

# Aufgabe: TOW

## Turm



BOI 2025, Tag 1. Speicherlimit: 256 MB.

2025.04.26

Um den schiefen Turm von Toruń ranken sich viele Legenden. Die Mauer des Turms ist ein Kreis mit  $N \geq 3$  gleichmäßig verteilten Türen (d.h. die Türen befinden sich in den Ecken eines regelmäßigen  $N$ -Ecks). Die Türen sind von 0 bis  $N - 1$  durchnummeriert, aber in einer **zufälligen Reihenfolge**. Schau dir den Abschnitt über die Bewertung an, um mehr Informationen darüber zu erhalten.

Laut einer der weniger bekannten Legenden musste jeder neue Bewohner des Turms eine bestimmte Herausforderung meistern. Das Ziel dieser Aufgabe war es, die Türen aufzulisten: Dabei sollte die Liste bei einer beliebigen Tür anfangen und dann in einer beliebigen Richtung (also im oder gegen den Uhrzeigersinn) an der kreisförmigen Wand des Turms entlanggehen. Insbesondere sollte jede Tür genau einmal aufgelistet werden.

Dies musste gemeistert werden, ohne den Turm tatsächlich zu sehen. Stattdessen konnte der neue Bewohner Fragen der folgenden Form stellen: "Gegeben drei verschiedene Türen  $x, y, z$ , welches Paar von Türen hat den geringsten Abstand zueinander:  $\{x, y\}$ ,  $\{y, z\}$ , oder  $\{z, x\}$ ?" Die Antwort auf eine solche Frage sind alle Paare (also Elemente der Liste  $\{x, y\}$ ,  $\{y, z\}$ ,  $\{z, x\}$ ) von Türen mit der geringsten euklidischen Distanz. Die Distanz ist ganz einfach die Länge der kürzesten Linie, die die beiden Türen miteinander verbindet. Schreibe ein Programm, das eine kleine Anzahl von solchen Fragen stellt und dann die Reihenfolge der Türen ermittelt.

## Interaktion

Dies ist eine interaktive Aufgabe. Du sollst ein Programm schreiben, das eine korrekte Lösung für die Aufgabe findet und mit dem Grader durch Lesen von der Standardeingabe und Schreiben auf die Standardausgabe kommuniziert.

Am Anfang der Interaktion soll dein Programm zwei Ganzzahlen  $t$  und  $k$  ( $1 \leq t \leq 100$ ,  $1 \leq k \leq 12\,000$ ) von der Standardeingabe lesen – die Anzahl der Testfälle und die maximal erlaubte durchschnittliche Anzahl an Fragen. Schau dir den Abschnitt über die Bewertung an, um über Letzteres mehr Informationen zu erhalten.

Für jeden Testfall soll dein Programm eine einzelne Ganzzahl  $n$  ( $3 \leq n \leq 500$ ) von der Standardeingabe einlesen – die Anzahl der Türen im Turm.

Dann soll dein Programm auf folgender Art und Weise Fragen stellen:

- Dein Programm soll eine einzelne Zeile der Form

?  $x$   $y$   $z$

auf die Standardausgabe schreiben, wobei  $x, y$  und  $z$  verschiedene Ganzzahlen sind ( $0 \leq x, y, z \leq n - 1$ ). Diese Zeile steht für eine einzelne Frage über die drei Türen  $x, y$  und  $z$ .

- Die Antwort wird das folgende Format haben:

$r$   
 $a_1$   $b_1$   
...  
 $a_r$   $b_r$

wobei die Ganzzahl  $r$  ( $1 \leq r \leq 3$ ) die Anzahl der Türpaare mit der geringsten Distanz ist. Jedes dieser Paare wird durch zwei Ganzzahlen  $a_i$  und  $b_i$  ( $a_i, b_i \in \{x, y, z\}$  mit  $a_i < b_i$ ) beschrieben.

Sobald du die Reihenfolge der Türen herausgefunden hast, sollst du eine einzelne Zeile in folgender Form auf die Standardausgabe ausgeben:

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

Dabei ist  $x_0, x_1, \dots, x_{n-1}$  die Reihenfolge der Türen, so wie es im Aufgabentext beschrieben wird. Bitte beachte, dass es genau  $2n$  verschiedene korrekte Antworten gibt, weil die Antwort an jeder Tür starten und dann in jede von zwei Richtungen gehen kann. Jede dieser Antworten wird vom Grader akzeptiert.

**Denk daran, den Output Buffer nach jeder Anfrage oder Antwort mithilfe von `cout.flush()` (oder `fflush(stdout)`, wenn du `printf` verwendest) in C++ bzw. `sys.stdout.flush()` in Python zu flushen.** Andernfalls wird dein Programm möglicherweise mit dem Verdict **Time Limit Exceeded** bewertet.

Nachdem du die Antwort an den Grader geschrieben hast, soll dein Programm unmittelbar mit dem nächsten Testfall fortfahren (oder die Interaktion beenden, falls schon alle Testfälle bearbeitet wurden).

