Рыцарский турнир

Для своей свадьбы с Беатриче д'Эсте в 1491 году граф Милана Людовико Сфорца попросил Леонардо организовать свадебную церемонию, включая большой рыцарский турнир, который бы длился 3 дня. Но самый популярный рыцарь опоздал…

Турнир

В начале турнира N рыцарей выстраивают в линию и их позиции нумеруются от 0 до N - 1 в порядке их расположения на линии. Распорядитель турнира начинает раунд, называя две позиции S и E, где $0 \leq S < E \leq (N - 1)$. Все рыцари, чьи позиции находятся между позициями S и E включительно, соревнуются между собой, после чего победитель продолжает участвовать в турнире и занимает свое место в линии, а проигравшие выбывают и покидают поле. После этого рыцари сдвигаются к началу линии, соблюдая относительный порядок в линии, и занимают позиции от 0 до N - (E - S) - 1. Распорядитель турнира начинает новый раунд, повторяя этот процесс, пока не останется только один рыцарь.

Леонардо знает, что все рыцари различаются по силе, которая задается как неповторяющийся рейтинг от 0 (самый слабый) до N - 1 (самый сильный). Он также знает точные команды, которыми распорядитель турнира будет начинать C раундов - он же Леонардо. Кроме того, он уверен, что в каждом из этих раундов победит рыцарь с наивысшим рейтингом.

Опоздавший рыцарь

Из N рыцарей N - 1 уже выстроенны в линию, не хватает только самого популярного рыцаря, который немного задерживается. Ранг этого рыцаря - R. Чтобы извлечь из этого выгоду, Леонардо хочет использовать его популярность и выбрать для него позицию в линии, которая максимизирует количество раундов, выигранных опоздавшим рыцарем. Необходимо принять во внимание, что раунды, в которых не принимает участия опоздавший рыцарь, нас не интересуют. Считаются только те раунды, в которых он принял участие и выиграл.

Пример

Есть 5 рыцарей (N = 5), N - 1 из них уже расставлены в линию и имеют рейтинги [1, 0, 2, 4]. Следовательно, опоздавний рыцарь будет иметь рейтинг R = 3. Распорядитель турнира собирается провести 3 раунда (C = 3), называя позиции (S, E) в начале каждого раунда в таком порядке: (1, 3), (0, 1), (0, 1).

Если Леонардо поставит опоздавшего рыцаря в начале линии, рейтинги рыцарей будут [3, 1, 0, 2, 4]. В первом раунде в позициях 1, 2, 3 с рейтингами 1, 0, 2. В нем победит рыцарь с рейтингом 2. Новая линия будет [3, 2, 4]. Следующий раунд будет между рыцарями в позициях 0, 1 с рейтингами 3 и 2, победит рыцарь с рейтингом R = 3, образовав линию [3, 4]. В последнем раунде будут участвовать рыцари в позициях 0 и 1, победит рыцарь с рейтингом 4. Таким образом, опоздавший рыцарь победит только в одном раунде (во втором).

Если Леонардо будет использовать другой вариант и поставит опоздавшего рыцаря между двумя другими с рейтингами 1 и 0, то исходная линия будет выглядеть так: [1, 3, 0, 2, 4]. Теперь в первом раунде соревнуются рыцари с рейтингами 3, 0, 2, и побеждает рыцарь с рейтингом R = 3. Получающаяся линия будет выглядеть как [1, 3, 4], и в следующем раунде (рыцарь с рейтингом 1 против рыцаря с рейтингом 3) снова побеждает рыцарь с рейтингом R = 3. В результатет получается линия [3, 4], где побеждает рыцарь с рейтингом 4. Таким образом, опоздавший рыцарь победит в двух раундах, и это наилучшее возможное расположение опоздавшего рыцаря, так как у него нет возможности выиграть больше двух раз.

Условие

Требуется написать программу, которая выбирает для опоздавшего рыцаря наилучшую позицию в линии, чтобы максимизировать количество выигранных им раундов, как хочет Леонардо. Точнее, необходимо реализовать процедуру с именем `GetBestPosition`(N, C, R, K, S, E), где:

- N - количество рыцарей;
- C - количество раундов, запланированных распорядителем турнира ($1 \le C \le N - 1$);
- R - рейтинг опоздавшего рыцаря; рейтинги всех рыцарей (как уже выстроенных в линию, так и опоздавшего) являются различными и выбирабтся из диапазона 0, …, N - 1, и рейтинг R опоздавшего рыцаря задается явно, хотя может быть вычислен;
- K - массив из N - 1 целых чисел, представляющих рейтинги N - 1 рыцаря, которые уже находятся на линии;
- S и E - два массива размера C: для каждого i от 0 до C - 1, включительно, в раунде i + 1 участвуют рыцари с номерами от S[i] до E[i], включительно. Гарантируется, что для каждого i, S[i] < E[i].

Эта процедура будет вызываться с правильными данными: E[i] будет меньше текущего количества рыцарей, которые могут участвовать в раунде i+1, и после всех C команд останется ровно один рыцарь.

`GetBestPosition`(N, C, R, K, S, E) должна возвращать наилучшую позицию P, в которую Леонардо должен расположить опоздавшего рыцаря ($0 \le P \le N - 1$). Если есть несколько равноценных позиций, необходимо вывести наименьшую. Номер позиции P можно определить как количество рыцарей, стоящих перед опоздавшим рыцарем в оптимальном решении. В частности, P = 0 обозначает, что опоздавший рыцарь находится

в начале линии, и P = (N - 1) обозначает, что он находится в конце.

Подзадача 1 [17 баллов]

Предполагается, что $N \leq 500$.

Подзадача 2 [32 балла]

Предполагается, что $N \leq 5\,000$.

Подзадача 3 [51 балл]

Предполагается, что $N \leq 100\,000$.

Детали реализации

Вам нужно сдать один файл, который называется `tournament.c`, `tournament.cpp` или `tournament.pas`. Этот файл должен реализовывать подпрограмму, описанную выше, используя такие сигнатуры.

программы на C/C++

```
int GetBestPosition(int N, int C, int R, int *K, int *S, int *E);
```

программы на Pascal

```
function GetBestPosition(N, C, R : LongInt; var K, S, E : array of LongInt) : LongInt;
```

Подпрограммы должны вести себя так, как описано выше. Конечно, вы можете реализовать собственные подпрограммы для внутреннего пользования. Ваши посылки не должны взаимодействовать ни со стандартными вводом/выводом, ни с другими файлами.

Пример проверяющего модуля

Предоставленный проверяющий модуль ожидает ввод в таком формате:

- первая строка: N, C, R;
- строки 2, …, N: K[i];
- строки N + 1, …, N + C: S[i], E[i].

Ограничения по времени и памяти

- Ограничение по времени: 1 секунда.
- Ограничение памяти: 256 мебибайт.