Лабораторная работа №5

Алгоритм программы

Пусть отрезок $S = [a, b]$, где a - начало отрезка, b - конец. Тогда принцип работы полшоамма таков: сортируем отрезки по началу каждого в возрастающем порядке, для S_i-го элемента смотрим a_{i+1}: если $a_i \leq a_{i+1} \leq b_i$, то объединяем отрезки S_i и S_{i+1}:

$$S = [a, max(b_i, b_{i+1})]$$

Если же для некоторого отрезка данное условие не выполняется, следовательно рассматриваемый отрезок не имеет пересечений с другими.

Код

Файл main: функция main

На вход получает количество аргументов и сами аргументы. Обрабатывает ввод и работает с файлами, вызывает остальные функции.

```
int main(int argc, char* argv[]) {
    char* input_path;
    char* output_path;
    bool flags[2] = {false,  // --fromfile
                     false}; // --tofile
    int input_path_index;
    int output_path_index;
    if (argc > 1) {
        for (int i = 1; i < argc; ++i) {
            if (strcmp("--fromfile", argv[i]) == 0) {
                input_path = argv[i + 1];
                flags[0] = true;
                ++i;
            }
            if (strcmp("--tofile", argv[i]) == 0) {
                output_path = argv[i + 1];
                flags[1] = true;
                ++i;
```

```
            }
        }
    }

    std::ifstream input(input_path);
    std::ofstream output(output_path);

    if (!input && flags[0]) {
        std::cerr << "Wrong input file name. Try again." << std::endl;
        exit(-1);
    }


    int T;
    if (flags[0]) input >> T;
    else std::cin >> T;


    float* lower = new float[T];
    float* upper = new float[T];
    float a, b;

    for (int i = 0; i < T; ++i) {
        if (flags[0]) input >> a >> b;
        else std::cin >> a >> b;
        lower[i] = a;
        upper[i] = b;
    }

    bubbleSort(lower, T, upper, T);

    int ans_c = 0;
    float* ans = parse(lower, T, upper, T, &ans_c);

    if (ans_c) {
        for (int i = 0; i < ans_c; i += 2) {
            if (flags[1]) {
                output << ans[i] << ' ' << ans[i + 1] << std::endl;
            }
            else {
                std::cout << ans[i] << ' ' << ans[i + 1] << std::endl;
            }
        }
    }
    else {
        if (flags[1]) {
            output << "NOTHING FOUND" << std::endl;
        }
        else std::cout << "NOTHING FOUND" << std::endl;
    }
    if (flags[0]) input.close();
    if (flags[1]) output.close();
}
```

Файл sort: функция bubbleSort

Принимает на вход два массива и сортирует первый по возрастанию “пузырьком” - последовательно сравнивает два соседних элемента и в случае, если значение первого болльше значения второго, меняет их местами.

```
void bubbleSort(float arr1[], int n1, float arr2[], int n2) {
  int i, j;
  float temp1, temp2;
  for (i = 0; i < n1 - 1; i++)
    for (j = 0; j < n1 - i - 1; j++)
      if (arr1[j] > arr1[j + 1]) {
                temp1 = arr1[j];
                arr1[j] = arr1[j + 1];
                arr1[j + 1] = temp1;

                temp2 = arr2[j];
                arr2[j] = arr2[j + 1];
                arr2[j + 1] = temp2;
            }
}
```

Файл parse: функция get_max

Принимает на вход два элемента и возвращает больший или равный второму по значению.

```
float get_max(float a, float b) {
    if (a >= b) return a;
    else return b;
}
```

Файл parse: функция parse

На вход принимает два массива и размер массива ответов - количества пересечений. Выполняет алгоритм, описанный выше.

```
float* parse(float lower[], int n1, float upper[], int n2, int* ans_len) {
    int ans_count = -2;

    float* ans = new float[*ans_len];
    bool intersected = false;
    for (int i = 0; i < n1 - 1; ++i) {
        if (lower[i] <= lower[i + 1] && lower[i + 1] <= upper[i]) {
            if  (!intersected) {
                ans_count += 2;
                *ans_len += 2;
            }
            ans[ans_count] = lower[i];
            ans[ans_count + 1] = get_max(upper[i], upper[i + 1]);

            lower[i + 1] = lower[i];
            upper[i + 1] = get_max(upper[i], upper[i + 1]);
            intersected = true;
        }
        else intersected = false;
    }
    return ans;
}
```

Тесты

Тесты былы реализованы с помощью языка программирования Python 3.10 и библиотек random, matplotlib. **tests.py** создает файл *tests.txt* с набором случайный отрезков в диапазоне $[-5; 5]$ в случайном количестве от 3, до 12.

```
from random import randint
print("_________")
print("TESTS:")


f = open("test.txt", "w")
n = randint(3, 12)
f.write(str(n) + "\n")
print(n)
values = []
for _ in range(n):
    a = randint(-50, 50)
    b = randint(a, 50)
    a /= 10
    b /= 10
    values += [[a, b]]
```

```
    f.write(str(a) + " " + str(b) + "\n")
f.close()
values.sort(key=lambda x: x[0])
[print(*elem) for elem in values]
print("_________")
```

Файл **plot.py** получает данные из *tests.txt,* на основе которых строит график с этими отрезками.

```
import matplotlib as mpl
import matplotlib.pyplot as plt
import numpy as np

colors = ["red", "green", "blue", "black", "yellow", "orange",
          "purple", "brown", "pink", "gray", "olive", "cyan"]

with open("test.txt", "r") as f:
    n = int(f.readline())
    values = [[float(x) for x in line.split()] for line in f.readlines()]

fig, ax = plt.subplots()
for i in range(n):
    a, b = values[i]
    col = colors.pop()
    ax.plot(np.linspace(a, b, 100), [i + 1] * 100, color=col)
    ax.plot([a] * 100, np.linspace(0, i + 1, 100), color=col)
    ax.plot([b] * 100, np.linspace(0, i + 1, 100), color=col)
plt.show()
```