Scontro di cavalieri

Per le sue nozze con Beatrice d'Este nel 1491, il Duca di Milano Ludovico Sforza chiese a Leonardo di organizzare le celebrazioni nuziali, tra cui un gran torneo di cavalieri della durata di tre giorni. Ma il cavaliere più popolare è in ritardo...

Torneo

In un torneo di cavalieri, gli N cavalieri sono inizialmente disposti in una linea, quindi le loro posizioni vengono numerate da 0 a N - 1 seguendo l'ordine nella linea. Ogni round è iniziato dal *mastro di torneo*, che chiama due posizioni S ed E (dove $0 \leq S < E \leq N - 1$) . Tutti i cavalieri le cui posizioni sono comprese tra S ed E (incluse) combattono: il vincitore continua nel torneo e riprende il suo posto nella linea, mentre gli sconfitti sono fuori dal gioco e lasciano il campo. Subito dopo, i cavalieri rimanenti si compattano verso l'inizio della linea, mantenendo il loro ordine relativo nella linea, in maniera che le loro posizioni risultanti siano da 0 a N - (E - S) - 1. Il mastro di torneo a questo punto inizia un altro round e questo processo viene ripetuto finché rimane soltanto un cavaliere.

Leonardo sa che tutti i cavalieri hanno un diverso valore di forza, rappresentato in valori distinti compresi tra 0 (il più debole) e N - 1 (il più forte). Inoltre, egli conosce i comandi esatti che il mastro di torneo chiamerà nei C round: lui è Leonardo, dopotutto... ed è sicuro che in ognuno di questi round vincerà il cavaliere con il valore più alto.

Il Cavaliere Ritardatario

N - 1 degli N cavalieri sono già disposti nella linea, manca solo il cavaliere più popolare. Questo cavaliere ha valore R ed è in ritardo. Per il beneficio del pubblico divertimento, Leonardo vuole sfruttare la sua popolarità e scegliere per lui una posizione nella linea in grado di massimizzare il numero di round che il cavaliere ritardatario vincerà. Notate che non siamo interessati ai round che non coinvolgono il cavaliere ritardatario: sono di interesse solo i round nei quali lui partecipa e vince.

Esempio

Nel caso di N = 5 cavalieri, gli N - 1 cavalieri che sono già disposti nella linea hanno, rispettivamente, valori pari a [1, 0, 2, 4]. Di conseguenza, il cavaliere ritardatario ha valore R = 3. Per i C = 3 round, il mastro di torneo ha intenzione di chiamare, rispettivamente, nei round le posizioni (S, E) in questo ordine: (1, 3), (0, 1), (0, 1).

Se Leonardo inserisce il cavaliere ritardatario nella prima posizione, i valori dei cavalieri nella linea saranno [3, 1, 0, 2, 4]. Il primo round coinvolge cavalieri (nelle posizioni 1, 2, 3) con valori 1, 0, 2,

lasciando quindi il cavaliere di valore 2 come vincitore. La nuova linea è [3, 2, 4]. Il round successivo vede opposti i cavalieri di valore 3 e 2 (nelle posizioni 0, 1), e il cavaliere di valore R = 3 vince, e la nuova linea è quindi [3, 4]. Il round finale (alle posizioni 0, 1) ha 4 come vincitore. Quindi, il cavaliere ritardatario vince solo un round (il secondo).

Viceversa, se Leonardo inserisce il cavaliere ritardatario tra quelli di valore 1 e 0, la linea varrà: [1, 3, 0, 2, 4]. Questa volta, il primo round coinvolge i cavalieri di valore 3, 0, 2, e il cavaliere di valore R = 3 vince. Nel round successivo la linea vale [1, 3, 4], e nel duello tra i cavalieri di valore 1 e 3 (posizioni 0 e 1) il cavaliere di valore R = 3 vince di nuovo. La linea finale vale [3, 4], dove il 4 vince. Quindi, in questo caso il cavaliere ritardatario vince due round: in realtà, questa posizione è la migliore possibile, e non c'è modo di inserire il cavaliere ritardatario nella linea in maniera da fargli vincere più di due round.

Descrizione del problema

Il tuo compito è quello di scrivere un programma che scelga la posizione migliore dove inserire il cavaliere ritardatario in modo da massimizzare il numero di round che vincerà, come Leonardo desidera. In particolare, devi implementare una funzione chiamata `GetBestPosition`(N, C, R, K, S, E), dove:

- N è il numero dei cavalieri;
- C è il numero di round chiamati dal mastro di torneo ($1 \leq C \leq N - 1$);
- R è il valore del cavaliere ritardatario; i valori di tutti i cavalieri (ovvero i valori di quelli già schierati nelle linea insieme con il valore del cavaliere ritardatario) sono distinti e scelti fra 0, …, N - 1, e il valore R è dato esplicitamente, nonostante sia possibile dedurlo;
- K è un array di N - 1 interi, che rappresentano i valori degli N - 1 cavalieri che sono già schierati nella linea;
- S e E sono due array di dimensione C: per ogni i compreso tra 0 e C - 1, inclusi, il round (i + 1) chiamato dal mastro del torneo coinvolgerà tutti i cavalieri nelle posizioni comprese tra S[i] ed E[i], incluse. Puoi assumere che, per ogni i, S[i] < E[i].

Le chiamate passate a questa funzione sono valide: E[i] è minore del numero corrente di cavalieri che rimangono per il round (i+1), e dopo tutti i C round sarà rimasto esattamente un cavaliere.

`GetBestPosition`(N, C, R, K, S, E) deve restituire la migliore posizione P dove Leonardo dovrebbe inserire il cavaliere ritardatario ($0 \leq P \leq N - 1$). Se ci sono diverse posizioni equivalenti, la funzione *deve restituire la minore* tra tutte queste. (La posizione P è la posizione del cavaliere ritardatario nella linea risultante, contando a partire da 0. In altre parole, P è il numero di cavalieri che nella linea sono prima del cavaliere ritardatario nella soluzione ottima. In particolare, P = 0 significa che il cavaliere ritardatario si trova all'inizio della linea, mentre P = N - 1 significa che si trova alla fine della linea.)

Subtask 1 [17 punti]

Puoi assumere che $N \leq 500$.

Subtask 2 [32 punti]

Puoi assumere che $N \leq 5\,000$.

Subtask 3 [51 punti]

Puoi assumere che $N \leq 100\,000$.

Dettagli implementativi

Devi inviare esattamente un file, chiamato `tournament.c`, `tournament.cpp` o `tournament.pas`. Questo file deve implementare la funzione descritta sopra con i seguenti prototipi.

Programmi C/C++

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

Programmi Pascal

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

Queste funzioni devono comportarsi come descritto sopra. Ovviamente sei libero di implementare altre funzioni per uso interno. Le tue sottoposizioni non devono interagire in alcun modo con l'input/output standard, né con nessun altro file.

Grader di esempio

Il grader di esempio fornito con l'ambiente del task si aspetta che l'input abbia il seguente formato:

- linea 1: N, C, R;
- linee 2, …, N: K[i];
- linee N + 1, …, N + C: S[i], E[i].

Limiti di tempo e memoria

- Limite di tempo: 1 secondo.
- Limite di memoria: 256 MiB.