Тренировки по алгоритмам 5.0 от Яндекса — Занятие 2 (Линейный поиск)

20 мар 2024, 21:12:24
старт: 6 мар 2024, 22:30:00
финиш: 20 мар 2024, 20:00:00
длительность: 13д. 21ч.
начало: 6 мар 2024, 22:30:00
конец: 20 мар 2024, 20:00:00

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 a_i единиц вверх по столбу, но при этом за ночь, потяжелев, съедет на b_i 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 ($1 \le n \le 5 \cdot 10^5$) — количество ягод у Васи. В последующих n строках описываются параметры каждой ягоды. В $i + 1$ строке дано два числа a_i и b_i ($0 \le a_i, b_i \le 10^9$) — 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	Вывод
3 1 5 8 2 4 4	10 2 3 1

Пример 2

Ввод	Вывод
2 7 6	10 2 1

Ввод	Вывод
7 4	

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.

Язык Kotlin 1.9.21 (JRE 21)

Набрать здесь | Отправить файл

```kotlin
fun main(args: Array<String>) {
    val count = readln().toInt()

    val berries = mutableListOf<Pair<Long, Long>>()
    for (i in 1..count) {
        val list = readln().split(" ")
        berries.add(Pair(list.first().toLong(), list.last().toLong()))
    }

    val indexedBerries = berries.mapIndexed { index, elem ->
        index + 1 to elem
    }.sortedByDescending { it.second.first - it.second.second }

    val positiveDiff = mutableListOf<Pair<Int, Pair<Long, Long>>>()
    val negativeDiff = mutableListOf<Pair<Int, Pair<Long, Long>>>()

    for (berry in indexedBerries) {
        if (berry.second.first - berry.second.second >= 0) positiveDiff.add(berry) else negativeDiff.add(berry)
    }

    val order = mutableListOf<Int>()
    var max = 0L
    var current = 0L

    for (positive in positiveDiff.sortedByDescending { it.second.first }) {
        current += positive.second.first
        if (current > max) {
            max = current
        }
        current -= positive.second.second
        order.add(positive.first)
    }

    for (negative in negativeDiff.sortedByDescending { it.second.first }) {
        current += negative.second.first
        if (current > max) {
            max = current
        }
```

Отправить

Предыдущая | Следующая