МИНИСТЕРСТВО НАУКИ ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА - Российский технологический университет»

РТУ МИРЭА

Отчет по выполнению практического задания №3

Тема:

«Определение эффективного алгоритма сортировки на основе эмпирического и асимптотического метода анализа»

Дисциплина: «Структуры и алгоритмы обработки данных»

Выполнил студент: Подоплелов А.С.

Группа: ИНБО-21-23

Москва - 2024

СОДЕРЖАНИЕ

ЦЕЛЬ РАБОТЫ

Получить навыки по анализу вычислительной сложности алгоритмов сортировки и определению наиболее эффективного алгоритма.

1. 1 УСЛОВИЕ ЗАДАНИЯ 1

Мой вариант обладает номером 7: «Пирамидальная сортировка».

Разработать алгоритм быстрой сортировки «Пирамидальная сортировка» на языке C++. Протестировать работоспособность алгоритма на примере массива n = 10, 100, 10_000, 100_000, 1_000_000. Привести словесное описание алгоритма и его блок-схему, а также программный код (с комментариями). Сделать «Эмпирическая оценку» эффективности алгоритма.

1. 2 УСЛОВИЕ ЗАДАНИЯ 2

Мой вариант обладает номером 7: «Простое слияние».

Разработать алгоритм быстрой сортировки «Простое слияние» на языке C++. Протестировать работоспособность алгоритма на примере массива n = 10, 100, 10_000, 100_000, 1_000_000. Привести словесное описание алгоритма и его блок-схему, а также программный код (с комментариями). Сделать «Асимптотический анализ» сложности алгоритма.

