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. 9382 _____ Иерусалимов Н.

Преподаватель _____ Фирсов М.А.

Санкт-Петербург

2020

Цель работы.

Разобраться в представленных ниже сортировках, понять чем они лучше своего прародителя . Рассмотреть разные варианты сортировок.

Задание.

Вариант 4.

4. Пузырьковая сортировка оптимизированная; сортировка чёт-нечет.

Описание основных функций.

void printDep(int depth)

Назначение: Выводит глубину работы алгоритма.

Int depth: Глубина погружениясортировки.

template<typename T> void BubleSort(T &array, size_t n)

Назначение: Шаблонный метод сортирует оптимизированной сортировкой.

T &array: шаблонная ссылка на массив элементов который надо отсортировать .

Size_t n: количество элементов в массиве.

template<typename T> void oddEvenSorting(T *array, size_t N)

Назначение: Производит сортировку элементов массива с помощью сортировки чет\нечет.

T &array: шаблонная ссылка на массив элементов который надо отсортировать .

Size_t N: количество элементов в массиве.

void writeToFile(string filename,int arg)

Назначение: Записывает в файл переданное ему число.

string filename: Куда записывать

int arg: Что записывать

void writeToFile(string filename,string arg1)

Назначение: Записывает в файл переданное ему строку.

string filename: Куда записывать

string arg1: Что записывать

void CheckTheInput(int* arr, bool choise, int len, bool choiseSort)

Назначение: Проверяет массив на соответствие сортировке.

int* arr:Массив для проверки.

bool choise: Запись в файл или в консоль. 1 – консоль 0 – в файл

int len: Длина массива.

bool choiseSort: Выбор сортировки, 1 – чет\нечет 0 – оптимизированая

int comp (const int *i, const int *j)

Назначение: компоратор для qsort.

const int *i: указатель на число от которого будем отнимать чтобы узнать какое меньше.

const int *j: указатель на число которое будем отнимать.

Возвращает: большое число из этих двух

Описание алгоритма.

После считывания массива, выбора сортировки и отлова ошибок на подобии пустого массива или одного элемента в нем запускается алгоритм сортировки. Есть два алгоритма сортировки чет\нечет(1) и оптимизированная сортировка пузырьком(2).

1)Алгоритм чет\нечет:

- на каждом проходе производится n/2 независимых сравнений соседних пар, так что никакой элемент пары не участвует в дальнейших сравнениях на данном проходе;

- Проходы делятся на четные и нечетные. На четных проходах обмен начинается с пары (a n-1,a n-2). На нечетном проходе производится сдвиг и начальной парой является пара (a n-2,a n-3). (Предполагается, что нумерация элементов массива начинается с нуля).

представляет вариацию алгоритма пузырьковой сортировки. В отличие от "пузырька", где на i -м проходе первые i элементов занимают свои места, в алгоритме "чет - нечет" элементы гарантировано занимают свои места после выполнения всех n проходов.

Для самого легкого элемента достаточно n - 1 проход для "всплытия" в вершину массива, так как на каждом проходе элемент поднимается вверх на одну позицию. Для следующего за ним элемента может понадобиться в самом неблагоприятном случае ровно N проходов.

2)Алгоритм оптимизированной сортировки пузырьком:

Алгоритм пузырьком состоит в повторяющихся проходах по сортируемому массиву. На каждой итерации последовательно сравниваются соседние элементы, и, если порядок в паре неверный, то элементы меняют местами. За каждый проход по массиву как минимум один элемент встает на свое место, поэтому необходимо совершить не более $n-1$ проходов, где n размер массива, чтобы отсортировать массив. Оптимизировать ее можно 2 способами:

1) Можно заметить, что после i-ой итерации внешнего цикла i последних элементов уже находятся на своих местах в отсортированном порядке, поэтому нет необходимости производить их сравнения друг с другом. Следовательно, внутренний цикл можно выполнять не до $n-2$, а до $n-i-2$.

2) Также заметим, что если после выполнения внутреннего цикла не произошло ни одного обмена, то массив уже отсортирован, и

продолжать что-то делать бессмысленно. Поэтому внутренний цикл можно выполнять не n−1 раз, а до тех пор, пока во внутреннем цикле происходят обмены.

Применив оба способа мы и получаем оптимизированную сортировку пузырьком

Пример работы программы.

