

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

Камакин Д.В.

Преподаватель

Фирсов М.А.

Санкт-Петербург

2020

Цель работы

Познакомиться с различными видами сортировки, реализовать бинго-сортировку.

Постановка задачи

Вариант 6.

Бинго-сортировка.

Описание алгоритма

Бинго-сортировка является улучшением классической сортировки выбором (ищем максимум в массиве, меняем его с последним элементом и проделываем то же самое с неотсортированной частью), которая учитывает не уникальные элементы. Такая сортировка имеет ряд преимуществ:

1. Отсутствует необходимость в памяти под стек.
2. Скорость при работе с массивами из не уникальных элементов значительно выше, чем у классической сортировки выбором.

Однако данная сортировка все равно не слишком быстрая и сильно зависит от вводимых данных:

Сложность в лучшем сценарии: $O(n+m^2)$, где n — общее количество элементов, m — количество уникальных значений

В худшем/среднем: $O(nm)$

Модификация заключается в том, что неупорядоченной части необходимо запоминать не только максимальный элемент, но и максимум для следующей итерации. Тогда, при наличии нескольких максимумов, не потребуются лишние итерации.

В начале программы пользователю предлагается выбрать способ считывания массива (клавиатура/файл). После считывания создаётся вектор, аналогичный данным пользователя, который сортируется при помощи стандартной сортировки `std::sort` для сравнения результата в дальнейшем.

Массив пользователя сортируется при помощи функции `sortArray()`, после чего результат сравнивается с вектором.

Описание функций:

1. **`int getAction()`** - запрашивает у пользователя действие и возвращает результат считывания.

2. **`template <typename T> void printArray(T *array, int size, char delimiter)`** — шаблонная функция для печати массива на консоль.

`T *array` — массив

`int size` — его размер

`char delimiter` — разделитель между элементами массива при выводе.

3. **`template <typename T> printArrayWithColor(T *array, int size, char delimiter, int redIndex, int redIndex_)`** - шаблонная функция для печати массива на консоль и окраски двух чисел в красный цвет.

`T *array` — массив

`int size` — размер

`char delimiter` — разделитель между элементами

4. **`template <typename T, typename D> void findMax(T *array, int size, D &max)`** — шаблонная функция для поиска максимума в массиве

`T *array` — массив

`int size` — размер

`D &max` — переменная, в которую будет записан максимум

5. **`template <typename T> void swapElements(T &left, T &right)`** — шаблонная функция для свапа двух элементов значениями

`T &left` — первый элемент

`T &right` — второй элемент

6. **`template <typename T> void sortArray(T *array, int size)`** — шаблонная функция с реализация бинго-сортировки.

`T *array` — массив

`int size` — размер

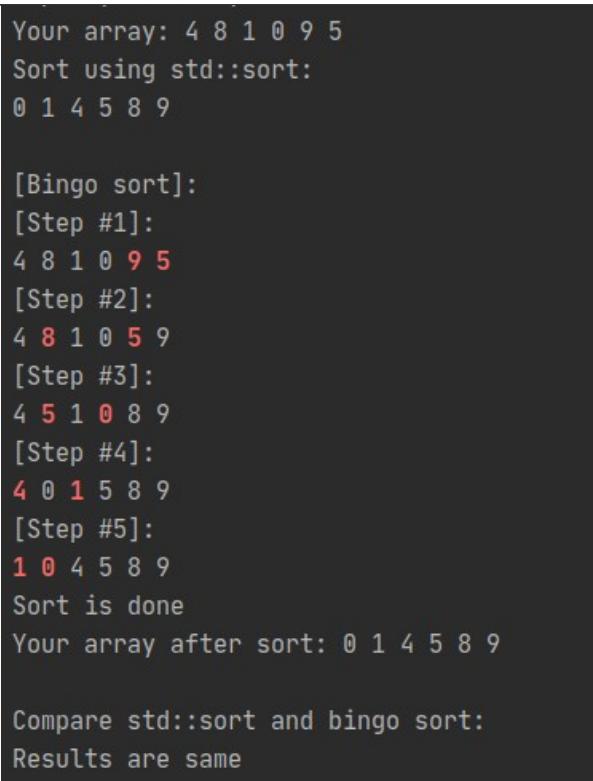
7. `template <typename T> bool compareVecArray(const T *array, vector<int> arr, int size)` — шаблонная функция для сравнение вектора и массива

`const int *array` — массив

`vector<int> arr` — вектор

`int size` — размер

Тестирование

Входные данные	Исходные данные
1. 1 6 4 8 1 0 9 5	 <pre> Your array: 4 8 1 0 9 5 Sort using std::sort: 0 1 4 5 8 9 [Bingo sort]: [Step #1]: 4 8 1 0 9 5 [Step #2]: 4 8 1 0 5 9 [Step #3]: 4 5 1 0 8 9 [Step #4]: 4 0 1 5 8 9 [Step #5]: 1 0 4 5 8 9 Sort is done Your array after sort: 0 1 4 5 8 9 Compare std::sort and bingo sort: Results are same </pre>
2. 1 10 9 9 8 8 7 7 4 3 2 5	2 3 4 5 7 7 8 8 9 9
3. 1 0	Wrong size. Try again
4. 1	

