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: Методы сортировки

Студент гр. 9382 _____ Герасев Г.А.

Преподаватель _____ Фирсов М.А.

Санкт-Петербург
2020

Цель работы.

Изучить алгоритмы сортировок.

Задание.

5. Гибрид сортировки пузырьком и сортировки выбором; Шейкерная сортировка.

Основные теоретические положения.

Шейкерная сортировка.

Сортировка проходит по массиву и область сортировки уменьшается с каждой итерацией. Появляется из наблюдения, что при сортировки пузырьком один из элементов «всплывает» к верхней границе и больше не перемещается.

Сортировка проходит сначала слева направо, после чего справа налево. Оба прохода являются сортировкой пузырьком. После каждого прохода уменьшается и увеличивается одна из границ, по которой проходит сортировка.

Если интервал между границами стал <=0 или ни за один из проходов ни один из элементов не был перемещен, то сортировка прекращается.

Функции и структуры данных.

Создан класс бинарного дерева, при инициализации которого создается пустое дерево заданной глубины. Для добавления значений в узлах создаются 2 метода, для добавления значения по адресу в дереве и по адресу в массиве.

Создается метод, считающий количество переданного элемента в дереве рекурсивным методом. (очевидно проще и быстрее было бы просто пройти по массиву, хранящему значения в дереве, воспользовавшись преимуществом такой структуры дерева, но условие лабораторной запрещает не использование рекурсии). Чтобы реализовать рекурсию отслеживается номер узла в массиве, и при рекурсивном переходе этот номер меняется на номер левого и правого поддерева.

Позже из этого метода очевидно выводится метод проверки существования 2х элементов в дереве.

Описание алгоритма.

Алгоритм был описан в теоретических положениях. В коде данный алгоритм реализуется идентично, за исключением выводов промежуточных результатов.

Достоинства и недостатки алгоритма:

Алгоритм работает от O(n) в лучшем до O(n^2) в худшем и среднем случае. Алгоритм несет исключительно обучающую функцию и в реальных задачах практически не применяется.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	5 5 4 3 2 1	Input mas -- 5 4 3 2 1 Expected output is -- 1 2 3 4 5 Go on new cycle Go right Swapping 4 and 5 Now mass is 4 5 3 2 1 Swapping 3 and 5 Now mass is 4 3 5 2 1 Swapping 2 and 5 Now mass is 4 3 2 5 1 Swapping 1 and 5 Now mass is 4 3 2 1 5 Go left Swapping 1 and 2 Now mass is 4 3 1 2 5	

		Swapping 1 and 3 Now mass is 4 1 3 2 5 Swapping 1 and 4 Now mass is 1 4 3 2 5 Go on new cycle Go right Swapping 3 and 4 Now mass is 1 3 4 2 5 Swapping 2 and 4 Now mass is 1 3 2 4 5 Go left Swapping 2 and 3 Now mass is 1 2 3 4 5 Actual output -- 1 2 3 4 5 Expected and actual results are the same	
2.	6 1 1 3 2 5 8	Input mas -- 1 1 3 2 5 8 Expected output is -- 1 1 2 3 5 8 Go on new cycle Go right Swapping 2 and 3 Now mass is 1 1 2 3 5 8 Go left Go on new cycle Go right Go left Actual output -- 1 1 2 3 5 8 Expected and actual results are the same	
3.	1 100	Input mas -- 100 Expected output is -- 100 Actual output -- 100 Expexted and actual results are the same	

4.	10 0 1 10 100 1000 10000 123 321 543 32	Input mas -- 0 1 10 100 1000 10000 123 321 543 32 Expected output is -- 0 1 10 32 100 123 321 543 1000 10000 Go on new cycle Go right Swapping 123 and 10000 Now mass is 0 1 10 100 1000 123 10000 321 543 32 Swapping 321 and 10000 Now mass is 0 1 10 100 1000 123 321 10000 543 32 Swapping 543 and 10000 Now mass is 0 1 10 100 1000 123 321 543 10000 32 Swapping 32 and 10000 Now mass is 0 1 10 100 1000 123 321 543 32 10000 Go left Swapping 32 and 543 Now mass is 0 1 10 100 1000 123 321 32 543 10000 Swapping 32 and 321 Now mass is 0 1 10 100 1000 123 32 321 543 10000 Swapping 32 and 123 Now mass is 0 1 10 100 1000 32 123 321 543 10000 Swapping 32 and 1000 Now mass is 0 1 10 100 32 1000 123 321 543 10000	