Таблица 1 – Пример работы

Вид сортировки	Входные данные	Выходные данные
Bubble sort optimized	961	Sort to 2 index /\array[0] > array[1]: 9 > 6 True /\array{9, 6, 1, } /\swap: array[0] -> array[1] /\new array{6, 9, 1, } Sort to 2 index /\array[1] > array[2]: 9 > 1 True /\array{6, 9, 1, } /\swap: array[1] -> array[2] /\new array{6, 1, 9, } Sort to 1 index /\/\array[0] > array[1]: 6 > 1 True /\/\array{6, 1, 9, } /\/\swap: array[0] -> array[1] /\/\new array{1, 6, 9, } Tumber of rounds->3 Sorted array is: 1 6 9 Sorted by qsort:1 6 9
Odd\Even	961	What indexes can we go by: 1) even/odd 2) odd/even

		Sorting start! _____ /\Split array into *odd/even* subgroups by index /\array[1] > array[2]: 6 > 1 True /\array{9, 6, 1, } /\swap: array[1] -> array[2] /\new array{9, 1, 6, } /\/\Split array into *even/odd* subgroups by index /\/\array[0] > array[1]: 9 > 1 True /\/\array{9, 1, 6, } /\/\swap: array[0] -> array[1] /\/\new array{1, 9, 6, } /\/\/\Split array into *odd/even* subgroups by index /\/\/\array[1] > array[2]: 9 > 6 True /\/\/\array{1, 9, 6, } /\/\/\swap: array[1] -> array[2] /\/\/\new array{1, 6, 9, } Tumber of rounds->3 Sorted array is: 1 6 9 Sorted by qsort:1 6 9

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 — Результаты тестирования

№	Входные данные	Выходные данные	Доп.Сообщения
1.	1	1	In array one element : 1
2.	963	369	Amount of elements: 3
3.	2 -6 -7 -2 1	-7 -6 -2 1 2	Amount of elements: 5
4.	“Enter”		Empty array!
5.	987654 87 65 54 32 21 10 12 11 0	0 10 11 12 21 32 54 65 87 987654	Amount of elements: 9
6.	n\	Eror: Only digit!	Process finished with exit code 0
7.	стрпо	Eror: Only digit!	Process finished with exit code 0

Для обоих типов сортировки результат тестирования один и тот же.

Выводы.

Разобрались в сортировках чет\нечет и оптимизированной пузырьковой, посмотрели чем они лучше своего прародителя . И рассмотрели разные варианты сортировки пузырьком.

