Университет ИТМО

Факультет программной инженерии и компьютерной техники

Лабораторная работа №2

по «Алгоритмам и структурам данных»

Timus

Выполнил:

Студент группы P3233
Богатов Александр Сергеевич

Преподаватели:

Косяков М.С.

Тараканов Д.С.

Санкт-Петербург

2022

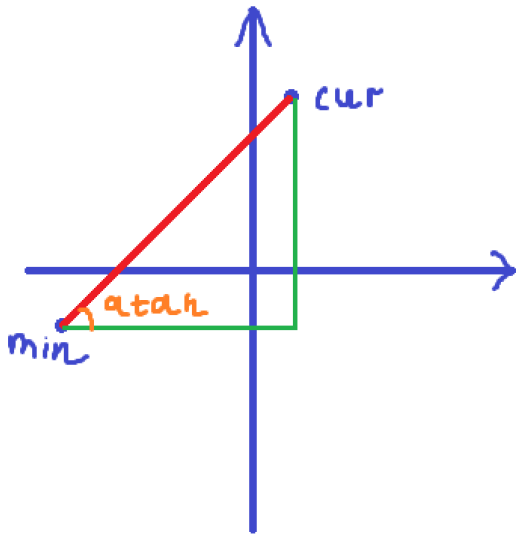
Задача 1207: медиана на плоскости

```
#include <iostream>
#include <cmath>
#include <map>
#include <algorithm>
#define PI 3.14159265358979323846264338327950288419716939937510582

using namespace std;


void merge(double * a, double * aux, int l, int m, int r) {
    int i = l;
    int j = m + 1;
    // for (int k = l; k <= r; ++k) {
    //     aux[k] = a[k];

    // }
    for (int k = l; k <= r; ++k) {
        if (i > m) {
            a[k] = aux[j++];
            continue;
        }
        if (j > r) {
            a[k] = aux[i++];
            continue;
        }
        if (aux[j] < aux[i]) {
            a[k] = aux[j++];
        }
        else {
            a[k] = aux[i++];
        }
    }
}


void merge_sort(double * a, double * aux, int l, int r) {
    if (l < r) {
        int m = (l + r) / 2;
        merge_sort(aux, a, l, m);
        merge_sort(aux, a, m + 1, r);
        merge(a, aux, l, m, r);
    }
}

void _sort(double* a, int N) {
    double* aux = new double[N];
    for (int k = 0; k < N; ++k)  {
        aux[k] = a[k];
    }
    merge_sort(a, aux, 0, N-1);
    delete[] aux;
}

int main()
{
    int n;
```

```
    cin >> n;
    int dots[n][2];
    int min_x = 1000001;
    int min_y = 1000001;
    int min_dot_idx = -1;
    int min_dot[1][2];
    double atans[n];
    double atans_copy[n];
    //int idxs[n];
    for (int i = 0; i< n; i++) {
        cin >> dots[i][0] >> dots[i][1];
    }

    for (int i = 0; i < n; i++) {
        if (dots[i][1] < min_y || dots[i][1] == min_y && dots[i][0] <
min_x) {
            min_dot_idx = i;
            min_x = dots[i][0];
            min_y = dots[i][1];
        }
    }

    min_dot[0][0] = dots[min_dot_idx][0];
    min_dot[0][1] = dots[min_dot_idx][1];

    for (int i = 0; i < n; i++) {
        double current_atan;
        if (dots[i][0] == min_dot[0][0])
            current_atan = PI / 2;
        else if (dots[i][1] == min_dot[0][1])
            current_atan = 0;
            // 3 + 1 / 1 + 1
        else current_atan = atan((double) (dots[i][1] - min_dot[0][1])
/ (double) (dots[i][0] - min_dot[0][0]));
        if (current_atan < 0) current_atan += 2 * PI;
        atans[i] = current_atan;
        atans_copy[i]=current_atan;
    }

    swap(atans[n-1], atans[min_dot_idx]);

    _sort(atans, n-1);
    cout << min_dot_idx + 1 << " ";
    for(int i = 0; i < n; i++) {
        if (atans_copy[i] == atans[(n-1)/2]) {
            cout << i+1 << endl;
            break;
        }
    }

    return 0;
}
```

Пояснение к примененному алгоритму:

Первой точкой всегда удобно выбирать самую нижнюю или самую нижнюю левую точку, из неё мы будем проводить линии к другим точкам и сравнивать углы относительно оси параллельной оси X. Необходимо найти средний угол.



Для всех точек найдем тот самый угол через функцию арктангенса, нормализуем значение прибавлением 2*Pi, если оно меньше нуля. Далее выкинем точку условного начала координат из массива и отсортируем массив углов с помощью алгоритма сортировки слиянием.

Также ведем копию неотсортированного массива углов, т.к. с помощью него в конце получаем индекс точки, через которую можно построить искомую линию.

Задача 1726: Кто в гости идет

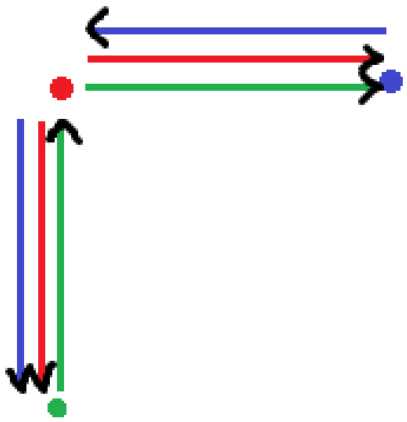
```
#include <iostream>
#include <cmath>
#include <algorithm>

using namespace std;

int main()
{
    long long n;
    cin >> n;
    int x[n];
    int y[n];
    long long distance_walked = 0;

    for (long long i = 0; i < n; i++) {
        cin >> x[i] >> y[i];
    }

    sort(x, x+n);
    sort(y, y+n);

    for (long long  i = 1; i < n; i++) {
        distance_walked += abs(x[i] - x[i-1] + y[i] - y[i-1]) * i *
(n-i);
    }
```

```
    cout << (distance_walked * 2)/(n*(n-1)) << endl;

    return 0;
}
```

Пояснение к примененному алгоритму:

Лирическое отступление: Несмотря на условие задачи, для n будем использовать тип long long. Честно не совсем уверен почему, но только после этого изменения Timus принял задачу. Предположу что проблема в вычислении делителя n*(n-1).

Нужно посчитать для каждого человека расстояние от его дома до остальных домов. Заведем отдельный массив для координат x и координат y для упрощения сортировки. Будем считать все расстояние, которое можно пройти по дорогам между домами. В цикле прибавляем к переменной расстояние пройденное параллельно оси X и оси Y между текущей и предыдущей точкой.



Нужно учитывать то, что по дорогам могу пройти несколько членов комитета. Каждое вычисленное расстояние умножим на i – число членов комитета, условно уже прошедших дорогу, и умножим на n-i – число тех, кому условно предстоит пройти дорогу.

На примере: т.к. массивы отсортированы, мы будем идти начиная с точки (10;10). Цикл начинается с рассмотра точек (10;10) и (10;20). Условный член комитета, за которым мы следим, уже находится в точке, в то время как остальные находятся дальше этой дороги, и чтобы попасть в его дом им нужно пройти по рассматриваемой дороге. Так, на следующем шаге мы будем смотреть уже за двумя членами комитета, которые будут идти вперед, а оставшемуся придется идти назад, чтобы посетить их дома.

В конце дистанцию умножим на 2, чтобы учитывать повторные проходы по дорогам члена комитета, и разделим на число всех дорог – n*(n-1).