РЕШЕНИЕ ЗАДАНИЯ 1

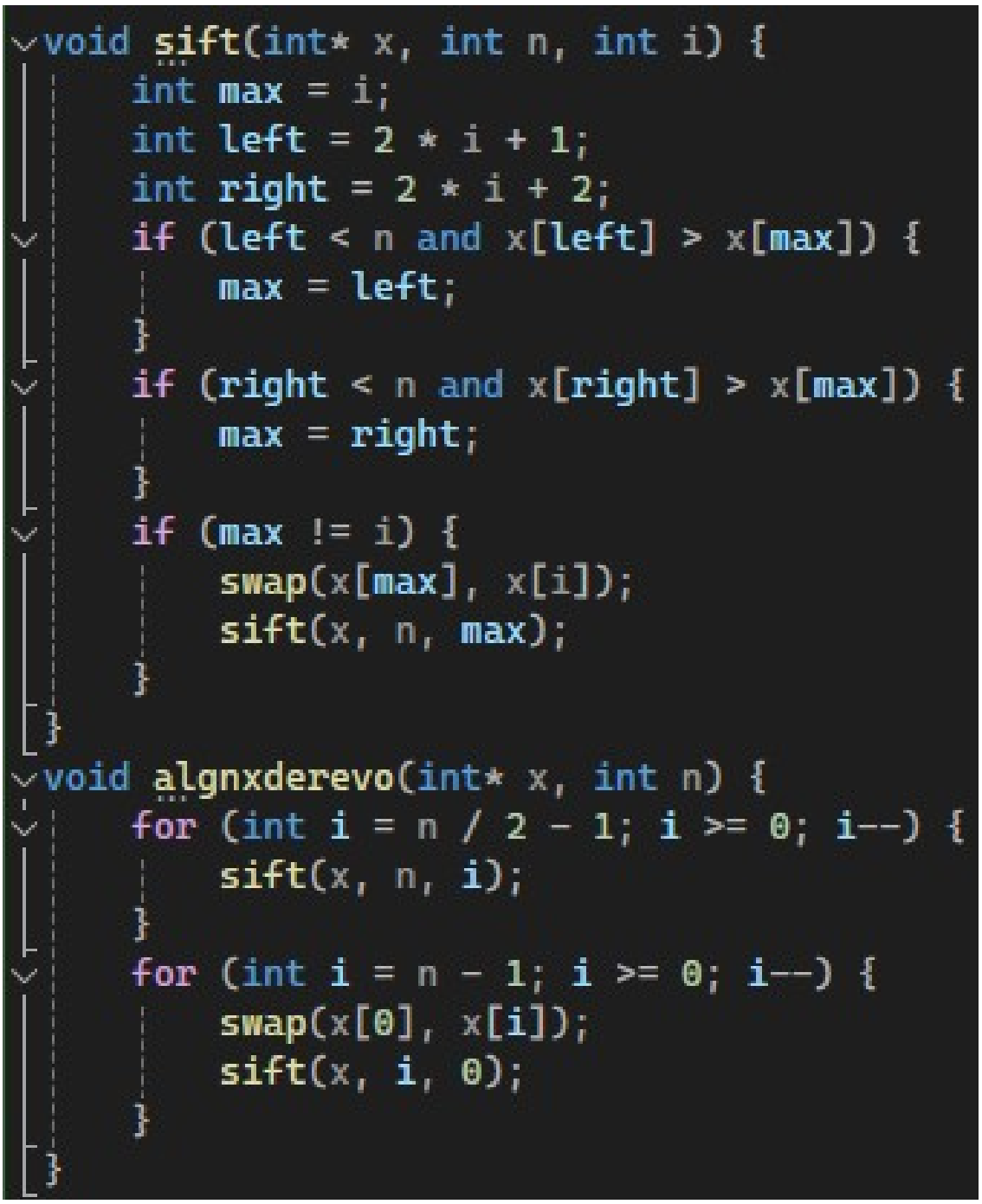
2.1 РЕАЛИЗАЦИЯ АЛГОРИТМА

Сортировка кучей, пирамидальная сортировка (англ. Heapsort) — алгоритм сортировки, использующий структуру данных двоичная куча. Это неустойчивый алгоритм сортировки с временем работы , где — количество элементов для сортировки, и использующий дополнительной памяти. Двоичная куча — это законченное двоичное дерево, в котором элементы хранятся в особом порядке: значение в родительском узле больше (или меньше) значений в его двух дочерних узлах. Поскольку двоичная куча — это законченное двоичное дерево, ее можно легко представить в виде массива, а представление на основе массива является эффективным с точки зрения расхода памяти. Если родительский узел хранится в индексе i, левый дочерний элемент может быть вычислен как 2i + 1, а правый дочерний элемент — как 2i + 2, при условии, что индексирование начинается с 0. Данный алгоритм состоит из функции «algnxderevo», которая включает в себя несколько действий:

1 – Построение кучи с помощью оператора цикла «for»;

2 - Один за другим извлекаем элементы из кучи с помощью цикла «for», вызываю рекурсивную функцию «sift».

Задача функции «sift» состоит в том, чтобы преобразовать в двоичную кучу поддерева с корневым узлом i, что является индексом в массиве. (См. Рисунок 1).



```
void sift(int* x, int n, int i) {
    int max = i;
    int left = 2 * i + 1;
    int right = 2 * i + 2;
    if (left < n and x[left] > x[max]) {
        max = left;
    }
    if (right < n and x[right] > x[max]) {
        max = right;
    }
    if (max != i) {
        swap(x[max], x[i]);
        sift(x, n, max);
    }
}
void algnxderevo(int* x, int n) {
    for (int i = n / 2 - 1; i >= 0; i--) {
        sift(x, n, i);
    }
    for (int i = n - 1; i >= 0; i--) {
        swap(x[0], x[i]);
        sift(x, i, 0);
    }
}
```

Рисунок 1 - Реализация алгоритма «Пирамидальной сортировки» на языке C++

Описание работы функций с помощью блок-схем (См. Рисунок 2-3).

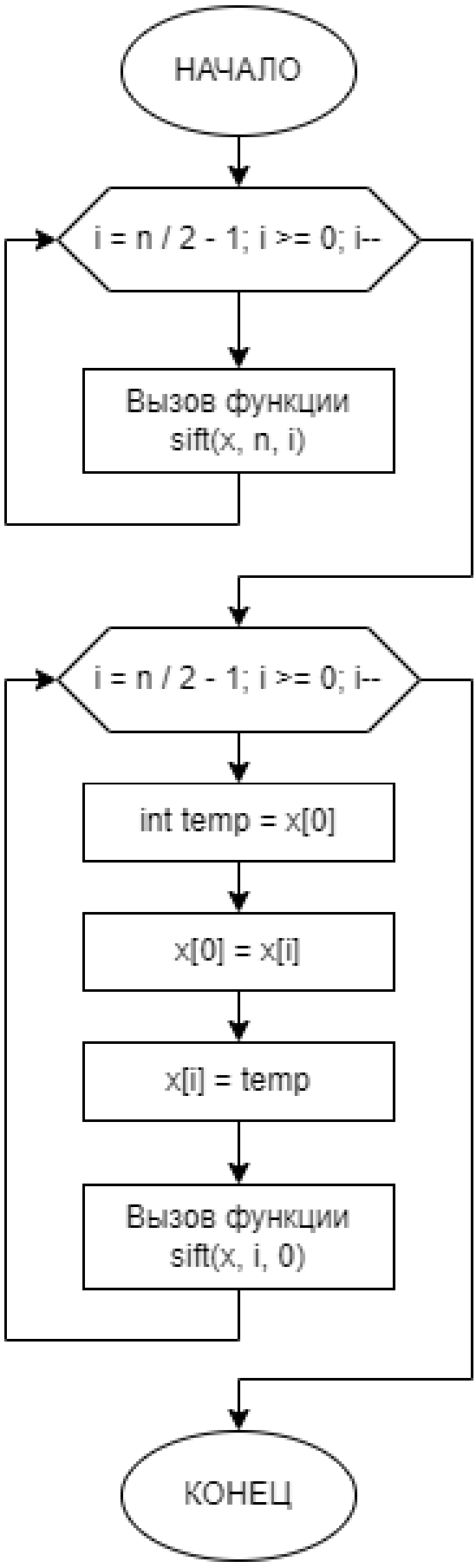


Рисунок 2 — Блок-схема функции «algnxderevo»

