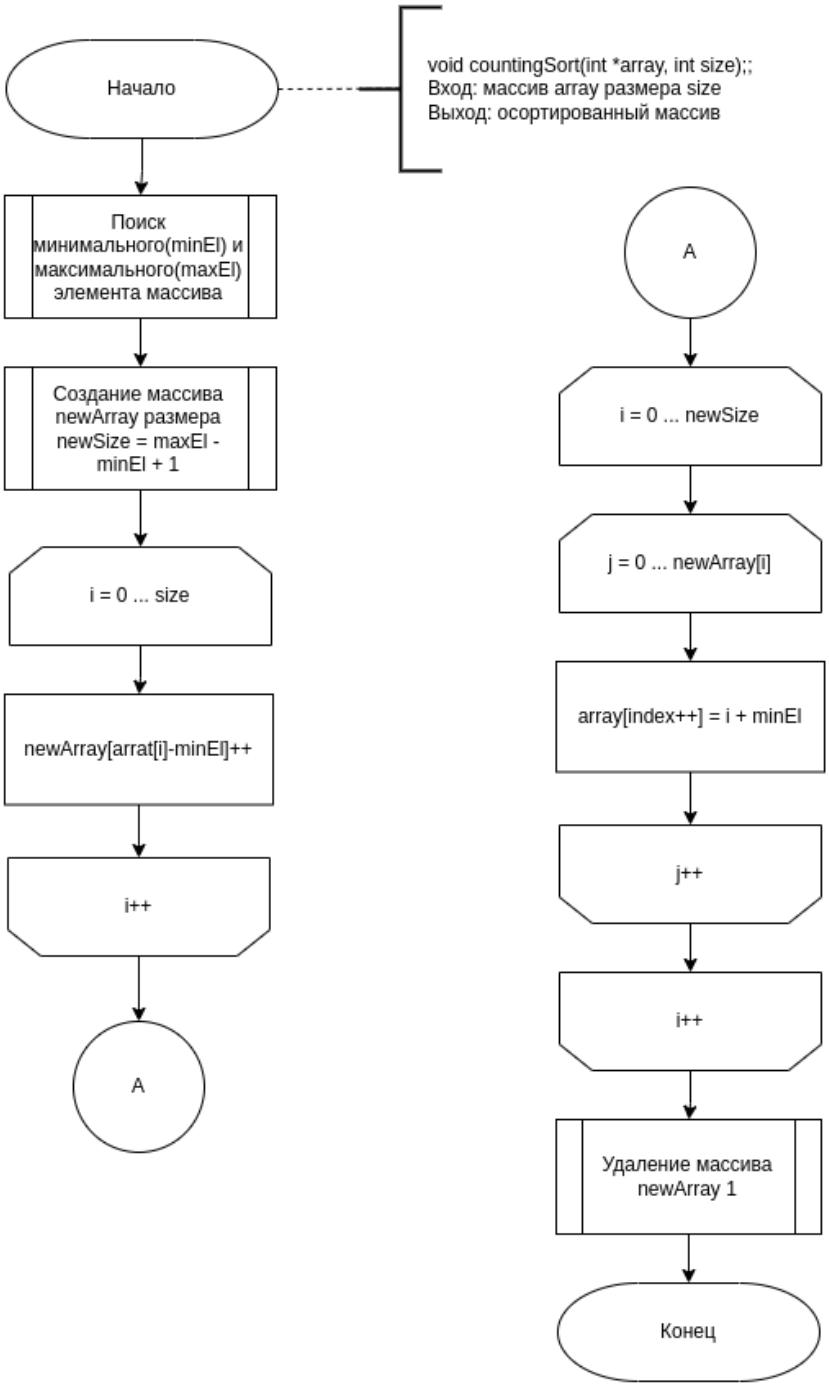

Рисунок 2.2 – Схема алгоритма сортировки подсчетом.

На рисунке 2.3 приведена схема алгоритма битонной сортировки.