ПРИЛОЖЕНИЕ С КОДОМ

Название файла:main.cpp

```
#include <windows.h>
#include <iostream>
#include <fstream>
#include <algorithm>
#include <vector>
#include <ctime>
#include <conio.h>
using namespace std;
void printDep(int depth){
    for(int i = 0; i < depth; ++i){
        cout<<"/\\";
    }
}

template < typename T>
void BubleSort(T &array, size_t n){
    int i = 0;
    bool t = true;
    int depth = 0;
    while (t) {
        t = false;
        ++depth;
        for(int j = 0; j < n-i-1;++j){
            cout<<"Sort to "<<n-i-1<<" index\n";
            if (array[j] > array[j + 1]){
                printDep(depth);
                cout<<"array["<<j<<"] > array["<<j+1<<"]: "<< array[j] <<"
> "<<array[j + 1]<<" True\n";
                printDep(depth);
                cout<<"array{";
                for(int out =0; out<n;++out){
                    cout<<array[out]<<", ";
                }
```

```
                cout<<"}\n";
                swap(array[j], array[j + 1]);
                printDep(depth);
                cout<<"swap: array["<<j<<"] -> array["<<j+1<<"]\n";
                printDep(depth);
                t = true;
                cout<<"new array{";
                for(int out =0; out<n;++out){
                    cout<<array[out]<<", ";
                }
                cout<<"}\n\n";
            }else {
                printDep(depth);
                cout<<"array["<<j<<"] > array["<<j+1<<"]: "<< array[j] <<"
> "<<array[j + 1]<<" Lie\n";
                printDep(depth);
                cout<<"move on!\n\n";
            }
        }
        i = i + 1;
    }
    cout<<"Tumber of rounds->"<<depth<<'\n';
}


template < typename T>
void oddEvenSorting(T &array, size_t N) {
    cout<<"\nSorting
start!\n_____________________________________________________________________\n";

    int depth = 0;
    for (size_t i = 0; i < N; i++) {
        ++depth;
        // (i % 2) ? 0 : 1 ?????????? 1, ???? i ??????, 0, ???? i ??
??????

        bool split = (i % 2) ? 0 : 1;
        for (size_t j = (i % 2) ? 0 : 1; j + 1 < N; j += 2) {

            if (array[j] > array[j + 1]) {
                printDep(depth);
                if(!split){
                    cout<< "Split array into *even/odd* subgroups by
index\n";
                }else {
                    cout<< "Split array into *odd/even* subgroups by
index\n";
                }

                printDep(depth);
                cout<<"array["<<j<<"] > array["<<j+1<<"]: "<< array[j] <<"
```

```
> "<<array[j + 1]<<" True\n";

                printDep(depth);
                cout<<"array{";
                for(int out =0; out<N;++out){
                    cout<<array[out]<<", ";
                }
                cout<<"}";
                cout<<'\n';
                std::swap(array[j], array[j + 1]);
                printDep(depth);
                cout<<"swap: array["<<j<<"] -> array["<<j+1<<"]\n";
                printDep(depth);
                cout<<"new array{";
                for(int out =0; out<N;++out){
                    cout<<array[out]<<", ";
                }
                cout<<"}";
                cout<<'\n';
                cout<<'\n';
            }else{
                printDep(depth);
                if(!split){
                    cout<< "Split array into *even/odd* subgroups by
index\n";
                }else {
                    cout<< "Split array into *odd/even* subgroups by
index\n";
                }
                printDep(depth);
                cout<<"array["<<j<<"] > array["<<j+1<<"]: "<< array[j] <<"
> "<<array[j + 1]<<" Lie\n";
                printDep(depth);
                cout<<"move on!\n\n";
            }
        }
    }
    cout<<"Tumber of rounds->"<<depth<<'\n';
}

void writeToFile(string filename,int arg) {
    ofstream output;
    output.open(filename, ios::app);
    output << arg;
    output.close();
}
void writeToFile(string filename,string arg1) {
    ofstream output;
    output.open(filename, ios::app);
    output << arg1;
    output.close();
```

```
}

void CheckTheInput(int* arr, bool &Exit, bool choise, int len, bool
choiseSort){
    if (len == 1 && arr[0] == '!') {
        Exit = false;
    } else if (len == 1) {
        cout << "In array one element : " << arr << "\n\n\n";
    } else if (len == 0) {
        cout << "\nAmount of elements: " << len << "\n";
        cout << "Empty array!\n\n\n";
    } else {
        if(choiseSort) {
            cout << "\nAmount of elements: " << len << '\n';
            cout << "What indexes can we go by:\n1) even/odd\n2)
odd/even\n";
            oddEvenSorting(arr, len);
        }else BubleSort(arr,len);
        if(!choise) {
            writeToFile("output.txt","\nSorted array is: ");
            for(int i=0;i<len;++i) {
                writeToFile("output.txt", arr[i]);
                writeToFile("output.txt", " ");
            }
        }
        cout << "Sorted array is: ";
        for(int i =0; i < len; ++i)cout<< arr[i]<<' ';


    }
}
int comp (const int *i, const int *j)
{
    return *i - *j;
}

int main() {
    bool Exit = true;
    bool choiseFileOrConsole = true;
    bool choiseSort;
    int len=0;
    int* copyArray;
    cout<<"Choise the sort, 1 - odd/even \t 0 - Bubble sort optimized;\n";
    cin >> choiseSort;
    cout << "Enter : 0 - File Input, 1 - Console input: ";
    cin >> choiseFileOrConsole;
    cout <<'\n';
    if(!choiseFileOrConsole){
        string input_filename;
        const string output_filename = "output.txt";
        ifstream in;
```

```
        ofstream out;

        out.open(output_filename);
        out << "";
        out.close();

        cout << "Enter the input file name: \n\n";
        cin >> input_filename;
        in.open(input_filename);

        if (in.is_open()) {
            in >> len;
            int* some = new int[len];
            copyArray = new int[len];
            writeToFile("output.txt", "Initial array: ");
            for(int i = 0; i < len; ++i){
                in >> some[i];
                writeToFile("output.txt", some[i]);
                writeToFile("output.txt", " ");
                copyArray[i] = some[i];
            }
            CheckTheInput(some,Exit,choiseFileOrConsole,len, choiseSort);

        }
        else {
            cout << input_filename << " doesn't exist!\n";
        }

    }else {
        cout<<"Array length: ";
        cin>>len;
        int* elem = new int[len];
        copyArray = new int[len];
        cout << "\nInput data or \'!\' to quit: \n";
        for(int i=0; i < len; ++i){
            cout<<i<<" elem: ";
            std::cin >> elem[i];
            copyArray[i] = elem[i];
            cout<<'\n';
        }
        CheckTheInput(elem, Exit,choiseFileOrConsole, len,choiseSort);


    }
    qsort(copyArray, len, sizeof(int),(int(*) (const void *, const void
*)) comp);
    cout<<"\nSorted by qsort:";
    for(int i = 0; i <len; ++i)cout<<copyArray[i]<<" ";
    cout << "\n_____________________________________________\n\n";
    getch();
    return 0;
```

```
}
```