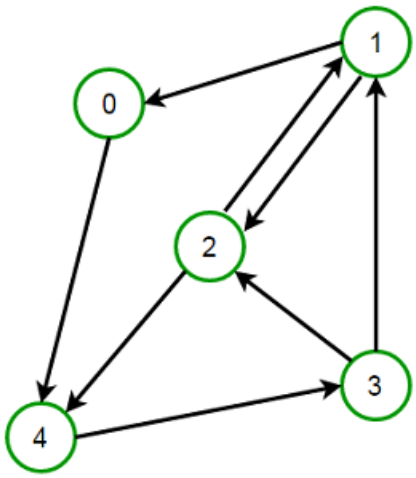
COMPLEJIDAD ALGORITMICA

PRACTICA DE EVALUACION PARCIAL – EA

PREGUNTA 1: Comprobar si un grafo está fuertemente conectado o no

Dado un grafo dirigido, comprueba si es fuertemente conexo o no. Se dice que un grafo dirigido es fuertemente conexo si todos los vértices son accesibles desde todos los demás vértices.

Por ejemplo, el siguiente gráfico está fuertemente conectado ya que existe un camino entre todos los pares de vértices:



Explique a través de un ejemplo utilizando los algoritmos: búsqueda primero en profundidad (DFS) o una búsqueda primero en amplitud (BFS) y guárdelo como archivo Pregunta1.py

Datos de entrada del programa:

Lista de aristas según el grafo

aristas = [(0, 4), (1, 0), (1, 2), (2, 1), (2, 4), (3, 1), (3, 2), (4, 3)]

total de nodos en el grafo

n = 5

Salida del programa:

1 (si es fuertemente conectado SCC)

0 (si no lo es)

Tip para solución posible:

Sin utilizar el Algoritmo de Kosaraju, una solución simple es realizar una búsqueda primero en profundidad (DFS) o una búsqueda primero en amplitud (BFS) comenzando desde cada vértice del gráfico. Si cada llamada DFS/BFS visita todos los demás vértices del gráfico, entonces el grafo está fuertemente conectado.

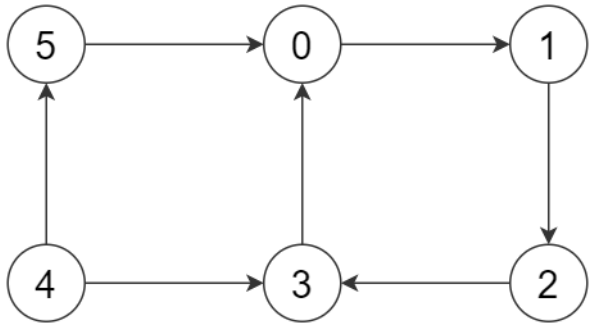
Otra solución es buscar la salida del programa aplicando el Algoritmo de Kosaraju.

PREGUNTA 2: Encontrar el vértice raíz de un grafo

Un vértice raíz de un gráfico dirigido es un vértice **u** con un camino dirigido de **u** hacia **v** para cada par de vértices **(u, v)** en el grafo. En otras palabras, todos los demás vértices del gráfico se pueden alcanzar desde el vértice raíz.

Un grafo puede tener múltiples vértices de raíz. Por ejemplo, cada vértice en un componente fuertemente conectado es un vértice raíz. En tales casos, la solución debería devolver cualquiera de ellos. Si el grafo no tiene vértices de raíz, la solución debería devolver -1. Presentar su solución en el archivo Pregunta2.py

El vértice raíz es 4 porque tiene una ruta a cada otro vértice en el siguiente gráfico:



Datos de entrada del programa:

Lista de aristas del grafo

aristas = [(0, 1), (1, 2), (2, 3), (3, 0), (4, 3), (4, 5), (5, 0)]

Número total de nodos en el grafo (0 a 5)

n = 6

Salida del programa:

4

(-1 si no encuentra un nodo raíz).

Tip para solución posible:

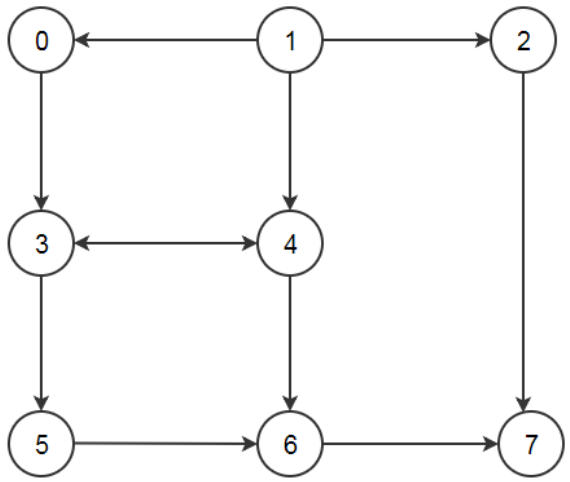
Una solución simple sería realizar un DFS (o BFS) en todos los vértices del grafo hasta que encontremos un vértice desde el cual se pueda llegar a todos los demás vértices.

Otra solución es realizar una clasificación topológica en el grafo, aunque puede que no funcione para los gráficos cíclicos.

PREGUNTA 3: Encuentra el camino entre los vértices dados en un gráfico dirigido

Dado un gráfico dirigido y dos vértices (por ejemplo, vértice de origen y de destino), determine si el vértice de destino es accesible desde el vértice de origen o no. Si existe una ruta desde el vértice de origen hasta el vértice de destino, imprímala. Implemente su respuesta en Pregunta3.py

Por ejemplo, existen dos caminos [0—3—4—6—7] y [0—3—5—6—7] de vértice 0 a vértice 7 en el siguiente grafo. Por el contrario, no hay camino desde el vértice 7 a ningún otro vértice.



Datos de entrada del programa:

Lista de aristas del grafo

aristas = [(0, 3), (1, 0), (1, 2), (1, 4), (2, 7), (3, 4), (3, 5), (4, 3), (4, 6), (5, 6), (6, 7)]

Número total de nodos en el grafo (etiquetados de 0 a 7)

n = 8

nodo_inicial = 0

nodo_final = 7

Salida del programa:

1 (si existe un camino desde el nodo_inicial al nodo_final)

0 (si no existe un camino desde el nodo_inicial al nodo_final)

Tip para solución posible:

Podemos usar el algoritmo **Breadth-first search (BFS)** para verificar la conectividad entre dos vértices en el gráfico de manera eficiente. La idea es iniciar la rutina BFS desde el vértice de origen y verificar si se alcanza el vértice de destino durante el recorrido. Si el vértice de destino no se encuentra en ningún punto, podemos decir que no es accesible desde el vértice de origen.