Bruņinieku turnīrs

Milānas hercogs Lodviko Sforza uz savām kāzām ar Beatrisi d'Esti 1491. gadā ir lūdzis Leonardo sarīkot svinības, kas iekļauj arī varenas trīs dienu ilgas bruņinieku cīņas. Taču tautā vispopulārākais bruņinieks ir aizkavējies.

Turnīrs

Bruņinieku turnīrā N bruņinieki nostājas vienā rindā un to pozīcijas tiek sanumurētas no 0 līdz N - 1. Turnīra vadītājs izsludina sacensību *kārtu*, nosaucot divas pozīcijas S un E (kur $0 \leq S < E \leq N - 1$). Tad visi bruņinieki, kuru pozīcijas ir starp S un E (ieskaitot), sacenšas savā starpā: uzvarētājs turpina savu dalību turnīrā un atgriežas atpakaļ savā vietā rindā, bet zaudētāji tiek izslēgti no turnīra un pamet norises vietu. Pēc tam atlikušie bruņinieki sabīdās ciešāk uz rindas sākumu, saglabājot relatīvo secību, tā, ka to rezultējošās pozīcijas ir no 0 līdz N - (E - S) - 1. Turnīra vadītājs izsludina jaunus raundus, atkārtojot procesu tik ilgi, kamēr turnīrā ir palicis tieši viens bruņinieks.

Leonardo zin, ka bruņinieki ir dažādi spēcīgi, kas tiek raksturots ar dažādiem rangiem no 0 (vājākais) līdz N - 1 (stiprākais). Viņš zin arī to, kādas tieši komandas turnīra vadītājs nosauks katrā no C turnīra raundiem (jo galu galā, to dara pats Leonardo). Visbeidzot, viņš ir pārliecināts par to, ka katrā raundā uzvarēs bruņinieks ar visaugstāko rangu no tiem, kas tajā piedalās.

Nokavējušais bruņinieks

N - 1 no N bruņiniekiem jau ir ieradušies un sastājušies rindā, tikai viena - vispopulārākā - bruņinieka vēl nav. Šī bruņinieka rangs ir R un viņš nedaudz kavē. Skatītāju priekam Leonardo vēlas izmantot šī bruņinieka popularitāti un izvēlēties viņam tādu pozīciju rindā, kas maksimizēs to raundu skaitu, kurus uzvarēs nokavējušais bruņinieks. Ievērojiet, ka mūs neinteresē tie raundi, kuros nokavējušais bruņinieks nemaz nepiedalās.

Piemērs

Turnīrā piedalās N = 5 bruņinieki. N - 1 bruņinieku, kas jau ir nostājušies rindā, rangi ir atbilstoši [1, 0, 2, 4]. Nokavējušā bruņinieka rangs ir R = 3. Turnīra C = 3 raundos vadītājs nosauks šādas (S, E) pozīcijas: (1, 3), (0, 1), (0, 1).

Ja Leonardo novietos nokavējušos bruņinieku rindas sākumā (0-tajā pozīcijā), bruņinieku rangi rindā būs attiecīgi [3, 1, 0, 2, 4]. Pirmajā raundā piedalīsies bruņinieki ar rangiem 1, 0, 2 (no pozīcijām 1, 2, 3); bruņinieks ar rangu 2 būs uzvarētājs. Jaunā bruņinieku rinda būs [3, 2, 4]. Nākamajā raundā bruņinieks ar rangu 3 cīnīsies pret bruņinieku ar rangu 2 (pozīcijas 0 un 1), un bruņinieks ar rangu R = 3 uzvarēs. Pēc šī raunda rinda būs [3, 4]. Pēdējā raundā bruņinieks ar rangu 4 būs uzvarētājs. Tātad, nokavējušais bruņinieks uzvarēs tikai vienā raundā (otrajā pēc

kārtas).

