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

ОТЧЕТ

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

Студент гр. 9382 _____ Сорокумов С. В.

Преподаватель _____ Фирсов М. А.

Санкт-Петербург
2020

Цель работы.

Реализовать сортировку Шелла, разобраться в её работе, а также изучить её положительные и отрицательные стороны.

Задание.

Вариант 18. Сортировка Шелла.

Описание алгоритма.

Изначально программа запрашивает у пользователя откуда он хочет подать данные из файла или из консоли, после чего происходит считывание данных и запись их в массив, затем программа создает вектор, который дублирует массив для сортировки шелла, это необходимо для стандартной сортировки, которая будет выполнена уже после сортировки Шелла. После считывания информации, вызывается сортировка Шелла, затем выводится результат её работы. Аналогично происходит со стандартной сортировкой.

Идея сортировки Шелла заключается в сравнение разделенных на группы элементов последовательности, находящихся друг от друга на некотором расстоянии. Изначально это расстояние равно **d** или **N/2**, где **N** — общее число элементов. На первом шаге каждая группа включает в себя два элемента расположенных друг от друга на расстоянии **N/2**; они сравниваются между собой, и, в случае необходимости, меняются местами. На последующих шагах также происходят проверка и обмен, но расстояние **d** сокращается на **d/2**, и количество групп, соответственно, уменьшается. Постепенно расстояние между элементами уменьшается, и на **d=1** проход по массиву происходит в последний раз.

Сортировка Шелла во многих случаях медленнее, чем быстрая сортировка, она имеет ряд преимуществ:

- отсутствие потребности в памяти под стек;

- отсутствие деградации при неудачных наборах данных — быстрая сортировка легко деградирует до $O(n^2)$, что хуже, чем худшее гарантированное время для сортировки Шелла.

В лучшем случае сортировка Шелла выполнится за O(n log(n)), тогда как в худшем случае только за $O(n^2)$.

Выбор длин промежутков.

Среднее время работы алгоритма зависит от длин промежутков — d, на которых будут находиться сортируемые элементы исходного массива ёмкостью N на каждом шаге алгоритма. Существует несколько подходов к выбору этих значений:

Первоначально используемая Шеллом последовательность длин промежутков: d1 = N/2, di = d(i-1)/2, dk = 1 в худшем случае, сложность алгоритма составит O(N^2);

Предложенная Хиббардом последовательность: все значения 2^i - 1 <= N; такая последовательность шагов приводит к алгоритму сложностью O(N^(3/2));

Предложенная Седжвиком последовательность: di = 9*2^i - 9*2^(i/2) + 1, если i четное и di = 8*2^i - 6*2^(i+1)/2 + 1, если i нечетное. При использовании таких приращений средняя сложность алгоритма составляет: O(N^(7/6)), а в худшем случае порядка O(N^(4/3)).

Описание функций алгоритма.

`void Shell(T* A, int n)` – Функция сортировки методом Шелла, принимает на вход:

`T* A` – шаблонный массив данных, который необходимо отсортировать.

`int n` – длина поданного массива.

Тестирование.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	После сортировки Шелла	После стандартной сортировки
1.	3 3 2 1	1 2 3	1 2 3
2.	1 1	1	1
3.	10 10 9 8 7 6 5 4 3 2 1	1 2 3 4 5 6 7 8 9 10	1 2 3 4 5 6 7 8 9 10
4	0 10 9 8 7 6 5 4 3 2 1	Некорректная длина	
5	10 -1 -2 -3 -4 -5 -6 -7 -8 -9 -10	-10 -9 -8 -7 -6 -5 -4 -3 -2 -1	-10 -9 -8 -7 -6 -5 -4 -3 -2 -1
6	10 1 1 1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1 1 1
7	7 5 7 3 99 56 43 81	3 5 7 43 56 81 99	3 5 7 43 56 81 99
	10 85 354 999 0 6 5 7 9 55 100	0 5 6 7 9 55 85 100 354 999	0 5 6 7 9 55 85 100 354 999

Выводы.

