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

ОТЧЕТ

По лабораторной работе №4
по дисциплине «Алгоритмы и структуры данных»
Тема: **Сортировки**

Студент гр. 9381 Судаков Е.В.

Преподаватель _____ Фирсов М.А.

_____

Санкт-Петербург
2020

1. Цель работы.

Познакомиться с принципами сортировок в компьютерных науках.

2. Задание.

Вариант 16.

16. Сортировка массивов слиянием – естественное слияние.

3. Основные теоретические положения.

Сортировка слиянием (англ. merge sort) — алгоритм сортировки, который упорядочивает списки (или другие структуры данных, доступ к элементам которых можно получать только последовательно, например — потоки) в определённом порядке. Эта сортировка — хороший пример использования принципа «разделяй и властвуй». Сначала задача разбивается на несколько подзадач меньшего размера. Затем эти задачи решаются с помощью рекурсивного вызова или непосредственно, если их размер достаточно мал. Наконец, их решения комбинируются, и получается решение исходной задачи.

Для алгоритма сортировки слиянием есть очень простая и эффективная модификация, которая называется «**естественная сортировка**» («Natural Sort»). Суть её в том, что нужно образовывать цепочки, и производить их слияние не в заранее определённом и фиксированном порядке, а анализировать имеющиеся в массиве данные.

4. Описание алгоритма

Алгоритм следующий:

1. разбить массив на цепочки уже отсортированных элементов (в худшем случае, когда элементы в массиве идут по убыванию, все цепочки будут

состоять из одного элемента);

2. произвести слияние цепочек по две.

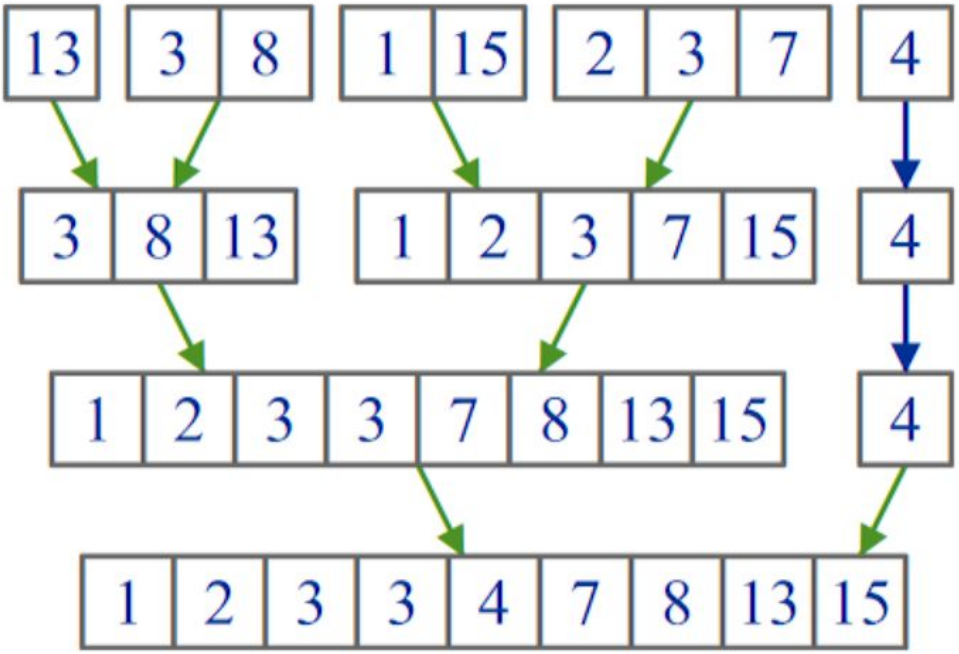

Рисунок 1. Пример работы алгоритма

4. Пример работы программы:

Основной тест №1:

Входные данные: 2 3 17 7 8 9 1 4 6 9 2 3 1 18

Выходные данные (с промежуточной информацией):

```
====================================
Iteration 1
Array :: 2 3 17 7 8 9 1 4 6 9 2 3 1 18

First series : 2 3 17
Second series : 7 8 9
Array before merging :: 2 3 17 7 8 9 1 4 6 9 2 3 1 18
merged : 2 3 7 8 9 17
Array after merging :: 2 3 7 8 9 17 1 4 6 9 2 3 1 18
```

First series : 1 4 6 9

Second series : 2 3

Array before merging :: 2 3 7 8 9 17 1 4 6 9 2 3 1 18

merged : 1 2 3 4 6 9

Array after merging :: 2 3 7 8 9 17 1 2 3 4 6 9 1 18

==========

==========

Iteration 2

Array :: 2 3 7 8 9 17 1 2 3 4 6 9 1 18

First series : 2 3 7 8 9 17

Second series : 1 2 3 4 6 9

Array before merging :: 2 3 7 8 9 17 1 2 3 4 6 9 1 18

merged : 1 2 2 3 3 4 6 7 8 9 9 17

Array after merging :: 1 2 2 3 3 4 6 7 8 9 9 17 1 18

==========

==========

Iteration 3

Array :: 1 2 2 3 3 4 6 7 8 9 9 17 1 18

First series : 1 2 2 3 3 4 6 7 8 9 9 17

Second series : 1 18

Array before merging :: 1 2 2 3 3 4 6 7 8 9 9 17 1 18

merged : 1 1 2 2 3 3 4 6 7 8 9 9 17 18

Array after merging :: 1 1 2 2 3 3 4 6 7 8 9 9 17 18

```
====================================

====================================

Iteration 4

Array :: 1 1 2 2 3 3 4 6 7 8 9 9 17 18

====================================



Data sorted by naturalMergeSort :: 1 1 2 2 3 3 4 6 7 8 9 9 17 18

Data sorted by std::sort :: 1 1 2 2 3 3 4 6 7 8 9 9 17 18

Arrays are equal


Process finished with exit code 0
```

Дополнительное тестирование :

Номер теста	Входные данные	Результат
2	5 4 3 2 0	Data sorted by naturalMergeSort :: 0 2 3 4 5 Data sorted by std::sort :: 0 2 3 4 5 Arrays are equal
3	a c d q equ jmp cmp ig	Data sorted by naturalMergeSort :: a c cmp d equ ig jmp q Data sorted by std::sort :: a c cmp d equ ig jmp q Arrays are equal
4	b a q c 0 /	Data sorted by naturalMergeSort :: / 0 a b c q Data sorted by std::sort :: / 0 a b c q Arrays are equal
5	Введите тип массива : (1) Integer (2) Float (3) String (4) Char **1** Введите, пожалуйста, количество элементов в массиве 2 Массив из консоли или из файла (0/1)?	Data sorted by naturalMergeSort :: 0 0 Data sorted by std::sort :: 0 0 Arrays are equal

	0 Введите массив : hello world	

4. Выполнение программы:

1. Ввести тип данных из предложенных программой
2. Ввести количество элементов для сортировки
3. Для ввода информации из файла необходимо ввести “1” на вопрос программы “Строка из консоли или из файла (0/1)?”.
4. Для ввода информации через консоль необходимо ввести “0” на вопрос программы “Строка из консоли или из файла (0/1)?”.

Рекомендуется работать только с консолью, так как она позволяет использовать цвета, что в данной работе активно используется при выводе информации.

5. Описание функций:

Функции описаны в исходном коде в стиле Javadoc:

```
template<typename T>

void printArray(vector<T>vec, int start = 0, int end = inf)
```

- Функция вывода шаблонного вектора.

@tparam T тип данных

@param vec вектор для вывода

@param start оптимальный старт

@param end оптимальный конец

template<typename T>

void printInfo(vector<T> &arr, int start, vector<T>first, vector<T>second) - Функция вывода некоторой промежуточной информации.

@tparam T тип данных

@param arr вектор, куда происходит слияние

@param start стартовая позиция для записи

@param first первая серия для слияния

@param second вторая серия для слияния

template<typename T> void merge(vector<T> &arr, int start, vector<T>first, vector<T>second) - Функция слияния двух серий в исходный массив.