		Swapping 32 and 100 Now mass is 0 1 10 32 100 1000 123 321 543 10000 Go on new cycle Go right Swapping 123 and 1000 Now mass is 0 1 10 32 100 123 1000 321 543 10000 Swapping 321 and 1000 Now mass is 0 1 10 32 100 123 321 1000 543 10000 Swapping 543 and 1000 Now mass is 0 1 10 32 100 123 321 543 1000 10000 Go left Go on new cycle Go right Go left Actual output -- 0 1 10 32 100 123 321 543 1000 10000 Expected and actual results are the same	
5.	20 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9	Input mas -- 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 Expected output is -- 0 0 1 1 2 2 3 3 4 4 5 5 6 6 7 7 8 8 9 9 Go on new cycle Go right Swapping 0 and 9 Now mass is 0 1 2 3 4 5 6 7 8 0 9 1 2 3 4 5 6 7 8 9 Swapping 1 and 9	

		Now mass is 0 1 2 3 4 5 6 7 8 0 1 9 2 3 4 5 6 7 8 9 Swapping 2 and 9 Now mass is 0 1 2 3 4 5 6 7 8 0 1 2 9 3 4 5 6 7 8 9 Swapping 3 and 9 Now mass is 0 1 2 3 4 5 6 7 8 0 1 2 3 9 4 5 6 7 8 9 Swapping 4 and 9 Now mass is 0 1 2 3 4 5 6 7 8 0 1 2 3 4 9 5 6 7 8 9 Swapping 5 and 9 Now mass is 0 1 2 3 4 5 6 7 8 0 1 2 3 4 5 9 6 7 8 9 Swapping 6 and 9 Now mass is 0 1 2 3 4 5 6 7 8 0 1 2 3 4 5 6 9 7 8 9 Swapping 7 and 9 Now mass is 0 1 2 3 4 5 6 7 8 0 1 2 3 4 5 6 7 9 8 9 Swapping 8 and 9 Now mass is 0 1 2 3 4 5 6 7 8 0 1 2 3 4 5 6 7 8 9 9 Go left Swapping 0 and 8 Now mass is 0 1 2 3 4 5 6 7 0 8 1 2 3 4 5 6 7 8 9 9 Swapping 0 and 7 Now mass is 0 1 2 3 4 5 6 0 7 8 1 2 3 4 5 6 7 8 9 9 Swapping 0 and 6	

		Now mass is 0 1 2 3 4 5 0 6 7 8 1 2 3 4 5 6 7 8 9 9 Swapping 0 and 5 Now mass is 0 1 2 3 4 0 5 6 7 8 1 2 3 4 5 6 7 8 9 9 Swapping 0 and 4 Now mass is 0 1 2 3 0 4 5 6 7 8 1 2 3 4 5 6 7 8 9 9 Swapping 0 and 3 Now mass is 0 1 2 0 3 4 5 6 7 8 1 2 3 4 5 6 7 8 9 9 Swapping 0 and 2 Now mass is 0 1 0 2 3 4 5 6 7 8 1 2 3 4 5 6 7 8 9 9 Swapping 0 and 1 Now mass is 0 0 1 2 3 4 5 6 7 8 1 2 3 4 5 6 7 8 9 9 Go on new cycle Go right Swapping 1 and 8 Now mass is 0 0 1 2 3 4 5 6 7 1 8 2 3 4 5 6 7 8 9 9 Swapping 2 and 8 Now mass is 0 0 1 2 3 4 5 6 7 1 2 8 3 4 5 6 7 8 9 9 Swapping 3 and 8 Now mass is 0 0 1 2 3 4 5 6 7 1 2 3 8 4 5 6 7 8 9 9 Swapping 4 and 8 Now mass is 0 0 1 2 3 4 5 6 7 1 2 3 4 8 5 6 7 8 9 9	

