E. Амбициозная улитка

|  |  |
| --- | --- |
| Ограничение времени | 5 секунд |
| Ограничение памяти | 256Mb |
| Ввод | стандартный ввод или input.txt |
| Вывод | стандартный вывод или output.txt |

Домашний питомец мальчика Васи — улитка Петя. Петя обитает на бесконечном в обе стороны вертикальном столбе, который для удобства можно представить как числовую прямую. Изначально Петя находится в точке 0.

Вася кормит Петю ягодами. У него есть n ягод, каждая в единственном экземпляре. Вася знает, что если утром он даст Пете ягоду с номером i, то поев и набравшись сил, за остаток дня Петя поднимется на ai единиц вверх по столбу, но при этом за ночь, потяжелев, съедет на bi единиц вниз. Параметры различных ягод могут совпадать.

Пете стало интересно, а как оно там, наверху, и Вася взялся ему в этом помочь. Ближайшие n дней он будет кормить Петю ягодами из своего запаса таким образом, чтобы максимальная высота, на которой побывал Петя за эти n дней была максимальной. К сожалению, Вася не умеет программировать, поэтому он попросил вас о помощи. Найдите, максимальную высоту, на которой Петя сможет побывать за эти n дней и в каком порядке Вася должен давать Пете ягоды, чтобы Петя смог её достичь!

Формат ввода

В первой строке входных данных дано число n (1≤n≤5⋅105) — количество ягод у Васи. В последующих n строках описываются параметры каждой ягоды. В i+1 строке дано два числа ai и bi (0≤ai,bi≤109) — то, насколько поднимется улитка за день после того, как съест i ягоду и насколько опуститься за ночь.

Формат вывода

В первой строке выходных данных выведите единственное число — максимальную высоту, которую сможет достичь Петя, если Вася будет его кормить оптимальным образом. В следующей строке выведите n различных целых чисел от 1 до n — порядок, в котором Вася должен кормить Петю (i число в строке соответствует номеру ягоды, которую Вася должен дать Пете в i день чтобы Петя смог достичь максимальной высоты).

Пример 1

| **Ввод**  Скопировать ввод | **Вывод**  Скопировать вывод |
| --- | --- |
| 3  1 5  8 2  4 4 | 10  2 3 1 |

Пример 2

| **Ввод**  Скопировать ввод | **Вывод**  Скопировать вывод |
| --- | --- |
| 2  7 6  7 4 | 10  2 1 |

Примечания

Во втором примере изначально улитка находится на высоте 0. Пусть сначала Петя накормит её второй ягодой, а затем первой. После того как она съест вторую ягоду, за день она поднимется на 7 (и окажется на высоте 7), а за ночь опустится на 4 (и окажется на высоте 3). После того как она съест первую ягоду, за день она поднимется на 7 (и окажется на высоте 10), а за ночь опустится на 6 (и окажется на высоте 4).

Таким образом, максимальная высота, на которой побывает улитка при данном порядке кормления, равна 10. Нетрудно видеть, что если Петя накормит улитку сначала первой ягодой, а затем второй, то максимальная высота, на которой побывает улитка, будет меньше.

using System;

using System.Collections.Generic;

public struct Berry

{

public int day;

public int night;

public int dif;

public int index;

public Berry(int day1, int night1, int dif1, int index1)

{

day = day1;

night = night1;

dif = dif1;

index = index1;

}

}

class Program

{

static void Main()

{

Int64 max = 0;

Int64 currentHight = 0;

int N = Convert.ToInt32(Console.ReadLine());

List<int> result = new List<int>();

// int[] result = new int[N];

Berry[] arrayGood = new Berry[N];

int lenGood = 0;

Berry[] arrayBad = new Berry[N];

int lenBad = 0;

int maxGood = 0;

int maxDif = 0;

int maxIndexGood = 0;

int maxBad = 0;

int maxIndexBad = 0;

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

{

string[] input1 = Console.ReadLine().Split();

int day = int.Parse(input1[0]);

int night = int.Parse(input1[1]);

if (day - night > 0)

{

Berry berry = new Berry(day, night, day - night, i);

if (night > maxGood)

{

maxGood = night;

maxIndexGood = lenGood;

}

arrayGood[lenGood] = berry;

lenGood++;

}

else

{

Berry berry = new Berry(day, night, day - night, i);

if (day > maxBad)

{

maxBad = day;

maxIndexBad = lenBad;

}

arrayBad[lenBad] = berry;

lenBad++;

}

}

for (int i = 0; i < lenGood; i++)

{

if (i != maxIndexGood)

{

currentHight += arrayGood[i].dif;

if (currentHight - arrayGood[i].night > max) max = currentHight - arrayGood[i].night;

result.Add(arrayGood[i].index + 1);

}

}

if (currentHight + arrayGood[maxIndexGood].day > max)

{

max = currentHight + arrayGood[maxIndexGood].day;

currentHight += arrayGood[maxIndexGood].dif;

result.Add(arrayGood[maxIndexGood].index + 1);

}

bool flag = true;

if (currentHight + arrayBad[maxIndexBad].day > max)

{

max = currentHight + arrayBad[maxIndexBad].day;

result.Add(arrayBad[maxIndexBad].index + 1);

flag = false;

}

for (int i = 0; i < lenBad; i++)

{

if (!(i == maxIndexBad && !flag))

{

result.Add(arrayBad[i].index + 1);

}

}

Console.WriteLine(max);

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

{

Console.Write(result[i] + " ");

}

}

}