Trabajo práctico 3 - Grafos

Programación 3 - TUDAI

Ejercicio 1.
Implemente en JAVA las clases GrafoDirigido y GrafoNoDirigido.

Ejercicio 2.
Implemente los recorridos Depth-First-Search y Breadth-First-Search.

Ejercicio 3.
Implemente un algoritmo que determine si un grafo dirigido tiene algún ciclo.

Ejercicio 4.
Escribir un algoritmo que, dado un grafo dirigido y dos vértices *i*, *j* de este grafo, devuelva el camino simple (sin ciclos) de mayor longitud del vértice *i* al vértice *j*. Puede suponerse que el grafo de entrada es acíclico.

Ejercicio 5.
Escriba un algoritmo que dado un grafo *G* y un vértice *v* de dicho grafo, devuelva una lista con todos los vértices a partir de los cuales exista un camino en *G* que termine en *v*.

Ejercicio 6.
Supongamos una conexión entre computadoras (1, ... ,*n*) que se encuentra modelada mediante un grafo. Se requiere, si existe, dar una conexión entre dos computadoras *a* y *b* existentes sabiendo que la computadora i está fuera de servicio.

Ejercicio 7.
Supongamos que una ciudad se encuentra modelada mediante un grafo, donde cada nodo es una esquina, y las aristas representan las calles. Diseñe un algoritmo tal que dadas dos esquinas, devuelva el camino más corto entre ambas de manera de caminar la menor cantidad de cuadras posible.

Ejercicio 8
Dados un grafo G con sus vértices rotulados con colores y dos vértices v1 y v2, escriba un algoritmo que encuentre un camino desde el vértice v1 al vértice v2 tal que no pase por vértices rotulados con el color rojo.

Ejercicio 9
Dado un grafo no orientado que modela las rutas de la provincia de Buenos Aires, devolver todos los caminos alternativos que se pueden tomar para ir desde la ciudad de Buenos Aires a la ciudad de Tandil, considerando que en el tramo Las Flores-Rauch está cortado al tránsito.

Ejercicio 10

Se dispone de un conjunto de tareas, donde cada tarea tiene un nombre, una descripción y una duración (medida en horas). Se sabe también que hay una dependencia en el orden posible en el cual se pueden ejecutar estas tareas y un tiempo de espera entre dos tareas consecutivas (también medido en horas). Por ejemplo, si la tarea B depende de la tarea A y tiene un tiempo de espera de 5 horas; significa que:

- B no puede ejecutarse antes que A y,
- B debe ejecutarse 5 horas después de haber finalizado la ejecución de A.

Objetivo

Implementar un algoritmo que obtenga la secuencia de ejecución crítica de estas tareas, es decir, la secuencia de tareas que resulta en el máximo tiempo empleado para su ejecución.

Por ejemplo: si partimos de la siguiente configuración podemos encontrar el camino crítico en la secuencia de tareas [0, 2, 5, 6, 10], ya que su tiempo de ejecución es la duración de cada tarea más el tiempo de espera entre cada par de tareas: 70 horas.