Problema del Metro de Byteville (metro)

Descripción

Bob el Programador, un exprogramador competitivo convertido en alcalde de Byteville, la capital de Byteland, necesita idear un plan para un nuevo metro de la ciudad. Ya se ha determinado que habrá cierta cantidad de estaciones y dónde estarán ubicadas.

Existen varias propuestas de líneas de metro, de las cuales Bob debe elegir un subconjunto. Cada propuesta consiste en dos estaciones conectadas directamente por la línea, junto con el costo de construcción de la línea. **Todos los costos de construcción de las líneas de metro son distintos entre sí.** Se sabe que para un buen sistema de metro, debe cumplirse la siguiente condición: debe haber **exactamente una** manera de viajar de cualquier estación a cualquier otra estación.

Además, Bob quiere que el costo total del metro (la suma de los costos de las líneas seleccionadas) sea lo más bajo posible.

Al definir formalmente el problema de esta manera, Bob recordó inmediatamente tareas similares de sus días de programación competitiva, pero no pudo recordar el algoritmo exacto que debía usar. Sin embargo, recordó un algoritmo que funciona de la siguiente manera:

1. Organizamos todas las propuestas de líneas de metro en una secuencia, en algún orden.
2. Recorremos las líneas una por una en esa secuencia, y debemos aceptar la propuesta para esa línea de metro si no entra en conflicto con las ya aceptadas.
3. Decimos que una línea entra en conflicto con las ya aceptadas si, al agregarla, existe más de una manera de viajar entre dos estaciones cualesquiera.

Bob no recuerda exactamente cómo elegir el orden de las propuestas (el primer paso). Pero sabemos que le gustan los números grandes y preferiría que estuvieran lo más cerca posible del inicio de la secuencia. Aunque el costo es lo más importante, también queremos que Bob esté feliz. Por lo tanto, tu tarea es encontrar el orden **lexicográficamente mayor** por costo de las líneas que, siguiendo el resto del algoritmo, aún produzca el metro más barato posible.

Implementación

Debes implementar la siguiente función en C++:

```
std::vector<int> orden_lineas_metro(
  int N, int M, std::vector<std::tuple<int,int,int>> lineas
)
```

O la siguiente función en Python:

```
def orden_lineas_metro(
  N: int, M: int, lineas: list[tuple[int,int,int]]
) -> list[int]:
```

Parámetros

- `N`: número de estaciones en Byteville.
- `M`: número de líneas de metro propuestas.
- `lineas`: un arreglo de M tuplas (X_i, Y_i, W_i) donde:
  - X_i, Y_i ($1 \leq X_i, Y_i \leq N, X_i \neq Y_i$) son las estaciones que conecta la línea i.
  - W_i ($1 \leq W_i \leq 10^9$, todos distintos) es el costo de construir la línea i.

Retorna

- Un arreglo de M enteros: los precios W_i de las líneas de metro propuestas, en el orden en que deben darse alalgoritmo.
- El orden debe garantizar que:
  1. Se construya un metro que permita viajar entre cualquier par de estaciones exactamente de una forma (es decir, un árbol).
  2. El costo total del metro sea mínimo.
  3. Entre los planes de costo mínimo, los números de línea más grandes aparecen lo antes posible (orden lexicográfico mayor).

Ejemplos

Ejemplo 1

En la siguiente llamada:

```
orden_lineas_metro(
  4, 5, [[1, 2, 50], [2, 3, 10], [2, 4, 20], [3, 4, 30], [1, 4, 40]]
)
```

Debe retornar `[40, 20, 50, 10, 30]`

El algoritmo elegirá la segunda, tercera y quinta línea propuesta, con un costo total de 70.

Ejemplo 2

En la siguiente llamada:

```
orden_lineas_metro(
  5, 5, [[2, 5, 12], [2, 4, 3], [1, 2, 50], [1, 4, 11], [3, 4, 36]]
)
```

Debe retornar `[36, 12, 11, 3, 50]` El algoritmo elegirá la primera, segunda, cuarta y quinta línea propuesta, con un costo total de 62.

Límites

- $1 \leq N, M \leq 3 \cdot 10^5$
- $1 \leq X_i, Y_i \leq N$
- $1 \leq W_i \leq 10^9$
- Se garantiza que la entrada admite al menos un plan válido de metro.
- $W_i \neq W_j$ para $i \neq j$
- $X_i \neq Y_i$ para $1 \leq i \leq M$
- Es posible que diferentes propuestas tengan la misma estación de inicio y fin (pero con distinto precio).

Subtareas

Subtarea 1 (7 puntos)

- $M \leq 9$

Subtarea 2 (11 puntos)

- Cada línea de metro propuesta pertenece a lo sumo a un ciclo.

Subtarea 3 (9 puntos)

- $N, M \leq 5000$
- y existe un plan de metro más barato tal que cada estación pertenece a lo sumo a dos líneas.

Subtarea 4 (12 puntos)

- Existe un plan de metro más barato tal que cada estación pertenece a lo sumo a dos líneas.

Subtarea 5 (20 puntos)

- $N, M \leq 5000$

Subtarea 6 (41 puntos)

- Sin restricciones adicionales.

> Un **ciclo** es una secuencia de estaciones $x_1, x_2, \ldots, x_k$ donde: para algún k, $x_1 = x_k$; cada otra estación no aparece más de una vez; y existe un plan de metro que conecta directamente x_i con x_{i+1} para todo $1 \leq i < k$.

Entrada

- Solo debes subir el archivo `metro.cpp` con tu solución.
- La entrada y salida serán manejadas por el archivo `grader.cpp` o `grader.py`.
- **No debes imprimir nada por la salida estándar** ni leer de la entrada estándar en tu archivo `metro.cpp`.
- No modifiques los archivos `grader.cpp` o `grader.py`, ya que el juez solo tomará en cuenta tu solución en `metro.cpp`.