CS2023 - Aula de Ejercicios Nº 11

Brenner H. Ojeda Rios

Semestre 2024-0

Se sugiere que cada estudiante trate de resolver los ejercicios de forma **individual** y luego los discuta en grupo.

Ejercicios

1. (5 pts) Hay un total de `numCourses` cursos que debe tomar, etiquetados de 0 a `numCourses - 1`. Se le proporciona una arreglo `requisitos` donde `prerrequisitos[i] =` $[a_i, b_i]$ indica que debe tomar el curso a_i primero si desea tomar el curso b_i.

   Por ejemplo, el par `[0, 1]` indica que debe realizar el curso 0 antes de poder realizar el curso 1. Los requisitos previos también pueden ser indirectos. Si el curso `a` es un requisito previo del curso `b` y el curso `b` es un requisito previo del curso `c`, entonces el curso `a` es un requisito previo del curso `c`.

   También se le proporciona un arreglo de consultas donde `consultas[j]` $= [u_j, v_j]$. Para la consulta j, debe responder si el curso u_j es un requisito previo del curso v_j o no.

   Devuelve un arreglo booleano `respuesta`, donde `respuesta[j]` es la respuesta a la j-ésima consulta.

   - **Ejemplo 1:**

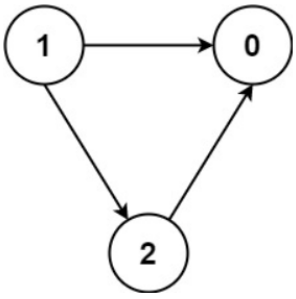
     

     `Input: numCourses = 2, prerequisites = [[1,0]], queries = [[0,1],[1,0]]`

     `Output: [false,true]`

     `Explicación:` El par `[1,0]` indica que debe realizar el curso 1 antes de poder realizar el curso 0. El curso 0 no es un requisito previo del curso 1, pero lo opuesto es verdad.

   - **Ejemplo 2:**

     `Input: numCourses = 3, prerequisites = [[1,2],[1,0],[2,0]], queries = [[1,0],[1,2]]`

     `Output: [true,true]`

   Restricciones: Sea $A = prerequisites.length$ y $B = numCourses$

   - $2 \leq$ `numCourses` ≤ 100
   - `prerequisites[i].length` $= 2$
   - $0 \leq ai, bi \leq n - 1$
   - $a_i \neq b_i$
   - Todos los pares `[`a_i`,` b_i`]` son únicos.
   - El grafo no tiene ciclos.
   - $1 \leq$ `queries.length` $\leq 10^4$
   - $0 \leq u_i, v_i \leq n - 1$
   - $ui \neq vi$
   - $0 \leq$ `A` $\leq (B * (B - 1)/2)$

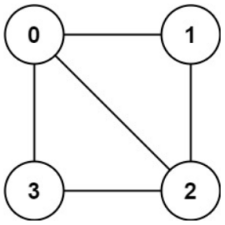
2. Hay un grafo no dirigido con n nodos, donde cada nodo está numerado entre 0 y $n-1$. Se le proporciona un grafo (arreglo 2D), donde el `grafo[u]` es un arreglo de nodos al que el nodo u es adyacente. Más formalmente, para cada v en el `grafo[u]`, hay una arista no dirigido entre el nodo u y el nodo v. El grafo tiene las siguientes propiedades:

   - No hay bucles (el `grafo[u]` no contiene u).
   - No hay aristas paralelas (el `grafo[u]` no contiene valores duplicados).
   - Si v está en el `grafo[u]`, entonces u está en el `grafo[v]` (el grafo no está dirigido).
   - Es posible que el grafo no esté conectado, lo que significa que puede haber dos nodos u y v tales que no haya un camino entre ellos.

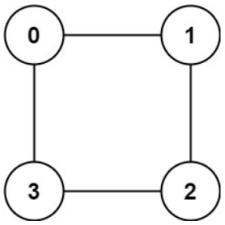
   Un grafo es bipartito si los nodos se pueden dividir en dos conjuntos independientes A y B de modo que cada arista del gráfico conecte un nodo del conjunto A y un nodo del conjunto B.

   Devuelve verdadero si y sólo si es bipartito.

   - **Ejemplo 1:**

     `Input: graph = [[1,2,3],[0,2],[0,1,3],[0,2]]`
     `Output: false`
     `Explicación:` No hay forma de dividir los nodos en dos conjuntos independientes de modo que cada arista conecte un nodo en uno y un nodo en el otro.
   - **Ejemplo 2:** `Input: graph = [[1,3],[0,2],[1,3],[0,2]]`

     `Output: true` (Podemos dividir los nodos en dos conjuntos: 0, 2 y 1, 3.)

   Restricciones:

   - `graph.length` = n
   - $1 \leq n \leq 100$
   - $0 \leq$ `graph[u].length` $< n$
   - $0 \leq$ `graph[u][i]` $\leq n-1$
   - `graph[u]` no contiene u.
   - Todos los valores de `graph[u]` son únicos.
   - Si `graph[u]` contine v, el `grafo[v]` contiene u.

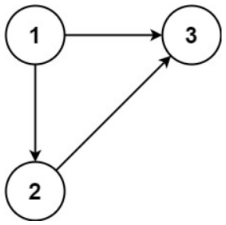
3. En este problema, un árbol con raíz es un grafo dirigido tal que hay exactamente un nodo (la raíz) para el cual todos los demás nodos son descendientes de este nodo, además cada nodo tiene exactamente un padre, excepto el nodo raíz que no tiene padres.

   La entrada dada es un grafo dirigido que comenzó como un árbol enraizado con n nodos (con valores distintos de 1 a n), con una arista dirigida adicional agregada. La arista agregada tiene dos vértices diferentes elegidos del 1 al n, y no era una arista que ya existiera.

   El grafo resultante se presenta como una arreglo 2D de aristas. Cada elemento de aristas es un par $[u_i, v_i]$ que representa una arista dirigida que conecta los nodos u_i y v_i, donde u_i es el padre de v_i.

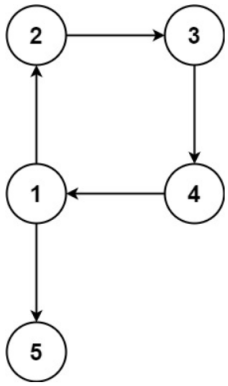
   Devuelve una arista que se puede eliminar para que el grafo resultante sea un árbol enraizado de n nodos. Si hay varias respuestas, devuelve la respuesta que aparece en último lugar en el arreglo 2D dado.

   - **Ejemplo 1:**


   ```
   Input: edges = [[1,2],[1,3],[2,3]]
   Output: [2,3]
   ```

   - **Ejemplo 2:**


   ```
   Input: edges = [[1,2],[2,3],[3,4],[4,1],[1,5]]
   Output: [4,1]
   ```

   Restricciones:

   - $n =$ `edges.length`
   - $3 \leq n \leq 10^3$
   - `edges[i].length` $= 2$
   - $1 \leq u_i, v_i \leq n$
   - $u_i \neq v_i$