

Zadanie: TOW

Wieża



BOI 2025, Dzień 1. Dostępna pamięć: 256 MB.

26.04.2025

Istnieje wiele legend dotyczących Krzywej Wieży w Toruniu. Ściana wieży to okrąg z $N \geq 3$ równomiernie rozmieszczonymi drzwiami (innymi słowy, drzwi są wierzchołkami foremnego N -kąta). Drzwi są ponumerowane od 0 do $N-1$, ale w **losowej kolejności**. Proszę zapoznać się z sekcją ocenianie, aby uzyskać więcej szczegółów na ten temat.

Jedna z mniej znanych legend opisuje, jak każdy nowy mieszkaniec wieży musiał ukończyć pewne wyzwanie. Celem wyzwania jest wymienienie numerów drzwi, zaczynając od pewnych drzwi, a następnie poruszając się dookoła okręgu (zgodnie z ruchem wskazówek zegara lub przeciwnie), odwiedzając każde drzwi dokładnie raz.

Należy to zrobić bez faktycznego oglądania wieży. Zamiast tego nowy mieszkaniec może zadawać pytania następującego typu: "Dla danych parami różnych drzwi x, y, z , która para drzwi jest najbliżej siebie: $\{x, y\}$, $\{y, z\}$, czy $\{z, x\}$?". Odpowiedzią na takie pytanie są wszystkie pary (spośród $\{x, y\}$, $\{y, z\}$ i $\{z, x\}$) drzwi z najmniejszą odległością euklidesową. Odległość to po prostu długość najkrótszego odcinka łączącego drzwi. Twoim zadaniem jest napisanie programu, który zada małą liczbę takich pytań, aby określić kolejność drzwi.

Interakcja

Jest to zadanie interaktywne. Powinieneś napisać program, który znajduje poprawne rozwiązanie zadania i komunikuje się z interaktorem poprzez wczytywanie danych ze standardowego wejścia i wypisywanie na standardowe wyjście.

Na początku interakcji Twój program powinien wczytać dwie liczby całkowite t i k ($1 \leq t \leq 100$, $1 \leq k \leq 12000$) ze standardowego wejścia, oznaczające liczbę przypadków testowych oraz maksymalną dozwoloną średnią liczbę zapytań, odpowiednio. Zobacz sekcję punktacji, aby uzyskać więcej informacji o drugiej wartości.

Dla każdego przypadku testowego Twój program powinien najpierw wczytać pojedynczą liczbę całkowitą n ($3 \leq n \leq 500$) ze standardowego wejścia, oznaczającą liczbę drzwi w wieży.

Następnie Twój program powinien zadawać pytania w następujący sposób:

- Twój program powinien wypisać pojedynczą linię postaci

? x y z

na standardowe wyjście, gdzie x, y i z są parami różnymi liczbami całkowitymi ($0 \leq x, y, z \leq n-1$). Ta linia reprezentuje pojedyncze pytanie dotyczące drzwi x, y i z .

- Odpowiedź zostanie podana jako:

r a_1 b_1 ... a_r b_r

gdzie r to liczba całkowita ($1 \leq r \leq 3$) reprezentująca liczbę par drzwi o najmniejszej odległości. Każda taka para jest opisana przez dwie liczby całkowite a_i i b_i ($a_i, b_i \in \{x, y, z\}$ oraz $a_i < b_i$).

Gdy już ustalisz kolejność drzwi, powinieneś wypisać pojedynczą linię postaci

! x_0 x_1 ... x_{n-1}

na standardowe wyjście, gdzie x_0, x_1, \dots, x_{n-1} to kolejność drzwi zgodnie z opisem w treści zadania. Należy zauważyć, że istnieje dokładnie $2n$ możliwych poprawnych odpowiedzi, ponieważ możesz wypisać kolejność zaczynając od dowolnych drzwi i idąc w dowolnym kierunku. Każda z tych odpowiedzi zostanie zaakceptowana.

Pamiętaj, że po każdym zapytaniu lub odpowiedzi musisz opróżnić bufor wyjściowy używając `cout.flush()` (lub `fflush(stdout)` jeśli używasz `printf`) w C++ lub `sys.stdout.flush()` w języku Python. W przeciwnym razie Twój program może otrzymać werdykt `Time Limit Exceeded`.

Po wypisaniu odpowiedzi, Twój program powinien natychmiast przejść do następnego przypadku testowego lub zakończyć interakcję, jeśli wszystkie przypadki testowe zostały przetworzone.

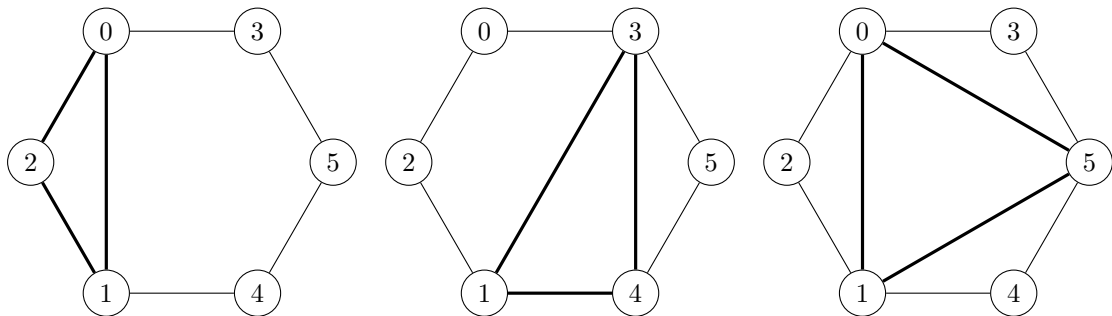
Twój program nie może otwierać żadnych plików ani korzystać z innych zasobów. Może używać standardowego strumienia błędów do celów debugowania, ale pamiętaj, że zapisywanie do tego strumienia zajmuje czas.

