D. Planificare (seatingplan)

Ceremonia de închidere a acestui EGOI va fi onorată de prezența a N invitați foarte importanți. Aceștia trebuie așezați în primul rând în exact o anumită ordine, ce corespunde tuturor conduitelor de protocol și bune maniere. Determinarea ordinii optime în care invitații trebuia așezați a însemnat pentru Noemi două nopți nedormite.

Veronica supraveghează ceremonia de închidere. Una dintre numeroasele ei responsabilități este să se asigure că locurile din primul rând au etichetele cu numele corecte. Există doar o mică problemă: Noemi nu i-a spus ordinea corectă de așezare, iar acum Noemi este de negăsit. Din fericire, Dorka, fotografa, are o aplicație care s-ar putea dovedi utilă.

Dorka a trebuit să își pregătească aparatele foto pentru a putea face niște fotografii specifice invitaților din primul rând. Pentru configurare, a avut nevoie să știe lățimea minimă necesară pentru fiecare fotografie, așa că Noemi i-a făcut o aplicație care îi oferă rapid informațiile de care are nevoie. Veronica vrea acum să folosească aplicația pentru a găsi aranjarea corectă a locurilor.

Cei N invitați importanți sunt numerotați de la 0 la $N-1$. Locurile din primul rând sunt, de asemenea, numerotate de la 0 la $N-1$, de la stânga la dreapta. Pentru fiecare I ($0 \leq I \leq N-1$), fie g_I invitatul care ar trebui să stea pe locul I, și fie s_I locul pe care ar trebui să stea invitatul I.

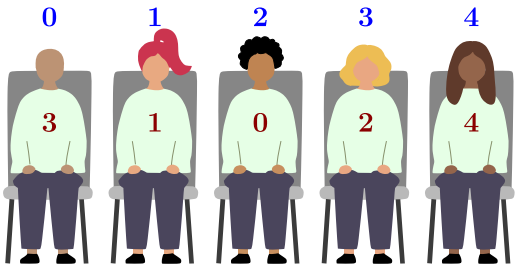

Figura 1: Un rând cu cinci invitați. Pentru acest rând, $g = [3, 1, 0, 2, 4]$ și $s = [2, 1, 3, 0, 4]$.

Aplicația funcționează astfel:

- Dorka introduce numerele I, J, K a trei invitați diferiți.
- Aplicația îi spune numărul minim de invitați care vor fi vizibili dacă toți cei trei invitați selectați sunt în fotografie. Formal, aplicația va afișa valoarea $\max(s_I, s_J, s_K) - \min(s_I, s_J, s_K) + 1$.

De exemplu, pentru situația prezentată în Figura 1:

- Invitații $I = 0$, $J = 2$ și $K = 4$ stau pe locurile $s_I = 2$, $s_J = 3$ și $s_K = 4$. Dacă Dorka îi selectează, aplicația va afișa valoarea $\max(2, 3, 4) - \min(2, 3, 4) + 1 = 3$.

  Cu alte cuvinte, cea mai îngustă fotografie care îi conține pe invitații 0, 2 și 4 are doar acești trei invitați.

- Invitații $I = 0$, $J = 4$ și $K = 3$ stau pe locurile $s_I = 2$, $s_J = 4$ și $s_K = 0$. Dacă Dorka îi selectează, aplicația va afișa valoarea $\max(2, 4, 0) - \min(2, 4, 0) + 1 = 5$.

  Cu alte cuvinte, o fotografie care îi conține pe cei trei invitați dați trebuie să îi conțină pe toți cei 5 invitați.

Ajut-o pe Veronica să determine ordinea corectă de așezare folosind aplicația lui Dorka. Mai exact, programul tău trebuie să determine și să afișeze secvența $g_0, g_1, ..., g_{N-1}$. Există întotdeauna exact două răspunsuri corecte (unul fiind oglinditul celuilalt), și îl poți afișa pe oricare dintre ele. Scorul tău va depinde de numărul de interogări pe care soluția ta le face către aplicație.

Implementare

⇨ Aceasta este o problemă interactivă. Programul tău va folosi intrarea și ieșirea standard pentru a comunica cu un grader în formatul descris mai jos.

Programul tău ar trebui să înceapă prin a citi de la intrare un număr întreg pozitiv T, numărul de teste care urmează în input.

Pentru fiecare test, programul tău ar trebui să înceapă prin a citi un număr întreg pozitiv N, numărul de locuri, care este și numărul de invitați.

Pentru a face o interogare, programul tău trebuie să afișeze o linie de forma „? I J K”, unde $0 \leq I, J, K \leq N-1$ sunt trei numere **distincte**.

După ce faci o interogare, programul tău trebuie să citească o linie care conține un număr întreg pozitiv, răspunsul la interogarea ta.

Pentru a răspunde cu o ordine de așezare corectă, programul tău trebuie să afișeze o linie de forma „! g_0 ⋯ g_{N-1}”.

După rezolvarea tuturor celor T cazuri de test, programul tău ar trebui să se termine normal.

Reține că grader-ul oficial folosit în CMS pentru a-ți testa soluția poate fi **adaptiv**. Adică, pentru unele cazuri de test, permutarea invitaților nu este determinată dinainte. În schimb, grader-ul își poate decide pe parcurs care va fi răspunsul dintre permutările posibile rămase în urma interogărilor specifice făcute de programul tău până la momentul respectiv.

Golirea buffer-ului (Flushing). Dacă nu folosești template-urile furnizate, asigură-te că golești buffer-ul de ieșire (flush) după afișarea fiecărei linii, altfel programul tău ar putea fi judecat ca fiind *Incorect* (*Not correct*). În Python, acest lucru se întâmplă automat dacă folosești `input()` pentru a citi linii sau poți folosi `print(..., flush=True)` pentru a forța golirea buffer-ului de ieșire. În C++, `cout << endl;` golește buffer-ul pe lângă afișarea unei linii noi; dacă folosești printf, folosește `fflush(stdout)`.

