Turniej rycerski

Przed swym ślubem z Beatrice d'Este w 1491 roku, Ludovico Sforza poprosił Leonarda da Vinci o przygotowanie atrakcji weselnych. Jedną z nich miał być wielki turniej rycerski, trwający całe trzy dni. Niestety, najbardziej popularny rycerz się spóźnia...

Turniej

W turnieju rycerskim bierze udział N rycerzy. Są oni ustawieni w szereg; ich pozycje numerujemy od 0 do N - 1, zgodnie z porządkiem w szeregu. Turniej składa się z pewnej liczby *rund*. Mistrz turnieju rozpoczyna rundę, ogłaszając dwie pozycje, S i E (gdzie $0 \leq S < E \leq N - 1$). W tym momencie wszyscy rycerze znajdujący się w szeregu na pozycjach od S do E (włącznie) walczą na kopie; zwycięzca pozostaje w turnieju i wraca na swoje miejsce w szeregu, podczas gdy wszyscy pozostali odpadają z turnieju i opuszczają pole gry. Następnie rycerze pozostający w turnieju przesuwają się ku początkowi szeregu (nie zmieniając względnego porządku), tj. przechodzą na pozycje od 0 do N - (E - S) - 1. Potem mistrz turnieju rozpoczyna kolejną rundę i powtarza tę procedurę, aż w szeregu pozostanie tylko jeden rycerz.

Leonardo wie, że wszyscy rycerze różnią się umiejętnościami, i potrafi przydzielić im parami różne rangi z zakresu od 0 (najsłabszy) do N - 1 (najsilniejszy). Zna on również dokładne polecenia, jakie wyda mistrz turnieju w każdej z C rund — w końcu to Leonardo… Jest on przy tym pewny, że w każdej rundzie wygra rycerz o największej randze.

Spóźniony rycerz

N - 1 spośród N rycerzy stoi już w szeregu, brakuje tylko najpopularniejszego zawodnika. Rycerz ten ma rangę R i przybędzie odrobinę spóźniony. Aby ucieszyć publikę, Leonardo chciałby wykorzystać jego popularność i ustawić go na pozycji w szeregu, która zmaksymalizuje liczbę wygranych przez niego rund. Zauważ, że nie liczymy rund, w których spóźniony rycerz nie walczy; interesują nas tylko rundy, w których bierze on udział i wygrywa.

Przykład

Rozważmy sytuację, w której N = 5 i N - 1 rycerzy ustawionych w szeregu ma kolejno rangi [1, 0, 2, 4]. Spóźniony rycerz ma zatem rangę R = 3. Przez C = 3 rundy mistrz turnieju zamierza wywołać kolejno następujące pary pozycji (S, E): (1, 3), (0, 1), (0, 1).

Jeżeli Leonardo umieści spóźnionego rycerza na pierwszej pozycji, rangi rycerzy w szeregu będą wynosiły kolejno [3, 1, 0, 2, 4]. W pierwszej rundzie walczą rycerze z pozycji 1, 2, 3, o rangach 1, 0, 2. W turnieju pozostaje więc rycerz o randze 2, a nowy szereg to [3, 2, 4]. W kolejnej rundzie walczą rycerze o rangach 3 i 2 (z pozycji 0, 1) i zwycięża rycerz o randze R = 3. Przed trzecią

rundą szereg to [3, 4]. W ostatniej walce (pozycje 0, 1) wygrywa rycerz o randze 4. Spóźniony rycerz wygrał zatem tylko jedną rundę (drugą).

Jeżeli zamiast tego Leonardo umieści spóźnionego rycerza pomiędzy tymi o rangach 1 i 0, szereg będzie wyglądał następująco: [1, 3, 0, 2, 4]. Tym razem w pierwszej rundzie biorą udział rycerze o rangach 3, 0, 2 i rycerz o randze R = 3 wygrywa. Na początku kolejnej rundy szereg to [1, 3, 4] i w walce (1 przeciwko 3) ponownie wygrywa rycerz o randze 3. Ostatni szereg to [3, 4] i trzecią rundę wygrywa rycerz o randze 4. Spóźniony rycerz wygrywa więc dwie rundy; okazuje się, że jest to najlepsze możliwe rozwiązanie, gdyż w tej sytuacji nie da się umieścić spóźnionego rycerza tak, aby wygrał więcej niż dwa razy.

Zadanie

Napisz program, który wybierze najlepszą pozycję dla spóźnionego rycerza, tak aby zmaksymalizować liczbę wygranych przez niego rund i zrealizować plan Leonarda.

Zaimplementuj funkcję `GetBestPosition`(N, C, R, K, S, E), gdzie:

- N to liczba rycerzy;
- C to liczba rund zaplanowanych przez mistrza turnieju ($1 \leq C \leq N - 1$);
- R to ranga spóźnionego rycerza; rangi wszystkich rycerzy (tych stojących w szeregu oraz tego spóźnionego) są parami różne i należą do zbioru 0, …, N - 1. Ranga R spóźnionego rycerza jest dana wprost, choć można ją wywnioskować na podstawie pozostałych danych;
- K to tablica N - 1 liczb całkowitych reprezentujących rangi N - 1 rycerzy, którzy stoją już w szeregu;
- S i E to dwie tablice rozmiaru C; dla każdego i od 0 do C - 1 włącznie, w (i + 1)-szej rundzie rozpoczętej przez mistrza turnieju wezmą udział wszyscy rycerze z pozycji od S[i] do E[i] włącznie. Możesz założyć, że dla każdego i zachodzi S[i] < E[i].

Możesz założyć, że wywołania opisanej funkcji będą poprawne: E[i] będzie zawsze mniejsze niż liczba rycerzy pozostałych do rundy (i + 1)-szej, a po wszystkich C rundach pozostanie dokładnie jeden rycerz.

`GetBestPosition`(N, C, R, K, S, E) musi zwrócić najlepszą pozycję P, na której Leonardo powinien umieścić spóźnionego rycerza ($0 \leq P \leq N - 1$). Jeżeli istnieje wiele równie dobrych pozycji, *zwróć najmniejszą z nich*. (Pozycja P to pozycja spóźnionego rycerza w wynikowym szeregu na początku turnieju, indeksowana od 0. Innymi słowy, P jest równe liczbie rycerzy stojących przed spóźnionym rycerzem w optymalnym rozwiązaniu. W szczególności P = 0 oznacza, że spóźniony rycerz stoi na początku szeregu, a P = N - 1 oznacza, że stoi na jego końcu.)

Podzadanie 1 [17 punktów]

- $N \leq 500$

Podzadanie 2 [32 punkty]

- $N \leq 5\,000$

Podzadanie 3 [51 punktów]

- $N \leq 100\,000$

Szczegóły implementacji

Należy zgłosić dokładnie jeden plik o nazwie `tournament.c`, `tournament.cpp` lub `tournament.pas`. Powinien on zawierać implementację opisanej powyżej funkcji.

Programy w C/C++

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

Programy w Pascalu

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

Funkcja powinna działać dokładnie tak, jak opisano powyżej. Twój program nie powinien korzystać ze standardowego wejścia, standardowego wyjścia lub jakichkolwiek plików.

Przykładowy moduł oceniający

Przykładowy moduł oceniający wczytuje dane w następującym formacie:

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

Ograniczenia

- Maksymalny czas działania: 1 sekunda.
- Dostępna pamięć: 256 MiB.