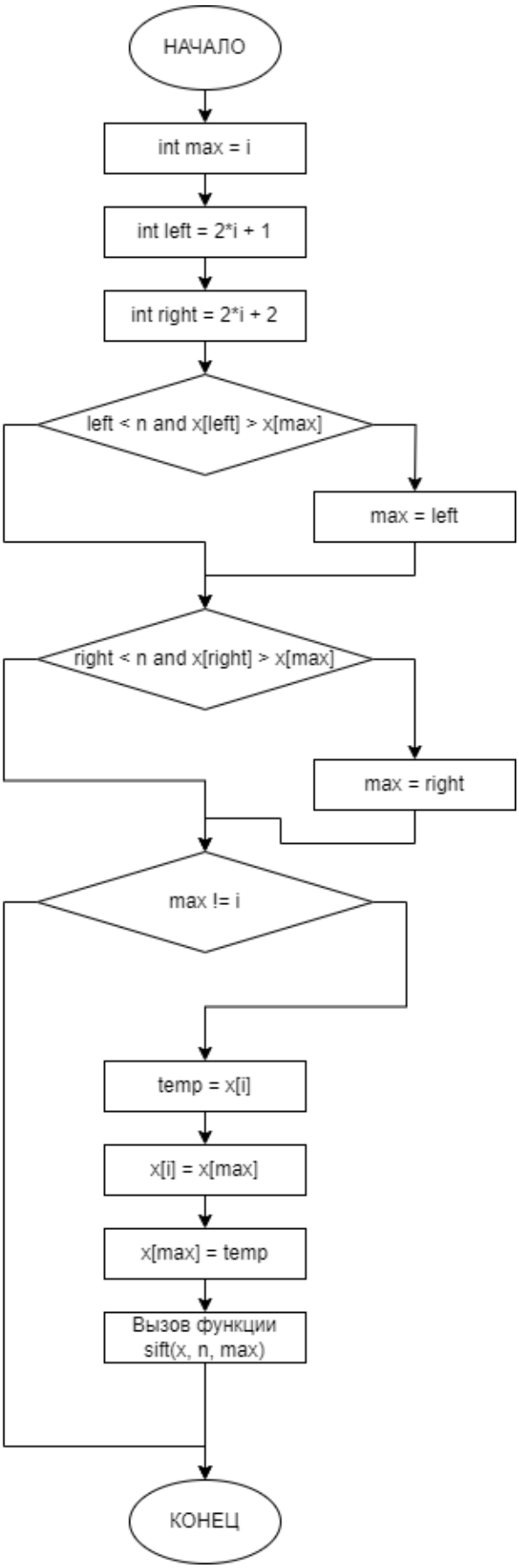


Рисунок 3 — Блок-схема функции «sift»

2.2 ЁМКОСТНАЯ СЛОЖНОСТЬ АЛГОРИТМА «ПИРАМИДАЛЬНАЯ СОРТИРОВКА»

Таблица 1 - Сводная таблица результатов алгоритма «Пирамидальная сортировка».

n	**T(n), мс**	**$T_п(n) = C_ф+M_ф$**
100	0,0005	460
1000	1	6907
10_000	1	92103
100_000	23	1151292
1_000_000	282	Невозможно измерить

Построение пирамиды занимает O(n*log n) операций, причем более точная оценка дает даже O(n) за счет того, что реальное время выполнения «sift» зависит от высоты уже созданной части пирамиды.

Вторая фаза занимает O(n*log n) времени: O(n) раз берется максимум и происходит просеивание бывшего последнего элемента. Плюсом является стабильность метода: среднее число пересылок (n*log n)/2, и отклонения от этого значения сравнительно малы.

На рисунке 4 изображен график функции роста алгоритма «Пирамидальная сортировка».

