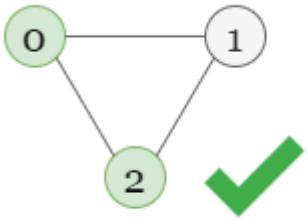
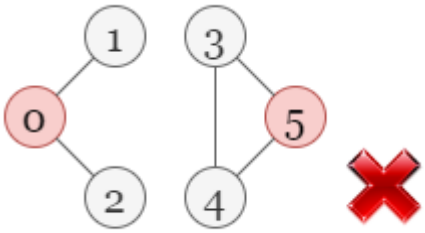
EXPLICACION

Tenemos un gráfico bidireccional en este problema, la tarea es: para los dos nodos `source` y `destination` en este gráfico, debemos verificar si existe una ruta desde `source` a `destination`.





Como se muestra en las imágenes de arriba:

- En el primer gráfico, existe un camino de `0` a `2`.
- En el segundo gráfico, no podemos encontrar un camino de `0` a `5`.

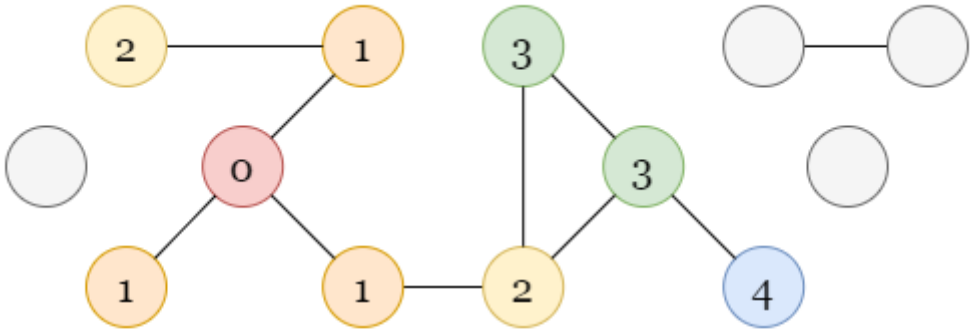
Ahora, tenemos que verificar si existe una ruta entre dos nodos dados. Esto nos indica que **se trata de un problema de recorrido de grafos** , en el que tenemos que iniciar el **recorrido desde un nodo y comprobar si podemos llegar al otro.** Existen dos métodos, búsqueda primero en amplitud (BFS) y búsqueda primero en profundidad (DFS) para recorridos de gráficos.

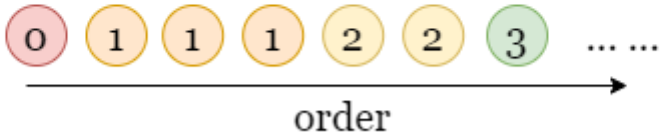
Una cosa más para notar, ya que tenemos que verificar si existe una ruta entre dos nodos dados, esto también sugiere que **dos nodos deben estar conectados** , por lo tanto, este **problema también se puede resolver usando Disjoint Set Union (DSU)** , donde verificamos si ambos nodos pertenecen al mismo grupo (por lo tanto, están conectados a través de alguna ruta) o no.

Veamos todos estos métodos en detalle uno por uno

En BFS, exploramos los nodos en el orden de su profundidad. Suponiendo que el nodo inicial tiene una profundidad de `0`, exploraremos todos los nodos en la profundidad actual (`d`) antes de pasar a todos los nodos en la siguiente profundidad (`d + 1`).

Aquí hay un ejemplo del orden en el que visitamos los nodos usando BFS, el nodo inicial está coloreado en rojo y tiene una profundidad de `0`. Los números representan la profundidad de cada nodo. Independientemente de la estructura específica, siempre visitamos el nodo de `depth = 0`, luego todos los nodos de `depth = 1`, todos los nodos de `depth = 2`y así sucesivamente.



