Torneio Medieval

Para seu casamento com Beatrice d'Este em 1491, o Duque de Milão Lodovico Sforza pediu que Leonardo organizasse as celebrações, incluindo um grande torneio medieval que durasse por três dias. Mas o cavaleiro mais popular está atrasado...

Torneio

Em um torneio medieval, os N cavaleiros são colocados em uma fila e então suas posições são numeradas de 0 a N - 1 obedecendo à ordem da fila. O mestre do torneio seleciona um *turno* definindo duas posições S e E (onde $0 \leq S < E \leq N - 1$). Todos os cavaleiros cujas posições estão entre S e E (inclusive) competem nesse turno: o vencedor continua no torneio e volta para a sua posição na fila, enquanto os perdedores são eliminados e deixam o campo de jogo. Após, os cavaleiros restantes juntam-se na direção do início da fila, preservando suas ordens relativas na fila, de tal forma que as posições resultantes são de 0 a N - (E - S) - 1. O mestre do jogo seleciona outro turno, repetindo esse processo até que reste apenas um cavaleiro.

Leonardo sabe que todos os cavaleiros têm capacidades distintas, representadas por inteiros distintos de 0 (o mais fraco) a N - 1 (o mais forte). Ele também sabe os comandos exatos que o mestre do torneio vai definir para os C turnos: afinal, ele é Leonardo ... e ele está certo de que em cada um desses turnos o cavaleiro com a maior capacidade vencerá.

Cavaleiro atrasado

N - 1 dos N cavaleiros estão já organizados na fila, apenas o cavaleiro mais popular está faltando. Esse cavaleiro tem capacidade R e está chegando um pouco atrasado. Para beneficiar os espectadores, Leonardo quer explorar a popularidade desse cavaleiro, escolhendo para ele uma posição na fila que maximize o número de turnos que o cavaleiro atrasado irá vencer. Note que não estamos interessados nos turnos que não involvem o cavaleiro atrasado, apenas nos turnos em que ele toma parte e vence.

Exemplo

Para N = 5 cavaleiros, os N - 1 cavaleiros que estão já organizados na fila têm capacidades [1, 0, 2, 4], respectivamente. Consequentemente, o cavaleiro atrasado tem capacidade R = 3. Para os C = 3 turnos, o mestre do torneio quer selecionar as posições (S, E) dos turnos nesta ordem: (1, 3), (0, 1), (0, 1).

Se Leonardo inserir o cavaleiro atrasado na primeira posição, as capacidades dos cavaleiros na fila serão [3, 1, 0, 2, 4]. O primeiro turno envolve cavaleiros (nas posições 1, 2, 3) com capacidades 1, 0, 2, e o vencedor é o cavaleiro com capacidade 2. A nova fila é [3, 2, 4]. O próximo turno é 3

contra 2 (nas posições 0, 1), e o cavaleiro com capacidade R = 3 vence, deixando a fila com [3, 4]. O turno final (nas posições 0, 1) tem 4 como vencedor. Assim, o cavaleiro atrasado vence apenas um turno (o segundo).

No entanto, se Leonardo inserir o cavaleiro atrasado entre os dois cavaleiros de capacidades 1 e 0, a fila fica assim: [1, 3, 0, 2, 4]. Dessa vez, o primeiro turno envolve 3, 0 2, e o cavaleiro com capacidade R = 3 vence. A próxima fila é [1, 3, 4], e no próximo turno (1 contra 3) o cavaleiro com capacidade R = 3 vence novamente. A fila final é [3, 4], e 4 vence. Assim, o cavaleiro atrasado vence dois turnos: esta é na realidade a melhor situação, pois não há maneira de o cavaleiro atrasado vencer mais do que duas vezes.

Enunciado

Sua tarefa é escrever um programa que escolhe a melhor posição para o cavaleiro atrasado de tal maneira que o número de turnos que ele vence é maximizado, como Leonardo deseja. Especificamente, você deve implementar uma função chamada `GetBestPosition`(N, C, R, K, S, E), onde:

- N é o número de cavaleiros;
- C é o número de turnos definido pelo mestre do torneio ($1 \leq C \leq N - 1$);
- R é a capacidade do cavaleiro atrasado; as capacidades de todos os cavaleiros (tanto daqueles já na fila como a do atrasado) são distintas, dentro do intervalo 0, …, N - 1, e a capacidade R do cavaleiro atrasado é dada explicitamente embora possa ser deduzida;
- K é um vetor de N - 1 inteiros, representando as capacidades dos N - 1 cavaleiros que já estão na fila;
- S e E são dois vetores de tamanho C: para cada i entre 0 e C - 1, inclusive, o (i + 1)-ésimo turno definido pelo mestre do torneio envolverá todos os cavaleiros da posição S[i] até a posição E[i], inclusive. Você pode assumir que para cada i, S[i] < E[i].

As chamadas a esta função são válidas: E[i] é menor do que o número corrente de cavaleiros no (i + 1)-ésimo turno, e após todos os C comandos restará exatamente um cavaleiro.

`GetBestPosition`(N, C, R, K, S, E) deve retornar a melhor posição P na qual Leonardo deve colocar o cavaleiro atrasado ($0 \leq P \leq N - 1$). Se há múltiplas posições equivalentes, *retorne a menor delas*. (A posição P é a posição, a partir do 0, do cavaleiro atrasado na fila. Em outras palavras, P é o número de outros cavaleiros posicionados antes do cavaleiro atrasado na solução ótima. Especificamente, P = 0 significa que o cavaleiro atrasado é o primeiro da fila, e P = N - 1 significa que ele é o último da fila.)

Sub-tarefa 1 [17 pontos]

Você pode supor que $N \leq 500$.

Sub-tarefa 2 [32 pontos]

Você pode supor que $N \leq 5\,000$.

Sub-tarefa 3 [51 pontos]

Você pode supor que $N \leq 100\,000$.

Detalhes de implementação

Você deve submeter exatamente um ficheiro, chamado `tournament.c`, `tournament.cpp` ou `tournament.pas`. Esse ficheiro deve implementar o subprograma descrito acima usando as assinaturas seguintes.

Programas C/C++

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

Programas Pascal

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

Esses subprogramas devem comportar-se como descrito acima. Obviamente, você é livre para implementar outros subprogramas de uso interno. Suas submissões não devem interagir de qualquer outro modo com entrada/saída padrão, nem com qualquer outro ficheiro.

Avaliador fornecido

O avaliador fornecido no ambiente da tarefa espera a entrada no seguinte formato:

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

Limites de Tempo e Memória

- Limite de Tempo: 1 segundo.
- Limite de Memória: 256 MiB.