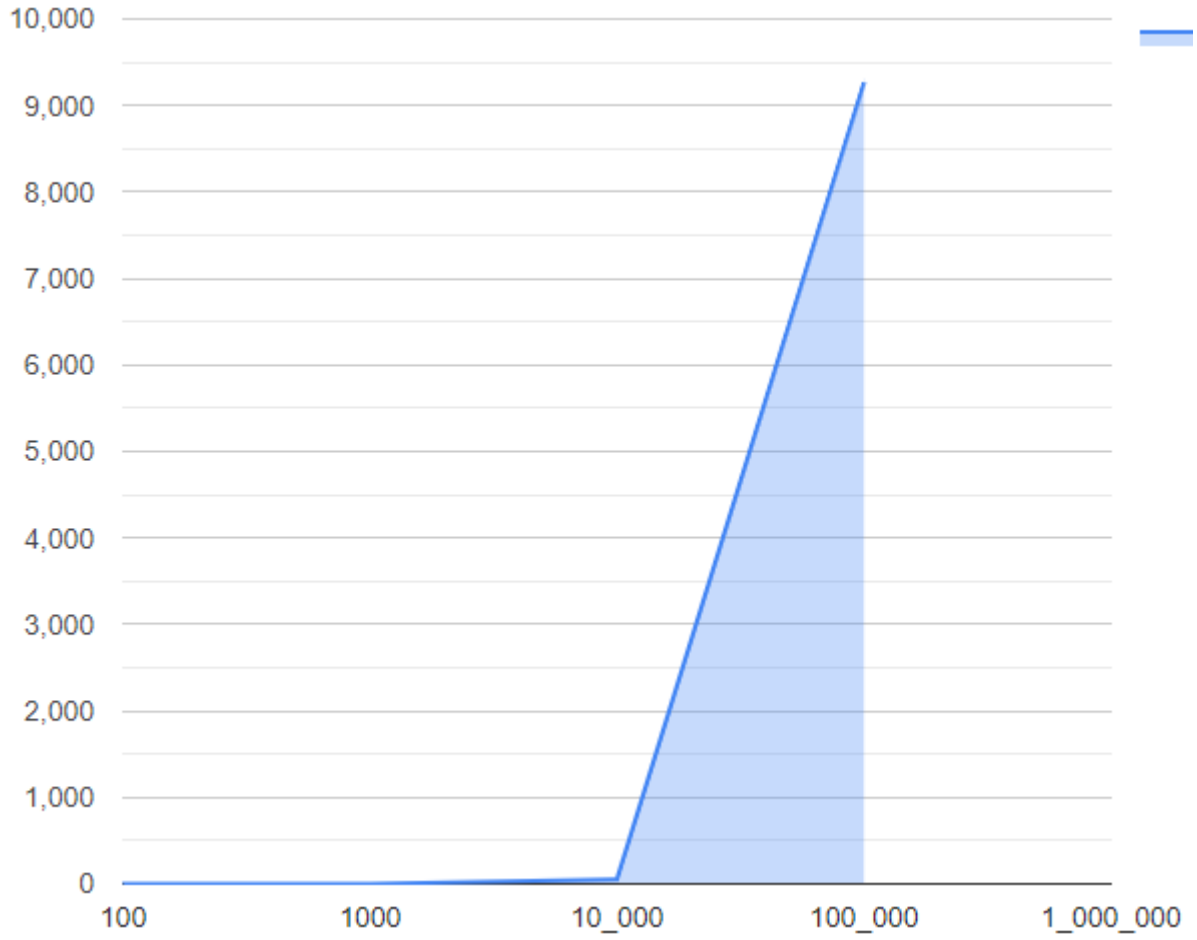


Рисунок 4 — Функция роста алгоритма «Пирамидальная сортировка»

Результаты работы алгоритма «Пирамидальная сортировка» при n = 10, n = 100, n = 1000, n = 10_000. (См. Рисунок 5-8).

```
ИСХОДНЫЙ МАССИВ.
8 0 4 6 9 8 5 1 3 8
The time: 0 ms
РЕЗУЛЬТАТ.
0 1 3 4 5 6 8 8 8 9
```

Рисунок 5 — Результат работы алгоритма «Пирамидальная сортировка» при n = 10

```
ИСХОДНЫЙ МАССИВ.
1 4 1 4 9 9 3 1 0 7 8 6 4 4 3 0 3 7 0 5 9 1 8 3 7 4 1 3 9 7 7 7 4 7 3 2 6 3 9 3 6 6 1 0 2 5 3 4 7 5 1 7 0 6 0 6 6 0 5 4
1 8 8 9 2 7 9 1 6 7 5 2 3 8 5 6 7 7 6 7 6 1 2 7 8 9 3 8 6 5 7 8 4 5 0 7 9 8 5 7
The time: 0 ms
РЕЗУЛЬТАТ.
0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 4 4 4 4 4 4 4 4 4 5 5 5 5 5 5 5 5 5 6 6 6 6 6 6 6 6
6 6 6 6 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 8 8 8 8 8 8 8 8 8 9 9 9 9 9 9 9 9 9
```