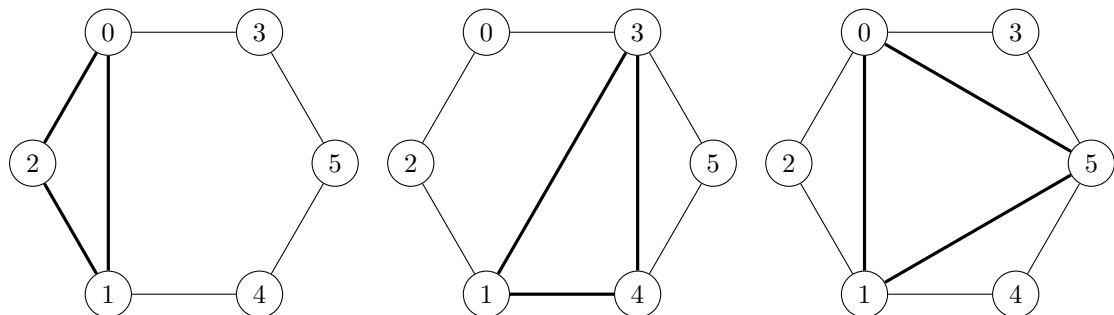
Dein Programm darf keine Dateien oder andere Ressourcen öffnen. Für die Fehlersuche kannst Du den Standard error-Stream verwenden. Bedenke aber, dass das Schreiben auf diesen Stream einige Zeit benötigt.

Beachte bitte auch, dass der Grader nicht adaptiv ist. Die anfängliche Reihenfolge der Türen steht für jeden Testfall also im Vorhinein fest und ändert sich während der Interaktion auch nicht.

## Beispielinteraktion

Angenommen, es gibt nur einen Testfall mit  $n = 6$  und die Reihenfolge der Türen ist 5, 3, 0, 2, 1, 4. Die Interaktion könnte dann folgendermaßen aussehen:

Grader	Dein Programm	Bemerkung
1 100		$t = 1$ und $k = 100$ .
6		Der Grader übermittelt die Anzahl der Türen im ersten Testfall.
	? 0 1 2	Dein Programm fragt, welches Paar von Türen die geringste Distanz hat.
2 0 2 1 2		Die Paare $\{0, 2\}$ und $\{1, 2\}$ haben die geringste Distanz.
	? 4 1 3	Dein Programm fragt, welches Paar von Türen die geringste Distanz hat.
1 1 4		Das Paar $\{1, 4\}$ hat die geringste Distanz.
	? 0 5 1	Dein Programm fragt, welches Paar von Türen die geringste Distanz hat.
3 0 5 0 1 1 5		Die Paare $\{0, 5\}$ , $\{0, 1\}$ , und $\{1, 5\}$ haben die geringste Distanz.
	! 4 5 3 0 2 1	Dein Programm gibt die (korrekte) Reihenfolge der Türen aus.



**Erklärung des Beispiels:** Das Bild zeigt die Anzahl der Türen mitsamt ihrer Nummern entlang der Wand des Turms. Im ersten Bild von links wird das Dreieck der Türen mit den Nummern 0, 1, 2 gezeigt. Das entspricht der ersten Anfrage deines Programms. Die Paare  $\{0, 2\}$  und  $\{1, 2\}$  haben die geringste Distanz. Im mittleren Bild ist das Dreieck der Türen 1, 4, 3 abgebildet. Es lässt sich klar erkennen, dass das Paar  $\{1, 4\}$  die geringste Distanz hat. Im dritten Bild von links wird das Dreieck der Türen mit den Nummern 0, 1, 5 gezeigt. Das entspricht der dritten Anfrage deines Programms. Offensichtlich haben alle Paare von Türen die gleiche Distanz voneinander.

Bitte beachte, dass die Folgen 0, 2, 1, 4, 5, 3 oder 5, 4, 1, 2, 0, 3 (und einige andere) ebenfalls korrekte Antworten für diesen Testfall sind.

## Bewertung

Die Bewertung für dieses Problem ist in Teilaufgaben aufgeteilt. Für jede Teilaufgabe gibt es genau einen Test und dieser enthält genau  $t = 100$  Testfälle. Für jeden Test wird die durchschnittliche Anzahl von Anfragen durch dein Programm berechnet. Dafür wird die Gesamtzahl deiner Anfragen über alle Testfälle durch die Anzahl der Testfälle geteilt. Ist dieser Durchschnitt für eine Teilaufgabe größer als  $k$ , so erhältst du für diese Teilaufgabe 0 Punkte. Andernfalls erhältst du die volle Punktzahl für diese Teilaufgabe, wenn es sich dabei um eine der ersten vier Teilaufgaben handelt.

In der letzten Teilaufgabe wird deine Punktzahl folgendermaßen berechnet: Es sei  $k^*$  die durchschnittliche Anzahl von Anfragen durch dein Programm. Dann wird die Punktzahl mithilfe der folgenden Formel berechnet:

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

Die Punktzahl steigt also linear von 0 bis 56, wenn  $k^*$  von 12000 auf 4200 sinkt.

Bitte beachte, dass du 0 Punkte für eine Teilaufgabe erhältst, wenn dein Programm in irgendeinem Testfall eine falsche Antwort gibt. Die Anzahl der Anfragen ist dann irrelevant.

Die zusätzlichen Beschränkungen für jede der Teilaufgaben sind in der untenstehenden Tabelle angegeben.

Teilaufgabe	Beschränkungen	Punkte
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	$k = 4500, n = 400$ , es gibt eine korrekte Antwort $x_0, \dots, x_{n-1}$ mit $x_i = i$ für $200 \leq i \leq 399$	22
5	$k = 12000, n = 500$	bis zu 56

Du kannst davon ausgehen, dass jeder Testfall durch die **uniform zufällige** Wahl von  $n$  aus allen möglichen Werten von  $n$  für die jeweilige Teilaufgabe generiert wurde. Anschließend wurde die Reihenfolge **uniform zufällig** aus allen möglichen Reihenfolgen der  $n$  Türen ausgewählt, insofern diese die Beschränkungen der jeweiligen Teilaufgabe erfüllen.