uniandes

Ingeniería de Sistemas y Computación

Diseño y análisis de algoritmos

Profesor: Mateo Sanabria Ardila

Taller DFS-BFS-Dijkstra-Bellman Ford

Fecha de entrega: 05/Abr | 2024-01 | Nota máxima: 50

Solucione los problemas propuestos usando el lenguaje de su preferencia (`python,java`). Debe ser entregado en los grupos del proyecto. Entregue un único archivo .zip. Todos los puntos valen lo mismo. **La solución a los problemas se debe basar en standard I/O, de no se asi el punto no es valido utilizando alguno de los algoritmos vistos en clase.**

Problema: Clausula transitiva

Dado un grafo dirigido y dos nodos del grafo, (G, N_1, N_2). Utilizando, **BFS** decida si es posible llegar de N_1 a N_2

Entrada

La primera línea de la entrada contiene un entero positivo T ($T \leq 1000$). Cada uno de los siguientes T casos contiene 2 líneas para cada caso: la primera línea contiene la representación del grafo en lista de adyacencia, por ejemplo la siguiente lista representa el grafo visto en clase:

$$[[1, 3, 2], [0, 3], [6, 0], [1, 4, 0, 6], [3, 5, 8, 9], [4], [3, 2, 7], [6], [4], [4]]$$

Note que el anterior grafo es un grafo no dirigido (como se vio en clase), los de este ejercicio serán dirigidos. La siguiente líneas contiene dos números enteros (Nodos) con el siguiente formato

$$0, 9$$

En este caso la respuesta seria **True**

Salida

Para cada caso, se debe imprimir una línea **True** o **False** según sea el caso.

Problema: Es árbol?

Dado un grafo ~~no dirigido~~ y utilizando **DFS** decida si el grafo dado es un árbol.

Entrada

La primera línea de la entrada contiene un entero positivo T ($T \leq 1000$). Cada uno de los siguientes T lineas contiene la representación del grafo en lista de adyacencia, por ejemplo la siguiente lista representa el grafo visto en clase:

$$[[1, 3, 2], [0, 3], [6, 0], [1, 4, 0, 6], [3, 5, 8, 9], [4], [3, 2, 7], [6], [4], [4]]$$

Salida

Para cada caso, se debe imprimir una línea **True** o **False** según sea el caso.

Problema: Aeropuertos

Hay N ciudades en el reino Wakanda y entre ellas K ciudades tienen aeropuerto. Allí las compañías aéreas están dispuestas a establecer vuelos directos entre M pares de ciudades. La monarquía piensa que no será rentable para ellos operar vuelos entre otros pares de ciudades. Tenga en cuenta que si hay aeropuertos en ambas ciudades de cualquiera de sus M pares, entonces ejecutarán vuelos entre ese par; de lo contrario, no. Debido a esta política de las compañías aéreas y también porque no todas las ciudades tienen aeropuerto, muchas personas no pueden viajar entre las ciudades deseadas. Por lo tanto, algunas personas del país han exigido que el rey haga algo para que puedan viajar a sus destinos. Hay Q demandas que llegaron al rey. Cada demanda es un par (x, y), lo que significa que algunas personas quieren una ruta entre la ciudad x y la ciudad y. El rey realmente quiere satisfacer las demandas, pero no quiere establecer demasiados aeropuertos. Entonces, para cada demanda (x, y), el rey quiere saber el número mínimo de aeropuertos que necesita establecer para establecer una ruta entre la ciudad x y la ciudad y.

Entrada

La primera línea de la entrada contiene X, el número de casos de prueba que es como máximo 10. Cada caso comienza con tres números N ($1 \leq N \leq 2,000$), M ($0 \leq M \leq 10,000$) y K ($1 \leq K \leq N$). Aquí, N es el número de ciudades, M es el número de rutas aéreas directas que las compañías aéreas están dispuestas a establecer y K es el número de ciudades que tienen aeropuerto. Luego habrá una línea que contiene K enteros que denotan el nombre de las ciudades que tienen aeropuerto (el nombre de la ciudad varía de 1 a N). Cada ciudad está separada por un solo espacio. Luego habrá M líneas, cada una de las cuales contiene un par 'a b'. Significa que las compañías aéreas están dispuestas a operar sus vuelos entre la ciudad a y la ciudad b. La siguiente línea contiene Q ($1 \leq Q \leq 50$), que es el número de demandas que llegaron al rey. Luego habrá Q líneas, cada una conteniendo un par 'x y', lo que significa que el rey debería construir el mínimo aeropuerto para establecer una ruta entre la ciudad x y la ciudad y. Hay una línea en blanco entre cada caso de prueba consecutivo.

Salida

Para cada caso de prueba, primero debe imprimir el número de caso (ver el ejemplo). Luego, debes imprimir Q líneas, cada una conteniendo el número mínimo de aeropuertos que el rey debe establecer para satisfacer esa demanda. Si es imposible satisfacer esa demanda, entonces imprime '-1'. Debes imprimir una línea en blanco después de cada caso de prueba.

Test in

```
1
6 4 4
1 2 5 6
1 2
3 5
2 4
4 5
3
1 2
1 3
1 6
```

Test out

```
Caso 1:
0
2
-1
```

Problema: Rutas

La ciudad de Dhaka se está volviendo cada vez más congestionada y ruidosa. Ciertas carreteras siempre permanecen bloqueadas debido a la congestión. Para que la gente evite las rutas más cortas, y por lo tanto las carreteras congestionadas, para llegar a su destino, la autoridad de la ciudad ha elaborado un nuevo plan. Cada intersección de la ciudad está marcada con un número entero positivo (≤ 20) que denota la actividad de la intersección. Cada vez que alguien va de una intersección (intersección de origen) a otra (intersección de destino), la autoridad de la ciudad recibe la cantidad $(actividad_de_destino - actividad_de_origen)^3$ del viajero. La autoridad te ha nombrado para averiguar la cantidad total mínima que gana cuando alguien va desde una cierta intersección (punto cero) a varias otras.

Entrada

Entrada La primera línea de cada caso de prueba contiene n (el número de intersecciones,$0 \leq n \leq 200$) seguido de n enteros que denotan la actividad de las intersecciones numeradas del 1 al n en ese orden. La siguiente línea contiene r, el número de carreteras en la ciudad. Cada una de las siguientes r líneas (una para cada carretera) contiene dos números de intersección (origen, destino) que la carretera correspondiente conecta (todas las carreteras son unidireccionales). La siguiente línea contiene el entero q, el número de consultas. Las siguientes q líneas contienen cada una un número de intersección. La entrada termina con EOF.

Salida

Salida Para cada conjunto de salidas, imprime 'Set #x' ($x = 1, 2, \cdots$) seguido de q líneas, una para cada consulta, cada una conteniendo la ganancia total mínima cuando uno viaja desde la intersección 1 (punto cero) hasta la intersección correspondiente. Sin embargo, para las consultas que dan menos de 3 unidades de ganancia total mínima, imprime un '?'.

Test in

```
5 6 7 8 9 10
6
1 2
2 3
3 4
1 5
5 4
```

```
4 5
2
4
5
```

Test out

```
Set #1
3
4
```