Рисунок 6 — Результат работы алгоритма «Пирамидальная сортировка» при n = 100

Рисунок 7 — Результат работы алгоритма «Пирамидальная сортировка» при n = 1000

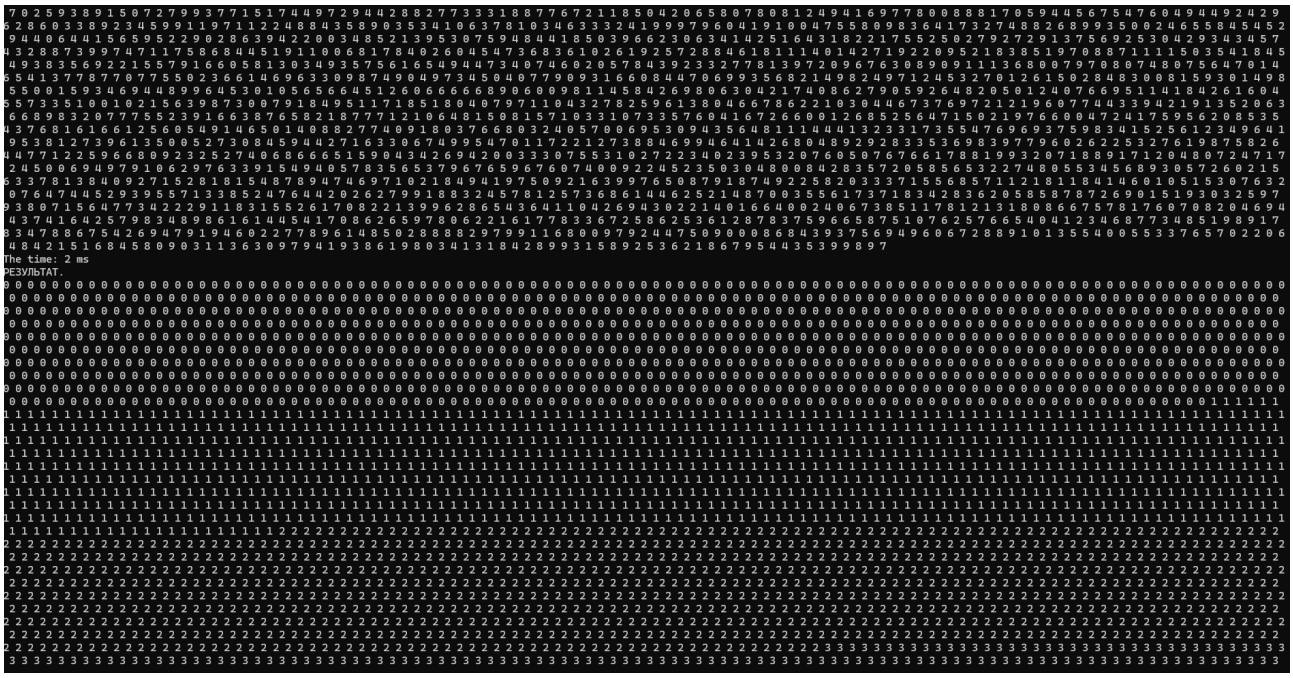


Рисунок 8 — Результат работы алгоритма «Пирамидальная сортировка» при n = 10000

2.3 АЛГОРИТМ СОРТИРОВКИ ПРОСТЫМ СЛИЯНИЕМ

Сортировка слиянием — алгоритм сортировки, который упорядочивает списки (или другие структуры данных, доступ к элементам которых можно получать только последовательно, например — потоки) в определённом порядке.

Слияние означает объединение двух (или более) последовательностей в одну упорядоченную последовательность при помощи циклического выбора элементов, доступных в данный момент.

Реализация алгоритма простого слияния на языке C++ (См. Рисунок 9).

```
void sort_direct_merge(int* a, int fsize) {
    if (fsize < 2)return;
    sort_direct_merge(a, fsize / 2);
    sort_direct_merge(&a[fsize / 2], fsize - (fsize / 2));
    int* buf = new int[fsize];
    int idbuf = 0, idl = 0, idr = fsize / 2;
    while ((idl < fsize / 2) && (idr < fsize))
        if (a[idl] > a[idr])
            buf[idbuf++] = a[idl++];
        else
            buf[idbuf++] = a[idr++];
    while (idl < fsize / 2) buf[idbuf++] = a[idl++];
    while (idr < fsize) buf[idbuf++] = a[idr++];
    for (idl = 0; idl < fsize; idl++)a[idl] = buf[idl];
    delete[]buf;
}
```

Рисунок 9 — Реализация алгоритма простого слияния на языке C++

Описание работы алгоритма «Простое слияние» с помощью блок-схем (См. Рисунок 10).

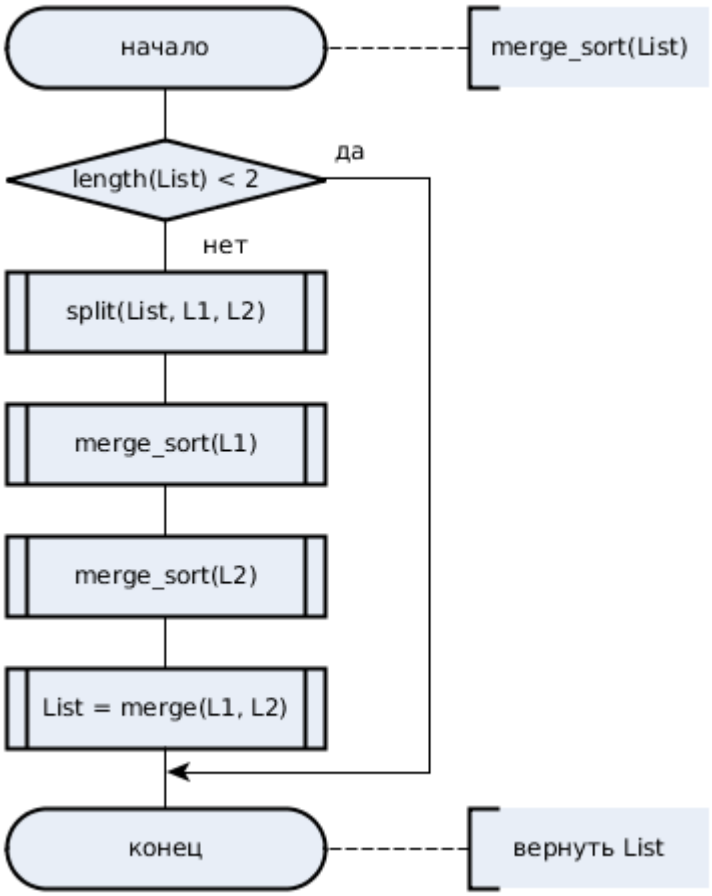


Рисунок 10 — Блок-схема алгоритма «Простое слияние»

Таблица 2 – Сводная таблица результатов алгоритма «Простым слиянием»

n	T(n)	$T_n = C_n + M_n$
100	0,0016	1059
1000	0,107	15201
10000	3	198069
100000	37	2441214
1000000	385	Невозможно измерить

Результаты работы алгоритма «Простое слияние» при n = 10, n = 100, n = 1000, n = 10_000. (См. Рисунок 11-14).

