**Algoritmo de Ordenación Topológica**

**Precondición:**

* Las restricciones están representadas como pares de números enteros **(i, j)** donde **i** precede a **j**.

**Postcondición:**

* Se devuelve una lista ordenada de las tareas que cumplen las restricciones.

**Algoritmo:**

1. Inicializar un diccionario **graph** donde las claves son las tareas y los valores son listas de tareas que dependen de ellas.
2. Inicializar un diccionario **in\_degree** con el grado de entrada de cada tarea (el número de tareas que dependen de ella).
3. Inicializar una cola vacía **queue**.
4. Inicializar una lista **topological\_order** para almacenar el resultado.
5. Llenar **graph** y **in\_degree** según las restricciones.
6. Para cada tarea **task**:
   1. Si el grado de entrada de **task** es 0, añadir **task** a **queue**.
7. Mientras **queue** no esté vacía:
   1. Sacar una tarea **current\_task** de **queue**.
   2. Añadir **current\_task** a **topological\_order**.
   3. Para cada tarea **neighbor** de **current\_task** en **graph**:
      1. Decrementar el grado de entrada de **neighbor**.
      2. Si el grado de entrada de **neighbor** es 0, añadir **neighbor** a **queue**.
8. Si la longitud de **topological\_order** es igual a **n**, devolver **topological\_order**, de lo contrario, no hay una ordenación válida.