IOI '12 P6 - Jousting Tournament

Time limit: 1.0s **Memory limit:** 256M

For his wedding with Beatrice d'Este in 1491, the Duke of Milan Lodovico Sforza asked Leonardo to orchestrate the wedding celebrations, including a great jousting tournament that lasted for three whole days. But the most popular knight is late...

Tournament

In a jousting tournament, the N knights are first arranged in a line and then their positions are numbered from 0 to $N-1$ following the line order. The joust master sets up a round by calling out two positions S and E (where $0 \leq S < E \leq N-1$). All the knights whose positions are between S and E (inclusive) compete: the winner continues in the tournament and goes back to his place in the line, whereas the losers are out of the game and leave the field. After that, the remaining knights pack together towards the beginning of the line, preserving their relative order in the line, so that their resulting positions are from 0 to $N-(E-S)-1$. The joust master calls out another round, repeating this process until just one knight remains.

Leonardo knows that all the knights have different strengths, represented as distinct ranks from 0 (weakest) to $N-1$ (strongest). He also knows the exact commands that the joust master will call out for the C rounds: he's Leonardo, after all... and he is certain that in each of these rounds the knight with the highest rank will win.

Late Knight

$N-1$ of the N knights are already arranged in the line, just the most popular knight is missing. This knight has rank R and is arriving a bit late. For the benefit of the entertainment, Leonardo wants to exploit his popularity and choose for him a position in the line that will maximize the number of rounds that the late knight will win. Note that we are not interested in the rounds that don't involve the late knight, just in the rounds he takes part in and wins.

Example

For $N=5$ knights, the $N-1$ knights that are already arranged in the line have ranks $[1,0,2,4]$, respectively. Consequently, the late knight has rank $R=3$. For the $C=3$ rounds, the joust master intends to call out the (S,E) positions per round, in this order: $(1,3)$, $(0,1)$, $(0,1)$.

If Leonardo inserts the late knight at the first position, the ranks of the knights on the line will be $[3,1,0,2,4]$. The first round involves knights (at positions 1, 2, 3) with ranks 1, 0, 2, leaving the knight with rank 2 as the winner. The new line is $[3,2,4]$. The next round is 3 against 2 (at positions 0, 1), and the knight with rank $R=3$ wins, leaving the line $[3,4]$. The final round (at positions 0, 1) has 4 as the winner. Thus, the late knight only wins one round (the second one).

Instead, if Leonardo inserts the late knight between those two of ranks 1 and 0, the line looks like this: $[1,3,0,2,4]$. This time, the first round involves 3, 0, 2, and the knight with rank $R=3$ wins. The next starting line is $[1,3,4]$, and in the next round (1 against 3) the knight with rank $R=3$ wins again. The final line is $[3,4]$, where 4 wins. Thus, the late knight wins two rounds: this is actually the best possible placement as there is no way for the late knight to win more than twice.

Statement

Your task is to write a program that chooses the best position for the late knight so that the number of rounds won by him is maximized, as Leonardo wants. Specifically, you have to implement a routine called `GetBestPosition(N, C, R, K, S, E)`, where:

- N is the number of knights;
- C is the number of rounds called out by the joust master ($1 \leq C \leq N-1$);
- R is the rank of the late knight; the ranks of all the knights (both those already lined up and the late one) are distinct and chosen from $0, \ldots, N-1$, and the rank R of the late knight is given explicitly even though it can be deduced;
- K is an array of $N-1$ integers, representing the ranks of the $N-1$ knights that are already on the starting line;
- S and E are two arrays of size C: for each i between 0 and $C-1$, inclusive, the $(i+1)^{\text{th}}$ round called by the joust master will involve all knights from position $S[i]$ to position $E[i]$, inclusive. You may assume that for each i, $S[i] < E[i]$.

The calls passed to this routine are valid: we have that $E[i]$ is less than the current number of knights remaining for the $(i+1)^{\text{th}}$ round, and after all the C commands there will be exactly one knight left.

`GetBestPosition(N, C, R, K, S, E)` must return the best position P where Leonardo should put the late knight ($0 \leq P \leq N-1$). If there are multiple equivalent positions, output the smallest one. (The position P is the 0-based position of the late knight in the resulting line. In other words, P is the number of other knights standing before the late knight in the optimal solution. Specifically, $P = 0$ means that the late knight is at the beginning of the line, and $P = N-1$ means that he is at the end of it.)

Subtask 1 [17 points]

You may assume that $N \leq 500$.

Subtask 2 [32 points]

You may assume that $N \leq 5\,000$.

Subtask 3 [51 points]

You may assume that $N \leq 100\,000$.

Implementation Details

You have to submit exactly one file. This file must implement the subprogram described above using the following signature.

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

This subprogram must behave as described above. Of course you are free to implement other subprograms for their internal use. Your submissions must not interact in any way with standard input/output, nor with any other file.

Sample Grader

The sample grader will expect input in the following format:

- line 1: N, C, R;
- lines $2, \ldots, N$: $K[i]$;
- lines $N+1, \ldots, N+C$: $S[i]$, $E[i]$.

Attachment Package

The sample grader along with sample test cases are available here.