Längste Besichtigung

Die Veranstalter der IOI 2023 haben keinen Plan! Insbesondere nicht für die Besichtigung des Freilichtmuseums Ópusztaszer. Aber vielleicht kannst du helfen ...

In Ópusztaszer kann man N Stationen besichtigen, nummeriert von 0 bis $N-1$.
Außerdem verbindet ein Netz aus (direkten) **Wegen**, die man in beide Richtungen nutzen kann, einige Stationen. Für jedes Paar von Stationen kann es höchstens einen Weg zwischen diesen Stationen geben. Dummerweise wissen die Veranstalter *nicht*, zwischen welchen Stationen es Wege gibt.

Für den Besichtigungsplan spielt die **Dichte** des Wegenetzes in Ópusztaszer eine Rolle. Die Dichte ist **mindestens** δ, wenn es zwischen je 3 unterschiedlichen Stationen mindestens δ Wege gibt. In anderen Worten: Für je drei Stationen (u, v, w) mit $0 \leq u < v < w < N$ gibt es für mindestens δ der Stationspaare $(u, v), (v, w)$ und (u, w) einen Weg zwischen den entsprechenden Stationen.

Die Veranstalter *wissen* immerhin, dass die Dichte des Wegenetzes mindestens D ist. D ist eine positive ganze Zahl und kann höchstens 3 sein.

Die Veranstalter versuchen nun, mit **Anfragen** an die Verwaltung von Ópusztaszer weitere Informationen über das Wegenetz zu erhalten. In jeder Anfrage müssen sie zwei (nicht leere) Arrays von Stationen benennen: $[A[0], \ldots, A[P-1]]$ und $[B[0], \ldots, B[R-1]]$. Die Stationen in den Arrays sind paarweise verschieden, also:

- $A[i] \neq A[j]$ für alle i, j mit $0 \leq i < j < P$;
- $B[i] \neq B[j]$ für alle i, j mit $0 \leq i < j < R$;
- $A[i] \neq B[j]$ für alle i, j mit $0 \leq i < P$ und $0 \leq j < R$.

Die Verwaltung ist zwar nicht planlos, aber lustlos. Auf eine Anfrage antwortet sie nur, ob es zwischen einer Station aus A und einer Station aus B einen Weg gibt.
Dazu betrachtet die Verwaltung alle Paare i und j mit $0 \leq i < P$ und $0 \leq j < R$. Wenn es für eines dieser Paare einen Weg zwischen den Stationen $A[i]$ und $B[j]$ gibt, antwortet sie `true`, ansonsten `false`.

Eine **Besichtigung** der Länge l ist eine Folge von l *verschiedenen* Stationen $t[0], t[1], \ldots, t[l-1]$, so dass es zwischen je zwei aufeinanderfolgenden Stationen $t[i]$ und $t[i+1]$ $(0 \leq i \leq l-2)$ einen Weg gibt. Eine Besichtigung der Länge l ist eine **längste Besichtigung**, wenn es keine Besichtigung gibt, die mindestens Länge $l+1$ hat.

Die Veranstalter sind planlos *und* lustlos und überlassen dir die ganze Arbeit: Du sollst die Anfragen an die Verwaltung von Ópusztaszer übernehmen und so eine längste Besichtigung des Museums bestimmen.

Implementierungsdetails

Implementiere die folgende Funktion:

```
int[] longest_trip(int N, int D)
```

- N: die Anzahl an Stationen in Ópusztaszer.
- D: die garantierte minimale Dichte des Wegenetzes.
- Diese Funktion soll ein Array $t = [t[0], t[1], \ldots, t[l-1]]$ zurückgeben, das eine längste Besichtigung beschreibt.
- Diese Funktion kann in jedem Testfall **mehrfach** aufgerufen werden.

