D. План за сядане (seatingplan)

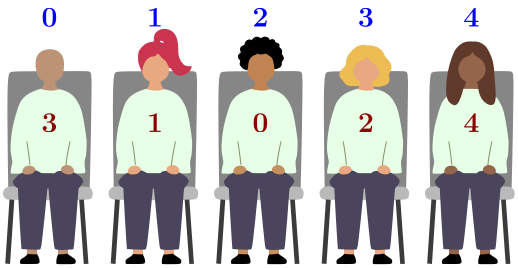
Ограничение по време: 4 секунди
Ограничение по памет: 1024 MiB

На церемонията по закриването на EGOI ще присъстват N важни гости. Всички те трябва да бъдат настанени на първия ред в много специфичен ред, който отговаря на всички нюанси на дипломатическия протокол. Определянето на правилния ред на сядане костваше на Ноеми две безсънни нощи.

Вероника ръководи церемонията по закриването. Една от многото ѝ отговорности е да се увери, че местата на първия ред имат правилните табелки с имена. Има само един малък проблем: Ноеми никога не ѝ е казала правилния ред на сядане, а сега Ноеми е в неизвестност. За щастие, фотографът Дорка има приложение, което може да бъде полезно.

Дорка трябваше да подготви камерите си, за да може да направи някои специфични снимки на гостите на първия ред. За настройката тя трябваше да знае колко широка ще бъде всяка снимка, затова Ноеми ѝ направи приложение, което бързо извежда информацията, от която тя се нуждае. Вероника сега иска да използва приложението, за да открие правилното разпределение на местата.

N-те важни гости са номерирани от 0 до $N-1$. Местата на първия ред също са номерирани от 0 до $N-1$, от ляво надясно. За всяко I $(0 \leq I \leq N-1)$, нека g_I означава госта, който трябва да седне на място I, а s_I означава мястото, на което трябва да седне гост I.



Фиг. 1: Ред с петима гости. За този ред, $g = [3, 1, 0, 2, 4]$ и $s = [2, 1, 3, 0, 4]$.

Приложението работи по следния начин:

- Дорка въвежда числата I, J, K на точно трима различни гости.
- Приложението ѝ казва минималния брой гости, които ще бъдат видими, ако и трите избрани гости са на снимката. Формално, приложението ще покаже стойността $\max(s_I, s_J, s_K) - \min(s_I, s_J, s_K) + 1$.

Например, вижте ситуацията, показана в Фиг. 1:

- Гости $I = 0$, $J = 2$ и $K = 4$ са на места $s_I = 2$, $s_J = 3$ и $s_K = 4$. Ако Дорка ги избере, приложението ще покаже стойността $\max(2, 3, 4) - \min(2, 3, 4) + 1 = 3$.

  С други думи, най-тясната снимка, която съдържа гости 0, 2 и 4, съдържа само тези три гости.

- Гости $I = 0$, $J = 4$ и $K = 3$ са на места $s_I = 2$, $s_J = 4$ и $s_K = 0$. Ако Дорка ги избере, приложението ще покаже стойността $\max(2, 4, 0) - \min(2, 4, 0) + 1 = 5$.

С други думи, снимка, която съдържа тримата дадени гости, трябва да съдържа всички 5 гости.

Помогнете на Вероника да определи правилния ред на сядане, използвайки приложението на Дорка. По-конкретно, вашата програма трябва да определи и изведе последователността $g_0, g_1, \ldots, g_{N-1}$. Винаги има точно два правилни отговора (единият е огледален на другия) и можете да изведете който и да е от тях. Вашият резултат ще зависи от броя на заявките към приложението, които вашата програма прави.

Имплементация

⇨ Това е интерактивна задача. Вашата програма ще използва стандартния вход и изход, за да комуникира с проверяващата система (grader) във формата, описан по-долу.

Вашата програма трябва да започне с четене на един ред от входа, съдържащ цяло положително число T, броя на тестовите примери, които следват.

За всеки тестов пример вашата програма трябва да започне с четене на един ред от входа, съдържащ цяло положително число N, броя на местата, който е и броят на гостите.

За да направите заявка, вашата програма трябва да изведе ред във формата „? I J K“, където $0 \leq I, J, K \leq N-1$ са три **различни** числа.

След като направите заявка, вашата програма трябва да прочете един ред, съдържащ едно положително число - отговора на вашата заявка.

За да отговорите с правилен ред на сядане, вашата програма трябва да изведе ред във формата „! g_0 ⋯ g_{N-1}“.

След решаването на всички T тестови примера вашата програма трябва да завърши нормално.

Обърнете внимание, че официалната проверяваща система, използвана в CMS за тестване на вашето решение, може да бъде **адаптивна**. Това означава, че за някои тестови примери пермутацията на гостите не е определена предварително. Вместо това, проверяващата система може да променя поведението си в зависимост от специфичните заявки, направени от вашата програма.