@tparam T тип данных

@param arr вектор, куда происходит слияние

@param start стартовая позиция для записи

@param first первая серия для слияния

@param second вторая серия для слияния

template<typename T> void naturalMergeSort(vector<T> &arr) - Функция сортировки массивов слиянием – естественное слияние.

@tparam T тип данных

@param arr вектор для сортировки

`template<typename T> void solution(vector<T> data)` - Программа запуска и сравнения двух сортировок.

@tparam T тип вектора для сортировки

@param data исходные данные(вектор)

`template<typename T> void readArray(vector<T> &arr)` - Функция вывода шаблонного вектора.

@tparam T Тип элементов вектора

@param arr собственно вектор для вывода

7. Вывод:

В ходе выполнения лабораторной работы была изучена и реализована на языке C++ модификации сортировки слиянием - естественное слияние.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Файл main.cpp:

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


#ifdef __linux__
    #define FIRST_COLOR   "\033[1m\033[31m"
    #define FAILURE_COLOR "\033[1m\033[31m"
    #define SUCCESS_COLOR "\033[1m\033[32m"
    #define SECOND_COLOR  "\033[1m\033[32m"
    #define INFO_COLOR    "\033[1m\033[36m"
    #define MERGED_COLOR  "\033[1m\033[33m"
    #define ITERATION_COLOR    "\033[1m\033[35m"


#elif __WIN32
    #define FIRST_COLOR   ""
    #define FAILURE_COLOR ""
    #define SUCCESS_COLOR ""
    #define SECOND_COLOR  ""
    #define INFO_COLOR    ""
    #define MERGED_COLOR  ""
    #define ITERATION_COLOR    ""
#endif


#define inf 1e5



//16. Сортировка массивов слиянием – естественное слияние.


using namespace std;
```

```
/**
 * Функция вывода шаблонного вектора
 * @tparam T тип данных
 * @param vec вектор для вывода
 * @param start оптимальный старт
 * @param end оптимальный конец
 */
template<typename T>
void printArray(vector<T>vec, int start = 0, int end = inf) {
    int startInd = max(start, 0);
    int endInd = min(end, int(vec.size()));
    for(int i = startInd; i < endInd; i++) cout << vec[i] << " ";
}



/**
 * Функция вывода некоторой промежуточной информации
 * @tparam T тип данных
 * @param arr вектор, куда происходит слияние
 * @param start стартовая позиция для записи
 * @param first  первая серия для слияния
 * @param second  вторая серия для слияния
 */
template<typename T>
void printInfo(vector<T> &arr, int start, vector<T>first, vector<T>second) {
    cout << FIRST_COLOR << "\nFirst series : ";
    printArray(first);
    cout << "\n";
    cout << SECOND_COLOR << "Second series : ";
    printArray(second);
```

```
    cout << "\n";
    cout << INFO_COLOR << "Array before merging :: ";
    printArray(arr, 0, start);
    cout << FIRST_COLOR;
    printArray(first);
    cout << SECOND_COLOR;
    printArray(second);
    cout << INFO_COLOR;
    printArray(arr, start + first.size() + second.size(), arr.size());
    cout << MERGED_COLOR << "\nmerged : ";
}



/**
 * Функция слияния двух серий в исходный массив
 * @tparam T тип данных
 * @param arr вектор, куда происходит слияние
 * @param start стартовая позиция для записи
 * @param first  первая серия для слияния
 * @param second  вторая серия для слияния
 */
template<typename T>
void merge(vector<T> &arr, int start, vector<T>first, vector<T>second) {
    printInfo(arr, start, first, second);
    int pos1(0), pos2(0), n1 = first.size(), n2 = second.size();
    while(pos1 + pos2 < n1 + n2) {
        if(pos2 >= n2 || (pos1 < n1 && first[pos1] < second[pos2])) {
            arr[start + pos1 + pos2] = first[pos1];
            cout << first[pos1] << " ";
            pos1++;
        }
```

