Национальный исследовательский университет ИТМО
(Университет ИТМО)

Факультет систем управления и робототехники

Дисциплина: Алгоритмы и структуры данных
Отчет по практической работе (1444 задача).

Студент:
Евстигнеев Дмитрий
Группа: *R3242*
Преподаватель:
Тропченко Андрей Александрович

Санкт-Петербург
2021

Цель: написать программу для решения задачи №1444 на сайте Timus Online

Задача:

1444. Накормить элефпотама

Ограничение времени: 0.5 секунды

Ограничение памяти: 64 МБ

Гарри Поттер сдаёт экзамен по предмету «Уход за магическими существами». Его задание — накормить карликового элефпотама. Гарри помнит, что элефпотамы отличаются прямолинейностью и невозмутимостью. Они настолько прямолинейны, что ходят строго по прямой, и настолько невозмутимы, что заставить их идти можно, только если привлечь его внимание к чему-нибудь действительно вкусному. И главное, наткнувшись на цепочку своих собственных следов, элефпотам впадает в ступор и отказывается идти куда-либо. По словам Хагрида, элефпотамы обычно возвращаются домой, идя в обратную сторону по своим собственным следам. Поэтому они никогда не пересекают их, иначе могут заблудиться. Увидев свои следы, элефпотам детально вспоминает все свои перемещения от выхода из дома (поэтому-то они и ходят только по прямой и лишний раз не меняют направление — так легче запоминать). По этой информации элефпотам вычисляет, в какой стороне расположена его нора, после чего поворачивается и идет прямо к ней. Эти вычисления занимают у элефпотама некоторое (довольно большое) время. А то, что некоторые невежды принимают за ступор, на самом деле есть проявление выдающихся вычислительных способностей этого чудесного, хотя и медленно соображающего животного!

Любимое лакомство элефпотамов — слоновьи тыквы, именно они и растут на лужайке, где Гарри должен сдавать экзамен. Перед началом испытания Хагрид притащит животное к одной из тыкв. Скормив элефпотаму очередную тыкву, Гарри может направить его в сторону любой оставшейся тыквы. Чтобы сдать экзамен, надо провести элефпотама по лужайке так, чтобы тот съел как можно больше тыкв до того, как наткнется на свои следы.

Исходные данные

В первой строке входа находится число N ($3 \le N \le 30000$) — количество тыкв на лужайке. Тыквы пронумерованы от 1 до N, причем номер один присвоен той тыкве, у которой будет стоять элефпотам в начале экзамена. В следующих N строках даны координаты всех тыкв по порядку. Все координаты — целые числа от −1000 до 1000. Известно, что положения всех тыкв различны, и не существует прямой, проходящей сразу через все тыквы.

Результат

В первой строке выхода вы должны вывести K — максимальное количество тыкв, которое может съесть элефпотам. Далее по одному числу в строке выведите K чисел — номера тыкв в порядке их обхода. Первым в этой последовательности всегда должно быть число 1.

Пример

исходные данные	результат
4 0 0 10 10 0 10 10 0	4 1 3 2 4

Принято системой (JUDGE_ID: 231802FR):

9376845	12:16:25 24 май 2021	Dmitry Evstigneev	1444. Накормить элефлотама	G++ 9.2 x64	Accepted

Решение на языке C++:

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

struct point {
    int x, y;
    int i;
} p[30000];
point pk;

int f(point& a, point& b) {
    if(a.x*b.y == a.y*b.x && a.x*b.x + a.y*b.y >= 0)
        // Points with the same angle are sorted outwards
        return a.x*a.x + a.y*a.y < b.x*b.x + b.y*b.y;
    return atan2(a.y, a.x) < atan2(b.y, b.x);
}

int main() {
    int n;
    std::ios::sync_with_stdio(false);
    std::cin >> n;

    for(int i = 0; i < n; i++) {
        int x, y;
        std::cin >> p[i].x >> p[i].y;
        p[i].i = i;
    }

    for(int i = n-1; i >= 0; i--)
        // Translate everything towards the first point
        p[i].x -= p[0].x, p[i].y -= p[0].y;

    std::sort(p, p+n, f);

    int s = 0;
    for(int i = 0; i < n-1; i++) {
        point p0 = p[0], p1 = p[i], p2 = p[i+1];
```

```
            int d1x = p1.x-p0.x, d2y = p2.y-p0.y, d1y = p1.y-p0.y, d2x = p2.x-
p0.x;
            int x = d1x*d2y - d1y*d2x, d = d1x*d2x + d1y*d2y;
            if(x < 0 || x == 0 && d < 0) { // Opposite turn, start at that point
instead
                s = i;
                break;
            }
        }

        std::cout << n << std::endl;
        std::cout << (p[0].i+1) << std::endl;
        for(int i = 0; i < n-1; i++)
            std::cout << (p[(s+i)%(n-1)+1].i+1) << std::endl;
}
```

Суть алгоритма:

Отсортируем точки по возрастанию и построим многоугольник в этом порядке. Если в какой-то момент многоугольник поворачивается на >180 градусов (что может привести к перекрытию линий), вместо этого сделаем начало многоугольника (после первой точки) сразу после поворота.