Longest Trip

¡Los organizadores de la IOI 2023 están en un gran problema! Han olvidado planear el viaje a Ópusztaszer. Pero tal vez no sea demasiado tarde...

Hay N atracciones en Ópusztaszer indexadas de 0 a $N-1$. Algunos pares de estas atracciones están conectadas por **calles bidireccionales**. Cada par de atracciones están conectadas a lo sumo por una calle. Los organizadores *no saben* cuales atracciones están conectadas por calles.

Decimos que la **densidad** de la red de calles en Ópusztaszer es **al menos** δ si cada 3 atracciones distintas tienen al menos δ calles entre ellas. En otras palabras, para cada tripleta de atracciones (u, v, w) tales que $0 \leq u < v < w < N$, entre los pares de atracciones $(u, v), (v, w)$ and (u, w) al menos δ de estos pares están conectados por una calle.

Los organizadores *conocen* un entero positivo D tal que la densidad de la red de calles es al menos D. Observa que el valor de D no puede ser mayor que 3

Los organizadores pueden hacer **llamadas** al informador telefónico de Ópusztaszer para obtener información acerca de los caminos entre ciertas atracciones. En cada llamada, dos arreglos no vacíos de atracciones $[A[0], \ldots, A[P-1]]$ y $[B[0], \ldots, B[R-1]]$ deben ser especificados. Las atracciones deben ser distintas entre sí. Es decir,

- $A[i] \neq A[j]$ para cada i y j tales que $0 \leq i < j < P$;
- $B[i] \neq B[j]$ para cada i y j tales que $0 \leq i < j < R$;
- $A[i] \neq B[j]$ para cada i y j tales que $0 \leq i < P$ y $0 \leq j < R$.

Para cada llamada, el informador reporta si hay un camino conectando una atracción de A y una atracción de B. Más precisamente, el informador itera sobre todos los pares i y j tales que $0 \leq i < P$ and $0 \leq j < R$. Si, para alguno de estos pares se cumple que las atracciones $A[i]$ y $B[j]$ están conectadas por un camino, el informador retorna `true`. En caso contrario, el informador retorna `false`.

Un **recorrido** de longitud l es una secuencia de atracciones *distintas* $t[0], t[1], \ldots, t[l-1]$, en la que para cada i entre 0 y $l-2$, inclusivo, se cumple que las atracciones $t[i]$ y $t[i+1]$ están conectadas por un camino.

Un recorrido de longitud l es llamado **recorrido más largo** si no existe ningún otro recorrido con una longitud de al menos $l+1$.

Tu tarea es ayudar a los organizadores a encontrar el recorrido más largo en Ópusztaszer haciendo llamadas al informador.

Detalles de implementación

Debe implementar el siguiente procedimiento:

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

- N: el número de atracciones en Ópusztaszer.
- D: la mínima densidad garantizada en la red de caminos.
- Este procedimiento debe retornar un arreglo $t = [t[0], t[1], \ldots, t[l-1]]$ que representa el recorrido más largo.
- Este procedimiento puede ser llamado **múltiples veces** en cada caso de prueba.

El procedimiento mencionado arriba puede hacer llamadas al siguiente procedimiento:

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

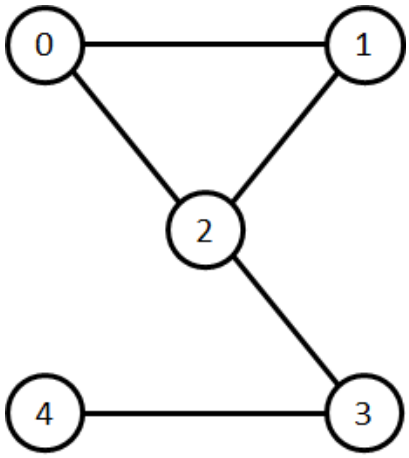
- A: un arreglo no vacío de atracciones distintas.
- B: un arreglo no vacío de atracciones distintas.
- A y B deben ser disjuntos.
- Este procedimiento retorna `true` si hay una atracción en A y una atracción en B conectadas por un camino. En caso contrario, retorna `false`
- Este procedimiento puede ser llamado a lo sumo $32\,640$ veces en cada invocación de `longest_trip`, y a lo sumo $150\,000$ veces en total.

El evaluador de ejemplo **no es adaptativo**. Cada envío es evaluado sobre el mismo conjunto de casos de prueba. Es decir, los valores de N y D, así como los pares de atracciones conectadas por caminos, ya están en cada caso de prueba.

Ejemplos

Ejemplo 1

Considera el escenario en el que $N = 5$, $D = 1$ y las conexiones de caminos son como se muestran en la siguiente figura:



El procedimiento `longest_trip` es llamado de la siguiente forma:

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

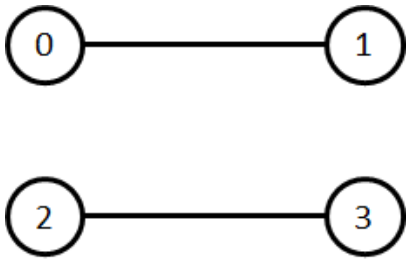
El procedimiento puede hacer llamadas a `are_connected` como sigue:

Llamada	Pares conectados por un camino	Valor de retorno
`are_connected([0], [1, 2, 4, 3])`	$(0, 1)$ y $(0, 2)$	`true`
`are_connected([2], [0])`	$(2, 0)$	`true`
`are_connected([2], [3])`	$(2, 3)$	`true`
`are_connected([1, 0], [4, 3])`	Ninguno	`false`

Después de la cuarta llamada, resulta que *ninguno* de los pares $(1, 4)$, $(0, 4)$, $(1, 3)$ and $(0, 3)$ está conectado por un camino. Como la densidad de la red es al menos $D = 1$, de la terna $(0, 3, 4)$ se puede ver que el par $(3, 4)$ debe estar conectado por un camino. De manera similar, se puede ver que las atracciones 0 y 1 deben estar conectadas.

