¡Jugando rápido!

Contribución de Agustín Santiago Gutiérrez

Descripción del problema

Monkey Island 2: LeChuck's Revenge es el videojuego favorito de Agustín, y uno de los mejores de todos los tiempos. Entre los eventos de gran popularidad en Internet, se encuentran aquellos en los que participan famosos *speedrunners*: Talentosos jugadores que baten permanentemente récords mundiales, al jugar y completar un videojuego lo más rápido posible.

En este caso, Agustín quiere romper el récord mundial de Monkey Island 2, completando el juego lo más rápido posible.

En Monkey Island 2 hay T tareas a realizar, que se identifican utilizando enteros entre 0 y $T-1$ inclusive. El juego es completado inmediatamente cuando se realiza la tarea 0 (elegir la última línea de diálogo en la escena final).

Obviamente, no es posible completar la tarea 0 al comienzo del juego, pues primero es necesario completar muchas otras tareas para alcanzar la escena final. Más precisamente, cada tarea **necesita** que se completen **ciertas otras tareas antes** de que sea posible completarla (por ejemplo, es necesario completar las acciones "abrir la cripta" y "conseguir poción mágica" **antes** de poder completar la acción "utilizar poción mágica dentro de la cripta").

Agustín conoce muy bien el juego, que tiene N "pantallas" o "sitios" visitables, que se identifican con enteros entre 0 y $N-1$.

Para cada una de las T tareas conoce:

- El sitio s_i donde se debe estar presente para poder completar la tarea número i.
- El tiempo t_i que insume completar la tarea número i, en segundos.
- Una lista r_i con los números de las tareas que se deben completar **antes** de poder completar la tarea i.

Además, en el juego hay M senderos directos, identificados con enteros entre 0 y $M-1$. El sendero j permite desplazarse **desde** el sitio a_j **hasta** el sitio b_j. El tiempo insumido en este desplazamiento es de c_j segundos.

Se sabe que el protagonista **comienza el juego en el sitio** 0, y que **es posible ganar el juego**, pues Agustín lo ha completado en múltiples ocasiones, aunque siempre utilizando un tiempo muchísimo mayor que el récord mundial.

Con toda esta información, tu programa debe calcular el mínimo tiempo posible (en segundos) hasta completar el juego, y una secuencia de *acciones* que permita completarlo en ese tiempo.

Las acciones pueden ser de dos tipos:

- Utilizar el sendero j (Lo que desplazará al protagonista desde el sitio a_j hasta el b_j).
- Completar la tarea i (Para lo cual, deberán cumplirse todas las condiciones explicadas).

Como el juego termina inmediatamente al completar la tarea 0, la última acción realizada siempre será "completar la tarea 0": por esta razón en la lista de acciones **se debe omitir esta última acción**.

De haber más de una secuencia ideal de acciones, cualquiera de ellas vale.

Descripción de la función

Se debe implementar la función
`speedrun(T,N :`ENTEROS`,`
`s, t :` ARREGLOS[T] `de` ENTEROS`,`
`r :` ARREGLO[T] `de`
ARREGLOS DE ENTEROS`,`
`M :` ENTERO`,`
`a,b,c :` ARREGLOS[M] `de` ENTEROS`,`
`acciones :` ARREGLO `de` ENTEROS `) :`
ENTERO LARGO

Sus parámetros son:

- `T` : la cantidad de tareas.
- `N` : la cantidad de sitios visitables.
- `s,t` : arreglos con los valores s_i y t_i.
- `r` : arreglo con las listas r_i. Para cada $0 \leq i < T$, los elementos del arreglo `r[i]` indican las tareas que hay que completar **antes** de poder realizar la tarea i.
- `M` : la cantidad de senderos.
- `a,b,c` : arreglos con los a_j, b_j y c_j.
- `acciones` : arreglo en el cual se deben cargar las acciones a realizar, en orden, para completar el juego lo más rápido posible. Cada acción se representa con un valor entero:
  - Un valor no negativo j indica la acción correspondiente a utilizar el sendero j.
  - Un valor negativo $-i$ indica la acción correspondiente a completar la tarea i.
  - Notar que no hace falta codificar la tarea 0, pues como se explicó antes, esta debe omitirse de la lista de acciones por ser siempre la acción final.

Evaluador

El evaluador local lee de la entrada estándar:

- Una línea con los valores T y N.
- $2T$ líneas, dos por cada tarea, en orden desde la tarea 0 hasta la tarea $T-1$. Para la i-ésima tarea:
  - La primera línea contiene tres enteros: s_i, t_i, y la cantidad de elementos en r_i.
  - La segunda línea contiene los elementos de r_i (posiblemente 0, en cuyo caso la línea está vacía).
- Una línea con el valor M.
- M líneas, cada una con a_j, b_j y c_j.

El evaluador local escribe en la salida estándar una primera línea con el valor devuelto por la función, y una segunda línea con el arreglo `acciones` calculado.

Cotas

$1 \leq T \leq 20$

$1 \leq N \leq 300$

$1 \leq M \leq 100.000$

$0 \leq s_i, a_j, b_j \leq N-1$

$1 \leq c_j, t_i \leq 1.000.000$

$0 \leq x \leq T-1$ para todo $x \in r_i$

Ejemplo

Si se ejecuta el evaluador con los siguientes datos:

```
5 4
1 3 2
1 2
1 2 0

2 8 0

0 1 0

0 5 1
2
5
0 1 3
1 0 4
0 2 1
2 3 2
3 0 2
```

Para una solución correcta, la salida del evaluador sería:

```
21
2 -2 3 4 0 -1
```

Que corresponde a la siguiente secuencia de acciones (tiempos entre corchetes):

1. Tomar el sendero 2 ($0 \to 2$) [1]
2. Completar la acción 2 [8]
3. Tomar el sendero 3 ($2 \to 3$) [2]
4. Tomar el sendero 4 ($3 \to 0$) [2]
5. Tomar el sendero 0 ($0 \to 1$) [3]
6. Completar la acción 1 [2]
7. Completar la acción 0 [3]

Subtareas

- r_i = [] para todo $i > 0$ (21 puntos)
- $s_i = 0$ para todo i (16 puntos)
- $r_i = [i + 1]$ para $i < T - 1$ (15 puntos)
- $T \leq 10$ (10 puntos)
- $T \leq 15$ (17 puntos)
- Sin más restricción (21 puntos)

En lo anterior, $[i + 1]$ representa la lista cuyo único elemento es $i + 1$, y [] representa la lista vacía.

Puntuación

Se recibe el 50 % del puntaje por devolver correctamente el mínimo tiempo posible, y el restante 50 % por además calcular una secuencia válida de acciones que permita alcanzar ese tiempo mínimo.