		Swapping 5 and 8 Now mass is 0 0 1 2 3 4 5 6 7 1 2 3 4 5 8 6 7 8 9 9 Swapping 6 and 8 Now mass is 0 0 1 2 3 4 5 6 7 1 2 3 4 5 6 8 7 8 9 9 Swapping 7 and 8 Now mass is 0 0 1 2 3 4 5 6 7 1 2 3 4 5 6 7 8 8 9 9 Go left Swapping 1 and 7 Now mass is 0 0 1 2 3 4 5 6 1 7 2 3 4 5 6 7 8 8 9 9 Swapping 1 and 6 Now mass is 0 0 1 2 3 4 5 1 6 7 2 3 4 5 6 7 8 8 9 9 Swapping 1 and 5 Now mass is 0 0 1 2 3 4 1 5 6 7 2 3 4 5 6 7 8 8 9 9 Swapping 1 and 4 Now mass is 0 0 1 2 3 1 4 5 6 7 2 3 4 5 6 7 8 8 9 9 Swapping 1 and 3 Now mass is 0 0 1 2 1 3 4 5 6 7 2 3 4 5 6 7 8 8 9 9 Swapping 1 and 2 Now mass is 0 0 1 1 2 3 4 5 6 7 2 3 4 5 6 7 8 8 9 9 Go on new cycle Go right Swapping 2 and 7	

		Now mass is 0 0 1 1 2 3 4 5 6 2 7 3 4 5 6 7 8 8 9 9 Swapping 3 and 7 Now mass is 0 0 1 1 2 3 4 5 6 2 3 7 4 5 6 7 8 8 9 9 Swapping 4 and 7 Now mass is 0 0 1 1 2 3 4 5 6 2 3 4 7 5 6 7 8 8 9 9 Swapping 5 and 7 Now mass is 0 0 1 1 2 3 4 5 6 2 3 4 5 7 6 7 8 8 9 9 Swapping 6 and 7 Now mass is 0 0 1 1 2 3 4 5 6 2 3 4 5 6 7 7 8 8 9 9 Go left Swapping 2 and 6 Now mass is 0 0 1 1 2 3 4 5 2 6 3 4 5 6 7 7 8 8 9 9 Swapping 2 and 5 Now mass is 0 0 1 1 2 3 4 2 5 6 3 4 5 6 7 7 8 8 9 9 Swapping 2 and 4 Now mass is 0 0 1 1 2 3 2 4 5 6 3 4 5 6 7 7 8 8 9 9 Swapping 2 and 3 Now mass is 0 0 1 1 2 2 3 4 5 6 3 4 5 6 7 7 8 8 9 9 Go on new cycle Go right Swapping 3 and 6 Now mass is 0 0 1 1 2 2 3 4 5 3 6 4 5 6	

		7 7 8 8 9 9 Swapping 4 and 6 Now mass is 0 0 1 1 2 2 3 4 5 3 4 6 5 6 7 7 8 8 9 9 Swapping 5 and 6 Now mass is 0 0 1 1 2 2 3 4 5 3 4 5 6 6 7 7 8 8 9 9 Go left Swapping 3 and 5 Now mass is 0 0 1 1 2 2 3 4 3 5 4 5 6 6 7 7 8 8 9 9 Swapping 3 and 4 Now mass is 0 0 1 1 2 2 3 3 4 5 4 5 6 6 7 7 8 8 9 9 Go on new cycle Go right Swapping 4 and 5 Now mass is 0 0 1 1 2 2 3 3 4 4 5 5 6 6 7 7 8 8 9 9 Go left Go on new cycle Go right Go left Actual output -- 0 0 1 1 2 2 3 3 4 4 5 5 6 6 7 7 8 8 9 9 Expected and actual results are the same	
6.	10 0 -1 -2 -3 -4 0 1 2 3 4	Input mas -- 0 -1 -2 -3 -4 0 1 2 3 4 Expected output is -- -4 -3 -2 -1 0 0 1 2 3 4 Go on new cycle Go right	

		Swapping -1 and 0 Now mass is -1 0 -2 -3 -4 0 1 2 3 4 Swapping -2 and 0 Now mass is -1 -2 0 -3 -4 0 1 2 3 4 Swapping -3 and 0 Now mass is -1 -2 -3 0 -4 0 1 2 3 4 Swapping -4 and 0 Now mass is -1 -2 -3 -4 0 0 1 2 3 4 Go left Swapping -4 and -3 Now mass is -1 -2 -4 -3 0 0 1 2 3 4 Swapping -4 and -2 Now mass is -1 -4 -2 -3 0 0 1 2 3 4 Swapping -4 and -1 Now mass is -4 -1 -2 -3 0 0 1 2 3 4 Go on new cycle Go right Swapping -2 and -1 Now mass is -4 -2 -1 -3 0 0 1 2 3 4 Swapping -3 and -1 Now mass is -4 -2 -3 -1 0 0 1 2 3 4 Go left Swapping -3 and -2 Now mass is -4 -3 -2 -1 0 0 1 2 3 4 Go on new cycle Go right Go left Actual output -- -4 -3 -2 -1 0 0 1 2 3 4 Expected and actual results are the same	

Выводы.

