uniandes

# Ingeniería de Sistemas y Computación

Diseño y análisis de algoritmos

Profesor: Mateo Sanabria Ardila

Parcial 2

Fecha de entrega: 22 Octubre 2024 | 2024-02 | Nota máxima: 100

1. **(25pts)** Mateo propone que una forma de saber si un grafo (con pesos positivos y sin nodos con conexiones a sí mismos) contiene ciclos es tomando la matriz de adyacencia del grafo, reemplazar la diagonal principal por $-1$ y ejecutar el algoritmo de Floyd-Warshall. Si en la diagonal de la matriz resultante hay un valor diferente de $-1$, es porque el grafo contiene ciclos. Explique por qué esta propuesta es correcta o incorrecta. Si es incorrecta, proponga una corrección.

2. **(25pts)** El profesor Adam tiene dos hijos que, lamentablemente, no se llevan bien. El problema es tan grave que no solo se niegan a caminar juntos hacia la escuela, sino que además cada uno se rehúsa a caminar por cualquier calle por la que el otro haya pasado ese día. Los niños no tienen problema en que sus caminos se crucen en una esquina. Afortunadamente, tanto la casa del profesor como la escuela están en esquinas, pero más allá de eso, no está seguro de si será posible que ambos vayan a la misma escuela. El profesor tiene un mapa de su ciudad. **Explique**, cómo formular el problema de determinar si ambos hijos pueden ir a la misma escuela como un problema de máximo flujo. **Su explicación debe ser clara y concisa, de lo contrario pondrá perder el punto.**

3. **(50pts)** Proponga una implementación (`Java,Python`) para la función `ways` que soluciona un caso del problema **¿Queda una segunda opción?**. La función recibe como parámetro el grafo (lista de aristas) asociado a la red del caso de prueba. La función debe dar solución al caso retornando el string correspondiente; inclusive si hay una respuesta numérica, la función debe retornar el string adecuado.

```
ways(network: list[(int, int, int)]) -> str
```

   Para la implementación puede suponer que ya implemento los algoritmos: `DFS, BFS, Union-Find`, `Floyd-Wars` y `Edmonds-Karp`. Sin embargo, se espera que quede claro cuáles son las entradas y salidas de cada algoritmo que decida utilizar. **Si esto no es claro o es incorrecto, su punto no será válido**. La complejidad de la solución no debe ser mayor a $O(EV)$, **de lo contrario no será válido el punto!**.

## ¿Queda una segunda opción?

Jesus, siendo el programador más talentoso de su tiempo, no puede pensar que las cosas sean tan simples. Recientemente, todos sus vecinos han decidido conectarse entre sí mediante una red (en realidad, todos quieren compartir una conexión de internet de banda ancha :-)). Pero él quiere minimizar el costo total del cable necesario, ya que es un poco exigente con los gastos del proyecto. Por alguna razón desconocida, también quiere que quede una segunda opción. Es decir, quiere saber el segundo mejor costo (si es que existe alguno que pueda ser igual al mejor costo) para el proyecto. Estoy seguro de que él es capaz de resolver el problema. Pero está muy ocupado con sus asuntos privados y seguirá así. Entonces, es tu turno de demostrar que eres un buen programador.

# Entrada

La entrada comienza con un número entero $t \leq 1000$, que indica el número de casos de prueba a manejar. Luego siguen $t$ conjuntos de datos, donde cada conjunto comienza con un par de enteros $v$ ($1 \leq v \leq 100$) y $e$ ($0 \leq e \leq 200$). $v$ denota el número de vecinos y $e$ denota el número de conexiones directas permitidas entre ellos. Las siguientes $e$ líneas contienen la descripción de las conexiones directas permitidas, donde cada línea tiene el formato 'inicio fin costo', donde inicio y fin son los dos extremos de la conexión y costo es el costo de la conexión. Todas las conexiones son bidireccionales y puede haber múltiples conexiones entre dos extremos.

# Salida

Puede haber tres casos en la salida:

1. No hay manera de completar la tarea.
2. Solo hay una manera de completar la tarea.
3. Hay más de una manera.

Imprime 'No way' para el primer caso, 'No second way' para el segundo caso y un número entero $c$ para el tercer caso, donde $c$ es el segundo mejor costo. La salida para cada caso debe comenzar en una nueva línea.