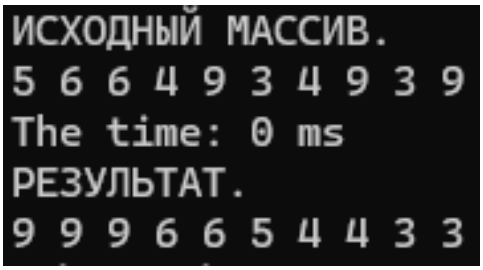


Рисунок 11 — Результат работы алгоритма «Простое слияние» при n = 10

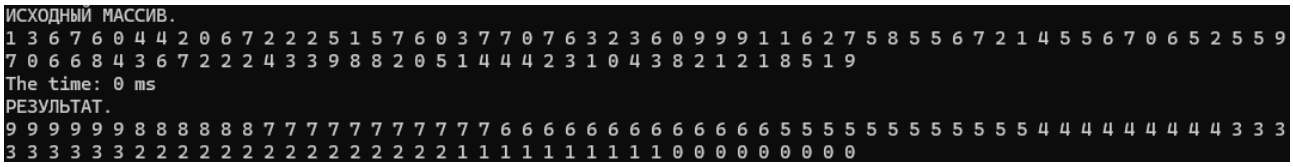


Рисунок 12 — Результат работы алгоритма «Простое слияние» при n = 100

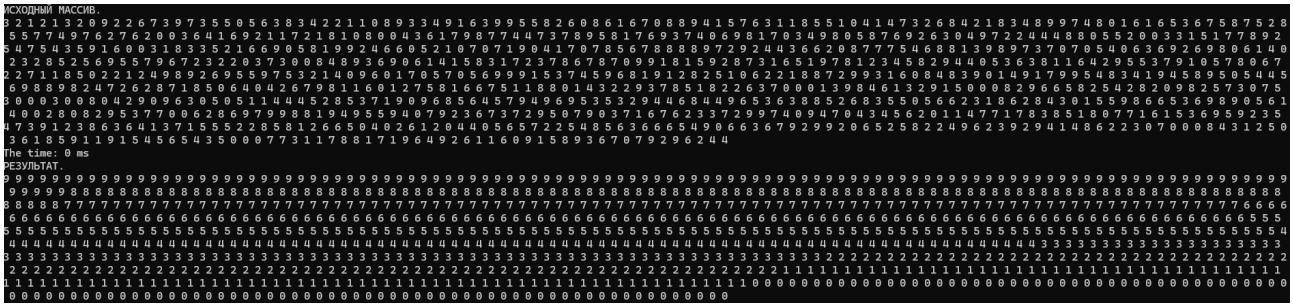


Рисунок 13 — Результат работы алгоритма «Простое слияние» при n = 1000

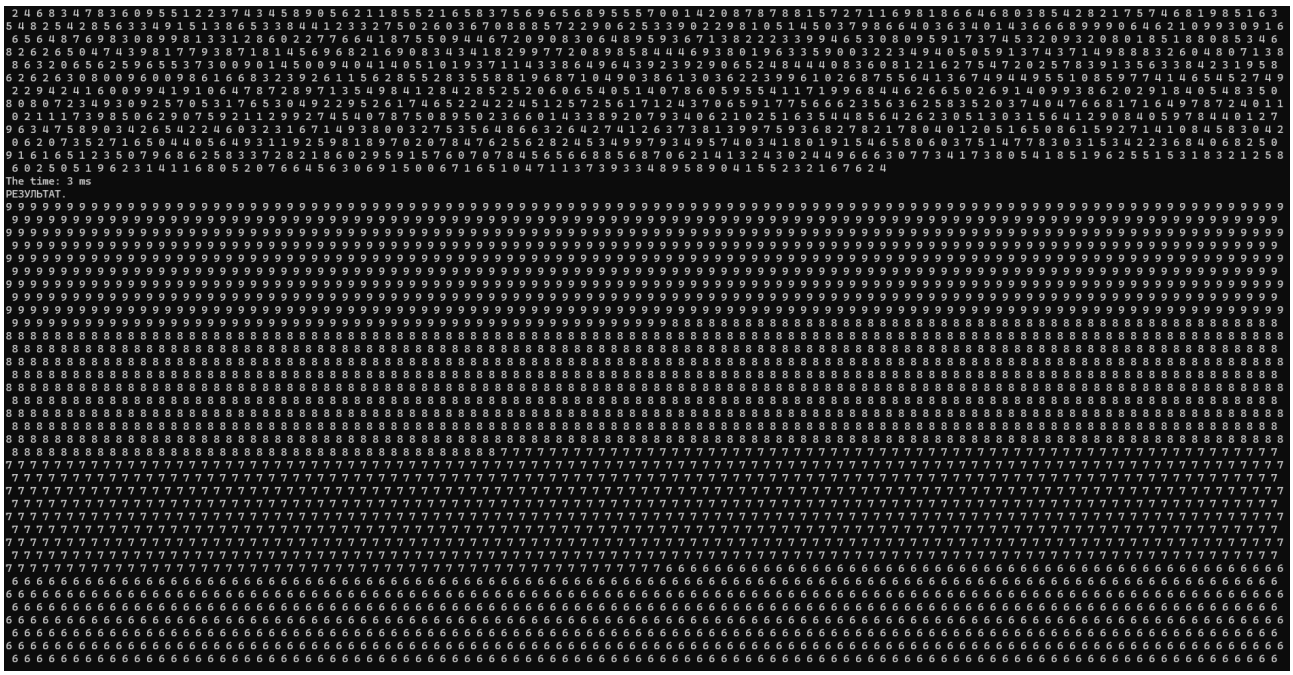


Рисунок 14 — Результат работы алгоритма «Простое слияние» при n = 10000

Для сравнения работы сортировок возьмем результаты работы алгоритма «Insertion Sort» и только что полученные данные.

Таблица 3 – Сводная таблица результатов для алгоритма «Insertion Sort»

n	T(n)	$\mathbf{T_n = C_n + M_n}$
100	0,0016	6040
1000	0,107	545348
10000	10,187	6085127653
100000	1045,23	497743278328
1000000	Невозможно измерить	Невозможно измерить

По трем таблицам результатов наших алгоритмов можем сделать вывод о том, что самый эффективный алгоритм «Пирамидальная сортировка». Второе место занимает алгоритм «Простое слияние».

