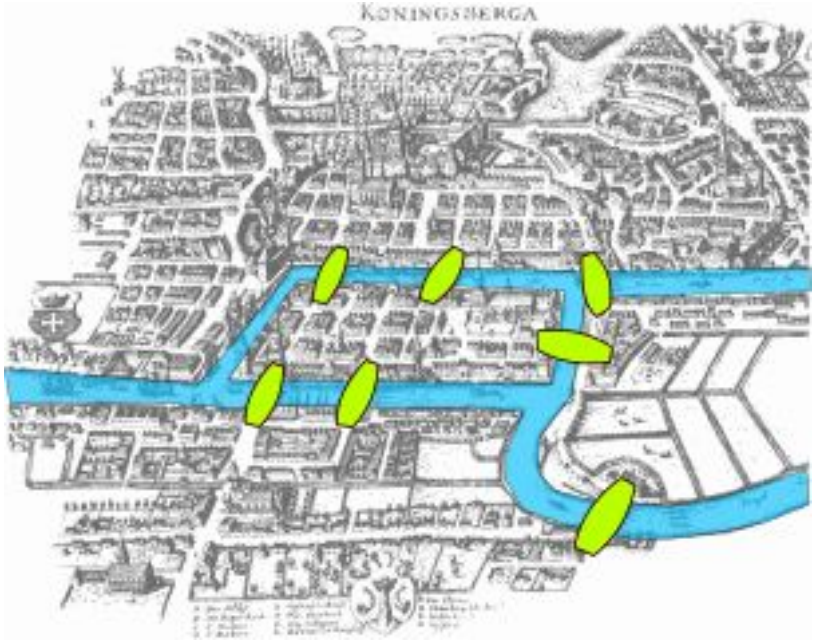
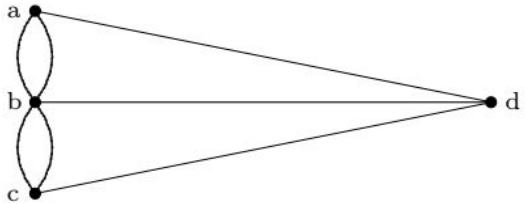
Introducción a grafos y union find

Que es un grafo?

Los habitantes de la ciudad hacían paseos dominicales tratando
de encontrar una forma de caminar por la ciudad, cruzando cada puente una sola vez, y regresando al lugar de partida.



Terminologia



Un grafo consiste de nodos (tambien llamados vertices) que están conectados con ejes.

La conexión entre un nodo y otro a través de múltiples nodos en el grafo se llama camino. La longitud de un camino siempre es la cantidad de ejes que tiene el camino.

Un grafo está conectado o es conexo si hay un camino entre cualquier par de nodos. Cada parte conectada de un grafo se llama componente

Terminologia (continuacion)

Un árbol es un grafo conexo que no contiene ciclos.

En un grafo dirigido, los ejes sólo pueden ser recorridos en una sola dirección.

En un grafo pesado, cada eje tiene un peso o costo. Pueden ser interpretados como longitudes y la longitud de un camino será la suma de los pesos de sus ejes.

Dos nodos son vecinos a adyacentes si hay un eje entre ellos. El grado de un nodo es el número de vecinos de un nodo.

Terminologia (continuacion)

Un grafo es dirigido cuando los ejes tiene una dirección.

Un grafo es pesado cuando los ejes tienen pesos.

Grado (degree) es la cantidad de vecinos de un nodo
In degree (grado de entrada) es la cantidad de ejes que llegan a un nodo.
Out degree (grado de salida) es la cantidad de ejes que salen del nodo.

Representaciones

- Lista de adyacencia
- Matriz de adyacencia
- Grafo implicito
- Lista de ejes

Lista de adyacencia

En la lista de adyacencia, a cada nodo x del grafo se le asigna un lista de adyacencia que consiste de nodos vecinos a x. Es la forma más popular de representar un grafo y muchos algoritmos pueden ser implementados eficientemente usandola.

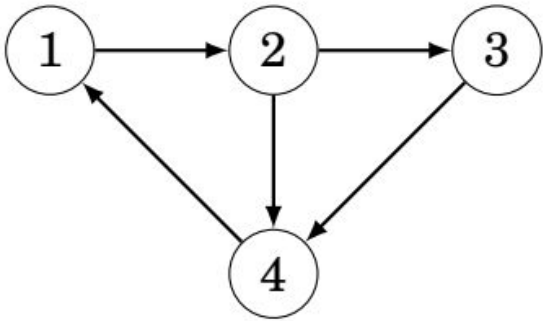
Una forma conveniente de almacenar una lista de adyacencia es declarar un arreglo de vectores o un vector de vectores:

```
vector<int> g[n];
vector<vector<int>> g(n);
vector<pair<int,int>> g[n];
```

Lista de adyacencia (continuacion)

Para almacenar el siguiente grafo, tendríamos que hacer las siguientes operaciones:

```
g[1].push_back(2);
g[2].push_back(4);
g[2].push_back(3);
g[3].push_back(4);
g[4].push_back(1);
```



Matriz de adyacencia

Una matriz de adyacencia indica los ejes que tiene el grafo, podemos ver si hay un eje entre dos nodos.
La matriz se almacena en un array de dos dimensiones.

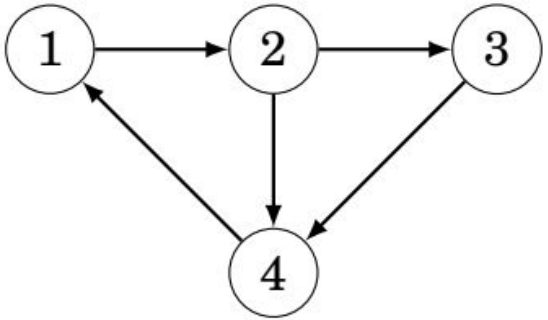
```
int g[n][n];
```

Donde cada valor m[a][b] indica si el grafo tiene un eje del nodo a al nodo b, si el nodo existe su valor será 1, si no su valor será 0.
En caso de que el grafo sea pesado, los valores serán los pesos de los ejes.

Matriz de adyacencia (continuacion)

La siguiente matriz representa al grafo de ejemplo:

1	1	0	0
0	1	1	1
0	0	1	1
1	0	0	1



Grafo implicito

En algunos problemas el grafo que se introduzca puede ser en forma de matriz, en estos casos el problema define cuales son los nodos y cuales son los ejes.
En el ejemplo los nodos son los caracteres **.** **A** y **B**
Los caracteres **#** representan paredes.
Los movimientos valos son arriba, abajo, izquierda y derecha.