TRABAJO PRACTICO N°3
Tema: Grafos

1.
G1=(V, A)
V(G1)={perro, gato, animal, vertebrado, ostra, crustáceo, invertebrado, cangrejo, perro de caza, mono, banana, dálmata, perro doméstico}
A(G1)={(vertebrado, animal), (invertebrado, crustáceo), (perro, vertebrado), (gato, vertebrado), (mono, vertebrado), (crustáceo, invertebrado), (cangrejo, crustáceo), (ostra, crustáceo), (perro de caza, perro), (dálmata, perro), (perro doméstico, perro)}

1.1. Dibujar el Grafo1 (dirigido).

1.2. Dibujar la matriz de adyacencia y la lista de adyacencia para el Grafo1.

1.3. Para decir si un elemento del Grafo1 tiene una relación "*x*" con otro elemento, buscar un camino entre ellos. Mostrar si las siguientes afirmaciones son ciertas, usando el dibujo o la matriz de adyacencia:

a) dálmata "*x*" perro
b) dálmata "*x*" vertebrado
c) dálmata "*x*" perro de caza
d) banana "*x*" invertebrado
e) ostra "*x*" invertebrado
f) mono "*x*" invertebrado

1.4. ¿Cuáles de las siguientes opciones describe mejor la relación representada "*x*" por las aristas en la pregunta anterior?. Explicar.

a) "tiene un"
b) "es un ejemplo de"
c) "es una generalización de"
d) "come"

2.
G2=(V, A)
V(G2)={Susana, Daniela, Miguel, Fernando, Juana, Sandra, Leonardo, Jessica, Brenda, Franco}
A(G2)={(Susana, Daniela), (Fernando, Brenda), (Sandra, Susana), (Leonardo, Franco), (Sandra, Franco), (Franco, Juana), (Leonardo, Jessica), (Jessica, Susana), (Miguel, Daniela), (Brenda, Leonardo), (Susana, Juana)}

2.1. Dibujar el Grafo2 (no dirigido).

2.2. Dibujar la matriz de adyacencia y la lista de adyacencia para el Grafo2.

2.3. Usando la matriz de adyacencia del Grafo2 del inciso anterior, y a partir de dos nombres, como por ejemplo Susana y Leonardo.

a) Describir el camino desde Susana a Leonardo.
b) Describir el camino desde Leonardo a Susana.
c) ¿Cuál es el camino mas largo de Leonardo a Susana?.
d) ¿Cuál es el camino mas corto de Leonardo a Susana?.

2.4. ¿Cuál de las siguientes opciones describe mejor la relación representada por las aristas entre los vértices del Grafo2?. Explicar.

a) "trabaja para"
b) "es el mejor amigo de"
c) "es el profesor de"

TRABAJO PRACTICO N°3
Tema: Grafos

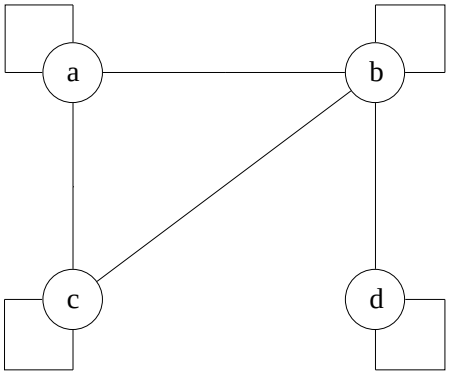
d) "trabaja con"

3. ¿Cuántos vértices tiene un grafo que contiene:
a) 16 aristas y todos los vértices de grado 2?.
b) 21 aristas, 3 vértices de grado 4 y los demás de grado 3?.

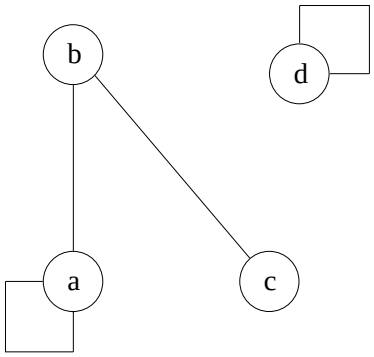
De ahora en adelante, los ejercicios deben realizarse en máquina. Deberán probar el código entregado e implementar todo lo que falte. Sería recomendable que todos los ejercicios relacionados se agrupen y se arme algún tipo de menú, para las cargas, eliminaciones, modificaciones, listados, etc.

