D. Sitzordnung (seatingplan)

An der Abschlusszeremonie dieser EGOI nehmen N wichtige Gäste teil. Diese müssen in der vordersten Reihe in einer ganz bestimmten Reihenfolge sitzen, die alle diplomatischen Feinheiten abdeckt. Noemi hat zwei schlaflose Nächte damit verbracht, die korrekte Sitzordnung festzulegen.

Veronica beaufsichtigt die Abschlusszeremonie. Eine ihrer vielen Aufgaben ist es, sicherzustellen, dass die Plätze in der vordersten Reihe die richtigen Namensschilder haben. Es gibt nur ein winziges Problem: Noemi hat ihr nie die korrekte Sitzordnung verraten und jetzt ist Noemi nirgendwo zu finden. Zum Glück hat Dorka, die Fotografin, eine App, die nützlich sein könnte.

Dorka musste ihre Kameras vorbereiten, um bestimmte Fotos von Gästen in der ersten Reihe zu machen. Für das Setup musste sie wissen, wie breit jedes Foto sein würde. Noemi hat ihr deswegen eine App gemacht, die schnell die Informationen ausgibt, die sie braucht. Veronica möchte die App nun nutzen, um die korrekte Sitzbelegung zu finden.

Die N wichtigen Gäste sind von 0 bis $N-1$ nummeriert. Die Plätze in der vordersten Reihe sind ebenfalls von links nach rechts von 0 bis $N-1$ nummeriert. Für jedes I $(0 \leq I \leq N-1)$ bezeichnet g_I den Gast, der auf Platz I sitzen soll, und s_I den Platz, auf dem Gast I sitzen soll.

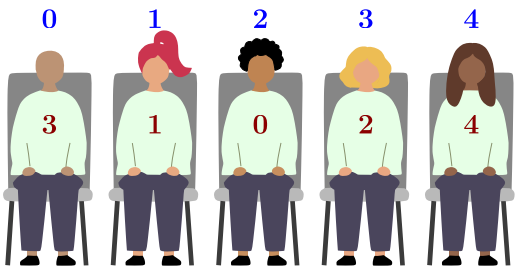

Abbildung 1: Eine Reihe mit fünf Gästen. Für diese Reihe gilt $g = [3, 1, 0, 2, 4]$ und $s = [2, 1, 3, 0, 4]$.

Die App funktioniert wie folgt:

- Dorka gibt die Nummern I, J, K von genau drei verschiedenen Gästen ein.
- Die App sagt ihr die minimale Anzahl von Gästen, die sichtbar sein werden, wenn alle drei ausgewählten Gäste auf dem Foto sind. Formal zeigt die App den Wert $\max(s_I, s_J, s_K) - \min(s_I, s_J, s_K) + 1$ an.

Zum Beispiel, siehe die Situation in Abbildung 1:

- Die Gäste $I = 0$, $J = 2$ und $K = 4$ sitzen auf den Plätzen $s_I = 2$, $s_J = 3$ und $s_K = 4$. Wenn Dorka diese auswählt, zeigt die App den Wert $\max(2, 3, 4) - \min(2, 3, 4) + 1 = 3$ an.

  Anders gesagt, das schmalste Foto, das die Gäste 0, 2 und 4 enthält, hat genau diese drei Gäste.

- Die Gäste $I = 0$, $J = 4$ und $K = 3$ sitzen auf den Plätzen $s_I = 2$, $s_J = 4$ und $s_K = 0$. Wenn Dorka diese auswählt, zeigt die App den Wert $\max(2, 4, 0) - \min(2, 4, 0) + 1 = 5$ an.

  Anders gesagt, ein Foto, das die drei angegebenen Gäste enthält, muss alle 5 Gäste enthalten.

Hilf Veronica dabei, die korrekte Sitzordnung mithilfe von Dorkas App zu bestimmen. Genauer gesagt sollte dein Programm die Sequenz $g_0, g_1, ..., g_{N-1}$ bestimmen und ausgeben. Es gibt immer genau

zwei richtige Antworten (eine ist die Umkehrung der anderen), und du darfst eine beliebige davon ausgeben. Deine Punktzahl hängt von der Anzahl der Anfragen ab, die deine Lösung an die App stellt.

