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: Сортировки

Студент гр. 9381 _____ Андрух И.А.

Преподаватель _____ Фирсов М.А.

Санкт-Петербург
2020

Цель работы.

Познакомиться со структурой данных бинарного дерева, реализовать его рекурсивную обработку на языке программирования C++.

Задание.

Вариант 10.

Быстрая сортировка, рекурсивная реализация. Процедура трёхчастного разделения. Деление производится не на две группы, а на три: <x, =x, >x.

Функции и структуры данных.

Для хранения чисел была создана структура array_list. Поля структуры:

- int* array – массив чисел;
- int capacity – размерность массива array;
- int count – количество элементов, записанных в массив array;

Для работы со списком использовались функции:

- void resize(int new_capacity) – изменяет размерность массива;
- array_list::array_list(int start_capacity) – создает массив заданной размерности без записанных туда значений;
- int& array_list::operator[] (int index) – позволяет обращаться к элементам поля array через объект типа array_list по индексу;
- void array_list::push_back(int element) – добавление элемента в конец массива;
- int array_list::size() – получение длины списка

Для логгирования, вывода промежуточных данных, была создана функция string log(array_list &list, int min, int max, int pivot, int depth).

Для реализации трёхпутевой сортировки чисел была создана функция void qsort3way(string& s, array_list& list, int l, int r, int depth).

Для вывода результата сортировки создана функция std::string print_list(array_list& list).

Описание алгоритма.

Программа запрашивает у пользователя путь до файла. Если файл не может быть открыт, выводится предупреждение. Из файла считывается строка чисел, числа записываются в элемент типа array_list list, в поле array. Введенные числа выводятся на экран для проверки.

Далее вызывается функция сортировки qsort3way(s, list, 0, list.size() - 1,0).

Трёхпутевая сортировка осуществляется следующим образом: выбирается опорный элемент(в нашем случае - pivot, генерируется случайным образом через rand()), массив чисел делится на три части. В центральной части находятся элементы, равные опорному, в левой – элементы, меньшие, чем опорный, а в правой – большие. Функция сортировки вызывается рекурсивно для левой и правой частей.

Перед рекурсивным вызовом выводятся промежуточные значения: крайние индексы сортируемого интервала, выбранный опорный элемент, глубина рекурсии.

Сложность алгоритма быстрой сортировки с процедурой трёхчастного разделения в худшем случае такая же, как и у обычной быстрой сортировки – O(n^2). Преимущество данной сортировки в том, что при малой частоте вариативности элементов сокращается количество итераций относительно быстрой сортировки с делением пополам.

В наиболее сбалансированном варианте при каждой операции разделения массив делится на две примерно одинаковые части, следовательно, сложность алгоритма будет равна O(n*$log_2 n$).

В самом несбалансированном варианте каждое разделение дает два подмассива размерами 1 и n-1, то есть при каждом рекурсивном вызове больший массив будет на 1 короче, чем в предыдущий раз. Такое может произойти, если в качестве опорного на каждом этапе будет выбран элемент либо наименьший, либо наибольший из всех обрабатываемых. Общее время работы составит O(n^2).

Тестирование.

№	Входные данные	Вывод
1	1 2 3 4 2 1	1 2 3 4 2 1 Executing algorithm... ---LOG--- 1 2 3 2 1 4 l:0 r:5 pivot: 4 depth: 0 1 2 2 1 3 4 l:0 r:4 pivot: 3 depth: 1 1 1 2 2 3 4 l:0 r:3 pivot: 1 depth: 2 1 1 2 2 3 4 l:2 r:3 pivot: 2 depth: 3 ---END--- Result: 1 1 2 2 3 4
2	44 122 67 1 67 33 0 89 12 6 34 25 25 78 122 3089 4456 367 13 0 7362 111 45 811	44 122 67 1 67 33 0 89 12 6 34 25 25 78 122 3089 4456 367 13 0 7362 111 45 811 Executing algorithm... ---LOG--- 0 13 1 6 0 12 25 25 89 34 33 67 78 122 3089 4456 367

		67 122 7362 111 45 811 44 l:0 r:23 pivot: 25 depth: 0 0 1 6 0 12 13 25 25 89 34 33 67 78 122 3089 4456 367 67 122 7362 111 45 811 44 l:0 r:5 pivot: 13 depth: 1 0 0 1 12 6 13 25 25 89 34 33 67 78 122 3089 4456 367 67 122 7362 111 45 811 44 l:0 r:4 pivot: 1 depth: 2 0 0 1 12 6 13 25 25 89 34 33 67 78 122 3089 4456 367 67 122 7362 111 45 811 44 l:0 r:1 pivot: 0 depth: 3 0 0 1 6 12 13 25 25 89 34 33 67 78 122 3089 4456 367 67 122 7362 111 45 811 44 l:3 r:4 pivot: 12 depth: 3 0 0 1 6 12 13 25 25 89 34 33 67 78 44 45 111 67 122 122 7362 367 811 4456 3089 l:8 r:23 pivot: 122 depth: 1 0 0 1 6 12 13 25 25 33 34 67 78 44 45 111 67 89 122 122 7362 367 811 4456 3089 l:8 r:16 pivot: 33 depth: 2 0 0 1 6 12 13 25 25 33 34 67 44 45 67 78 89 111 122 122 7362 367 811 4456 3089

