ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА»
(СПбГУТ)
Кафедра информационных управляющих систем

ОТЧЁТ

по лабораторной работе №4

по дисциплине «Структура и алгоритмы обработки данных в информационных системах и сетях»

Выполнил: студент группы ИСТ-813, /Кравец А.Ю./

«04» декабря 2020 г. _____/А.Ю. Кравец/

Принял: ст. преподаватель Антонов В.В.

«05» декабря 2020 г. _____/В.В. Антонов/

Санкт-Петербург
2020

Задание: описать алгоритмы пирамидальной сортировки (Heap sort), поразрядной сортировки (Radix sort), реализовать их программно, провести сравнение быстродействия алгоритмов.

1 Описание работы алгоритмов

1.1 Пирамидальная сортировка (Heap sort)

Алгоритм основан на абстрактной структуре данных, называемой пирамидой (heap).

Пирамида есть бинарное дерево высоты k, в котором:

- Все узлы имеют глубину k или $k-1$ – дерево сбалансированное;
- Уровень $k-1$ заполнен полностью, а уровень k заполен слева направо;
- Каждый элемент меньше, либо равен родителю.

Пирамиду можно представить в виде одномерного массива A, где для родителя $A[i]$ левый и правый дочерние элементы хранятся соответственно в $A[2i+1]$ и $A[2i+2]$ – если нумерация элементов массива начинается с нуля.

Алгоритм пирамидальной сортировки имеет вид:

1. Преобразование исходного массива к пирамиде:
2. Взятие корня пирамиды: первый и последний элемент несортированной части массива меняются местами;
3. Несортированная часть массива уменьшается на 1 – количество элементов пирамиды уменьшается на 1;
4. Добавления элемента, который занял место корня на шаге 2, в пирамиду;
5. Возвращение к шагу 2.

Добавление элемента в пирамиду реализуется следующим образом:

0. Новый элемент занимает место корня пирамиды.
1. Выбирается наибольший из потомков;
2. Если наибольший потомок не превосходит по значению родителя – операция завершается;

3. Если наибольший потомок превосходит по значению родителя:
   3.1. Новый элемент и его потомок меняются местами;
   3.2. Если индекс нового элемента меньше значения половины размера массива, выполняется шаг 1 для нового элемента.

Преобразование исходного массива в пирамиду (Шаг 1 основного алгоритма) реализуется следующим образом:

0. Элементы массива, индексы которых больше или равны значению половины размера массива – т.е. не имеющие потомков, уже являются сформированной частью пирамиды.
1. Элемент, расположенный левее начала сформированной части пирамиды, добавляется в пирамиду.
2. Граница сформированной части пирамиды сместилась на один элемент влево.
3. Возвращение к шагу 1.

Сложность алгоритма: $O(nlog_2 n)$.

1.2 Поразрядная сортировка (Radix sort)

Имеется:

k – количество разрядов в максимальном числе;

m – количество возможных значений разряда: для чисел – основание системы счисления.

Алгоритм поразрядной сортировки имеет следующий вид:

0. Значение $i = 1$.
1. Выполнение сортировки подсчетом элементов по $i-$му справа разряду.
2. Увеличение i на 1.
3. Если $i \leq k$: переход к шагу 1. Иначе – завершение работы алгоритма.

Алгоритм сортировки подсчетом имеет следующий вид:

1. Создание массива счетчиков возможных значений разряда – номер счетчика соответствует значению разряда;

2. Проход по исходному массиву от начала до конца, для каждого выбранного разряда увеличивая на один значение счетчика с соответствующим номером;
3. Увеличение каждого счетчика на значение, равное сумме всех предшествующих ему в массиве счетчиков;
4. Расстановка элементов исходного массива в выходной массив согласно массиву счетчиков: счетчик соответствующего разряда после шага 3 хранит позицию элемента в массиве.

Сложность алгоритма: $O(k(n+m))$, где k – количество разрядов в максимальном числе, m – основание системы счисления.

2 Сравнение быстродействия алгоритмов

Для сравнения замерялось время сортировки массивов со случайными значениями элементов: в одном случае – каждый элемент массива есть девятизначное десятичное число, а в другом случае – каждый элемент массива есть четырехзначное десятичное число. Полученные значения для алгоритмов пирамидальной сортировки и поразрядной сортировки представлены в таблице 1.