Implementierung

⇨ Dies ist ein interaktives Problem. Dein Programm verwendet die Standardeingabe und -ausgabe, um mit einem Grader in dem unten beschriebenen Format zu kommunizieren.

Dein Programm sollte mit dem Einlesen einer Zeile beginnen, die eine positive Ganzzahl T enthält, die Anzahl der folgenden Testfälle.

Für jeden Testfall sollte dein Programm mit dem Einlesen einer Zeile beginnen, die eine positive Ganzzahl N enthält, die Anzahl der Plätze, was auch der Anzahl der Gäste entspricht.

Um eine Anfrage zu stellen, sollte dein Programm eine Zeile der Form „? I J K“ ausgeben, wobei $0 \leq I, J, K \leq N-1$ drei **verschiedene** Nummern sind.

Nachdem du eine Anfrage gestellt hast, sollte dein Programm eine Zeile mit einer positiven Ganzzahl einlesen, die die Antwort auf deine Anfrage ist.

Um mit einer korrekten Sitzordnung zu antworten, sollte dein Programm eine Zeile der Form „! g_0 … g_{N-1}“ ausgeben.

Nachdem alle T Testfälle gelöst wurden, sollte dein Programm normal beendet werden.

Beachte, dass der offizielle Grader, der in CMS verwendet wird, um deine Lösung zu testen, **adaptiv** sein kann. Das heisst, für manche Testfälle steht die Permutation der Gäste nicht im Voraus fest. Stattdessen kann der Grader sein Verhalten ändern, abhängig von den spezifischen Anfragen, die dein Programm stellt.

Flushing. Wenn du nicht die bereitgestellten Vorlagen verwendest, stelle sicher, dass du die Standardausgabe nach dem Ausgeben jeder Zeile flushst, da dein Programm sonst als *Not correct* bewertet werden könnte. In Python passiert dies automatisch, wenn du `input()` verwendest, um Zeilen zu lesen, und du kannst `print(..., flush=True)` verwenden, um das Flushen zu erzwingen. In C++ flusht `cout << endl;` zusätzlich zum Drucken eines Zeilenumbruchs; wenn du printf verwendest, benutze `fflush(stdout)`.

Einschränkungen

- $1 \leq T \leq 10$.
- N ist 5 (nur im Beispiel), 8, 40 oder 2000.
- Für jeden Testfall darfst du maximal 10 000 Anfragen stellen.

Bewertung

Dein Programm wird auf mehreren Testfällen getestet, die in Teilaufgaben gruppiert sind. Um die Punkte für eine Teilaufgabe zu erhalten, musst du alle darin enthaltenen Tests korrekt lösen.

- **Teilaufgabe 0** [**0 Punkte**]: Beispiel ($N = 5$).
- **Teilaufgabe 1** [**9 Punkte**]: $N = 8$.
- **Teilaufgabe 2** [**11 Punkte**]: $N = 2000$, und die Gäste 0 und 1 sitzen nebeneinander.
- **Teilaufgabe 3** [**15 Punkte**]: $N = 40$.
- **Teilaufgabe 4** [**65 Punkte**]: $N = 2000$.

Für die Teilaufgaben 1 und 2 erhält jede Lösung, die alle Testfälle korrekt löst, die volle Punktzahl.

Für die Teilaufgaben 3 und 4 muss deine Lösung alle Testfälle korrekt lösen, um überhaupt Punkte zu erhalten. Deine Punktzahl hängt dann von Q_s ab, der grössten Anzahl an Anfragen, die deine Lösung für einen Testfall benötigt hat. Sei $X_s = \max(1, Q_s/N)$. Die Punktzahlen für die Teilaufgaben 3 und 4 werden dann wie folgt berechnet:

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

Der Wert von score_s wird pro Teilaufgabe auf die nächste Ganzzahl gerundet, und deine Gesamtpunktzahl ist die Summe dieser Werte. Um eine volle Punktzahl zu erreichen, musst du Teilaufgabe 3 in maximal 55 Anfragen und Teilaufgabe 4 in maximal 2597 Anfragen lösen. Beispielwerte für Q_s und die Punktzahlen für die Teilaufgaben 3 und 4 sind hier dargestellt.