Die obige Funktion kann die folgende Funktion aufrufen:

```
bool are_connected(int[] A, int[] B)
```

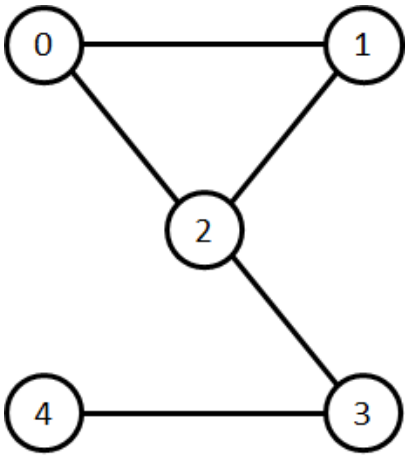
- A: ein nicht leeres Array von verschiedenen Stationen.
- B: ein nicht leeres Array von verschiedenen Stationen.
- A und B sollten keine gemeinsame Station haben.
- Diese Funktion gibt `true` zurück, falls es eine Station in A und eine Station in B gibt, zwischen denen es einen Weg gibt. Ansonsten gibt die Funktion `false` zurück.
- In jedem Aufruf von `longest_trip` kann diese Funktion höchstens $32\,640$ Mal aufgerufen werden, und insgesamt kann die Funktion höchstens $150\,000$ Mal aufgerufen werden.
- Die Gesamtlänge aller Arrays A und B, die dieser Funktion über alle Aufrufe hinweg übergeben werden, darf $1\,500\,000$ nicht überschreiten.

Der Grader ist **nicht adaptiv**. Jede Einsendung wird auf den gleichen Testfällen bewertet. Das heißt, dass sowohl die Werte von N und D als auch die Paare an Stationen, zwischen denen es Wege gibt, für jeden Aufruf von `longest_trip` innerhalb eines Testfalls von Beginn an festgelegt sind.

Beispiele

Beispiel 1

Betrachte ein Szenario mit $N = 5$, $D = 1$, bei dem es zwischen den Stationen diese Wege gibt:



Die Funktion `longest_trip` wird also so aufgerufen:

```
longest_trip(5, 1)
```

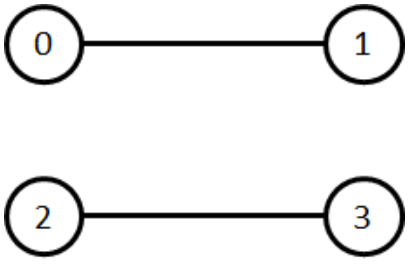
Diese Funktion kann beispielsweise `are_connected` nacheinander so aufrufen:

Aufruf	Paare mit Wegen	Rückgabewert
`are_connected([0], [1, 2, 4, 3])`	$(0, 1)$ und $(0, 2)$	`true`
`are_connected([2], [0])`	$(2, 0)$	`true`
`are_connected([2], [3])`	$(2, 3)$	`true`
`are_connected([1, 0], [4, 3])`	keine	`false`

Nach dem vierten Aufruf ist klar, dass es für *keines* der Stationspaare $(1, 4)$, $(0, 4)$, $(1, 3)$ und $(0, 3)$ einen Weg gibt. Da die Dichte des Netzwerkes mindestens $D = 1$ ist, können wir aus der Betrachtung der Stationen $(0, 3, 4)$ schließen, dass es für das Paar $(3, 4)$ einen Weg gibt. Aus ähnlichen Gründen muss es zwischen den Stationen 0 und 1 einen Weg geben.

Ab diesem Punkt lässt sich schließen, dass $t = [1, 0, 2, 3, 4]$ eine Besichtigung der Länge 5 ist und dass keine Besichtigung mit einer Länge größer als 5 existiert. Deshalb kann die Funktion `longest_trip` das Array $[1, 0, 2, 3, 4]$ zurückgeben.

Betrachte nun ein anderes Szenario mit $N = 4$, $D = 1$, bei dem es zwischen den Stationen diese Wege gibt:



Die Funktion `longest_trip` wird also so aufgerufen:

```
longest_trip(4, 1)
```

In diesem Szenario ist die Länge einer längsten Besichtigung 2. Nach ein paar Aufrufen von `are_connected` kann deshalb die Funktion `longest_trip` eines der Arrays $[0, 1]$, $[1, 0]$, $[2, 3]$ oder $[3, 2]$ zurückgeben.

Beispiel 2

Teilaufgabe 0 enthält einen zusätzlichen Beispieltestfall mit $N = 256$ Stationen. Dieser Testfall ist im Anhang enthalten, den du vom Wettbewerbssystem runterladen kannst.