2. 4 ДОПОЛНИТЕЛЬНЫЕ ПРОГОНЫ

Дополнительные прогоны сортировок на убывающем и возрастающем порядке элементов представлены на рисунках 8-9, при n = 100.

Таблица 3 — Сравнение алгоритма «Пирамидальная сортировка», при n = 100, когда задан массив в возрастающем и убывающем порядках элементов

n	T(n), мс	$T_п(n) = C_ф+M_ф$
100(При возрастании)	0,203	2405
100(При убывании)	0,704	5607

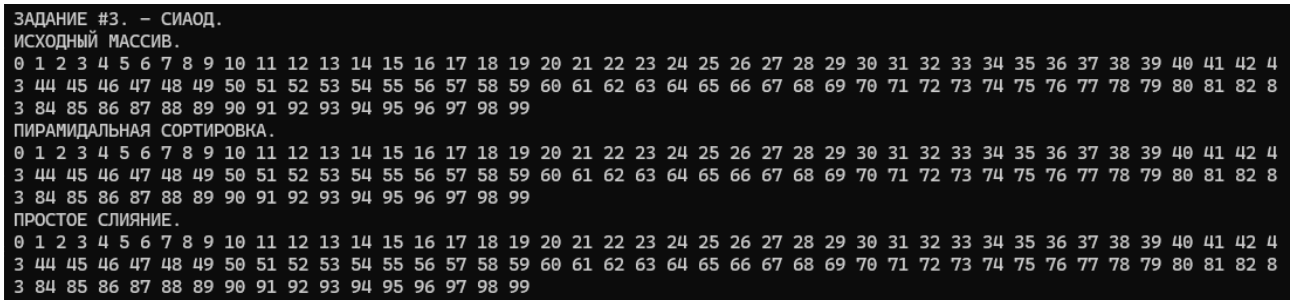


Рисунок 15 — Примеры работы алгоритмов при n = 100, когда элементы стоят в возрастающем порядке

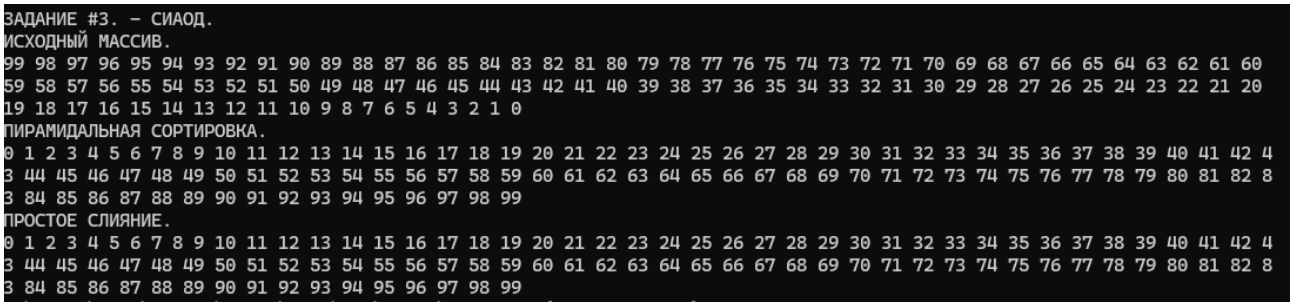


Рисунок 16 — Примеры работы алгоритмов при n = 100, когда элементы стоят в убывающем порядке

Заметим, что при сортировке убывающих элементов требуется большая ёмкостная сложность выполнения алгоритма, чем при возрастающем порядке элементов заданного массива.