4. Cargar mediante matriz de adyacencia y lista de adyacencia los siguientes grafos:

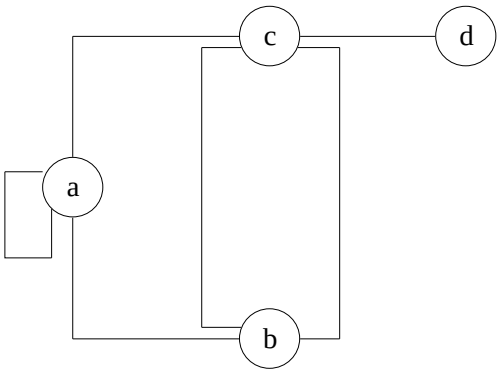
a.-



b.-

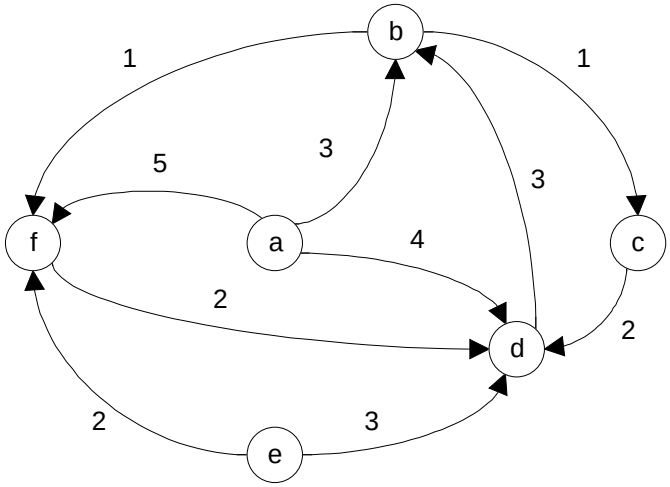


c.-

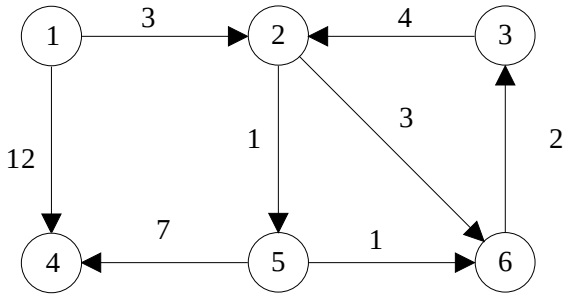


5. Realizar la misma tarea que el ejercicio anterior, incluyendo el manejo de costes para cada tipo de implementación, a partir de la siguiente figura:

TRABAJO PRACTICO N°3
Tema: Grafos



6. Usando la matriz de adyacencia del siguiente grafo, describir el camino a partir de *1*.



a) Usando una estrategia primero en profundidad (depth-first).
b) Usando una estrategia primero en anchura o amplitud (breadth-first).

7. A partir del grafo dirigido del ejercicio anterior:
a) Emplear el algoritmo de **Dijkstra** para encontrar los caminos más cortos que van del vértice "**1**" a los otros vértices.
b) Utilizar el algoritmo de **Floyd** para encontrar las distancias más cortas entre todos los pares de puntos. Construir también la matriz P que permita recuperar los caminos más cortos.

8. Supóngase que G=(V, A) es un grafo conexo, dónde cada arista (u, v) de A tienen un costo asociado c(u, v). Un **árbol abarcador de costo mínimo** para G es un árbol que conecta todos los vértices de V; su **costo** es la suma de los costos de las aristas del árbol.

Una aplicación típica de los árboles abarcadores de costo mínimo tiene lugar en los diseños de redes de comunicación. Los vértices del grafo representan ciudades, y las aristas las posibles líneas de comunicación entre ellas. El costo asociado a una arista representa el costo de seleccionar esa línea para la red. Un árbol abarcador de costo mínimo representa una red que comunica todas las ciudades en un costo minimal.
Existen dos técnicas populares para construir un árbol abarcador de costo mínimo a partir de un grafo ponderado G=(V, A); una de ellas se conoce como algoritmo de ***Prim***. Supóngase que V={1, 2, 3, ..., n}. El algoritmo de *Prim* comienza cuando se asigna a un conjunto U un valor inicial {1}, en el cual "crece" un árbol abarcador, arista por arista. En cada paso localiza la arista mas corta (u, v) que conecta U y V-U, y después agrega v, el vértice en V-U, a U. Este proceso se repite hasta U=V.