Q_s	55	56	60	70	80	100	150	10000
score_3	15	14	13	11	10	8	6	3

Q_s	2597	2800	3000	4000	5000	6000	8000	10000
score_4	65	58	53	35	26	21	14	11

Beispiele

Grader	Lösung
`1` `5`	
	`? 0 2 4`
`3`	
	`? 3 0 1`
`3`	
	`? 0 4 3`
`5`	
	`! 3 1 0 2 4`

Erklärung

Die Beispiel-Eingabe enthält einen Testfall ($T = 1$) mit $N = 5$ Gästen. Die versteckte Konfiguration der Gäste in diesem Testfall entspricht Abbildung 1.

Die erste Anfrage der Beispiel-Lösung ist 0, 2, 4. Die Antwort 3 auf diese Anfrage sagt uns, dass diese Gäste in irgendeiner unbekannten Reihenfolge auf drei aufeinanderfolgenden Plätzen nebeneinander sitzen.

Die Antwort 3 auf die zweite Anfrage sagt uns dasselbe über die Gäste 3, 0 und 1.

Wir können nun ableiten, dass Gast 0 in der Mitte sitzen muss, mit den Gästen 2 und 4 auf der einen Seite und den Gästen 1 und 3 auf der anderen Seite.

Nach der dritten Anfrage können wir uns bereits sicher sein, dass die Gäste entweder in der Reihenfolge $[3, 1, 0, 2, 4]$ oder in der umgekehrten Reihenfolge $[4, 2, 0, 1, 3]$ sitzen müssen. Wir können eine der beiden Reihenfolgen ausgeben.

Code-Vorlagen und Details zur Bewertung in CMS

Wir empfehlen stark, die bereitgestellten Code-Vorlagen für C++ und Python zu verwenden. Diese überprüfen, ob die Kommunikation mit dem Grader erfolgreich war, und beenden das Programm sauber, falls dies nicht der Fall war.

Wenn du die bereitgestellten Vorlagen nicht verwendest, zeigt CMS bei einer falschen Lösung möglicherweise die Bewertung inkorrekt an. Zum Beispiel erhältst du statt „Output isn't correct" möglicherweise „Execution killed by signal" oder „Execution timed out (wall clock limit exceeded)".

Wir empfehlen ausserdem das Test-Tool (siehe unten), um deine Lösung lokal zu testen, bevor du sie einreichst. Das Test-Tool überprüft die Ausgaben deiner Lösung und meldet die Verwendung ungültiger Abfragen.

Test-Tool

Um das Testen deiner Lösung zu erleichtern, stellen wir ein einfaches Tool bereit, das du von CMS herunterladen kannst. Das Tool zu nutzen ist optional. Beachte, dass der offizielle Grader auf CMS sich von dem Test-Tool unterscheidet.

Um das Tool zu verwenden, benötigst du eine Eingabedatei. Du kannst die bereitgestellte Beispiel-Eingabedatei `seatingplan.input0.txt` verwenden oder eine eigene erstellen. Die Eingabedatei sollte mit einer Zeile beginnen, die die Anzahl T der Testfälle enthält, und dann pro Testfall zwei Zeilen haben: eine Zeile mit der Anzahl N und dann eine Zeile mit den Zahlen $g_0, g_1, ..., g_{N-1}$.

Für Python-Programme, z.B. `seatingplan.py` (normalerweise ausgeführt als `pypy3 seatingplan.py`), führe das Test-Tool wie folgt aus:

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

Für C++-Programme, kompiliere zuerst deine Lösung:

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

und führe dann das Test-Tool aus:

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