Был изучен алгоритм шейк-сортировки, была создана программа, которая создает и проверяет отсортированный массив.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.cpp

```
#include <iostream>
#include <bits/stdc++.h>
#include <string.h>

using namespace std;

// No templatres bc i made them in lab3

void greetingMessage()
{
    cout << "Please, input size of massive and data\n";
    cout << "The programm will sort data 2 different ways -- \n";
    cout << "By C++ sort and self-written shaker sort\n";
}

void copyMas(int* from, int sizeFrom, int* to, int sizeTo)
{
    // Can lead to unexpected results on different size masses
    for(int i=0; i<sizeFrom && i<sizeFrom; i++)
    {
        to[i] = from[i];
    }
}

void printMas(int* mas, int size)
{
    for (int i=0; i<size; i++)
    {
        cout << mas[i] << ' ';
    }
    cout << '\n';
}

void compareResults(int* mas1, int size1, int* mas2, int size2)
{
    // For strange test only, so output is in the function
    bool res = true;
    for(int i=0; i<size1 && i<size2; i++)
    {
        if (mas1[i] != mas2[i])
            res = false;
    }

    if (size1 != size2)
        res = false;

    if (res) {
        cout << "Expected and actual results are the same\n";
    } else {
        cout << "Expected and actual results are NOT the same\n";
    }
}
```

```
void swap(int* a, int* b)
{
   int temp = *a;
   *a = *b;
   *b = temp;
}

void shakerSort(int* mas, int size)
{
   int left = 0;
   int right = size-1;
   bool swappedOnCycle = true;

   while ((left < right) && swappedOnCycle)
   {
      cout << "Go on new cycle\n";
      swappedOnCycle = false;
      cout << "  Go right\n";
      for (int i=left; i<right; i++)
      {
         if (mas[i] > mas[i+1])
         {
            swap(&mas[i], &mas[i+1]);
            cout << "   Swapping " << mas[i] << " and " << mas[i+1] << '\n';
            cout << "   Now mass is ";
            printMas(mas, size);
            swappedOnCycle = true;
         }
      }
      right--;

      cout << "  Go left\n";
      for (int i=right; left<i; i--)
      {
         if (mas[i-1] > mas[i])
         {
            swap(&mas[i-1], &mas[i]);
            cout << "   Swapping " << mas[i-1] << " and " << mas[i] << '\n';
            cout << "   Now mass is ";
            printMas(mas, size);
            swappedOnCycle = true;
         }
      }
      left++;
   }
}

void outputResult(int* inputMas, int size)
{
   // All mass handle here
   cout << "Input mas -- ";
   printMas(inputMas, size);

   int expectedResult[size];
   copyMas(inputMas, size, expectedResult, size);
   sort(expectedResult, expectedResult+size);
   cout << "Expected output is -- ";
   printMas(expectedResult, size);
```

```
    int actualResult[size];
    copyMas(inputMas, size, actualResult, size);
    shakerSort(actualResult, size);
    cout << "Actual output -- ";
    printMas(actualResult, size);

    compareResults(expectedResult, size, actualResult, size);
}

void stdInputCase()
{
    // Standart input
    // Will continue untill size of mass <= 0
    cout << "The input will continue untill size of mass <= 0\n";
    int size;
    cin >> size;
    while (size > 0)
    {
        int inputMas[size] = { 0 };
        for (int i=0; i<size; i++)
        {
            cin >> inputMas[i];
        }

        outputResult(inputMas, size);
        cout << "-1 to stop\n";
        cin >> size;
    }
}

void fileInputCase(string path)
{
    // File input case
    ifstream inFile;
    inFile.open(path);

    int size;

    while (inFile >> size)
    {
        int mas[size];
        for (int i=0; i<size && inFile >> mas[i]; i++);
        cout << "\n\n\n";
        outputResult(mas, size);
    }
}

int main(int argc, char *argv[])
{
    if (argc>= 2) // Arguments case
    {
        string flag(argv[1]);
        string path(argv[2]);
        if (flag.compare("-f") == 0)
            fileInputCase(path); // No obvious way to overload the function
        return 0;
    }
    greetingMessage();
    stdInputCase();
```

```
    return 0;
}
```