Taču, ja Leonardo novietos nokavējušos bruņinieku starp bruņiniekiem ar rangiem 1 un 0, rinda izskatīsies šādi: [1, 3, 0, 2, 4]. Šoreiz, pirmajā raundā piedalīsies 3, 0, 2, un bruņinieks ar rangu R = 3 uzvarēs. Pēc tam rinda būs [1, 3, 4], un nākamajā raundā (1 pret 3) bruņinieks ar rangu R = 3 atkal uzvarēs. Visbeidzot rinda būs [3, 4] un pēdējā raundā uzvarēs bruņinieks ar rangu 4. Kopumā nokavējušais bruņinieks uzvarēs divus raundus. Šis arī ir vislabākais nokavējušā bruņinieka novietojums, jo viņam nav iespējams uzvarēt vairāk kā divas reizes, sākot turnīru jebkurā citā pozīcijā.

Uzdevums

Jūsu uzdevums ir uzrakstīt programmu, kas izvēlas labāko nokavējušā bruņinieka sākuma pozīciju, kas maksimizē to raundu skaitu, kurus viņš uzvar. Precīzāk, Jums jāimplementē funkcija ar nosaukumu `GetBestPosition`(N, C, R, K, S, E), kur:

- N ir bruņinieku skaits;
- C ir turnīra vadītāja izsludināto raundu skaits ($1 \leq C \leq N - 1$);
- R ir nokavējušā bruņinieka rangs; visu bruņinieku (gan tos, kas stāv rindā, gan nokavējušā) rangi ir atšķirīgi un no intervāla 0, .., N - 1, un rangs R ir dots atsevišķi, lai gan tā vērtība var tikt izsecināta no rindā stāvošo bruņinieku rangiem;
- K ir masīvs ar N - 1 skaitli, kas raksturo N - 1 sākuma rindā stāvošo bruņinieku rangus;
- S un E ir divi masīvi ar izmēru C: katram i starp 0 un C - 1, ieskaitot, (i+1)-ajā turnīra vadītāja izsludinātajā raundā piedalīsies bruņinieki, kas atrodas pozīcijās starp S[i] un E[i], ieskaitot. Jūs varat pieņemt, ka S[i] < E[i] katram i.

Funkcijai padotie turnīra raundu izsaukumi ir pareizi: ir zināms, ka E[i] ir mazāks par bruņinieku skaitu, kas vēl palikuši turnīrā pēc i raundiem, kā arī tas, ka pēc visām C komandām palicis pāri būs tieši viens bruņinieks.

Funkcijai `GetBestPosition`(N, C, R, K, S, E) ir jāatgriež labākā pozīcija P, kurā Leonardo vajag novietot nokavējušo bruņinieku ($0 \leq P \leq N - 1$). Ja ir vairākas vienādi labas pozīcijas, *izvadiet mazāko no tām*. (Pozīcija P ir 0-indeksēta nokavējušā bruņinieka pozīcija rezultējošajā rindā. Citiem vārdiem, P ir citu pirms viņa stāvošo bruņinieku skaits optimālajā risinājumā. Precīzāk, P = 0 nozīmē, ka nokavējušais bruņinieks atrodas rindas sākumā, P = N - 1 nozīmē, ka viņš atrodas rindas beigās.)

Apakšuzdevums 1 [17 punkti]

Jūs varat pieņemt, ka $N \leq 500$.

Apakšuzdevums 2 [32 punkti]

Jūs varat pieņemt, ka $N \leq 5\,000$.

Apakšuzdevums 3 [51 punkti]

Jūs varat pieņemt, ka $N \leq 100\,000$.

Implementācijas detaļas

Jums ir jāiesūta viens fails ar nosaukumu `tournament.c`, `tournament.cpp` vai `tournament.pas`. Šajā failā jābūt realizētai augstāk aprakstītajai apakšprogramai ar šādu signatūru.

C/C++ programmas

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

Pascal programmas

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

Apakšprogrammai jādarbojas iepriekš aprakstītajā veidā. Protams, varat implementēt arī citas apakšprogrammas. Iesūtītās programmas nedrīkst neko rakstīt / lasīt no standarta izvada / ievada vai kāda cita faila.

Paraugvērtētājs

Piedāvātā paraugvērtētāja ievaddatiem jābūt šādā formā:

- 1. rinda: N, C, R;
- Katrā no nākamajām N-1 rindām: K[i];
- Katrā no nākamajām C rindām: S[i], E[i].

Izpildes laika un atmiņas ierobežojumi

- Izpildes laika ierobežojums: 1 sekunde.
- Atmiņas ierobežojums: 256 MiB.