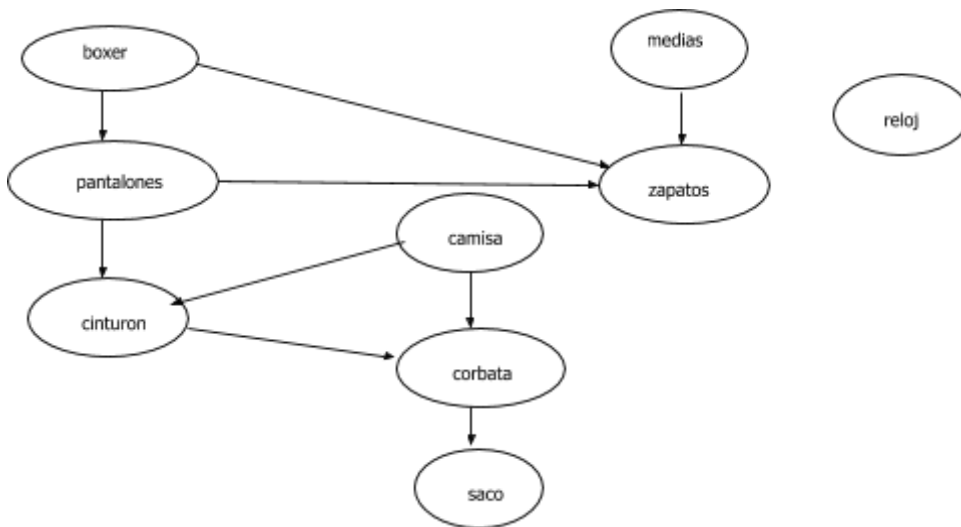


Programación III
TEMA 8: Grafos
Práctica nº 8 - B

Ejercicio 1

La organización topológica (o “sort topológico”) de un DAG (grafo dirigido acíclico) es un proceso de asignación de un orden lineal a los vértices del DAG de modo que si existe una arista (v,w) en el DAG, entonces v aparece antes de w en dicho ordenamiento lineal.

- a) El siguiente DAG surge cuando el Profesor Miguel se viste a la mañana. El profesor debe ponerse ciertas prendas antes que otras (por ejemplo las medias antes que los zapatos). Otras prendas pueden ponerse en cualquier orden (es el caso de las medias y los pantalones). Una arista dirigida (v,w) en el dag indica que la prenda v debe ser puesta antes que la prenda w . Indique al profesor un posible orden de colocación de prendas.



- b) Implemente en JAVA una clase llamada **OrdenTopologico** ubicada dentro del paquete **prog3.grafos.util** cumpliendo la siguiente especificación:

ordenTopologico(Grafo<T> grafo): ListaGenerica<Vertice<T>> // Retorna una lista de vértices con un orden topológico del *grafo* recibido como parámetro.

- c) Calcule el tiempo de ejecución para el método del inciso anterior.

Ejercicio 2

- a) Implemente en JAVA una clase llamada **Dijkstra** ubicada dentro del paquete **prog3.grafos.util**, cumpliendo la siguiente especificación:

dijkstraSinHeap (Grafo<T> grafo, Vertice<T> v): Costo [] // Este vector almacena los datos del camino mínimo desde el origen v a cada uno de los restantes vértices del grafo. El vector es de dimensión igual a la cantidad de vértices, cada posición del mismo representa la información obtenida para el vértice con igual posición. **Costo** es un objeto que contiene el costo mínimo de acceder al vértice y la posición del Vértice por el cual hay que pasar previamente, a fin de poder rearmar el camino mínimo.

Implemente dijkstra sin utilizar Heap.

- b) Calcule el tiempo de ejecución para el método del inciso anterior.

- c) Implemente el método **dijkstraConHeap** (**Grafo<T> grafo, Vertice<T> v**): **Costo []**. Valen las mismas anotaciones realizadas en el inciso a.
- d) Calcule el tiempo de ejecución para el método del inciso anterior.
- e) Mostrar mediante un ejemplo que el algoritmo de dijkstra falla cuando existen en el grafo aristas de costo negativo.

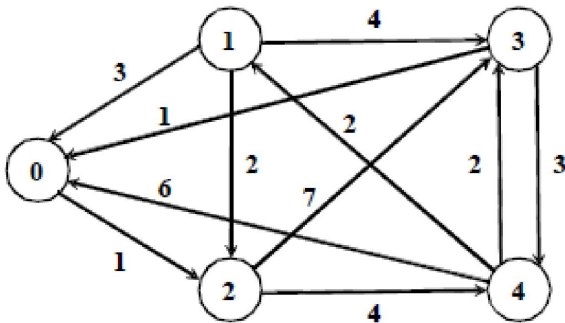
Ejercicio 3

Modifique el método dijkstra (elijá uno de los dos implementados) del ejercicio 2 para que además contabilice el número de diferentes caminos mínimos desde el vértice origen al resto de los vértices del grafo. Agregue el nuevo método **dijkstraTodosMinimos** (**Grafo<T> grafo, Vertice<T> v**): **CostoTodosMinimos []** a la clase Dijkstra. Qué información contendrían los objetos de la clase *CostoTodosMinimos* ?

Ejercicio 4

Dado el grafo orientado que se muestra en la figura, se pide:

- a) Calcular mediante el algoritmo de Floyd, el camino mínimo desde cualquier vértice (origen) al vértice (destino) 0 de dicho grafo, es decir el camino mínimo desde los vértices 1, 2, 3 y 4 al vértice 0. Mostrar para ello, la secuencia de matrices de costos (D) y la secuencia de matrices de vértices intermedios o de paso (A).
- b) Calcular además, cuál es el vértice del grafo que más veces se considera como vértice intermedio o de paso al aplicar el algoritmo de Floyd.



- c) Implemente en JAVA una clase llamada **Floyd** ubicada dentro del paquete **prog3.grafos.util**, cumpliendo la siguiente especificación:

floyd (Grafo<T> grafo): Costo [][] // Esta matriz almacena la información del camino mínimo entre cada par de vértices del grafo. La matriz es $n \times n$ donde n = cantidad de vértices. *Costo* es un objeto que contiene el costo mínimo y la posición del Vértice por el cual hay que pasar previamente.

Ejercicio 5

Se desea mantener un conjunto de antenas situadas estratégicamente por una zona determinada. Se conoce cuál es el costo de ir de una antena a otras antenas cercanas. El equipo de mantenimiento trata de optimizar las rutas de visita a las antenas de forma que el costo de mantener las antenas sea mínimo.

El mapa de antenas junto con el coste de ir de unas antenas a otras lo representaremos en la siguiente matriz:

Facultad de Informática - UNLP
Año 2020

	Antena 1	Antena 2	Antena 3	Antena 4	Antena 5	Antena 6	Antena 7
Antena 1	0	7	2	6	9		8
Antena 2	7	0		3			
Antena 3	2		0		6		
Antena 4	6	3		0			3
Antena 5	9		6		0	3	
Antena 6					3	0	2
Antena 7	8			3		2	0

Cuando no aparece valor entre dos antenas es porque no se puede llegar directamente desde una a la otra.
Se pide:

- (a) ¿Qué algoritmo se puede aplicar para calcular el costo mínimo para ir desde la antena 1 hasta la antena 7?
Mostrar el árbol de caminos de longitud mínima desde la antena 1.