Beschränkungen

- $3 \leq N \leq 256$
- Die Summe der N über alle Aufrufe von `longest_trip` ist maximal $1\,024$ in jedem Testfall.
- $1 \leq D \leq 3$

Teilaufgaben

1. (5 Punkte) $D = 3$
2. (10 Punkte) $D = 2$
3. (25 Punkte) $D = 1$. Sei $l^\star$ die Länge einer längsten Besichtigung. Die Funktion `longest_trip` muss keine Besichtigung der Länge $l^\star$ zurückgeben. Stattdessen soll die Funktion eine Besichtigung zurückgeben, deren Länge mindestens $\left\lceil \frac{l^\star}{2} \right\rceil$ ist.
4. (60 Punkte) $D = 1$

In Teilaufgabe 4 wird deine Punktzahl basierend auf der Anzahl an Aufrufen der Funktion `are_connected` innerhalb eines einzelnen Aufrufs von `longest_trip` bestimmt. Sei q die maximale Anzahl an Aufrufen von `are_connected` über alle Aufrufe von `longest_trip` für alle Testfälle dieser Teilaufgabe. Deine Punktzahl für diese Teilaufgabe wird nach der folgenden Tabelle berechnet:

Bedingung	Punktzahl
$2\,750 < q \leq 32\,640$	20
$550 < q \leq 2\,750$	30
$400 < q \leq 550$	45
$q \leq 400$	60

Falls in irgendeinem der Testfälle die Aufrufe der Funktion `are_connected` nicht den Vorgaben aus den Implementierungsdetails entsprechen, wird die Punktzahl deiner Lösung für diese Teilaufgabe 0 sein.

Beispielgrader

Sei C die Anzahl an Szenarios, also die Anzahl an Aufrufen von `longest_trip`. Der Beispielgrader liest die Eingabe im folgenden Format:

- Zeile 1: C

Dann folgen die Beschreibungen von C Szenarien.

Der Beispielgrader liest die Beschreibung jedes Szenarios im folgenden Format:

- Zeile 1: $N\ D$
- Zeile $1+i$ ($1 \leq i < N$): $U_i[0]\ U_i[1]\ \ldots\ U_i[i-1]$

Hier ist jedes U_i ($1 \leq i < N$) ein Array der Größe i, das beschreibt, für welche Stationspaare es einen Weg gibt. Für all i und j mit $1 \leq i < N$ und $0 \leq j < i$ gilt:

- falls es zwischen den Stationen j und i einen Weg gibt, dann soll $U_i[j]$ den Wert 1 haben;
- falls es zwischen den Stationen j und i keinen Weg gibt, dann soll $U_i[j]$ den Wert 0 haben.

In jedem Szenario, bevor `longest_trip` aufgerufen wird, überprüft der Beispielgrader, ob die Dichte im Straßennetzwerk mindestens D ist. Falls diese Bedingung nicht erfüllt ist, gibt er die Nachricht `Insufficient Density` aus und terminiert.

Falls der Beispielgrader einen fehlerhaften Aufruf von `are_connected` erkennt, gibt er `Protocol Violation: <MSG>` aus, wobei `<MSG>` eine der folgenden Fehlermeldungen ist:

- `invalid array`: in einem Aufruf von `are_connected` gilt für mindestens eines der Arrays A und B:
  - das Array ist leer, oder
  - es enthält ein Element, das keine ganze Zahl zwischen 0 und $N-1$ (eingeschlossen) ist, oder
  - es enthält ein Element mindestens zweimal.
- `non-disjoint arrays`: in einem Aufruf von `are_connected` haben die Arrays A und B mindestens eine gemeinsame Station.
- `too many calls`: die Anzahl an Aufrufen von `are_connected` überschreitet $32\,640$ innerhalb eines Aufrufs von `longest trip` oder überschreitet $150\,000$ insgesamt.
- `too many elements`: die Gesamtanzahl an Stationen, die an `are_connected` über alle Aufrufe übergeben wurden, überschreitet $1\,500\,000$.

Seien ansonsten $t[0], t[1], \ldots, t[l-1]$ die Elemente des Array, das durch `longest_trip` in einem Szenario zurückgegeben wird, für irgendein nichtnegatives l. Der Beispielgrader gibt drei Zeilen für dieses Szenario im folgenden Format aus:

- Zeile 1: l
- Zeile 2: $t[0]$ $t[1]$ $\ldots$ $t[l-1]$
- Zeile 3: die Anzahl an Aufrufen von `are_connected` in diesem Szenario

Am Ende gibt der Beispielgrader folgendes aus:

- Zeile $1 + 3 \cdot C$: die maximale Anzahl an Aufrufen von `are_connected` über alle Aufrufe von `longest_trip`