МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №4
по дисциплине «Алгоритмы и структуры данных»
Тема: Алгоритмы сортировки

Студент гр. 9303 _____ Махаличев Н.А.

Преподаватель _____ Филатов Ар.Ю.

Санкт-Петербург
2020

Цель работы.

Написание алгоритма сортировки в соответствие с вариантом задания, теоретическая оценка сложности алгоритма

Задание.

Вариант 13.

Реализовать алгоритм пирамидальной сортировки

Описание алгоритма.

Пирамидой называется такое двоичное дерево, что

$$a[i] \leq a[2i+1]$$

$$a[i] \leq a[2i+2]$$

а a[0] – минимальный размер пирамиды.

Алгоритм пирамидальной сортировки следующий:

1. Выстраиваются элементы пирамиды по следующему условию:

   $$a[i] \geq a[2i+1]$$

   $$a[i] \geq a[2i+2]$$

   где $i \in [0, \frac{size}{2} - 1]$ (size – количество элементов)
2. Меняем местами последний элемент пирамиды с первым, уменьшая её размер на 1.
3. Применить пункт первый на уменьшенной пирамиде.
4. Повторять второе и третье действия, пока размер пирамиды больше 1.

Таким образом элементы массива выстраиваются от меньшего к большему.

Чтобы элементы сортировались в порядке уменьшения, используется аналогичный алгоритм, но в первом пункте элементы выстраиваются по условию

$$a[i] \leq a[2i+1]$$

$$a[i] \leq a[2i+2]$$

Сложность алгоритма:

1. Сложность выстраивания элементов функций согласно пункту 1 – O(log(n));
2. Сложность 2-4 пунктов в худшем случае – O(n), в среднем – O(n/2);
3. Общая сложность алгоритма в худшем случае – O(n log(n)), в среднем – O(n/2 log(n)).

Достоинства:

1. Требуется всего лишь O(1) дополнительной памяти для перестановки;
2. Небольшая сложность алгоритма в среднем и худшем случае, эффективен для больших значений n.

Недостатки:

1. Не учитывается частичная упорядоченность;
2. На частично иди достаточно отсортированных массивах работает столько же, сколько на неотсортированных;
3. Работает последовательно, не распараллеливается.

Выполнение работы.

Создана функция PyramidSort(T pyramid[], int size, int sort_type), выполняющая основной алгоритм сортировки, описанный ранее. Функция принимает на вход указатель на массив данных, который необходимо отсортировать, его размер, а также способ сортировки, который необходимо совершить (от меньшего к большему или наоборот).

Функции SortByMax(T pyramid[], int index, int size, int size_of_array) и SortByMin(T pyramid[], int index, int size, int size_of_array) сортируют пирамиду согласно пункту 1 описанного алгоритма.

Код программы см. в приложении А. Тестирование программы см. в приложении Б.

Выводы.

В процессе выполнения лабораторной работы был изучен алгоритм пирамидальной сортировки, определена его сложность, достоинства и недостатки.

Была разработана программа, реализовывающая алгоритм пирамидальной сортировки на сгенерированном наборе случайных чисел и выводящая результат работы данного алгоритма.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Файл main.cpp

```
#include <iostream>

using namespace std;

template <typename T>
void SortByMax(T pyramid[], int index, int size, int size_of_array){
    for (int i = 0; i < size_of_array; i++){
        cout << pyramid[i] << ' ';
    }
    cout << endl;
    int left_root = 2 * index + 1;
    int right_root = 2 * index + 2;
    int max = index;
    if ((left_root < size) && (pyramid[left_root] > pyramid[max])){
        max = left_root;
    }
    if ((right_root < size) && (pyramid[right_root] > pyramid[max])){
        max = right_root;
    }
    if (max != index){
        swap(pyramid[index], pyramid[max]);
        SortByMax(pyramid, max, size, size_of_array);
    }
}

template <typename T>
void SortByMin(T pyramid[], int index, int size, int size_of_array){
    for (int i = 0; i < size_of_array; i++){
        cout << pyramid[i] << ' ';
    }
    cout << endl;
    int left_root = 2 * index + 1;
    int right_root = 2 * index + 2;
    int min = index;
    if ((left_root < size) && (pyramid[left_root] < pyramid[min])){
        min = left_root;
    }
    if ((right_root < size) && (pyramid[right_root] < pyramid[min])){
        min = right_root;
    }
    if (min != index){
        swap(pyramid[index], pyramid[min]);
        SortByMin(pyramid, min, size, size_of_array);
    }
}

template <typename T>
void PyramidSort(T pyramid[], int size, int sort_type){
    for (int i = (size / 2 - 1); i >= 0; i--){
        if (sort_type == 1){
            SortByMax(pyramid, i, size, size);
        } else {
```

```
            if (sort_type == 2){
                SortByMin(pyramid, i, size, size);
            } else {
                cout << "Wrong choice" << endl;
                return;
            }
        }
    }
    for (int i = (size - 1); i >= 0; i--){
        swap(pyramid[0], pyramid[i]);
        if (sort_type == 1){
            SortByMax(pyramid, 0, i, size);
        } else {
            SortByMin(pyramid, 0, i, size);
        }
    }
}

int main(){
    int quantity = 0;
    cout << "Amount of numbers = ";
    cin >> quantity;
    int array[quantity];
    for (int i = 0; i < quantity; i++){
        array[i] = (rand() % (2 * quantity)) - quantity;
    }
    cout << "Generated array: ";
    for (int i = 0; i < quantity; i++){
        cout << array[i] << ' ';
    }
    cout << endl;
    int choice = 0;
    cout << "The way to sort ('1' to min->max; '2' to max->min): ";
    cin >> choice;
    PyramidSort(array, quantity, choice);
    cout << "Result: ";
    for (int i = 0; i < quantity; i++){
        cout << array[i] << ' ';
    }
    cout << endl;
    return 0;
}
```

ПРИЛОЖЕНИЕ Б

ТЕСТИРОВАНИЕ

Таблица Б.1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1	Amount of numbers = 7 Generated array: -6 -3 2 -2 -6 -4 3 The way to sort ('1' to min->max; '2' to max->min): 1	Result: -6 -6 -4 -3 -2 2 3	Получен ожидаемый вывод.
2	Amount of numbers = 11 Generated array: 6 -1 6 2 -10 4 -11 -5 -8 -10 -3 The way to sort ('1' to min->max; '2' to max->min): 2	Result: 6 6 4 2 -1 -3 -5 -8 -10 -10 -11	Получен ожидаемый вывод
3	Amount of numbers = 1 Generated array: 0 The way to sort ('1' to min->max; '2' to max->min): 1	Result: 0	Получен ожидаемый вывод
4	Amount of numbers = 2 Generated array: 1 0 The way to sort ('1' to min->max; '2' to max->min): 2	Result: 1 0	Получен ожидаемый вывод
5	Amount of numbers = 5 Generated array: -2 1 2 0 -2 The way to sort ('1' to min->max; '2' to max->min): 2	Result: 2 1 0 -2 -2	Получен ожидаемый вывод