**Вариант 23**

Пусть на плоскости с евклидовой метрикой задано N красных и N синих точек (красные точки имеют номера от 1 до N, а синие — от N + 1 до 2N). Существует N отрезков, соединяющих эти точки таким образом, что каждый отрезок соединяет точки различного цвета и каждая точка — концевая только для одного отрезка. Необходимо определить, является ли заданное соединение минимальным по длине (сумма длин всех отрезков) среди всех возможных соединений, удовлетворяющих заданным свойствам.

Формат входных данных

В первой строке находится натуральное число N (1 ≤ N ≤ 100). Следующие N строк файла

содержат по два целых числа: координаты красных точек (в первой строке — координаты

красной точки с номером 1, во второй — с номером 2 и т. д.). Затем идут N строк файла, в

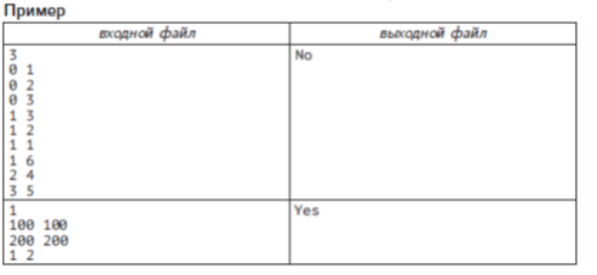
каждой из которых указаны два целых числа: координаты синих точек (первая из этих строк

— координаты синей точки с номером N + 1, вторая — с номером N + 2 и т. д.). Каждая из

следующих N строк файла задаёт отрезки соединения и содержит по два числа, определяющих начальные и конечные номера точек отрезков. Координаты красных и синих точек по модулю не превосходят 1000.

Формат выходных данных

Выведите в единственной строке сообщение Yes, если заданное соединение является минимальным по длине. Если оно таковым не является, то выведите сообщение No.



**Решение задачи:**

Для решения данной задачи можно воспользоваться графовым подходом. Мы можем представить точки и отрезки соединения в виде графа, где точки будут вершинами, а отрезки — ребрами.

Алгоритм решения через графы:

1) Создать пустой граф.

2) Добавить вершины в граф для каждой красной и синей точки.

3) Добавить ребра в граф для каждого отрезка соединения. Ребро должно соединять точки различного цвета.

4) Вычислить сумму длин всех ребер в графе.

5) Проверить, является ли данная сумма минимальной среди всех возможных соединений с заданными свойствами.

**Решение С++:**

#include <iostream>

#include <vector>

#include <cmath>

#include <limits>

#include <fstream>

std::ifstream fin("input.txt");

struct Point {

int x;

int y;

};

bool isMinLength(const std::vector<Point>& redPoints, const std::vector<Point>& bluePoints, const std::vector<std::pair<int, int>>& segments) {

int N = redPoints.size();

std::vector<std::vector<int>> graph(2 \* N + 1);

for (int i = 0; i < N; ++i) {

graph[i + 1].reserve(N);

for (int j = 0; j < N; ++j) {

graph[i + 1].push\_back(N + j + 1);

graph[N + j + 1].push\_back(i + 1);

}

}

double totalLength = 0.0;

for (const auto& segment : segments) {

int start = segment.first;

int end = segment.second;

int redIndex = start > N ? start - N : start;

int blueIndex = end > N ? end - N : end;

double length = std::sqrt(std::pow(redPoints[redIndex - 1].x - bluePoints[blueIndex - 1].x, 2) +

std::pow(redPoints[redIndex - 1].y - bluePoints[blueIndex - 1].y, 2));

totalLength += length;

}

double minLength = std::numeric\_limits<double>::max();

for (const auto& segment : segments) {

int start = segment.first;

int end = segment.second;

int redIndex = start > N ? start - N : start;

int blueIndex = end > N ? end - N : end;

double length = std::sqrt(std::pow(redPoints[redIndex - 1].x - bluePoints[blueIndex - 1].x, 2) +

std::pow(redPoints[redIndex - 1].y - bluePoints[blueIndex - 1].y, 2));

if (length < minLength) {

minLength = length;

}

}

return totalLength == minLength;

}

int main() {

int N;

fin >> N;

std::vector<Point> redPoints(N);

std::vector<Point> bluePoints(N);

std::vector<std::pair<int, int>> segments(N);

for (int i = 0; i < N; ++i) {

fin >> redPoints[i].x >> redPoints[i].y;

}

for (int i = 0; i < N; ++i) {

fin >> bluePoints[i].x >> bluePoints[i].y;

}

for (int i = 0; i < N; ++i) {

fin >> segments[i].first >> segments[i].second;

}

if (isMinLength(redPoints, bluePoints, segments)) {

std::cout << "Yes" << std::endl;

}

else {

std::cout << "No" << std::endl;

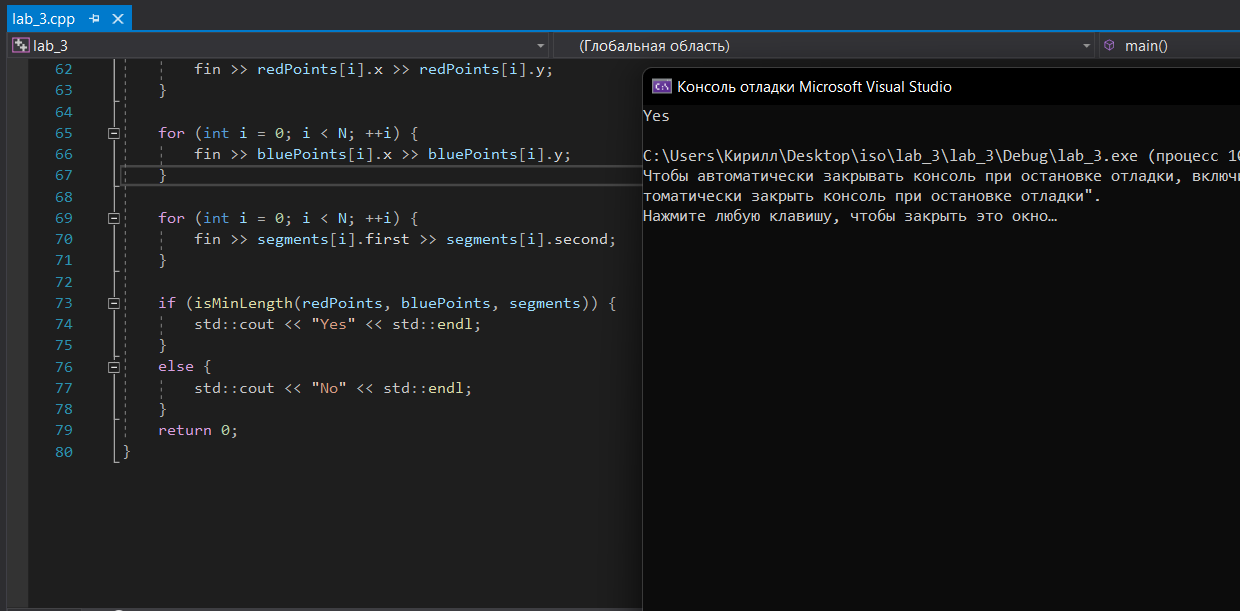
}

return 0;

}

Чтение из файла “input.txt”.

Вывод осуществляется в консоль.

****