Pac-Man Go

Contribución de Guido Tagliavini Ponce

Descripción del problema

Los desarrolladores de Pac-Man están muy preocupados por la caída de la popularidad de su juego, luego del auge en los últimos años del polémico Pokémon GO. Es por esto que están planeando sacar una nueva entrega del clásico juego de arcade. En el nuevo lanzamiento, titulado Pac-Man GO, Pac-Man ya no tiene hambre, pues ya se cansó de comer tantos puntos durante tantos años del juego original. En cambio, ahora sólo le interesa escapar de los fantasmas.

Más precisamente, ahora el escenario tiene una salida, y el objetivo de Pac-Man es llegar a ella, sin ser capturado por un fantasma. **Si en algún instante Pac-Man y un fantasma se encuentran en una misma posición, Pac-Man pierde automáticamente**. A su vez, si Pac-Man llega a posicionarse sobre la salida **sin que allí haya un fantasma**, y sin haberse cruzado nunca antes con un fantasma, gana **inmediatamente** y el juego termina.

Para la primera versión de Pac-Man GO, los desarrolladores quieren que haya turnos. En el primer turno, Pac-Man puede moverse a cualquier posición contigua, o quedarse quieto. En el segundo turno, cada fantasma puede elegir moverse a una posición contigua, o quedarse quieto. Así sucesivamente, Pac-Man y los fantasmas juegan en forma alternada. El laberinto consta de **N** posiciones, numeradas con enteros entre **1** y **N** inclusive, y **M** pares de posiciones contiguas. Es posible llegar desde cualquier posición hasta cualquier otra mediante alguna secuencia de movimientos entre posiciones contiguas.

Los fantasmas inicialmente se encuentran en las posiciones $\mathbf{F_1}, \cdots, \mathbf{F_K}$ (no empiezan todos juntos en un recinto, a diferencia del Pac-Man original). Pac-Man comienza en la posición **P**. El sitio de salida es la posición **S**. Los programadores de Pac-Man GO quieren determinar si Pac-Man puede ganar el juego con alguna secuencia de órdenes adecuada, **independientemente de cómo jueguen los fantasmas**. Se te pide escribir un programa que determine una secuencia de órdenes para Pac-Man que le permita ganar el juego, sin importar cómo jueguen los fantasmas. Además, como el juego da puntos extra por cada turno que se sobrevive jugando al mismo, se desea que la secuencia de órdenes tenga **la mayor longitud posible**.

La secuencia de movimientos de Pac-Man debe estar prefijada, es decir, **la secuencia no debe depender de los movimientos de los fantasmas**.

Descripción de la función

Se debe implementar la función `pacmango(a,b,P,S,F, movimientos)`. Sus parámetros son:

- `a,b`: Arreglos de **M** enteros cada uno. Para cada **i**, las posiciones `a[i]` y `b[i]` del laberinto son contiguas entre sí. Nunca se dará dos veces el mismo par de posiciones.
- `P`: La posición inicial de PacMan
- `S`: La posición de la salida
- `F`: Arreglo de **K** enteros, con las posiciones iniciales de los fantasmas
- `movimientos`: Arreglo en el cual escribir una lista de números de posiciones a las cuales moverse a continuación en cada turno, para lograr el objetivo explicado.

Al dar la secuencia en `movimientos`, se puede indicar la misma posición en la que se encuentra Pac-Man en un cierto turno, para ordenarle quedarse quieto.

Si es posible ganar, debe ser una secuencia de órdenes lo más larga posible que asegure ganar el juego.

Si es imposible ganar, se debe producir en `movimientos` una lista de órdenes lo más larga posible que permita a Pac-Man cumplir con todas ellas y sobrevivir todos esos turnos (quizás siendo atrapado tras el movimiento de los fantasmas que sigue al último movimiento de la lista, pero no antes), sin importar lo que hagan los fantasmas.

De haber más de una secuencia de órdenes que satisfaga todo lo pedido, cualquiera de ellas vale.

La función debe retornar la longitud del arreglo `movimientos` si se calcula correctamente de acuerdo a las indicaciones anteriores.

Evaluador

El evaluador local lee de la entrada estándar con el siguiente formato:

- Una línea con los enteros **N**, **M**, **K**
- Una línea con los enteros **P**, **S**
- Una línea con **K** enteros $\mathbf{F_1}, \cdots, \mathbf{F_K}$
- **M** líneas, cada una con **a**[**i**], **b**[**i**]

Escribe a la salida estándar la respuesta retornada por la función, y en una segunda línea, el valor almacenado en `movimientos`.

Restricciones

- $\mathbf{2} \leq \mathbf{N} \leq \mathbf{100.000}$
- $\mathbf{M} \leq \mathbf{200.000}$
- $\mathbf{1} \leq \mathbf{K}$
- $\mathbf{P} \neq \mathbf{S}$
- $\mathbf{F_i} \neq \mathbf{P}$
- $\mathbf{F_i} \neq \mathbf{F_j}$ si $\mathbf{i} \neq \mathbf{j}$
- $\mathbf{1} \leq \mathbf{P}, \mathbf{S}, \mathbf{F_i}, \texttt{a[i]}, \texttt{b[i]} \leq \mathbf{N}$
- $\texttt{a[i]} \neq \texttt{b[i]}$

Ejemplos

Si la entrada fuera:

```
10 11 3
5 2
7 6 8
1 2
2 3
4 3
6 5
4 5
5 7
7 8
9 8
9 10
10 3
7 9
```

La salida correcta será:

```
3
4 3 2
```

Si en cambio la entrada fuera:

```
5 4 1
2 1
5
1 2
2 3
3 4
4 5
```

La salida correcta será:

```
3
2 2 1
```

Puntuación

Se recibe el 25 % del puntaje por el correcto valor de retorno de la función, y el 75 % restante por además dar una secuencia correcta en el arreglo `movimientos`.

Subtareas

1. `a[i]+1 == b[i]` (4 puntos)
2. No es posible ganar (16 puntos)
3. $\mathbf{N} \leq \mathbf{50}$ (20 puntos)
4. $\mathbf{N} \leq \mathbf{800}$ (12 puntos)
5. Sin más restricción (48 puntos)