20	1 2 2 2 3 4 4 5 12 12 23 23 25 43 56 65 76 234 435
12 2 3 4 2 5 546 65	546
23 43 234 25 435 2	
4 1 23 56 76 12	
5. 1	1
1	
1	
6. 1	
2	1 2
2 1	

Выводы

Были изучены различные виды сортировки, реализована бинго-сортировка на языке программирования C++ с использованием шаблонов.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.cpp

```
#include <iostream>
#include <vector>
#include <fstream>

#define RED_COLOR "\033[1;31m"
#define RESET_COLOR "\033[0m"

using namespace std;

// Reads an action from the keyboard
int getAction() {
    int action = 0;

    cout << "Choose one of the following options: " << '\n' <<
        "1. Read from the keyboard" << '\n' <<
        "2. Read from the file" << '\n' <<
        "3. Exit" << '\n' <<
        "Your choice: ";

    cin >> action;
    return action;
}

// Outputs an array with delimiter
template < typename T >
void printArray(T *array, int size, char delimiter) {
    for (auto i = 0; i < size; i++)
        cout << array[i] << delimiter;
}
```

```

        // Outputs an array of any element and paints redIndex and
redIndex_ in red
        template < typename T >
        void printArrayWithColors(T *array, int size, char delimiter, int
redIndex, int redIndex_) {
            for (auto i = 0; i < size; i++) {
                if (i == redIndex || i == redIndex_) {
                    cout << RED_COLOR << array[i] << RESET_COLOR <<
delimiter;
                } else {
                    cout << array[i] << ' ';
                }
            }
        }
    }
}

```

```

// Finds the maximum in an array and writes in &max
template < typename T, typename D >
void findMax(T *array, int size, D &max) {
    for (auto i = size - 1; i >= 0; i--)
        if (array[i] > max)
            max = array[i];
}

```

```

// Swaps two elements by reference
template < typename T >
void swapElements(T &left, T &right) {
    T temp = left;
    left = right;
    right = temp;
}

```

```

// Bingo sort
template < typename T >
void sortArray(T *array, int size) {
    int stepCount = 0;

    // First iteration

```

```

    auto max = size - 1;
    T nextValue = array[max];

    // max is now in nextValue
    findMax(array, max, nextValue);

    // if more than one max skips them all
    while (max && array[max] == nextValue) max--;

    // Other iterations

    while (max) {
        T value = nextValue;
        nextValue = array[max];

        for (auto i = max - 1; i >= 0; i--) {
            if (array[i] == value) {
                cout << "[Step #" << ++stepCount << "]: \n";
                printArrayWithColors(array, size, ' ', i, max);
                cout << '\n';

                swapElements(array[i], array[max]);

                max--;
            } else if (array[i] > nextValue)
                nextValue = array[i]; // max
        }

        // skips maximum
        while (max && array[max] == nextValue)
            max--;
    }
    cout << "Sort is done" << '\n';
}

template <typename T>
bool compareVecArray(const T *array, vector<int> arr, int size) {
    for (auto i = 0; i < size; i++)
        if (array[i] != arr[i])

```



```

        return false;
    return true;
}

int main() {
    int action, size = 0;
    int *array;

    string input;
    ifstream file;
    string fileName;

    while ((action = getAction()) != 3) {
        size = 0;
        switch(action) {
            case 1:
                cout << "Input size of your array: ";
                cin >> size;

                if (!size) {
                    cout << "Wrong size. Try again" << "\n\n";
                    continue;
                }

                array = new int[size];

                cout << "Input your array: ";
                for (auto i = 0; i < size; i++)
                    cin >> array[i];
                break;
            case 2:
                cout << "Input the path to your file: ";
                cin >> fileName;
                file.open(fileName);

                if (!file.is_open()) {
                    cout << "Wrong file" << "\n\n";
                    continue;
                }

```

```

        file >> size;

        if (!size) {
            cout << "Wrong size. Try again" << "\n\n";
            continue;
        }

        array = new int[size];

        for (auto i = 0; i < size; i++)
            file >> array[i];

        file.close();
        break;
    default:
        cout << "Wrong input. Try again" << "\n\n";
        continue;
}

cout << "Your array: ";
printArray(array, size, ' ');
cout << '\n';

cout << "Sort using std::sort: \n";
vector<int> arr (array, array + size);
sort(arr.begin(), arr.end());
for (auto i = 0; i < size; i++)
    cout << arr[i] << ' ';
cout << "\n\n";

cout << "[Bingo sort]: " << '\n';
sortArray(array, size);
cout << "Your array after sort: ";
printArray(array, size, ' ');
cout << "\n\n";

cout << "Compare std::sort and bingo sort: \n";

```

```
        cout << (compareVecArray(array, arr, size) ? "Results are  
same" : "Results are different");  
  
        cout << "\n\n";  
        delete [] array;  
    }  
  
    cout << "Exiting the program" << '\n';  
    return 0;  
}
```