```
        else {
            arr[start + pos1 + pos2] = second[pos2];
            cout << second[pos2] << " ";
            pos2++;
        }
    }
    cout << "\n" <<INFO_COLOR << "Array after merging :: ";
    printArray(arr);
    cout << "\n\n";
}


/**
 * Функция сортировки массивов слиянием – естественное слияние.
 * @tparam T тип данных
 * @param arr вектор для сортировки
 */
template<typename T>
void naturalMergeSort(vector<T> &arr){
    int n = arr.size();
    int cnt = 1;
    while(true)  {
        int start1 = 1, start2 = -1, end1, end2 = 0;
        vector<T>s1, s2;
        cout << ITERATION_COLOR << "=====================================\n";
        cout << "Iteration " << cnt++ << INFO_COLOR << "\nArray :: ";
        printArray(arr);
        cout << "\n";
        while(true) {
            s1.clear(); s2.clear();
            start1 = end2; end1 = start1;
```

```
            s1.push_back(arr[end1++]); // первая серия
            while(end1 < n && arr[end1] >= arr[end1-1]){
                s1.push_back(arr[end1++]);
            }

            if(end1 == n)
                break;

            start2 = end1; end2 = start2;

            s2.push_back(arr[end2++]); // вторая серия
            while(end2 < n && arr[end2] >= arr[end2-1]) {
                s2.push_back(arr[end2++]);
            }

            merge(arr, start1, s1, s2);

            if(end2  == n) break;
        }
        cout << ITERATION_COLOR << "=====================================\n";
        if(start1 == 0 && end1 == n) break;
    }
}



/**
 * Программа запуска и сравнения двух сортировок
 * @tparam T тип вектора для сортировки
 * @param data исходные данные(вектор)
 */
```

```
template<typename T>

void solution(vector<T> data) {

    vector<T>stdSortData = data;

    naturalMergeSort(data);

    cout << INFO_COLOR <<  "\n\nData sorted by " <<  ITERATION_COLOR << "naturalMergeSort
:: " << MERGED_COLOR;

    printArray(data);

    cout << INFO_COLOR << "\nData sorted by" << ITERATION_COLOR << " std::sort :: " <<
MERGED_COLOR;

    sort(stdSortData.begin(), stdSortData.end());

    printArray(stdSortData);

    if(data == stdSortData) {

        cout << SUCCESS_COLOR << "\nArrays are equal\n";

    }

    else {

        cout << FAILURE_COLOR << "\nArrays are not equal\n";

    }

    assert(data == stdSortData);

}




/**

 * Функция вывода шаблонного вектора

 * @tparam T Тип элементов вектора

 * @param arr собственно вектор для вывода

 */

template<typename T>

void readArray(vector<T> &arr) {

    for(auto &it : arr) cin >> it;

}
```

```
int main() {
    int type;
    cout << "Введите тип массива :\n"
            "(1) Integer\n"
            "(2) Float\n"
            "(3) String\n"
            "(4) Char\n";
    cin >> type;
    vector<int>iv;
    vector<float>fv;
    vector<string>sv;
    vector<char>cv;
    int n;
    cout << "Введите, пожалуйста, количество элементов в массиве\n";
    cin >> n;
    int f;
    cout << "Массив из консоли или из файла (0/1)?\n";
    cin >> f;
    if (f == 1) {
        freopen("input.txt", "r", stdin);
        //Если работать на windows, то можно использовать вывод в файл. Если нет, то в
файл
        //будет выводить абракадабра с цветами.
//        freopen("output.txt", "w", stdout);
    }
    else cout << "Введите массив :\n";
    switch (type) {
        case 1:
            iv = vector<int>(n);
            readArray(iv);
            solution(iv);
            break;
```

```
        case 2:
            fv = vector<float>(n);
            readArray(fv);
            solution(fv);
            break;
        case 3:
            sv = vector<string>(n);
            readArray(sv);
            solution(sv);
            break;
        case 4:
            cv = vector<char>(n);
            readArray(cv);
            solution(cv);
            break;
    }
    return 0;
}
```