		l:9 r:16 pivot: 78 depth: 3 0 0 1 6 12 13 25 25 33 34 44 45 67 67 78 89 111 122 122 7362 367 811 4456 3089 l:9 r:13 pivot: 67 depth: 4 0 0 1 6 12 13 25 25 33 34 44 45 67 67 78 89 111 122 122 7362 367 811 4456 3089 l:9 r:11 pivot: 44 depth: 5 0 0 1 6 12 13 25 25 33 34 44 45 67 67 78 89 111 122 122 7362 367 811 4456 3089 l:15 r:16 pivot: 89 depth: 4 0 0 1 6 12 13 25 25 33 34 44 45 67 67 78 89 111 122 122 3089 367 811 4456 7362 l:19 r:23 pivot: 4456 depth: 2 0 0 1 6 12 13 25 25 33 34 44 45 67 67 78 89 111 122 122 367 811 3089 4456 7362 l:19 r:21 pivot: 3089 depth: 3 0 0 1 6 12 13 25 25 33 34 44 45 67 67 78 89 111 122 122 367 811 3089 4456 7362 l:19 r:20 pivot: 367 depth: 4 ---END---

		Result: 0 0 1 6 12 13 25 25 33 34 44 45 67 67 78 89 111 122 122 367 811 3089 4456 7362
3	21 21 21 21 21 21 21 21 21 21 21 21	21 21 21 21 21 21 21 21 21 21 21 Executing algorithm... ---LOG--- 21 21 21 21 21 21 21 21 21 21 21 l:0 r:11 pivot: 21 depth: 0 ---END--- Result: 21 21 21 21 21 21 21 21 21 21 21
4	2 44 72 56 44 90 12 56 44 76 72 44 2 12 72 90	2 44 72 56 44 90 12 56 44 76 72 44 2 12 72 90 Executing algorithm... ---LOG--- 2 2 12 12 90 44 56 44 76 72 44 56 72 72 90 44 l:0 r:15 pivot: 12 depth: 0 2 2 12 12 90 44 56 44 76 72 44 56 72 72 90 44 l:0 r:1 pivot: 2 depth: 1 2 2 12 12 44 44 44 44 72 76 56 72 72 90 56 90 l:4 r:15 pivot: 44 depth: 1 2 2 12 12 44 44 44 44 72 76 56 72 72 56 90 90 l:8 r:15 pivot: 90 depth: 2

		2 2 12 12 44 44 44 44 56 56 72 72 72 76 90 90 l:8 r:13 pivot: 72 depth: 3 2 2 12 12 44 44 44 44 56 56 72 72 72 76 90 90 l:8 r:9 pivot: 56 depth: 4 ---END--- Result: 2 2 12 12 44 44 44 44 56 56 72 72 72 76 90 90
5	23	23 Executing algorithm... ---LOG--- ---END--- Result: 23
6		Executing algorithm... ---LOG--- ---END--- Result:
7	-3 -7 -1 -4 -77 -11	-3 -7 -1 -4 -77 -11 Executing algorithm... ---LOG--- -11 -7 -77 -4 -1 -3

		l:0 r:5 pivot: -4 depth: 0 -77 -11 -7 -4 -1 -3 l:0 r:2 pivot: -11 depth: 1 -77 -11 -7 -4 -3 -1 l:4 r:5 pivot: -1 depth: 1 ---END--- Result: -77 -11 -7 -4 -3 -1
8	-3 -7 56 -1 -4 12 -77 -7 44 13 0 -11	-3 -7 56 -1 -4 12 -77 -7 44 13 0 -11 Executing algorithm... ---LOG--- -77 56 -1 -4 12 -7 -7 44 13 0 -11 -3 l:0 r:11 pivot: -77 depth: 0 -77 -3 -1 -4 12 -7 -7 13 0 -11 44 56 l:1 r:11 pivot: 44 depth: 1 -77 -3 -4 -11 -7 -7 -1 0 13 12 44 56 l:1 r:9 pivot: -1 depth: 2 -77 -11 -7 -7 -4 -3 -1 0 13 12 44 56 l:1 r:5 pivot: -7 depth: 3 -77 -11 -7 -7 -4 -3 -1 0 13 12 44 56

		l:4 r:5 pivot: -4 depth: 4 -77 -11 -7 -7 -4 -3 -1 0 12 13 44 56 l:7 r:9 pivot: 13 depth: 3 -77 -11 -7 -7 -4 -3 -1 0 12 13 44 56 l:7 r:8 pivot: 0 depth: 4 ---END--- Result: -77 -11 -7 -7 -4 -3 -1 0 12 13 44 56

Демонстрация работы программы.

```
Введите путь до файла
test.txt
1 2 3 4 2 1
Executing algorithm...
---LOG---
1 2 3 2 1 4
 l:0 r:5 pivot: 4 depth: 0

1 2 2 1 3 4
 l:0 r:4 pivot: 3 depth: 1

1 1 2 2 3 4
 l:0 r:3 pivot: 1 depth: 2

1 1 2 2 3 4
 l:2 r:3 pivot: 2 depth: 3


---END---

Result:
1 1 2 2 3 4
```

Выводы.