Interaktor w tym zadaniu nie jest adaptacyjny, co oznacza, że początkowa kolejność drzwi jest ustalona z góry w każdym przypadku testowym i nie zmienia się podczas interakcji.

Przykładowa interakcja

Przypuśćmy, że mamy tylko jeden przypadek testowy z $n = 6$, a kolejność drzwi to 5, 3, 0, 2, 1, 4. Interakcja może wyglądać następująco:

Interaktor	Twój program	Komentarz
1 100		$t = 1$ i $k = 100$.
6		Interaktor podaje liczbę drzwi w pierwszym przypadku testowym.
	? 0 1 2	Twój program pyta, która para drzwi jest najbliższej.
2 0 2 1 2		Pary drzwi $\{0, 2\}$ i $\{1, 2\}$ są najbliższej.
	? 4 1 3	Twój program pyta, która para drzwi jest najbliższej.
1 1 4		Para $\{1, 4\}$ jest najbliższej.
	? 0 5 1	Twój program pyta, która para drzwi jest najbliższej.
3 0 5 0 1 1 5		Pary $\{0, 5\}$, $\{0, 1\}$, i $\{1, 5\}$ są najbliższej.
	! 4 5 3 0 2 1	Twój program poprawnie wypisuje kolejność numerów drzwi.



Wyjaśnienie przykładu: Powyższe obrazki pokazują drzwi z ich numerami wzdłuż ścian wieży. Na pierwszym obrazku od lewej pokazany jest trójkąt utworzony przez drzwi o numerach 0, 1, 2, odpowiadający pierwszemu zapytaniu twojego programu. Możemy zauważyć, że pary $\{0, 2\}$ i $\{1, 2\}$ są najbliższej siebie. Na środkowym obrazku pokazany jest trójkąt utworzony przez drzwi o numerach 1, 4, 3, odpowiadający drugiemu zapytaniu twojego programu. Wyraźnie widać, że para $\{1, 4\}$ jest najbliższej. Na trzecim obrazku od lewej pokazany jest trójkąt utworzony przez drzwi o numerach 0, 1, 5, odpowiadający trzeciemu zapytaniu twojego programu. Wyraźnie widać, że wszystkie pary drzwi są jednakowo blisko siebie.

Należy zauważyć, że sekwencje 0, 2, 1, 4, 5, 3 lub 5, 4, 1, 2, 0, 3 (i kilka innych) również byłyby poprawnymi odpowiedziami w tym przypadku.

Ocenianie

Ocenianie tego zadania jest podzielone na podzadania. Dla każdego podzadania istnieje dokładnie jeden test, zawierający dokładnie $t = 100$ przypadków testowych. Dla każdego testu, liczona jest średnia liczba pytań zadanych przez Twój program poprzez wzięcie łącznej liczby pytań i podzielenie jej przez liczbę przypadków testowych. Jeśli ta średnia jest większa niż k dla danego podzadania, dostaniesz wynik 0 za to podzadanie. W przeciwnym wypadku dla podzadań od 1 do 4, dostaniesz pełną liczbę punktów.

W ostatnim podzadaniu, Twój wynik będzie obliczany w sposób podany poniżej. Niech k^* oznacza średnią liczbę zapytań zadanych przez Twój program. Liczba punktów, którą otrzyma on za to podzadanie jest obliczana w następujący sposób:

$$\left\lceil 56 \cdot \min \left(1, \frac{12000 - k^*}{7800} \right) \right\rceil,$$

oznacza to, że Twój wynik rośnie liniowo od 0 do 56 gdy k^* maleje od 12000 do 4200.

Jeśli Twój program poda niepoprawną odpowiedź dla któregośkolwiek przypadku testowego, otrzymasz 0 punktów za to podzadanie niezależnie od liczby zadanych zapytań.

Dodatkowe ograniczenia dla każdego podzadania opisuje tabela poniżej.

Podzadanie	Ograniczenia	Punkty
1	$k = 8000, 3 \leq n \leq 9$	6
2	$k = 4500, 40 \leq n \leq 50$	7
3	$k = 3000, 90 \leq n \leq 100$	9
4	$k = 4500, n = 400$, istnieje poprawny wynik x_0, \dots, x_{n-1} gdzie $x_i = i$ dla $200 \leq i \leq 399$	22
5	$k = 12\,000, n = 500$	aż do 56

Ponadto, możesz założyć, że każdy przypadek testowy został wygenerowany poprzez najpierw wybranie n **z rozkładem jednostajnym** spośród wszystkich wartości n spełniających ograniczenia danego podzadania, a następnie wybranie kolejności drzewi **z rozkładem jednostajnym** spośród wszystkich kolejności n drzewi spełniających ograniczenia danego podzadania.