Peajes de autopista

En Japón, las ciudades están conectadas por una red de autopistas. Esta red consiste de N ciudades y M autopistas. Cada autopista conecta un par de ciudades distintas. No hay dos autopistas que conecten el mismo par de ciudades. Las ciudades se numeran desde 0 hasta $N-1$, y las autopistas se numeran desde 0 hasta $M-1$. Puede manejar en una autopista en ambas direcciones y puede viajar de cualquier ciudad a cualquier otra ciudad usando las autopistas.

Se cobra un peaje por transitar en cada autopista. Este peaje depende de las condiciones de **tráfico** en la autopista. El tráfico puede ser **ligero** o **pesado**. Cuando el tráfico es ligero, el peaje cuesta A yen (moneda Japonesa). Cuando el tráfico es pesado, el peaje cuesta B yen. Se garantiza que $A < B$. Note que usted sabe los valores de A y B.

Usted tiene una máquina que, dadas las condiciones de tráfico de todas las autopistas, calcula el menor peaje total que se debe pagar para viajar entre las ciudades S y T ($S \neq T$), bajo las condiciones de tráfico especificadas.

Sin embargo, la máquina es sólo un prototipo. Los valores de S y T están fijos (es decir, codificados directamente en la máquina). Usted desconoce estos valores pero le gustaría determinarlos. Para hacerlo, usted planea especificar varias condiciones de tráfico a la máquina y usar los valores de peaje que retorna para deducir S y T. Ya que especificar las condiciones de tráfico es costoso, no desea usar la máquina muchas veces.

Detalles de implementación

Debe implementar la siguiente función:

```
find_pair(int N, int[] U, int[] V, int A, int B)
```

- `N`: el número de ciudades.
- `U` y `V`: arreglos de tamaño M, donde M es el número de autopistas de la red. Para cada i ($0 \leq i \leq M-1$), la autopista i conecta las ciudades `U[i]` y `V[i]`.
- `A`: el peaje de una autopista cuando el tráfico es ligero.
- `B`: el peaje de una autopista cuando el tráfico es pesado.
- Esta función es llamada exactamente una vez para cada caso de prueba.
- Note que el valor de M es el largo de los arreglos y puede ser obtenido como se

indica en las notas de implementación.

La función `find_pair` puede llamar a la siguiente función:

```
int64 ask(int[] w)
```

- El largo de `w` debe ser M. El arreglo `w` describe las condiciones de tráfico.
- Para cada i ($0 \leq i \leq M-1$), `w[i]` determina la condición de tráfico en la autopista i. El valor de `w[i]` debe ser 0 ó 1.
  - `w[i] = 0` significa que el tráfico en la autopista i es ligero.
  - `w[i] = 1` significa que el tráfico en la autopista i es pesado.
- Esta función retorna el menor peaje total para viajar entre las ciudades S y T, bajo las condiciones de tráfico especificadas por `w`.
- Esta función puede ser llamada a lo sumo 100 veces (para cada caso de prueba).

`find_pair` debe llamar a la siguiente función para reportar la respuesta:

```
answer(int s, int t)
```

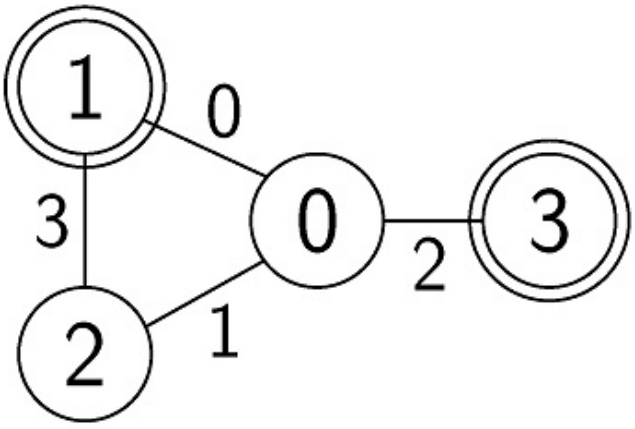
- `s` y `t` debe ser el par S y T (el orden no es importante).
- Esta función debe ser llamada exactamente una vez.

Si alguna de las condiciones anteriores no se satisface, su programa será juzgado como **`Wrong Answer`**. De lo contrario, su programa será juzgado como **`Accepted`** y su puntuación será calculada por el número de llamadas a `ask` (ver Subtareas).

Ejemplo

Sea $N = 4$, $M = 4$, $U = [0, 0, 0, 1]$, $V = [1, 2, 3, 2]$, $A = 1$, $B = 3$, $S = 1$, y $T = 3$.

