**UNIVERSIDAD NACIONAL DE CÓRDOBA**

**Facultad de Ciencias Exactas Físicas y Naturales**

Parcial Nº3

**Profesores:**

Ayme

Wolfmann

**Integrantes:**

Barbiani, Gianfranco

Gutierrez, Cristian

Sepúlveda, Federico

**Año:** 2013

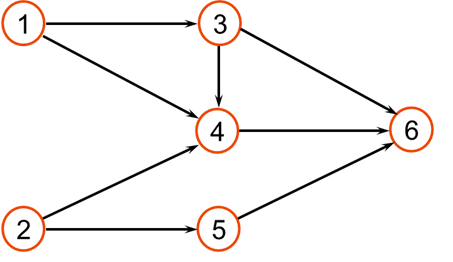
**Enunciando:** Implementar un ordenamiento topológico de las materias de la carrera de ingeniería en computación. Para ello deberá definir una clase grafo dirigido, cuyos vértices representen las materias y cada arista represente la correlatividad entre las materias. El grafo deberá implementarse con una lista de adyacencia, no con una matriz. Como resultado deberá imprimirse una lista de las materias ordenadas.   
Adicionalmente deberá obtener 4 listas adicionales con ordenamientos diferentes utilizando búsqueda en profundidad y otras 4 utilizando búsqueda en amplitud. Para estos últimos 8 casos, deberá utilizar otro grafo cuyos vértices representen el estado de avance de la carrera y las aristas, el haber aprobado una materia.

**Marco teórico:**

**Ordenamiento topológico:**

El algoritmo de ordenamiento topológico se aplica a un grafo dirigido. La condición es que el grafo no tenga ciclos. Por ejemplo una lista de materias de una carrera se puede realizar con grafos donde las materias son los vértices y las aristas son las correlaciones entre dichas materias. Para realizar un ordenamiento topológico se aplica una modificación del algoritmo de búsqueda en profundidad, la cual se va aplicando a través de cada nodo en forma recursiva hasta que llega a un nodo final, una vez ahí se empieza a guardar en la pila y comienza a volver en la recursión.

Ejemplo:



Si se empieza por 1 el ordenamiento topológico, luego se va a 3 luego a 6 como no se puede seguir avanzando se guarda 6 en la pila se vuelve a 3 y se va hacia 4 como no se puede seguir avanzando ya que 6 ya fue visitado se guarda a 4 en la pila, se vuelve a 3 como 4 y 6 ya fueron visitados no se puede avanzar entonces se guarda a 3 en la pila y vuelve a 1, como 4 y 3 ya fueron visitados no se puede avanzar se guarda a 1 en la pila. Va 2, como 4 ya fue visitado se va para 5 como 6 ya fue visitado no se avanza guardo 5 en la pila vuelve a 2 como 4 y 5 ya fueron visitados se guarda a 2 en la pila y finaliza la recursión.

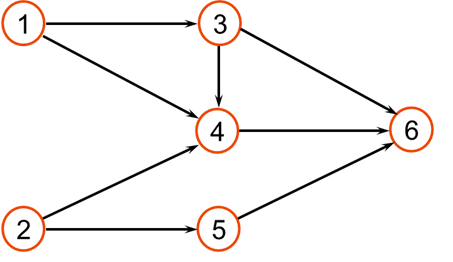
Luego se vacía la pila quedando el ordenamiento topológico de la siguiente manera:

2- 5- 1- 3- 4- 6

**Búsqueda en profundidad:**

Es el algoritmo que se usa en el ordenamiento topológico con la diferencia que para el caso del ordenamiento se trabaja con una pila para ir almacenando los nodos una vez que ya no se pueda avanzar. En este caso no se usa una pila si no que se usa una cola

Ejemplo:

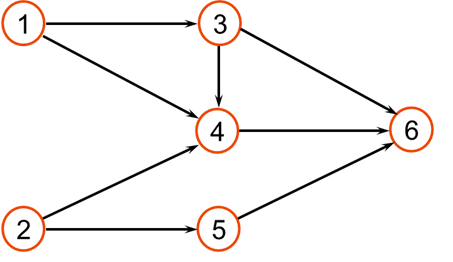


Si la búsqueda en profundidad se inicia en 1, se almacena el 1, luego se avanza hacia 3 se guarda el 3, luego se va hacia el 6 se almacena el 6, luego se vuelve a 3 y avanza a 4 se guarda 4, se vuelve a 3 luego vuelve a 1 como no se puede avanzar a 3 y 4 porque ya fueron visitados, finaliza la recursión. Al desencolar se tiene: 1-3-6-4

**Búsqueda en amplitud:**

Este algoritmo revisa en cada nodo vecino a un nodo raíz, luego avanza hacia uno de los nodos vecinos y revisa todos los nodos vecinos a este cuando finaliza vuelve al nodo raíz y avanza hacia otro vecino de este que aún no haya sido visitado, luego se revisan todos los vecinos de este nodo y así continua iterativamente hasta revisar a cada nodo del grafo.

Ejemplo:



Si la búsqueda en amplitud comienza de 1, se guarda a 1 en la cola, avanza a 3, se guarda 3 en la cola se vuelve a 1 y va hacia 4, se guarda 4 en la cola, vuelve a 1 va 3 luego a 6 se guarda 6 en la cola, vuelve a 3 vuelve a 1 finaliza las iteraciones. Al desencolar se obtiene: 1-3-4-6.

**Implementación:** Se utilizaron 3 clases grafos, una para ordenamiento topológico otra para búsqueda por profundidad y otra para búsqueda por amplitud.

A la clase ordenamiento topológico se le pasa la cantidad de vértices cuando es creada luego se le cargan las aristas por medio del método addArista(u,v) donde u y v son vértices entonces esta sería la relación entre los mismo (u que va a v). Luego para imprimir por pantalla la lista de los vértices ordenados topológicamente se llama al método topologicalSort().

A la clase de búsqueda por profundidad se le pasa el número de elemento hasta el cual va a poder llegar, después se le cargan las aristas por medio del método addArista(v,w). Luego para que realice la búsqueda por profundidad se llama al método DFS(v) donde v es el vértice donde se comienza la búsqueda, dentro del método se llama a un método auxiliar para realizar la recursividad una vez dentro del método auxiliar se imprime en pantalla los vertices.

La clase búsqueda por amplitud se la crea por medio de constructor Grafo\_2(V) donde V es el numero de vértice al puede llegar como máximo, luego se cargan las aristas por medio del método addArista(v,w). Finalmente se realiza la búsqueda por amplitud por medio del método BFS(s) donde s es el vértice por donde se va a comenzar la búsqueda luego la cual imprime la lista de vértices en pantalla.