2. 5 ВЫВОД ПО ПЕРВОМУ ЗАДАНИЮ

Пирамидальная сортировка имеет сложность O(n*log n), что означает, что время выполнения алгоритма увеличивается логарифмически пропорционально количеству элементов, что является очень быстрым и эффективным для больших массивов данных. В целом, пирамидальная сортировка является эффективным и надёжным алгоритмом сортировки, который широко

применяется в различных областях программирования и инженерии для сортировки массивов данных.

РЕШЕНИЕ ЗАДАНИЯ 2

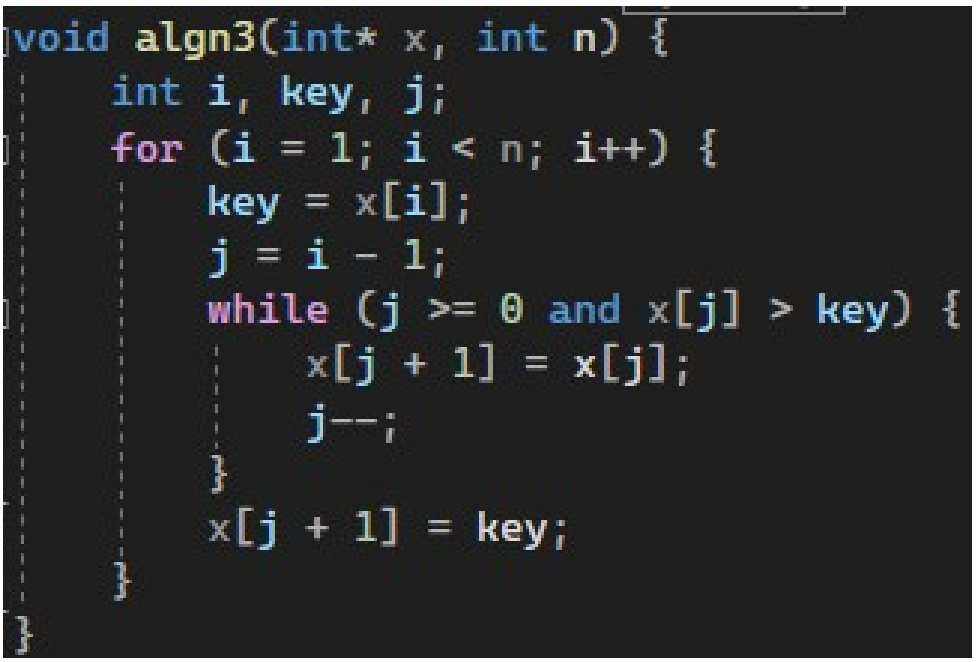
2.1 АЛГОРИТМ ИЗ ПРЕДЫДУЩЕГО ЗАДАНИЯ

Таблица 4 – Сводная таблица результатов для алгоритма «Insertion Sort» из предыдущего задания

n	T(n)	$T_n = C_n + M_n$
100	0,0016	6040
1000	0,107	545348
10000	10,187	6085127653
100000	1045,23	497743278328
1000000	Невозможно измерить	Невозможно измерить

Рисунок 10 – Функция роста сортировки «Вставками»

Реализация данного алгоритма на языке C++ (См. Рисунок 11).



```
void algn3(int* x, int n) {
    int i, key, j;
    for (i = 1; i < n; i++) {
        key = x[i];
        j = i - 1;
        while (j >= 0 and x[j] > key) {
            x[j + 1] = x[j];
            j--;
        }
        x[j + 1] = key;
    }
}
```

Рисунок 11 – Реализация алгоритма «Вставками» на языке C++

2.2 СРАВНЕНИЕ ТРЕХ АЛГОРИТМОВ

Таблица 5 — Сводная таблица результатов асимптотической сложности алгоритмов

Алгоритмы	Асимптотическая сложность алгоритма			
	Наихудший случай(сверху)	Наилучший случай(снизу)	Средний случай(точная оценка)	Емкостная сложность
Простой	$O(n^2)$	O(n)	Невозможно определить	n
Усовершенствованный	O(n*log n)	O(n)	Невозможно определить	n*log n
Быстрый	$O(n^2)$	O(n)	Невозможно определить	n

2.3 ВЫВОД ПО ВТОРОМУ ЗАДАНИЮ

Реализовав и протестировав три алгоритма, были найдены и внесены в таблицу 5 «Асимптотические сложности» всех алгоритмов.

ВЫВОД

В результате проделанной работы были изучены два новых алгоритма:

- Пирамидальная сортировка;

- Простое слияние.

Также заново рассмотрен простой алгоритм «Вставками». Произведено сравнение и выяснилось, что простой алгоритм эффективен для массивов малых размеров, а усовершенствованный — для массивов больших размеров.

СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ

1. Структуры данных и проектирование программ : Пер. с англ. / Р. Круз. — М.: БИНОМ. Лаборатория знаний, 2017. — 766 с.

2. Полный справочник по C++ : Пер. с англ. / Г. Шилдт. — М.: ООО "И.Д.Вильямс", 2016. — 796 с.: ил. — Предм. указ.: с. 787-796