TRABAJO PRACTICO N°3
Tema: Grafos

Una forma sencilla de encontrar la arista de menor costo entre U y V-U en cada paso, es por medio de dos arreglos; uno, **masCercano[i]**, que da el vértice en U que esté más cercano a i en V-U. El otro, **menorCosto[i]**, que da el costo de la arista (i, masCercano[i]).
En cada paso se revisa menorCosto para encontrar algún vértice, como k, en V-U que esté más cercano a U. Se imprime la arista (k, masCercano[k]). Entonces se actualizan los arreglos menorCosto y masCercano, teniendo en cuenta el hecho de que k ha sido agregada a U.
Se supone que C es un arreglo de N x N tal que C[i, j] es el costo de la arista (i, j). Si la arista (i, j) no existe, se supone que C[i, j] tiene un valor grande apropiado.
Si se encuentra otro vértice k para el árbol abarcador, se hace que menorCosto[k] sea *infinito*, un valor muy grande, de modo que este vértice ya no se considera en los recorridos siguientes para incluirlo en U. El valor infinito es mayor que el costo de cualquier arista o que el costo asociado a una arista no existente.

El algoritmo es el siguiente:

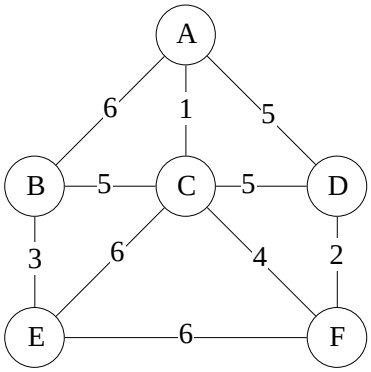
```
public void prim()
        {
        int[] menorCosto=new int[cantVert];
        int[] masCercano=new int[cantVert];
        int[][] matrizAdya=//asignarle la matriz de adyacencia
        int i, j, k, min;

        for(i=1; i<cantVert; i++)
                {
                menorCosto[i]=matrizAdya[0][i];
                masCercano[i]=0;
                }
        for(i=1; i<cantVert; i++)
                {
                min=menorCosto[1];
                k=1;
                for(j=2; j<cantVert; j++)
                        if(menorCosto[j] < min)
                                {
                                min=menorCosto[j];
                                k=j;
                                }
                //imprimir la arista de masCercano[k] a k
                System.out.println(...);
                menorCosto[k]=INFINITO;
                for(j=1; j<cantVert; j++)
                        if(matrizAdya[k][j] < menorCosto[j] & menorCosto[j] < INFINITO)
                                {
                                menorCosto[j]=matrizAdya[k][j];
                                masCercano[j]=k;
                                }
                }
        }
```

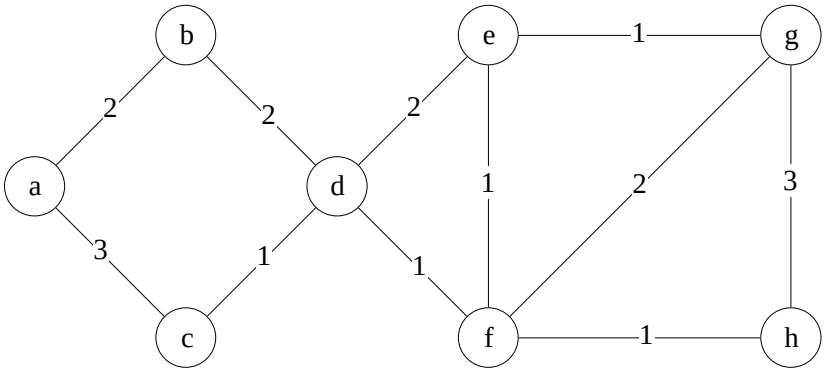
Encuentre un árbol abarcador de costo mínimo para los siguientes grafos:

TRABAJO PRACTICO N°3
Tema: Grafos

a.-



b.-

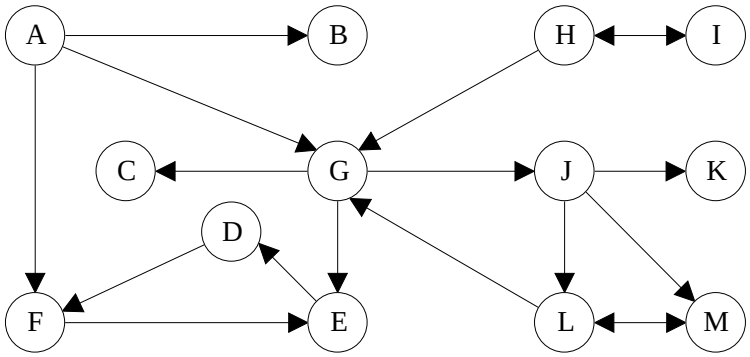


9. Se tiene un grafo dirigido que representa las rutas que conectan diferentes ciudades. Realizar un programa que reciba dicha estructura y determine si existe un camino entre dos ciudades dadas. El programa debe permitir hacer consultas, como los haría una aerolínea.