Таблица 1 – Время выполнения алгоритмов пирамидальной сортировки, поразрядной сортировки

Размер массива, эл	Время выполнения, с			
	Heap Sort		Radix Sort	
	Число десятичных разрядов максимального элемента			
	4	9	4	9
2,50E+06	1,359	1,328	0,547	1
5,00E+06	1,688	1,906	0,89	1,328
7,50E+06	2,484	2,735	0,968	1,813
1,00E+07	3,485	3,843	1,047	2,469
1,25E+07	4,469	4,969	1,391	3,031
1,50E+07	5,531	6,141	1,563	3,641
1,75E+07	6,703	7,407	1,922	4,219
2,00E+07	7,86	8,688	2,078	4,797
2,25E+07	8,828	9,86	2,422	5,25
2,50E+07	9,907	11,266	2,688	5,844

Продолжение таблицы 1

2,75E+07	11,05	12,594	2,844	6,375
3,00E+07	12,079	13,891	3,172	6,937
3,25E+07	13,079	15,141	3,297	7,516
3,50E+07	14,219	16,673	3,61	8,079
3,75E+07	15,266	17,782	3,859	8,891
4,00E+07	15,907	19,392	4,109	9,469
4,25E+07	17,36	20,939	4,375	10,063
4,50E+07	18,158	22,017	4,565	10,345
4,75E+07	19,283	23,923	4,922	11,125
5,00E+07	20,439	25,314	5,203	11,626

Для сравнения были составлены графики по таблице 1 для случаев, когда массив состоит из девятизначных чисел (рисунок 1); когда массив состоит из четырехзначных чисел (рисунок 2).

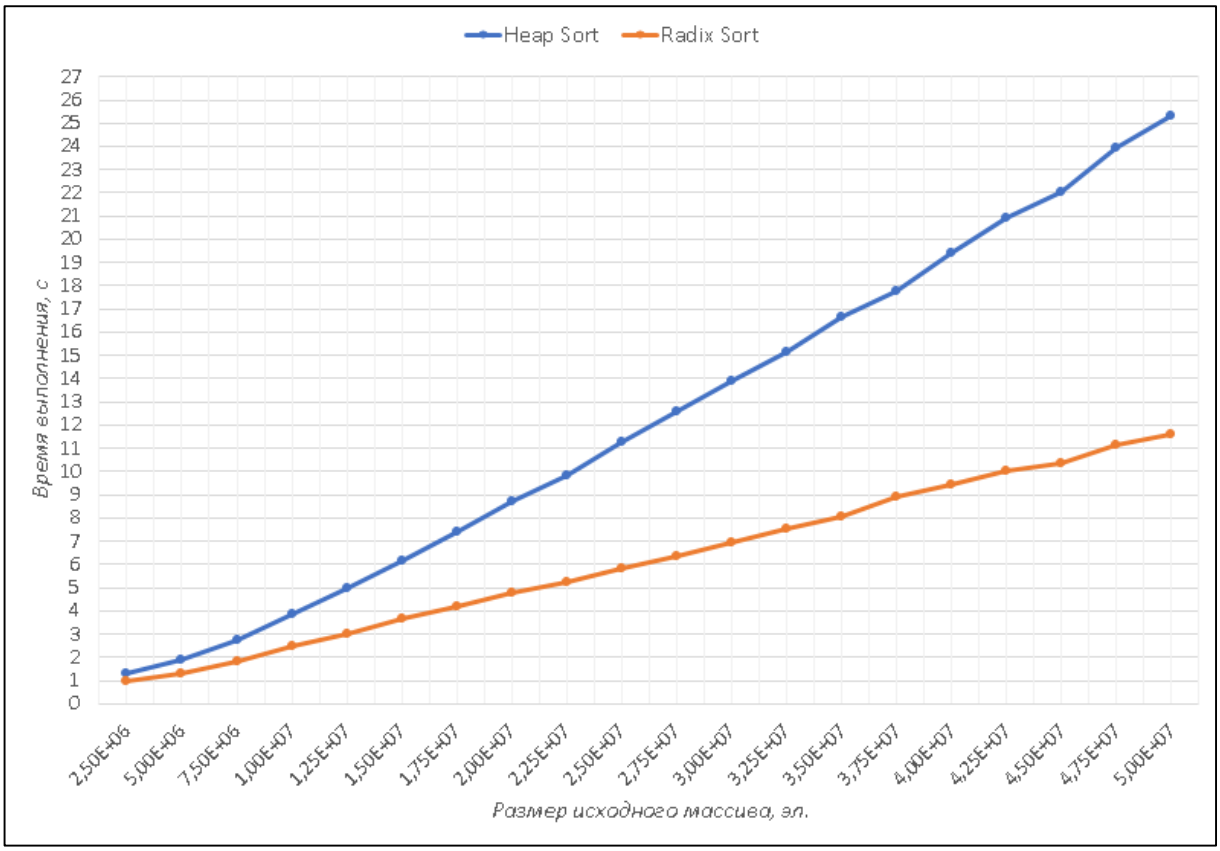

Рисунок 1 – Время выполнения Heap Sort и Radix Sort для массивов, элементы которых – девятизначные десятичные числа

