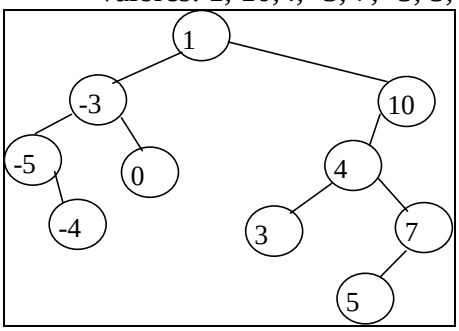
ELO320
Tercer Certamen 27/06/2001

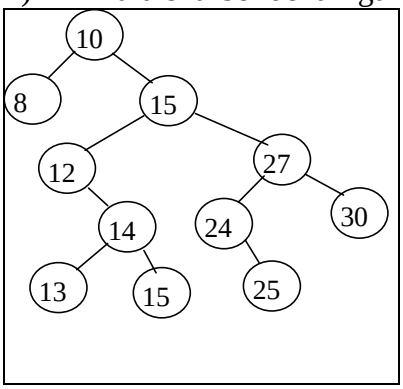
TODAS LAS PREGUNTAS VALEN 20 puntos => total 100 puntos
120 minutos. Las respuestas estarán publicadas en la página del curso al término del certamen.

1.- Considere las siguientes preguntas sobre árboles de búsqueda binaria.

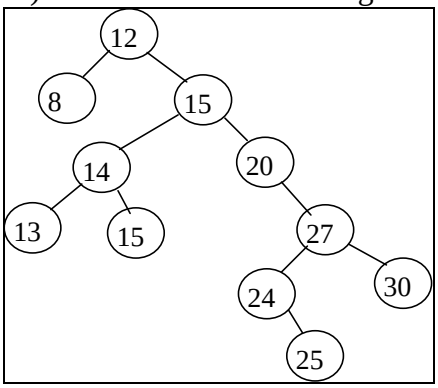
i) Partiendo de un árbol vacío muestre el árbol de búsqueda binaria resultante luego de insertar los valores: 1, 10,4, -3, 7, -5, 5, -4, 3, 0.



ii) Para el árbol de la figura 1, dibujar el árbol resultante al eliminar el nodo 20.



iii) Para el árbol de la figura 1, dibujar el árbol resultante al eliminar el nodo 10.

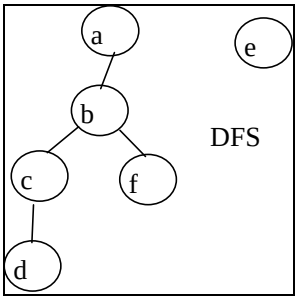


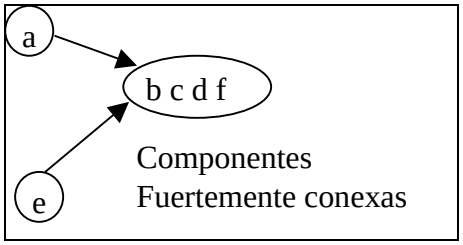
2.a.- Muestre los árboles (o foresta) obtenida al visitar los nodos del grafo de la figura 2 usando el algoritmo DFS (Depth-first search) y recorriendo los nodos y arcos en orden alfabético. Ignore el peso de cada arco.

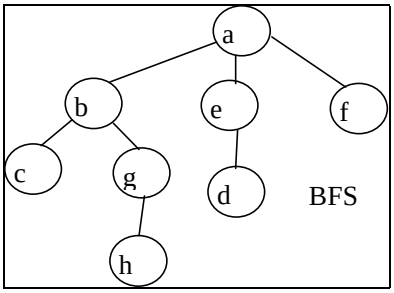
2.b.- Obtenga las componentes fuertemente conexas para el grafo de la figura 2. Si puede hacerlo directamente, hágalo. Ignore los pesos de cada arco.

2.c.- Muestre el árbol obtenido al visitar los nodos del grafo de la figura 3 usando el algoritmo BFS (Breadth-first search) y comenzando en ***a***. Ignore los pesos de cada arco.