Был изучен принцип трёхпутевой быстрой сортировки и реализован алгоритм этой сортировки на языке c++.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.cpp

```
#include <iostream>
#include <string>
#include <fstream>
#include <cstdlib>
#include <sstream>
using  namespace std;


struct array_list
{
    int* array;  //массив чисел
    int capacity;//размерность массива array
    int count; //количество непустых элементов масси
ва array

    void resize(int new_capacity);
    array_list(int start_capacity=1);
    int& operator[] (int index);

    void push_back(int element);
    int size();
};


void array_list::resize(int new_capacity) //изменение размерно
сти массива
{
    int *arr = new int[count];
    for (int i = 0; i < count; ++i)
    {
        arr[i] = array[i];  //переписываем значения масси
ва в вспомогательный
    }
    delete [] array;         //удаляем старый массив
    array = new int[new_capacity]; //выделяем массив нужного
размера
    for (int i = 0 ; i < count; ++i)
    {
        array[i] = arr[i];    //переписываем значения в ну
жный массив
    }
    delete [] arr;        //очищаем память под вспомогате
льный массив
    capacity = new_capacity;    //изменяем поле класса - раз
мерность массива
}

array_list::array_list(int start_capacity)  //создание списка ч
исел
{
    capacity = start_capacity;
    count = 0;
    array = new int[capacity];
}
```

```
int& array_list::operator[] (int index) //функция позволяет обращаться к элементам поля array через объект типа array_list по индексу
{
    return array[index];
}


void array_list::push_back(int element) //добавление элемента в конец массива
{
    if (capacity == count)
    {
        resize(count + 8);  //изменяем размер массива
    }
    array[count] = element;  //вставляем элемент
    count++;
}

int array_list::size() //получение длины списка
{
    return count;
}



string log(array_list &list, int min, int max, int pivot, int depth) //функция для вывода промежуточного результата
{
    string s = ""; //строка для вывода промежуточных данных
    for (int i = 0; i < list.size(); i++)
    {
        s += to_string(list[i]) + ' ';//добавлене элементов в строку
    }
    s += '\n';
    s += " l:" + to_string(min) + " r:" + to_string(max) + " pivot: " + to_string(pivot) + " depth: " + to_string(depth) +"\n\n"; //добавляем к строке промежуточные сведения
    return s;
}


void qsort3way(string& s, array_list& list, int l, int r, int depth) //функция трёхпутевой сортировки
{
    if (l >= r)
    {
        return;
    }
    int lt = l; // lt - индекс, по которому запишется очередное значение, меньше опорного
    int gt = r;  //gt - индекс, по которому запишется очередное значение, больше опорного
    int pivot = list[l + (rand() % (r - l))];//опорный элемент генерируется рандомным образом
    int i = l; //проходим по массиву слева направо
    while (i <= gt)
    {
```

```
            if (list[i] < pivot)  //если значение меньше чем опорный элемент, записываем его по индексу lt, lt смещаем вправо
            {
                int t = list[lt];
                list[lt] = list[i];
                list[i] = t;
                lt += 1;
                i += 1;
            }
            else if (list[i] > pivot)  //если значение больше чем опорный элемент, записываем его по индексу gt, gt смещаем влево
            {
                int t = list[gt];
                list[gt] = list[i];
                list[i] = t;
                gt -= 1;
            }
            else
            {
                i += 1;
            }
        }
        s += log(list, l, r, pivot, depth);
        qsort3way(s, list, l, lt - 1, depth + 1);  //рекурсивный вызов для левой части
        qsort3way(s, list, gt + 1, r, depth + 1);  //рекурсивный вызов для правой части
    }



    std::string print_list(array_list& list)  //вывод итогового результата
    {
        string s;
        for (int i = 0; i < list.size(); i++)
        {
            s += std::to_string(list[i]) + " ";
        }
        return s;
    }

    int main() {
        string input;
        ifstream file;
        string filename;
        cout << "Введите путь до файла\n";
        cin >> filename;
        file.open(filename); //открываем файл
        if (!file.is_open()){
            cout << "Файл не может быть открыт!\n";
        }
        getline(file,input); //считываем строку из файла
        file.close();           //закрываем файл
        array_list list = array_list();
        int elem;
        istringstream str(input);
        while(str >> elem) list.push_back(elem);
```

```
        cout << print_list(list);
        cout << endl << "Executing algorithm..." << endl;
        string s;            //строка для вывода промежуточных значений
        qsort3way(s, list, 0, list.size() - 1,0);
        cout << "---LOG---" << endl << s << endl << "---END---" << endl;
        cout << endl << "Result: " << endl << print_list(list) << endl;
        return 0;
    }
```