В ходе выполнения лабораторной работы была реализована сортировка Шелла, так же было приведено ее описание и положительные и отрицательные стороны.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.cpp

```
#include <windows.h>
#include <iostream>
#include <fstream>
#include <algorithm>
#include <vector>

template <typename T>
void shell(T* A, int n){ //сортировка Шелла
    int count, d;
    d=n/2; // Присвоение в d половины длины массива
    while (d>0){ // выполнение цикла пока d > 0
        for (int i=0; i<n-d; i++){
            std::cout << std::endl;
            std::cout << "Passage " << i+1  << ", d = " << d << std::endl;
            for (int j = i; j >= 0; j--){
                std::cout << "Comparison of elements " << A[j] << " и " <<
A[j + d] << std::endl;
                if (A[j]>A[j+d]){ // сравнение двух элементов
                    /*
                     * Замена элементов если условие верно
                     */
                    std::cout << "Replacing elements " << A[j] << " и " <<
A[j + d] << std::endl;
                    count = A[j];
                    A[j] = A[j + d];
                    A[j + d] = count;
                    std::cout << "Array after replacement: ";
                    for (int k = 0; k < n; k++){
                        std::cout << A[k] << ' ';
                    }
                    std::cout << std::endl<<std::endl;
                }
                else{
                    std::cout   << A[j] << " и " << A[j + d] << " no
replacement required " << std::endl;
                }
                //std::cout << std::endl;
            }
        }
        d=d/2;
    }
}

//главная функция

int main(){
```

```
    SetConsoleOutputCP(CP_UTF8);
    int n;
    bool file;
    std::cout << "Write f - if reading from file, any other - if console"
<< std::endl;
    char symbol;
    std::cin >> symbol;
    std::string input;
    /*
     * Считывание название файла из командной строки, после чего
     * Считывание размера массива из файла
     */
    if (symbol == 'f'){
        std::cout << "Enter the path to the file:" << std::endl;
        std::cin >> input;
        std::ifstream fin;
        fin.open(input);
        if (!fin){
            std::cout << "error" << std::endl;
            system("pause");
            std::cout << "To continue the program, press any key ...";
            std::cin.get();
            exit(1);
        }
        fin >> n;
    } else {
        /*
         * Считывание размера массива из консоли
         */
        std::cout<<"Array size > " ;
        std::cin>>n;
        std::cout << std::endl;
    }
    if (n <= 0){
        std::cout << "Incorrect length" << std::endl;
        system("pause");
        std::cout << "To continue the program, press any key ...";
        std::cin.get();
        exit(1);
    }

    int* arrayForShell= new int[n]; //объявление динамического массива
    std::vector<int> arrayForStdSort; // создание вектора для стандартной
сортировки

    /*
     * Считывание массива из файла
     */
    if (symbol == 'f'){
        std::ifstream fin;
        fin.open(input);
```

```
        fin >> n;
        for (int i=0; i<n; i++){
            fin>>arrayForShell[i];
        }
        std::cout << "The array fed: ";
        for (int i=0; i<n; i++){
            std::cout << arrayForShell[i] << ' ';
        }
        std::cout << std::endl;
    } else{
        /*
        * Считывание массива из консоли
        */
        for (int i=0; i<n; i++){
            std::cout<<i+1<<"     item     >     "     <<     std::endl;
std::cin>>arrayForShell[i];
        }

    }
    /*
     * Занесение элементов в массив для стандартной сортировки
     */
    for (int i = 0; i < n; i++){
        arrayForStdSort.push_back(arrayForShell[i]);
    }

    std::cout << std::endl << "Shell sort call" << std::endl << std::endl;
    shell(arrayForShell, n); //Вызов сортировки Шелла

    /*
     * Вывод отсортированного массива
     */
    std::cout<<"Resulting array: ";
    for (int i=0; i<n; i++) {
        std::cout<<arrayForShell[i]<<" "; //вывод массива
    }
    std::cout << std::endl;
    std::cout << std::endl << "Calling the standard sort " << std::endl;
    std::sort(arrayForStdSort.begin(),  arrayForStdSort.end());  //Вызов
стандартной сортировки
    std::cout<<"Array after standard sort: ";
    for (int i=0; i<n; i++) {
        std::cout<<arrayForStdSort[i]<<" "; //вывод массива
    }
    std::cout << std::endl;
    delete [] arrayForShell; //освобождение памяти
    system("pause");
    std::cout << "To continue the program, press any key ...";
    std::cin.get();
    return 0;
}
```