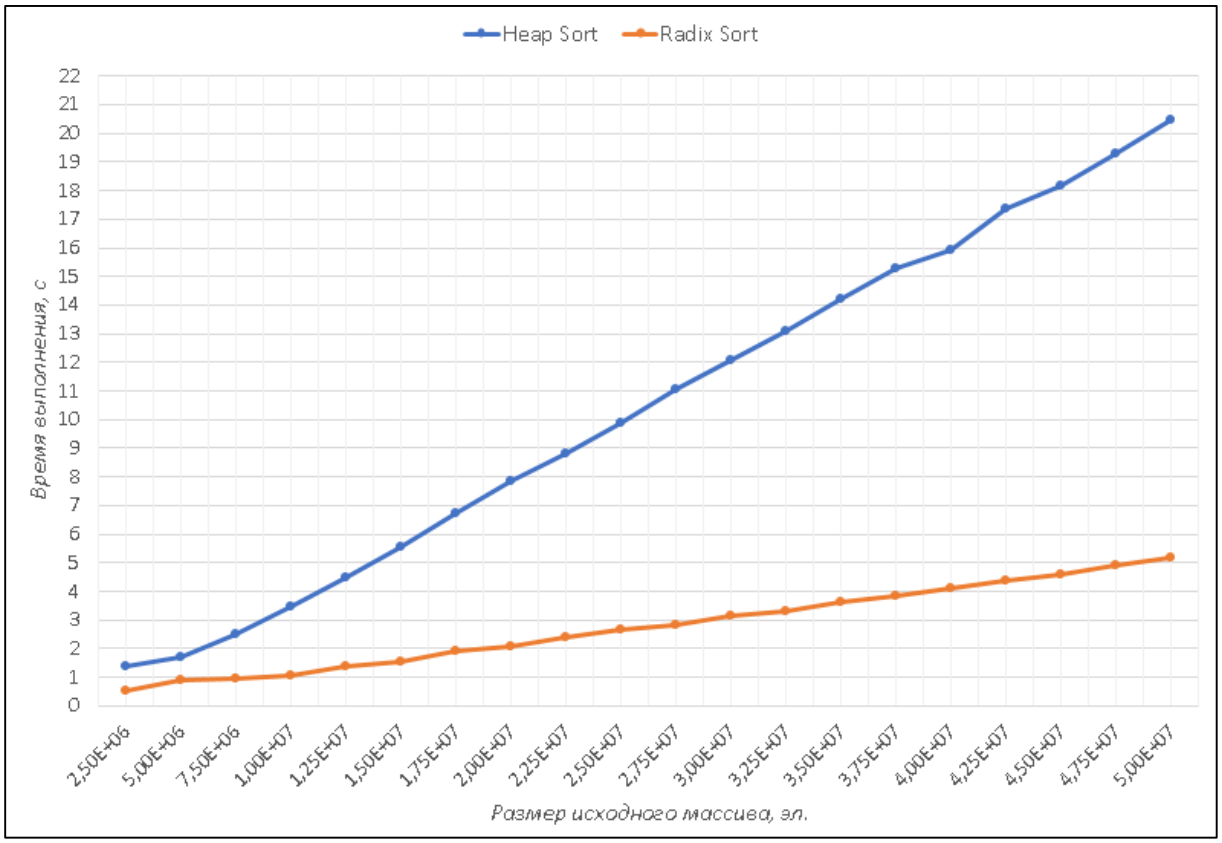

Рисунок 2 – Время выполнения Heap Sort и Radix Sort для массивов, элементы которых – четырехзначные десятичные числа

Вывод:

В данной работе были рассмотрены алгоритмы пирамидальной сортировки и поразрядной сортировки.

Данные алгоритмы были реализованы программно. Были получены данные о времени выполнения каждого алгоритма при различных размерах исходного массива для случаев, когда массив состоит из девятизначных чисел; когда массив состоит из четырехзначных чисел.

Исходя из собранных данных (рисунок 1 и 2) ранжирование алгоритмов в порядке убывания быстродействия имеет вид:

1. Поразрядная сортировка (Radix sort: $O(k(n+m))$);
2. Пирамидальная сортировка (Heap sort: $O(nlog_2 n)$).

Причем, разница в быстродействии алгоритмов увеличилась с уменьшением количества разрядов элементов исходного массива. Данное поведение соответствует результату сравнения сложностей алгоритмов: при

$m \ll n$ и $k < \log_2 n$ разница в быстродействии поразрядной сортировки и пирамидальной сортировки пропорциональна величине $\frac{\log_2 n}{k}$.