10. Una raíz de un grafo dirigido es un vértice **r** tal que todo vértice del grafo dirigido pueda alcanzarse por un camino dirigido desde r. Escribir un programa para determinar si un grafo dirigido posee raíz y si es así, cuál es el vértice.

11. En un grafo dirigido, un subgrafo $G_i=(V_i, A_i)$ es un componente *fuertemente conectado* si $\forall$ v, w $\in$ V_i existe un camino de v a w y de w a v en G_i.

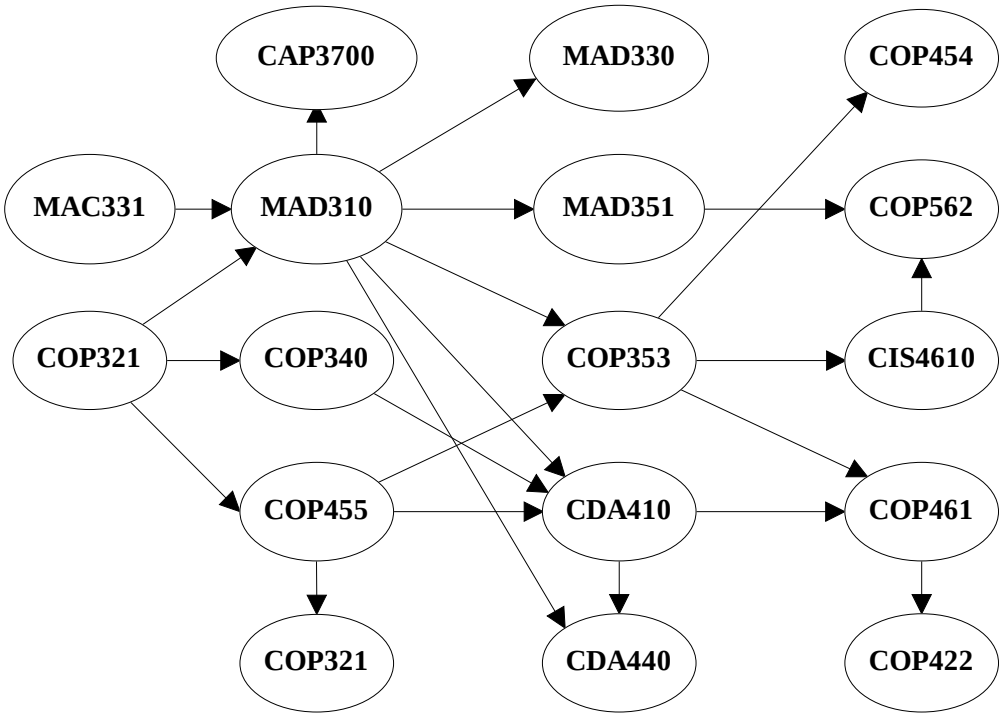
a) Encuentre los componentes fuertemente conectados de la siguiente figura:



b) Implemente el algoritmo para cualquier grafo dirigido.

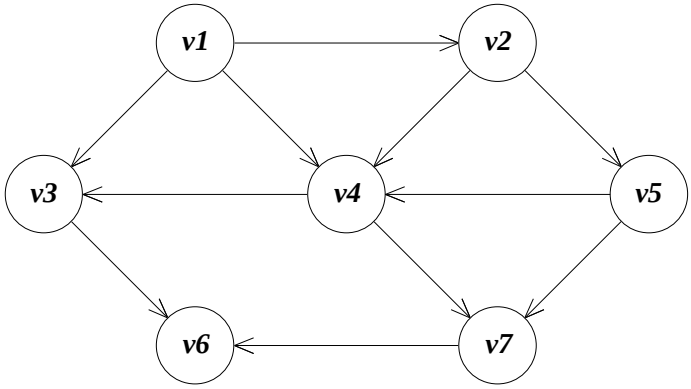
TRABAJO PRACTICO N°3
Tema: Grafos

14. Ordenación topológica.
La ordenación topológica es una ordenación de los vértices de un grafo dirigido acíclico, tal que si hay un camino de v_i a v_j, entonces v_j aparece después de v_i en la ordenación. En el grafo del cuadro 1 se representa la estructura de requisitos previos de los cursos en una universidad estatal de Miami. Una arista dirigida *(v, w)* indica que el curso *v* debe cubrirse antes de intentar cursar *w*. Una ordenación topológica de esos cursos es cualquier secuencia de cursos que no viole el requerimiento de requisitos.