El evaluador llama `find_pair(4, [0, 0, 0, 1], [1, 2, 3, 2], 1, 3)`.



En la figura de arriba, la arista con número i corresponde a la autopista i. Algunas

llamadas posibles a `ask` y los valores de retorno correspondientes son listados a continuación:

Llamada	Retorno
`ask([0, 0, 0, 0])`	2
`ask([0, 1, 1, 0])`	4
`ask([1, 0, 1, 0])`	5
`ask([1, 1, 1, 1])`	6

Para la llamada `ask([0, 0, 0, 0])`, el tráfico de cada autopista es ligero y, por lo tanto, el peaje de cada autopista es 1. El camino más barato de $S = 1$ a $T = 3$ es $1 \to 0 \to 3$. El peaje total para este camino es 2. Por lo tanto, esta función retorna 2.

Para una respuesta correcta, la función `find_pair` debe llamar `answer(1, 3)` ó `answer(3, 1)`.

El archivo `sample-01-in.txt` en el paquete zip adjunto corresponde a este ejemplo. Otros ejemplos de entradas también están disponibles en el zip.

Restricciones

- $2 \leq N \leq 90\,000$
- $1 \leq M \leq 130\,000$
- $1 \leq A < B \leq 1\,000\,000\,000$
- Para cada $0 \leq i \leq M - 1$
  - $0 \leq U[i] \leq N - 1$
  - $0 \leq V[i] \leq N - 1$
  - $U[i] \neq V[i]$
- $(U[i], V[i]) \neq (U[j], V[j])$ y $(U[i], V[i]) \neq (V[j], U[j])$ $(0 \leq i < j \leq M - 1)$
- Puede viajar de cualquier ciudad a cualquier otra ciudad utilizando las autopistas.
- $0 \leq S \leq N - 1$
- $0 \leq T \leq N - 1$
- $S \neq T$

En este problema, el evaluador NO es adaptativo. Esto significa que S y T están fijos al inicio de la ejecución del evaluador y no dependen de las preguntas realizadas por su solución.

Subtareas

1. (5 puntos) ($S = 0$ ó $T = 0$), $N \leq 100$, $M = N - 1$
2. (7 puntos) ($S = 0$ ó $T = 0$), $M = N - 1$

3. (6 puntos) $M = N - 1$, $U[i] = i$, $V[i] = i + 1$ $(0 \leq i \leq M - 1)$
4. (33 puntos) $M = N - 1$
5. (18 puntos) $A = 1$, $B = 2$
6. (31 puntos) Sin restricciones adicionales

Asuma que su programa es juzgado como **`Accepted`**, y realiza X llamadas a `ask`. Entonces su calificación P para el caso de prueba, dependiendo del número de la subtarea, se calcula de la siguiente manera:

- Subtarea 1. $P = 5$.
- Subtarea 2. Si $X \leq 60$, $P = 7$. De lo contrario $P = 0$.
- Subtarea 3. Si $X \leq 60$, $P = 6$. De lo contrario $P = 0$.
- Subtarea 4. Si $X \leq 60$, $P = 33$. De lo contrario $P = 0$.
- Subtarea 5. Si $X \leq 52$, $P = 18$. De lo contrario $P = 0$.
- Subtarea 6.
  - Si $X \leq 50$, $P = 31$.
  - Si $51 \leq X \leq 52$, $P = 21$.
  - Si $53 \leq X$, $P = 0$.

Note que su calificación para cada subtarea es el mínimo de las calificaciones para los casos de prueba en la subtarea.

Evaluador de ejemplo

El evaluador de ejemplo lee la entrada en el siguiente formato:

- línea 1: N M A B S T
- línea $2 + i$ $(0 \leq i \leq M - 1)$: $U[i]$ $V[i]$

Si su programa es juzgado como **`Accepted`**, el evaluador de ejemplo imprime `Accepted: q`, siendo `q` la cantidad de llamadas a `ask`.

Si su programa es juzgado como **`Wrong Answer`**, imprime `Wrong Answer: MSG`, donde `MSG` es uno de los siguientes:

- `answered not exactly once`: La función `answer` no fue llamada exactamente una vez.
- `w is invalid`: El largo de `w` dado a `ask` no es M o `w[i]` no es ni 0 ni 1 para algún i $(0 \leq i \leq M - 1)$.
- `more than 100 calls to ask`: La función `ask` es llamada más de 100 veces.
- `{s, t} is wrong`: La función `answer` es llamada con una pareja incorrecta `s` y `t`.