UT7 - PD5

Ejercicio 1

codigo que determine si hay ciclos en un grafo

Algoritmo determina si hay ciclo ~~tiene Ciclos DFS~~

```
Función tiene Ciclos (grafo)
    n = grafo.numeroDeVertices()
    estados = array[n]

    Para cada vertice en grafo hacer
        Si estados[v] == 0:
            Si DFS(v, estados, grafo) == true:
                Devolver true
            Fin si
        Fin si
    Fin Para
    Devolver false

Función DFS(v, estados, grafo)
    estados[v] = 1

    Para cada vecino u de v en grafo hacer
        Si estado[u] == 1
            Devolver true
        Fin si
        Si estados[u] == 0
            Si DFS(u, estados, grafo) == true
                Devolver true
            Fin si
        fin si
    Fin Para
    Estados[v] = 2
    Devolver false
```

Orden de Tiempo de Ejecución de este Algoritmo: El orden del tiempo de ejecución de este algoritmo, es de O(n). Esto debido a que el algoritmo se ejecuta v. veces, siendo v el número de vertices. Este algoritmo recorre todos los vertices de un grafo, viendo sai estan visitados o no, determinando así si hay ciclos o no.

Ejercicio 2

Lenguaje Natural: El algoritmo debe de recibir un grafo dirigido y acíclico por parametro, y apartir de esto es que calcula una lista de vertices en orden topologico. Para esto es que visitara cada vertice.

Precondiciones: El grafo pasado por parametro debe ser dirigido y acíclico (DAG).

Postcondiciones: - se obtiene una lista de vértices en orden topológico
- No modifica el arbol inical

Algoritmo OrdenTopológico

```
Funcion ordenación Topologica (grafo)
    n = grafo.numeroDeVertices()
    visitados = array [n]
    ordenacion = lista vacía

    Para cada vertice v en grafo: hacer
        si vertice [v] == false
            DFS (v, visitado, ordenación, grafo)
        Fin si
    fin para
    ordenación = invertir (ordenación)
    Devolver ordenación
```

```
Función DFS (v, visitados, ordenacion, grafo)
    visitados [v] = verdadero

    Para cada vecino u de v en grafo hacer
        Si visitados [u] == false
            DFS (u, visitados, ordenacion, grafo)
        Fin Si
    Fin Para
    agregar Inicio (v, ordenación)
```

~~Cómo hacías para obtener todas las ordenaciones topológicas existentes?~~

Ejercicio 3

Algoritmo Determinar si es conexo

```
Función es Conexo (grafo): Booleano
    n = grafo.numeroDeVertices()

    visitados = array [n]
    Para i = 0 hasta n-1 hacer
        visitados [i] = false
    Fin Para

    DFS (0, visitados, grafo)

    Para cada v en grafo
        si visitados [v] == false
            Devolver false
        Fin si
    Fin para

    grafoTranspuesto = transponer (grafo)
```

```
Para i = 0 hasta n-1
    visitados[i] = false
Fin Para

DFS(0, visitados, grafoTranspuesto)

Para cada v en grafo:
    si visitados[v] == false
        Devolver false
    fin si
fin Para

Devolver Verdadero

Función Transponer(grafo): grafo
grafoTranspuesto = nuevo grafo(grafo.numeroDeVertices())

Para cada v en grafo:
    Para cada vecino u de v en grafo:
        grafoTranspuesto.agregaArista(u, v)
    Fin Para
Fin Para

Devolver grafoTranspuesto
```

Analisis tiempo de ejecución - El tiempo de ejecución de este algoritmo es de N^2 en el peor caso, esto debido a que dependiendo de la cantidad de vertices en el grafo (v) y la cantidad de vecinos de es vertice (u), es que lo recorre y lo va agregando.