ELO320
Tercer Certamen 27/06/2001





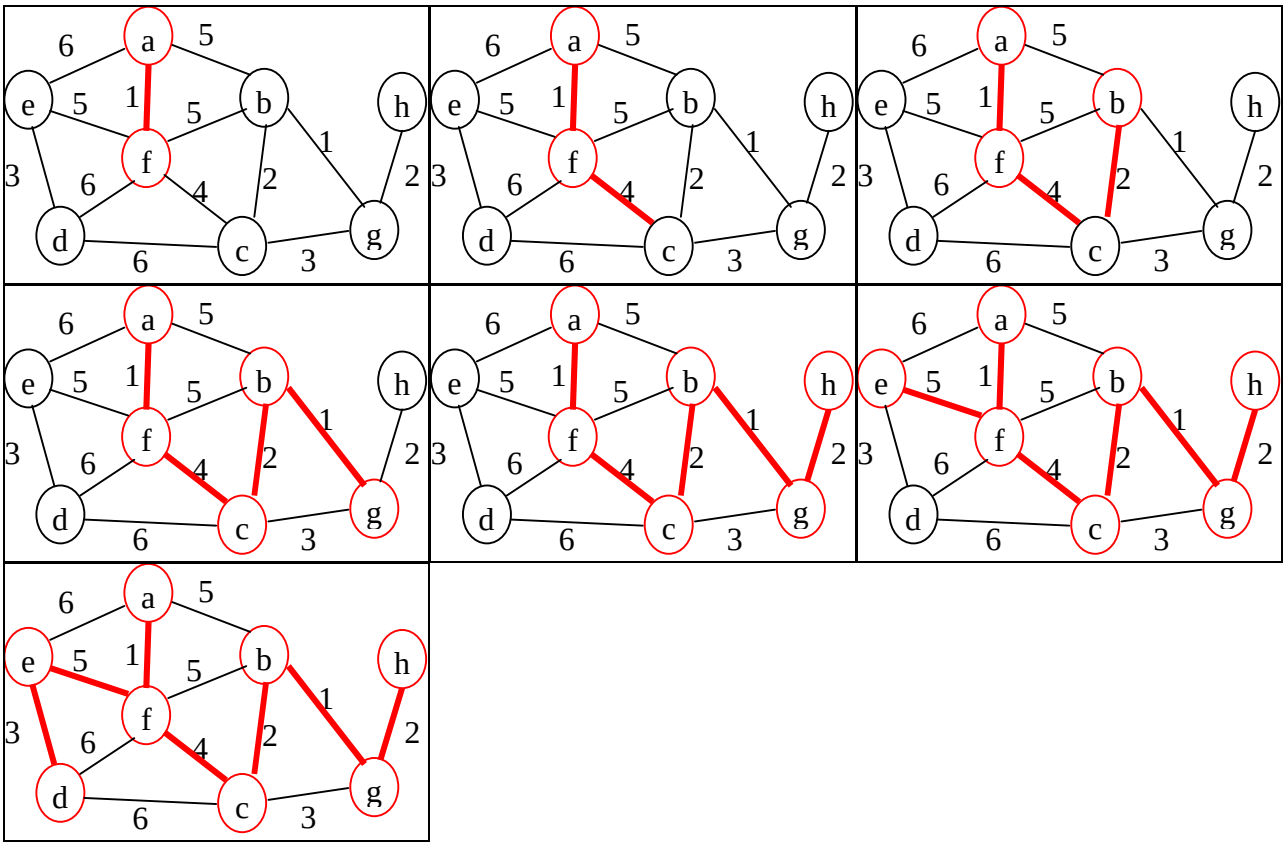


3.- Si los nodos y arcos del grafo de la figura 3 son considerados en orden alfabético, liste la secuencia de arcos a medida que son incorporados al árbol de expansión de peso mínimo (minimum spanning tree) cuando se usa:

i) Algoritmo de Prim

ii) Algoritmo de Kruskal

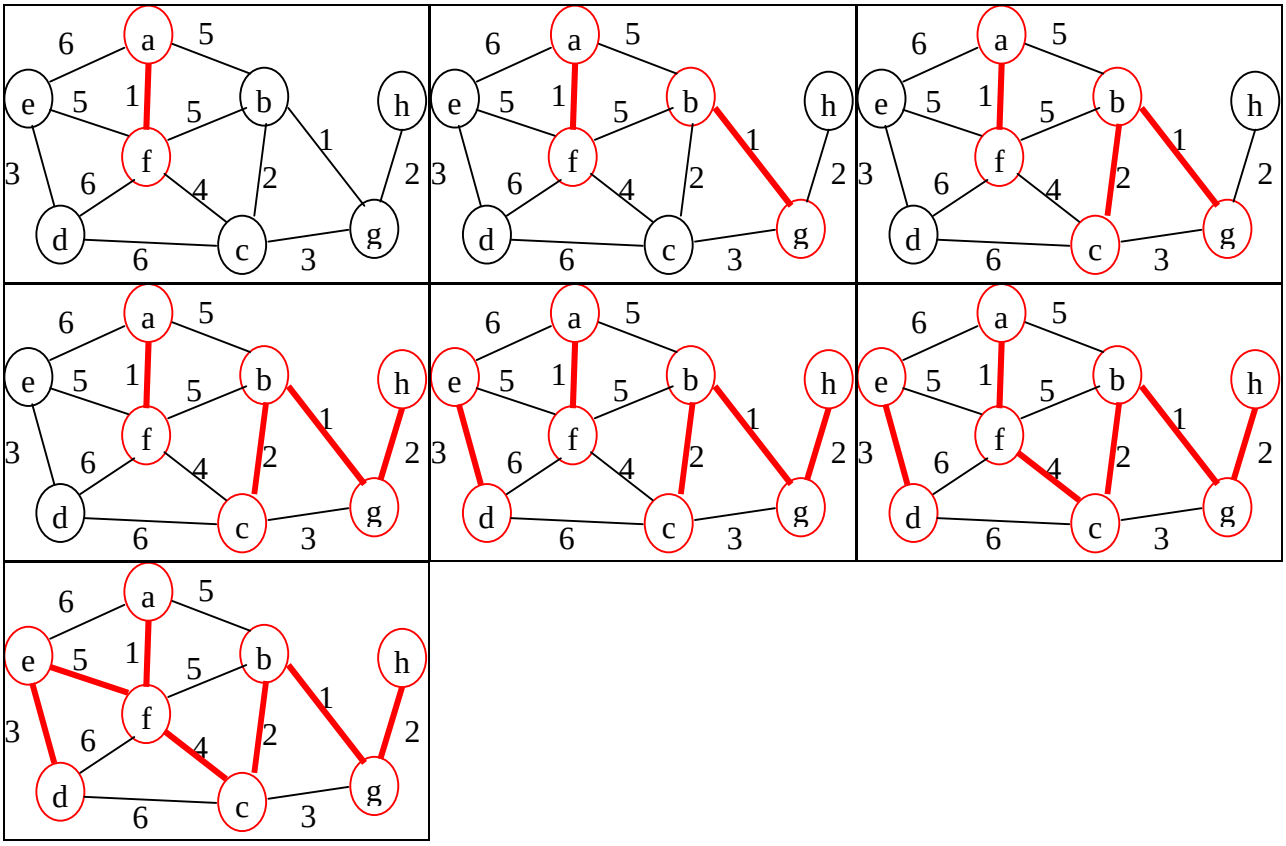
Prim:



Luego el orden de los arcos es: (a,f), (f,c), (c,b), (b,g), (g,h), (f,e), (e,d) y el peso mínimo es: 18

Kruskal:

ELO320
Tercer Certamen 27/06/2001



Luego el orden de los arcos es: (a,f), (b,g), (b,c), (g,h), (e,d), (f,c), (f,e) y el peso mínimo es: 18

4.- Use el algoritmo de Dijkstra sobre el grafo de la figura 2 partiendo del nodo ***a***. Se pide:

i) Mostrar los valores de los arreglos **d** y **p** luego que son relajados los arcos del nodo ***c*** en el algoritmo. Suponga que los vértices están en orden alfabético en los índices del arreglo.

ii) El valor final para el arreglo **d** y arreglo **p**.

i)d: <0, 3, 4, 4, infinito, 5>
p: <NIL, a, b, a, NIL, a>

ii) d: <0, 3, 4, 4, infinito, 5>
p: <NIL, a, b, a, NIL, a>

5.- Sea G=(V,E) un grafo en el que cada arco ***(u,v)*** de E tiene asociado un valor ***r(u,v)***, el cual es un número real en el rango $0 \leq r(u,v) \leq 1$ que representa la seguridad de un canal de comunicación desde el vértice ***u*** al vértice ***v***. Interpretamos ***r(u,v)*** como la probabilidad que el canal desde ***u*** a ***v*** no será monitoreado por el bando contrario y suponemos que estas probabilidades son independientes. Proponga un algoritmo eficiente para encontrar el camino más seguro entre dos vértices ***a*** y ***b*** dados.

Obs: Si X e Y son dos eventos independientes,
Probabilidad{X e Y ocurren} = Probabilidad{ocurra X}*Probabilidad{ocurra Y}

Este problema se puede resolver adaptando el algoritmo de Dijkstra. En este caso el arreglo d, en lugar de distancia, representa la seguridad del canal desde la fuente hasta el nodo en cuestión y el arreglo p indica la ruta a seguir. La adaptación básicamente modifica la función de relajación y detiene el algoritmo una vez que se incorpora el nodo destino al arreglo S.

ELO320
Tercer Certamen 27/06/2001

```
RelaxExam (u,v, w){
        if (d[v] < d[u] * w(u,v) ){
                d[v] = d[u] * w(u,v);
                p[v] = u;
        }
}

Dijkstra_Exam(G, w, a, b) {
        for (cada vértice v en V[G] ) {
                d [v] = 0;
                p [v] = NIL;
        }
        d [a] = 1;
        S = {};   /* S Contiene el arreglo de los nodos cuyo camino más corto ya ha sido encontrado */
        Q = V [G];
        while (Q != {}) {
                u = Extract_Max(Q);
                if (u == b)
                        return; /* El trabajo de aquí en adelante no cambia la situación para b*/
                S = S ∪ {u};
                for ( cada vértice v en adj[u] )
                        Relax(u,v, w);
        }
}
```