Constrângeri

- $1 \leq T \leq 10$.
- N va fi 5 (doar exemplu), 8, 40, sau 2000.
- Pentru fiecare test, poți folosi cel mult 10 000 interogări.

Punctaj

Programul tău va fi rulat pe mai multe date de intrare grupate în subtask-uri. Pentru a obține scorul pentru un subtask, trebuie să rezolvi corect toate testele pe care le conține.

- **Subtask-ul 0 [0 puncte]**: Exemplu ($N = 5$).
- **Subtask-ul 1 [9 puncte]**: $N = 8$.
- **Subtask-ul 2 [11 puncte]**: $N = 2000$, și invitații 0 și 1 stau unul lângă altul.
- **Subtask-ul 3 [15 puncte]**: $N = 40$.
- **Subtask-ul 4 [65 puncte]**: $N = 2000$.

Pentru subtask-urile 1 și 2, orice soluție care rezolvă corect toate testele va primi punctajul maxim.

Pentru subtask-urile 3 și 4, soluția ta trebuie să rezolve corect toate testele pentru a primi puncte, iar scorul tău va depinde de Q_s, numărul maxim de interogări pe care soluția ta a trebuit să le facă pentru a rezolva un test. Fie $X_s = \max(1, Q_s/N)$. Scorurile pentru subtask-urile 3 și 4 sunt calculate astfel:

$$\text{scor}_3 = \min\left(15, 3 + \frac{19}{X_s^{1.5}}\right), \qquad \text{scor}_4 = \min\left(65, 3 + \frac{91}{X_s^{1.5}}\right)$$

Valoarea scor_s este rotunjită la cel mai apropiat întreg pe subtask, iar scorul tău total este suma acestora. Pentru a obține punctajul maxim, trebuie să rezolvi subtask-ul 3 în cel mult 55 de interogări și subtask-ul 4 în cel mult 2597 de interogări. Valori exemplificative pentru Q_s și scorurile pentru subtask-urile 3 și 4 sunt prezentate mai jos.

Q_s	55	56	60	70	80	100	150	10000
scor_3	15	14	13	11	10	8	6	3

Q_s	2597	2800	3000	4000	5000	6000	8000	10000
scor_4	65	58	53	35	26	21	14	11

Exemple de intrare/ieșire

Grader	Soluție
1 5	
	? 0 2 4
3	
	? 3 0 1
3	
	? 0 4 3
5	
	! 3 1 0 2 4

Explicație

Exemplul de intrare conține un singur test ($T = 1$) cu $N = 5$ invitați. Configurația ascunsă a invitaților din acest caz de test corespunde cu Figura 1.

Prima interogare făcută de soluția exemplu este 0, 2, 4. Răspunsul 3 la această interogare ne spune că acești invitați stau, într-o ordine necunoscută, pe trei locuri consecutive, unul lângă altul.

Răspunsul 3 la a doua interogare ne spune același lucru despre invitații 3, 0 și 1.

Acum putem deduce că invitatul 0 trebuie să stea la mijloc, cu invitații 2 și 4 pe o parte și invitații 1 și 3 pe cealaltă parte.

După a treia interogare, putem fi deja siguri că invitații trebuie să stea fie în ordinea $[3, 1, 0, 2, 4]$, fie în ordinea inversă $[4, 2, 0, 1, 3]$. Putem afișa oricare dintre aceste ordini.

Template-uri de cod și detalii de evaluare în CMS

Recomandăm insistent utilizarea template-urilor de cod furnizate pentru C++ și Python. Acestea verifică dacă comunicarea cu evaluatorul a avut succes și se termină elegant atunci când nu a avut.

Dacă nu folosești template-ul de cod oferit, în unele cazuri când soluția ta nu este corectă, CMS-ul ar putea afișa alt verdict decât cel potrivit. De exemplu, în loc de „Output isn't correct", verdictul afișat ar putea fi „Execution killed by signal" sau „Execution timed out (wall clock limit exceeded)".

De asemenea, recomandăm utilizarea instrumentului de testare local (vezi mai jos) pentru a-ți testa soluția înainte de a o trimite. Instrumentul de testare verifică ceea ce afișează soluția ta și raportează eventuale interogări invalide.

Instrument de testare

Pentru a facilita testarea soluției tale, oferim un instrument simplu pe care îl poți descărca din CMS. Instrumentul este opțional. Reține că evaluatorul oficial din CMS este diferit de instrumentul de testare.

Pentru a folosi instrumentul, ai nevoie de un fișier de intrare. Poți folosi exemplul de intrare furnizat `seatingplan.input0.txt` sau poți face unul propriu. Fișierul de intrare ar trebui să înceapă cu o linie care are numărul T de teste, apoi ar trebui să aibă două linii per test: o linie cu numărul N și apoi o linie cu numerele $g_0, g_1, ..., g_{N-1}$.

Pentru programele Python, de exemplu `seatingplan.py` (rulată de obicei ca `pypy3 seatingplan.py`) rulează instrumentul de testare după cum urmează:

```
python3 testing_tool.py pypy3 seatingplan.py < seatingplan.input0.txt
```

Pentru programele C++, mai întâi compilează soluția:

```
g++ -DEVAL -std=gnu++20 -O2 -pipe -static -s -o seatingplan seatingplan.cpp
```

și apoi rulează instrumentul de testare:

```
python3 testing_tool.py ./seatingplan < seatingplan.input0.txt
```