En este punto, se puede concluir que $t = [1, 0, 2, 3, 4]$ es un recorrido de longitud 5, y que no existe un recorrido de longitud mayor que 5. Por lo tanto, el procedimiento `longest_trip` puede retornar $[1, 0, 2, 3, 4]$.

Considera otro escenario en el cual $N = 4$, $D = 1$, y los caminos entre atracciones son como se muestra en la siguiente figura.



El procedimiento `longest_trip` es llamado de la siguiente manera:

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

En este escenario, la longitud del recorrido más largo es 2. Por lo tanto, después de unas cuantas llamadas al procedimiento `are_connected`, el procedimiento `longest_trip` puede retornar alguno de entre $[0, 1]$, $[1, 0]$, $[2, 3]$ o $[3, 2]$

Ejemplo 2

La subtarea 0 contiene un caso de prueba de ejemplo adicional con $N = 256$ atracciones. Este caso de prueba está incluido en el paquete adjunto que tu puedes descargar desde el sistema de la competencia.

Restricciones

- $3 \leq N \leq 256$
- La suma de N en todas las llamadas a `longest_trip` no excede $1\,024$.
- $1 \leq D \leq 3$

Subtareas

1. (5 puntos) $D = 3$
2. (10 puntos) $D = 2$
3. (25 puntos) $D = 1$. Sea $l^\star$ la longitud del recorrido más largo. El procedimiento `longest_trip` no tiene que retornar un recorrido de longitud $l^\star$. En cambio, debe retornar un recorrido con una longitud de al menos $\left\lceil \frac{l^\star}{2} \right\rceil$.
4. (60 puntos) $D = 1$

Si en alguno de los casos de prueba las llamadas al procedimiento `are_connected` no cumplen con las restricciones descritas en los Detalles de Implementación, o el arreglo retornado por `longest_trip` es incorrecto, el puntaje de su solución para esa subtarea será 0.

En la subtarea 4 su puntaje es determinado basado en el número de llamadas al procedimiento `are_connected` realizadas en una sola invocación de `longest_trip`.

Sea q el máximo número de llamadas entre todas las invocaciones de `longest_trip` sobre cada caso de prueba de la subtarea.

Tu puntaje para esta subtarea es calculado de acuerdo a la siguiente tabla:

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

Evaluador de ejemplo

Sea C el número de escenarios, es decir, el número de llamadas a `longest_trip`. El calificador leerá la entrada con el siguiente formato:

- línea 1: C

Las descripciones de los C escenarios sigue.

El Evaluador de ejemplo lee la descripción de cada escenario con el siguiente formato:

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

Aquí, cada U_i ($1 \leq i < N$) es un arreglo de tamaño i, que describe cuales pares de atracciones están conectadas por un camino. Para cada i y j tales que $1 \leq i < N$ y $0 \leq j < i$:

- Si las atracciones j e i están conectadas por un camino, entonces el valor de $U_i[j]$ debe ser 1;
- Si no hay un camino conectando las atracciones j e i, entonces el valor de $U_i[j]$ debe ser 0.

En cada escenario, antes de llamar a `longest_trip`, el evaluador de ejemplo chequea si la densidad de la red de caminos es al menos D. Si no se cumple esta condición, imprime el mensaje `Insufficient Density` y termina.

Si el evaluador de ejemplo detecta una violación de protocolo, su salida será `Protocol Violation: <MSG>`. donde `<MSG>` es uno de los siguientes mensajes de error:

- `invalid array`: en una llamada a `are_connected`, al menos uno de los arreglos A y B
  - está vacío, o
  - contiene un elemento que no es un entero entre 0 y $N-1$, inclusivo, o
  - contiene el mismo elemento al menos dos veces.

- `non-disjoint arrays`: en una llamada a `are_connected`, los arreglos A y B no son disjuntos.
- `too many calls`: el número de llamadas hechas a `are_connected` excede $32\,640$ en la invocación actual de `longest trip`, o excede $150\,000$ en total.
- `too many elements`: el número de atracciones pasadas a `are_connected` sobre todas las llamadas excede $1\,500\,000$.

En caso contrario, sean $t[0], t[1], \ldots, t[l-1]$ los elementos del arreglo retornado por `longest_trip` en un escenario para algún entero no negativo l. El calificador de ejemplo imprimirá tres líneas para este escenario con el siguiente formato:

- línea 1: l
- línea 2: $t[0]\ t[1]\ \ldots\ t[l-1]$
- línea 3: el número de llamadas a `are_connected` en este escenario

Finalmente, el evaluador de ejemplo imprime:

- línea $1 + 3 \cdot C$: el número máximo de llamadas a `are_connected` sobre todas las llamadas a `longest_trip`