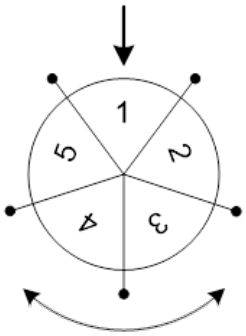
Тренировки по алгоритмам 5.0 от Яндекса — Занятие 2 (Линейный поиск)

20 мар 2024, 21:12:31
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

F. Колесо Фортуны

Ограничение времени	1 секунда
Ограничение памяти	64Mb
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

Развлекательный телеканал транслирует шоу «Колесо Фортуны». В процессе игры участники шоу крутят большое колесо, разделенное на сектора. В каждом секторе этого колеса записано число. После того как колесо останавливается, специальная стрелка указывает на один из секторов. Число в этом секторе определяет выигрыш игрока.
Юный участник шоу заметил, что колесо в процессе вращения замедляется из-за того, что стрелка задевает за выступы на колесе, находящиеся между секторами. Если колесо вращается с угловой скоростью v градусов в секунду, и стрелка, переходя из сектора X к следующему сектору, задевает за очередной выступ, то текущая угловая скорость движения колеса уменьшается на k градусов в секунду. При этом если $v \le k$, то колесо не может преодолеть препятствие и останавливается. Стрелка в этом случае будет указывать на сектор X.



Юный участник шоу собирается вращать колесо. Зная порядок секторов на колесе, он хочет заставить колесо вращаться с такой начальной скоростью, чтобы после остановки колеса стрелка указала на как можно большее число. Колесо можно вращать в любом направлении и придавать ему начальную угловую скорость от a до b градусов в секунду.

Требуется написать программу, которая по заданному расположению чисел в секторах, минимальной и максимальной начальной угловой скорости вращения колеса и величине замедления колеса при переходе через границу секторов вычисляет максимальный выигрыш.

Формат ввода

Первая строка входного файла содержит целое число n — количество секторов колеса ($3 \le n \le 100$).
Вторая строка входного файла содержит n положительных целых чисел, каждое из которых не превышает 1000 — числа, записанные в секторах колеса. Числа приведены в порядке следования секторов по часовой стрелке. Изначально стрелка указывает на первое число.

Третья строка содержит три целых числа: a, b и k ($1 \le a \le b \le 10^9$, $1 \le k \le 10^9$).

Формат вывода

В выходном файле должно содержаться одно целое число — максимальный выигрыш.

Пример 1

Ввод	Вывод
5 1 2 3 4 5 3 5 2	5

Пример 2

Ввод	Вывод
5 1 2 3 4 5 15 15 2	4

Пример 3

Ввод	Вывод
5 5 4 3 2 1 2 5 2	5

Примечания

В первом примере возможны следующие варианты: можно придать начальную скорость колесу равную 3 или 4, что приведет к тому, что стрелка преодолеет одну границу между секторами, или придать начальную скорость равную 5, что позволит стрелке преодолеть 2 границы между секторами. В первом варианте, если закрутить колесо в одну сторону, то выигрыш получится равным 2, а если закрутить его в противоположную сторону, то — 5. Во втором варианте, если закрутить колесо в одну сторону, то выигрыш будет равным 3, а если в другую сторону, то — 4.
Во втором примере возможна только одна начальная скорость вращения колеса — 15 градусов в секунду. В этом случае при вращении колеса стрелка преодолеет семь границ между секторами. Тогда если его закрутить в одном направлении, то выигрыш составит 4, а если в противоположном направлении, то — 3.

Наконец, в третьем примере оптимальная начальная скорость вращения колеса равна 2 градусам в секунду. В этом случае стрелка вообще не сможет преодолеть границу между секторами, и выигрыш будет равен 5.

Язык Kotlin 1.9.21 (JRE 21)

Набрать здесь | Отправить файл

```
fun main(args: Array<String>) {
    val n = readln().toInt()
    val nums = readln().split(" ").map { it.toInt() }
    val lastRow = readln().split(" ").map { it.toInt() }

    var a = lastRow[0]
    val b = lastRow[1]
    val k = lastRow[2]

//    val elementsA = a / k
//    val elementsB = b / k

    val elementsA = customDivide(a, k)
    val elementsB = customDivide(b, k)

    val roundsA = customDivide(elementsA, n)
    val roundsB = customDivide(elementsB, n)

    val formattedElementsA = elementsA - roundsA * n
    val formattedElementsB = elementsB - roundsB * n

//    nums[formattedElementsA]
//    nums[nums.size - formattedElementsA]

    var max = 0

    if (roundsA == roundsB) {
        for (i in formattedElementsA..formattedElementsB) {
            if (i == n) {
                if (nums[0] > max) max = nums[0]
            } else {
                if (nums[i] > max) max = nums[i]
            }
        }

        for (i in n - formattedElementsB..n - formattedElementsA) {
            if (i == n) {
                if (nums[0] > max) max = nums[0]
```

Отправить

Предыдущая

Следующая