Изчистване на буфера. Ако не използвате предоставените шаблони, уверете се, че изчиствате стандартния изход след отпечатване на всеки ред, в противен случай програмата ви може да бъде оценена като *Not correct*. В Python това се случва автоматично, ако използвате `input()` за четене на редове или можете да ползвате `print(..., flush=True)`, за да изчистите буфера принудително. В C++, `cout << endl;` изчиства буфера в допълнение към отпечатването на нов ред; ако използвате printf, използвайте `fflush(stdout)`.

Ограничения

- $1 \leq T \leq 10$.
- N ще бъде 5 (само за пример), 8, 40 или 2000.
- За всеки тестов пример можете да направите най-много 10 000 заявки.

Оценяване

Вашата програма ще бъде тествана с няколко тестови примера, групирани в подзадачи. За да получите точки за подзадача, трябва правилно да решите всички тестове, които тя съдържа.

- **Подзадача 0 [0 точки]**: Пример ($N = 5$).
- **Подзадача 1 [9 точки]**: $N = 8$.
- **Подзадача 2 [11 точки]**: $N = 2000$ и гостите 0 и 1 седят един до друг.
- **Подзадача 3 [15 точки]**: $N = 40$.

- **Подзадача 4 [65 точки]**: $N = 2000$.

За подзадачи 1 и 2 всяко решение, което правилно решава всички тестови примери, ще получи всички точки.

За подзадачи 3 и 4 вашето решение трябва правилно да реши всички тестови примери, за да получи точки, и вашият резултат ще зависи от Q_s, най-големият брой заявки, които вашето решение е трябвало да направи, за да реши даден тестов пример. Нека $X_s = \max(1, Q_s/N)$. Резултатите за подзадачи 3 и 4 се изчисляват по следния начин:

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

Стойността на score_s се закръгля до най-близкото цяло число за всяка подзадача, а общият ви резултат е сборът от тях. За да получите пълен брой точки, трябва да решите подзадача 3 с най-много 55 заявки и подзадача 4 с най-много 2597 заявки. Примерни стойности на Q_s и резултати за подзадачи 3 и 4 са показани по-долу.

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

Примерни входове/изходи

Грейдър	Решение
1 5	
	? 0 2 4
3	
	? 3 0 1
3	
	? 0 4 3
5	
	! 3 1 0 2 4

Обяснение

Примерният вход съдържа един тестов пример ($T = 1$) с $N = 5$ гости. Скритата конфигурация на гостите в този тестов пример съответства на Фиг. 1.

Първата заявка, направена от примерното решение, е 0, 2, 4. Отговорът 3 на тази заявка ни казва, че тези гости седят, в някакъв неизвестен ред, на три последователни места едно до друго.

Отговорът 3 на втората заявка ни казва същото за гостите 3, 0 и 1.

Сега можем да заключим, че гост 0 трябва да седи по средата, с гости 2 и 4 от едната страна и гости 1 и 3 от другата страна.

След третата заявка вече можем да сме сигурни, че гостите трябва да седят или в реда $[3, 1, 0, 2, 4]$, или в обратния ред $[4, 2, 0, 1, 3]$. Можем да изведем който и да е от двата реда.

Примерни шаблони и детайли за оценяването в CMS

Силно препоръчваме да ползвате предложените шаблонни програми за C++ и Python. Те проверяват дали комуникацията с оценяващата програма е успешна и терминират в противен случай.

Оценяващата програма, комуникираща с решението ви, съобщава първата грешка, която срещне и терминира след нея. Ако не ползвате предложените шаблонни програми, това може да предизвика вашето решението да се срине или да продължава да чака за отговор. Ако това се случи, бихте получили непредсказуемо съобщение в CMS, примерно *Execution timed out (wall clock limit exceeded)*.

Допълнително, препоръчваме да ползвате инструмента за тестване локално преди да изпратите решение. Той проверява изхода на решението и съобщава нарушения на протокола.

Инструмент за тестване

За да улесним тестването на вашето решение, предоставяме прост инструмент, който можете да изтеглите от CMS. Ползването на този инструмент не е задължително. Имайте предвид, че официалната проверяваща система в CMS е различна от инструмента за тестване.

За да използвате инструмента, ви е необходим входен файл. Можете да използвате предоставения примерен вход `seatingplan.input0.txt` или да направите свой собствен. Входният файл трябва да започва с ред, съдържащ броя T на тестовите примери, и след това трябва да има по два реда за всеки тестов пример: един ред с броя N и един ред с числата $g_0, g_1, ..., g_{N-1}$.

За Python програми, да кажем `seatingplan.py` (обикновено се изпълнява като `pypy3 seatingplan.py`) стартирайте инструмента за тестване по следния начин:

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

За C++ програми, първо компилирайте вашето решение:

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

и след това стартирайте инструмента за тестване:

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