El Segundo Camino X77211_es

Sabemos calcular el camino de coste mínimo entre dos vértices cualquiera (si existe), en un grafo dirigido etiquetado con números positivos, gracias al **algoritmo de Dijkstra**, que todos deberíamos conocer.
En este problema queremos calcular el coste del **segundo** camino de coste mínimo, de modo que este camino no comparta *ningúna* arista con el camino de coste mínimo original, que se ha calculado sin ninguna restricción.
Así pues, el problema es, dados:

- un grafo G dirigido y etiquetado (con números enteros positivos),
- dos vértices s y e,

Escribir una función **`segundo_mejor_camino(G,s,e)`** que calcule el coste del camino de coste mínimo en G entre s y e, y que no pasa por *ningúna* arista del conjunto de aristas E_{min} (donde E_{min} es el conjunto de aristas del camino de coste mínimo en G entre s y e, calculado sin ninguna restricción).

Precondición

G es un grafo dirigido y etiquetado con números enteros positivos. Si N es el número de vértices de G, $0 \leq s < N$ y $0 \leq e < N$

Entrada

La entrada al programa será el grafo G y los vértices s y e.
Primero tenemos un número n, que es el número de vértices del grafo (denominados, por tanto, con los números $0 \ldots (n-1)$, seguido de un número m, que es el número de aristas del grafo. A continuación tenemos m grupos de tres números u, v, w, significando la arista entre u y v con etiqueta w. Está claro que $0 \leq u, v < n$, y que $w > 0$. Por último tenemos dos números s y e, que son los vértices inicial y destino respectivamente, y $0 \leq s, e < n$.
Véanse los ejemplos que forman el juego de pruebas público.

Observaciones

Se debe descargar el archivo **`code.py`** (icono de la serpiente). Este archivo es un programa con **todo** lo necesario para ejecutar los juegos de prueba públicos. Sólo falta, claro, la función que se pide en el enunciado. Este archivo debe completarse con el código que falta, y eso, **todo**, es lo que se ha de enviar al Jutge como solución.

En el archivo **`code.py`** hay una versión modificada del algoritmo de Dijkstra. La modificación está pensada para hacer sencilla la solución a este problema. Está convenientemente comentado, pero entender y saber utilizar esta versión modificada de Dijkstra forma parte de la solución de éste problema.

También hay la función para leer grafos que ya se ha utilizado varias veces en las sesiones de laboratorio. No hace falta, por tanto, preocuparse por leer la entrada. Ya está hecho.

La eficiencia y calidad de la solución se tendrán en cuenta en la corrección manual.

Ejemplo de entrada 1

```
5 7
0 1 2
0 2 5
1 2 1
1 3 4
2 3 2
2 4 3
3 4 2
0 4
```

Ejemplo de salida 1

```
9
```

Ejemplo de entrada 2

```
7 10
0 1 7
0 2 7
0 3 2
1 4 3
2 1 5
2 5 8
3 2 4
3 5 6
5 4 4
5 6 3
0 6
```

Ejemplo de salida 2

```
-1
```

Ejemplo de entrada 3

```
6 10
1 0 6
1 5 15
3 4 3
3 1 8
4 0 20
0 5 5
0 2 1
5 1 10
4 1 2
2 3 4
3 5
```

Ejemplo de salida 3

```
23
```

Ejemplo de entrada 4

```
2 1
0 1 1000
1 0
```

Ejemplo de salida 4

```
-1
```

Ejemplo de entrada 5

```
3 3
0 2 100
0 1 40
1 2 60
0 2
```

Ejemplo de salida 5

```
100
```

Información del problema

Autoría: Jordi Delgado

Generación: 2026-01-25T19:36:43.126Z