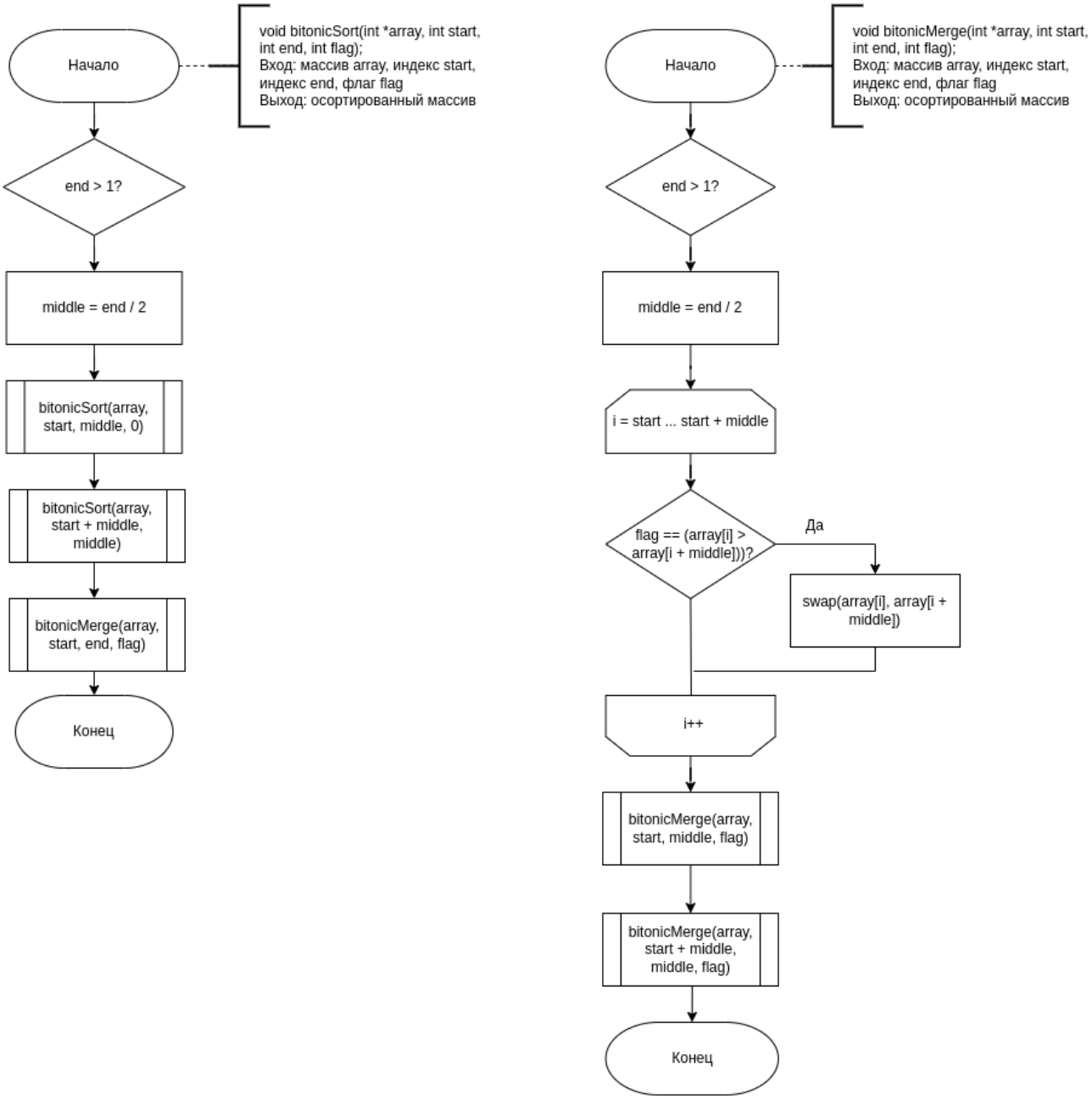

Рисунок 2.3 – Схема алгоритма битонной сортировки.

2.2 Модель вычислений

Для последующего вычисления трудоемкости необходимо ввести модель вычислений.

Операции из списка (2.1) имеют трудоемкость 1:

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

Трудоемкость оператора выбора `if условие then A else B` рассчитывается как:

$$f_{if} = f_{\text{условия}} + \begin{cases} f_A, & \text{если условие выполняется,} \\ f_B, & \text{иначе.} \end{cases} \tag{2.2}$$

Трудоемкость цикла рассчитывается как:

$$f_{for} = f_{\text{инициализации}} + f_{\text{сравнения}} + N \cdot (f_{\text{тела}} + f_{\text{инкремента}} + f_{\text{сравнения}}) \tag{2.3}$$

Трудоемкость вызова функции равна 0.

2.3 Трудоемкость алгоритмов

Далее размер массива обозначается как $size$.

2.3.1 Алгоритм сортировки слиянием

2.3.2 Алгоритм сортировки подсчетом

Трудоемкость алгоритма сортировки подсчетом состоит из:

- трудоемкость поиска минимального элемента в массиве:

$$f_{min} = 2 + size \cdot (f_{\text{тела}} + 2) \tag{2.4}$$

- трудоемкость поиска максимального элемента в массиве:

$$f_{max} = 2 + size \cdot (f_{\text{тела}} + 2) \tag{2.5}$$

- трудоёмкость каждой итерации цикла поиска минимального и максимального элемента:

$$f_{\text{тела}} = 4 + \begin{cases} 0, & \text{в лучшем случае,} \\ 2, & \text{в худшем случае.} \end{cases} \tag{2.6}$$

- трудоемкость заполнения временного массива:

$$f_{\text{доп}} = 2 + size \cdot (4 + 2) = 2 + 6 \cdot size \tag{2.7}$$

- трудоемкость внешнего цикла заполнения массива:

$$f_{\text{массив}} = 2 + newSize \cdot (f_{\text{внутренний}} + 2) \tag{2.8}$$

- трудоемкость внутреннего цикла заполнения массива:

$$f_{\text{внутренний}} = 3 + size \cdot (3 + 4) = 3 + 7 \cdot size \tag{2.9}$$

- суммарная трудоёмкость внешнего и внутреннего циклов:

$$f_{\text{суммарная}} = 2 + newSize \cdot (5 + 7 \cdot size) \tag{2.10}$$

Трудоёмкость в лучшем (массив отсортирован, все элементы совпадают) случае:

2.3.3 Алгоритм битонной сортировки

Литература

[1] Дональд Кнут. Сортировка и поиск. Вильямс, 2000. Т. 3 из Искусство программирования. с. 834.

[2] Язык программирования C++[Электронный ресурс]. Режим доступа: `https://isocpp.org/`(дата обращения: 20.09.2022)

[3] Операционная система Ubuntu 22.04 LTS[Электронный ресурс]. Режим доступа: `https://releases.ubuntu.com/jammy/`(дата обращения: 20.09.2022)