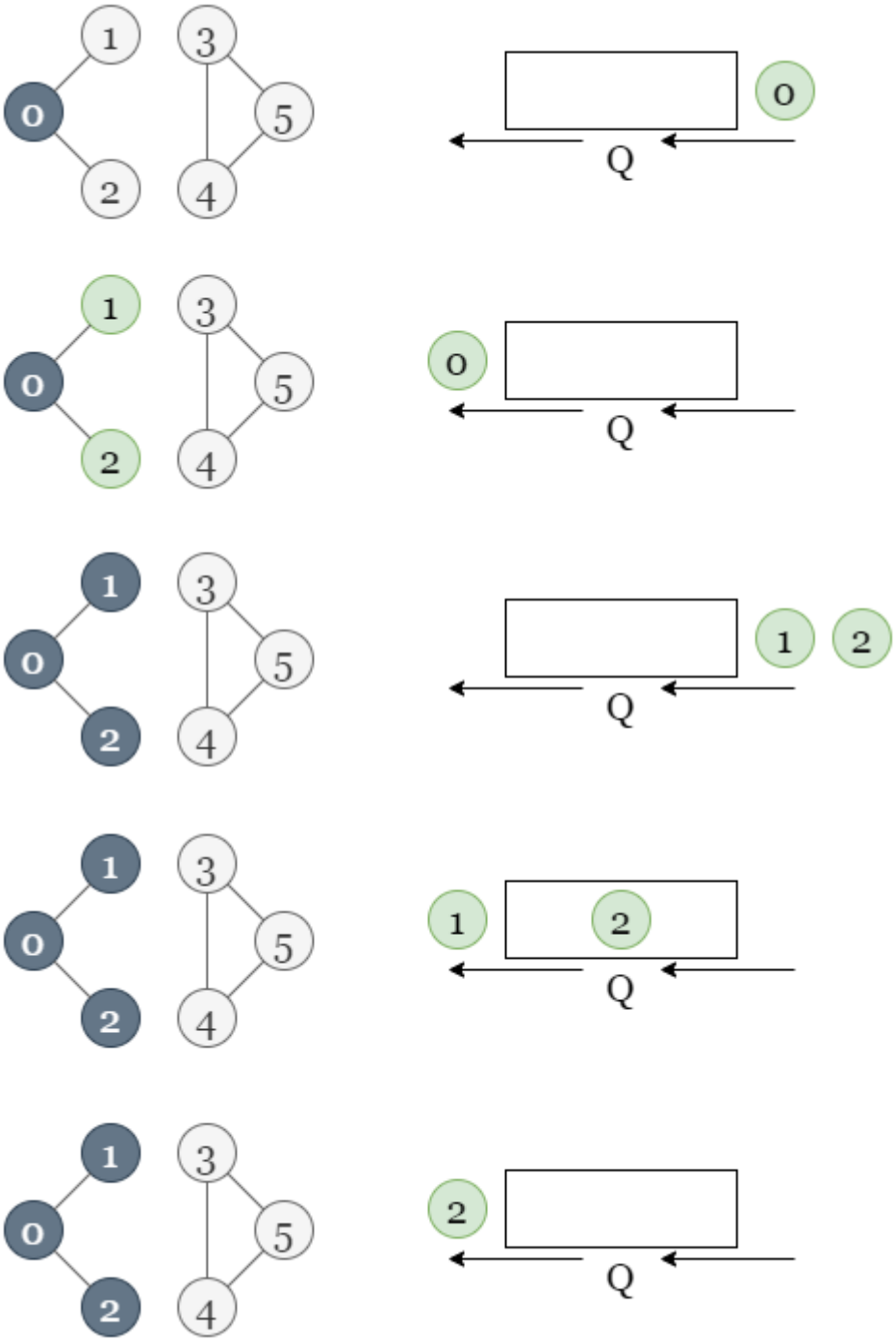


Volviendo a este problema, comenzamos con el nodo sourcecon depth = 0, luego marcamos todos sus nodos vecinos no visitados con depth = 1para ser visitados pronto, una vez que visitamos un nodo con depth = 1, marcamos todos sus nodos vecinos no visitados con depth = 2también.

Podemos usar una cola queuecomo contenedor para almacenar todos los nodos a visitar sin mezclar el orden. Dado que la operación en la cola se realiza en el orden Primero en entrar, Primero en salir (FIFO), nos permite explorar todos los nodos con la profundidad actual, antes de pasar a los nodos con mayor profundidad.

Una vez que agregamos un nodo a queue, lo marcamos inmediatamente como **visitado** para evitar que queueotros nodos lo agreguen nuevamente más adelante.

Si nos encontramos con el nodo destinationdurante el proceso, significa que existe un camino de sourcea destination. De lo contrario, indica que no podemos encontrar ese camino.



Como se muestra en la figura anterior, el nodo 0se **visita** mientras que el nodo no 5se **visita** . Por lo tanto, no hay camino de 0a 5.