PC06 ¡Que se enfría la cena!

Cada semana, mi grupo de amigos se reúne para cenar todos juntos y hablar de la vida. Esta vez me toca a mi organizar la cena, pero últimamente ha habido un montón de caos en la ciudad y muchas de las calles están cortadas, por lo que mis amigos no pueden venir por su camino habitual (de hecho, no podemos asegurar que todos puedan llegar).

Necesito un programa que, dada la descripción de las calles y el tiempo que se tarda en recorrer cada una, calcule cuántos de mis amigos podrán llegar a mi casa y cuánto tiempo tardarán el más rápido y el más lento en hacerlo, para así tener los aperitivos listos cuando llegue el primero y que la cena no se haya enfriado para cuando llegue el último.

Entrada

La entrada está compuesta por diversos casos de prueba, cada uno compuesto por varias líneas.

La primera línea de cada caso tiene 4 dígitos: El número N de cruces de la ciudad $1 < N \leq 10.000$, el número M de carreteras que no han sido cortadas $0 \leq M \leq 1.000.000$, el número X de amigos que quieren venir a cenar a mi casa $0 < X \leq 1.000$, y el cruce Z en el que se encuentra mi casa.

Las siguientes M líneas poseen 3 números: El cruce de origen A, el cruce destino B, y el tiempo C que se tarda en recorrer la calle. Cabe destacar que existen calles de sentido único y que no todas las calles tardan lo mismo en recorrerse en cada sentido.

La última línea, contiene X dígitos, correspondientes al cruce donde viven mis amigos.

Salida

Para cada caso de prueba, el programa escribirá una única línea en la que se muestre el número de amigos que pueden llegar a mi casa, seguido del tiempo que tardarán el primero y el último.

Si ninguno de mis amigos puede llegar a la cena, se imprimirá “Se cancela la cena!” en pantalla.

Entrada de ejemplo

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

2 3 15
3 4 10
1 2
3 3 2 0
0 1 10
0 2 5
1 2 1
1 2

Salida de ejemplo

2 17 25

Se cancela la cena!

Costes de las soluciones y caso TLE

El coste de la solución eficiente e ineficiente se encuentran explicados en su archivo con extensión .cpp correspondiente, adjuntados con este documento.

Para resumir, la solución eficiente posee una complejidad de O((V+A)*Log(A)) correspondiente a una única ejecución del algoritmo de Dijkstra, mientras que la complejidad del algoritmo ineficiente es del orden de O(X*(V+A)*Log(A)), correspondiente a la ejecución del algoritmo de Dijkstra tantas veces como amigos que quieren acudir a la cena.

En cuanto al tiempo máximo de ejecución del problema, y entendiendo que puede variar por muchos factores, en las pruebas realizadas el tiempo que se tardan en ejecutar todos los casos de prueba generados y adjuntos en la entrega ronda los 50 segundos para el caso eficiente, mientras que para el ineficiente tarda más de 8 minutos. Por tanto, hemos decidido que el tiempo límite de ejecución del algoritmo sea de 1 minuto y medio, margen que consideramos suficiente para asegurar que por muy ineficientemente que se codifique la solución óptima (Una única iteración del algoritmo de Dijkstra en el grafo inverso) y solo se muestre TLE cuando se codifique la subóptima (Una ejecución del algoritmo por cada cruce del que parten los amigos).