Cuadro 1

Esta claro que una ordenación topológica no es posible si el grafo tiene un ciclo. Por otra parte, la ordenación topológica no es única; cualquier ordenación legal servirá. En el grafo del cuadro 2, tanto *v1, v2, v5, v4, v3, v7, v6* como v1, v2, v5, v4, v7, v3, v6 son ordenaciones topológicas.



Cuadro 2

Un algoritmo sencillo para encontrar una ordenación topológica consiste en encontrar primero cualquier vértice al que no entren aristas. Entonces podemos imprimir el vértice, y eliminarlo del grafo junto con sus aristas. Después se aplica esta misma estrategia al resto del grafo.

TRABAJO PRACTICO N°3
Tema: Grafos

Para formalizar esto, definimos **el grado de entrada** de un vértice *v* como el número de aristas *(u, v)*. Calculamos los grados de entradas de todos los vértices en el grafo. Suponiendo que en un arreglo *gradoEnt* se inicia y que el grafo esta en una lista de adyacencia, podemos aplicar el algoritmo para generar una ordenación topológica.

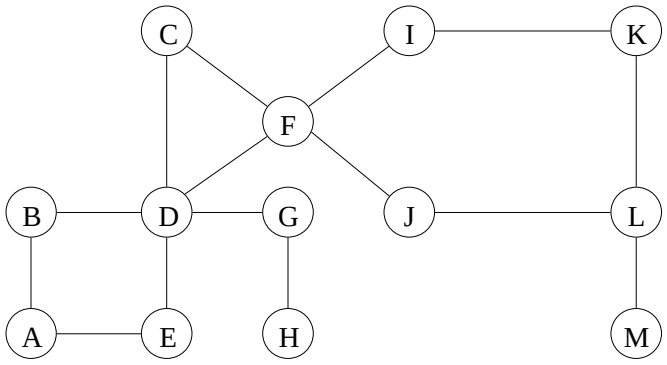
Se pide:
Escriba un programa que permita generar un grafo dirigido mediante listas de adyacencia y efectúe una ordenación topológica utilizando la estrategia descripta. (Puede suponer que el grafo se ingresará sin ciclos).

15. Con el algoritmo Dijkstra se calculan los caminos mínimos desde un vértice inicial a los demás vértices del grafo. Escribir las modificaciones necesarias para que teniendo como base el algoritmo de Dijkstra se calculen los caminos mínimos entre todos los pares de vértices. Analizar las diferencias con el algoritmo Floyd.

16. El algoritmo de Dijkstra no se puede aplicar a un grafo valorado en el que alguna arista tiene un factor de peso negativo. El algoritmo Bellman-Ford resuelve el problema cuando hay aristas negativas, aunque no lo resuelve cuando hay un ciclo en el grafo con peso negativo. En caso de haber un ciclo con peso negativo lo indica y termina.
Implementar el algoritmo, cuyo pseudocódigo se encuentra en el apunte de Grafos y probarlo exhaustivamente.

17. Se dice que un vértice de un grafo conexo es un *punto de articulación* del mismo si al suprimir ese vértice y todas las aristas que inciden en él, el grafo resultante deja de ser conexo. Por ejemplo, un grafo en forma de anillo NO TIENE puntos de articulación, mientras que todo nodo que no sea hoja de un árbol es punto de articulación.

a) Determinar los vértices que son puntos de articulación en el siguiente grafo:



b) Dado un grafo conexo no dirigido, *G=(V, A)* diseñar un algoritmo que indique qué vértices del grafo son puntos de articulación. Indicar qué implementación del grafo es la que proporciona un algoritmo más eficiente.

18. Un circuito de Euler en un grafo dirigido es un ciclo en el cual toda arista es visitada exactamente una vez. Se puede demostrar que un grafo dirigido tiene un circuito de Euler, sí y solo sí, es fuertemente conexo y todo vértice tienen iguales sus grados de entrada y de salida.

a) Dibujar un grafo en el que se puede encontrar un circuito de Euler.
b) Escribir un programa en el que se represente mediante lista de adyacencias un grafo. Implemente el algoritmo para encontrar, si existe, un circuito de Euler.