Tiempo: 110 Minutos (de 10:00 a 11:50), todas las preguntas tiene igual puntaje.

1. Sea $h(k,i) = (h_1(k) + i*h_2(k)) \bmod 13$; con $h_1 = k \bmod 13$ y $h_2 = 1 + (k \bmod 11)$.
a) ¿Cuantas entradas tiene la tabla hash?
b) Determine las entradas de la tabla en las que se ubicarán las siguientes claves cuando son ingresadas en orden.
17, 72, 25, 90, 20. Incluya su desarrollo.
c) Si ahora eliminamos el 17, e insertamos 59, ¿En qué entrada queda el 59? ¿Hay algún cambio para alguno de los otros datos?

a)13, van del 0 al 12.
b)

Clave	h_1	h_2	h(k,0)	h(k,1)	h(k,2)	Ubicación en tabla
17	4		4			4
72	7		7			7
25	12		12			12
90	12	3	12	2		2
20	7	10	7	4	1	1

c) Se remueve 17, la entrada se marca borrada.

Clave	h_1	h_2	h(k,0)	h(k,1)	h(k,2)	Ubicación en tabla
17	4		4			4 (Borrada)
72	7		7			7
25	12		12			12
90	12	3	2			2
20	7	10	4	1		1
59	7	5	7	12	4	4

2.- Se tienen los siguientes socios que están solicitando el uso de una cancha de tenis por los bloques indicados abajo. El administrador se le ha pedido que las asigne de forma que el mayor número de socios pueda jugar.

Socio	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13
Inicio	7	6	5	2	2	3	3	11	7	6	9	1	9
Término	10	9	7	5	2	4	6	12	8	7	9	3	12

Solicitud de bloques inclusive.
a) ¿Qué socios terminan jugando?
b) Suponga ahora que se dispone un varias canchas C1, C2,.. Cn y el administrador debe hacer una asignación tal que se use el mínimo número de canchas para que todos los socios puedan jugar. ¿Donde queda jugando cada socio?

Primero ordenamos las solicitudes por orden de término:

Socio	S5	S12	S6	S4	S7	S3	S10	S9	S2	S11	S1	S8	S13
Inicio	2	1	3	2	3	5	6	7	6	9	7	11	9
Término	2	3	4	5	6	7	7	8	9	9	10	12	12

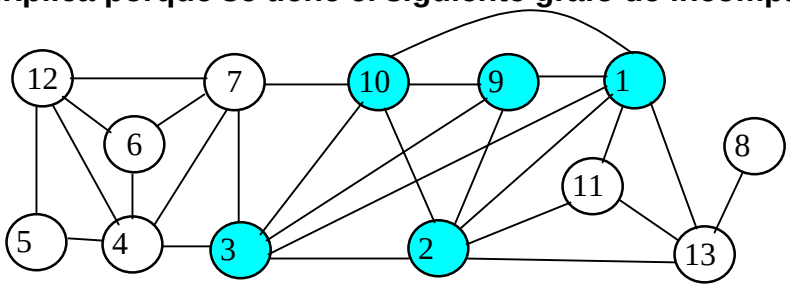
Luego se procede a la asignación:

Socio	S5	S12	S6	S4	S7	S3	S10	S9	S2	S11	S1	S8	S13
Inicio	2	1	3	2	3	5	6	7	6	9	7	11	9
Término	2	3	4	5	6	7	7	8	9	9	10	12	12
Juega?	si	no	si	no	no	si	no	no	no	si	no	si	no

b) Ya asignamos una cancha ahora tomamos los socios que quedan para asignar la otra cancha. Una asignación posible es:

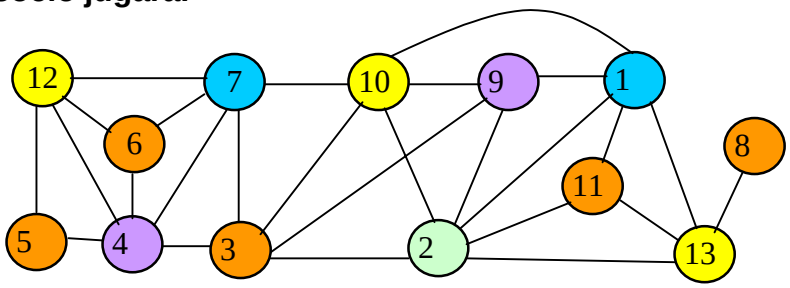
Socio	S5	S12	S6	S4	S7	S3	S10	S9	S2	S11	S1	S8	S13
Inicio	2	1	3	2	3	5	6	7	6	9	7	11	9
Término	2	3	4	5	6	7	7	8	9	9	10	12	12
Cancha1	si		si			si				si		si	
Cancha2		si					si						si
Cancha3				si				si					
Cancha4					si						si		
Cancha5									si				

Se requieren 5 canchas.
Esto también se explica porque se tiene el siguiente grafo de incompatibilidad:



Se puede identificar que hay varias pandillas de nodos incompatibles. Como el tamaño de la pandilla mayor es cinco, el problema no puede ser resuelto con menos de cinco canchas.

En el contexto de grafos este problema es conocido como Coloración del grafo de intervalos. No se trata de cualquier grafo sino de aquel derivado de un grupo de actividades incompatibles. En este caso se busca colorear el grafo con el mínimo número de colores sin coincidencia de colores adyacentes. El color corresponde a la cancha donde el socio jugará.



Obviamente esta no es la única asignación posible.

3.- Proponga un algoritmo que determine si un grafo no dirigido G=(V,E) contiene o no ciclos.

Este problema puede ser resuelto considerando una variante de la búsqueda en profundidad. Si en el proceso de búsqueda llegamos a un nodo ya visitado entonces hemos encontrado un ciclo. Luego el algoritmo queda:

```
bool DFS_TieneCiclos(G) {   /* pseudo-código */
        for ( cada vértice u ∈ V[G]) {
                color [u] =Blanco;
                p[u] = NULL;
        }
        tiempo = 0;
        for (cada vértice  u ∈ V[G])
                if (color[u] == Blanco)
                        if (DFS_visit(u)) return true;
        return false;
}
bool DFS_visit_Ciclo (u)  /* pseudo-código */
        color [u]= Plomo;  /* Vértice Blanco u es visitado, ingresamos a su sub-árbol */
        d[u] = ++tiempo; /* el tiempo avanza  cada vez que "entramos o salimos" de un nodo*/
        for ( cada v ∈ Adj [u] ) {  /* explora arcos (u,v) */
                if (color [v] == Blanco) {
                        p [v] = u;
                        if (DFS_visit_Ciclo(v)) return true;
                } else return true;    /* ENCONTRÉ EL CICLO*/
        }
        color [u] = Negro;   /* ennegrezca u, salimos de su sub-árbol */
        f [u] = ++tiempo;
        return false;
}
```

Otras soluciones:
Idea1:
Buscar las componentes fuertemente conexas.
Si alguna componente fuertemente conexa contiene más de un nodo, hay un ciclo.
Idea 2:
Asignar peso -1 a cada arco y aplicar Bellman-Ford. Si no hay convergencia, hay ciclos. Habría que aplicar Bellman-ford desde un nodo para cada componente conexa del grafo.

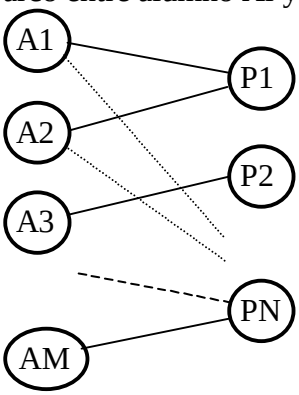
4.- Considere un grafo G=(V,E) **sin ciclos** donde los pesos asociados a los arcos corresponden a personas en la ruta esperando por el autobus para llegar al novo ***v***. Suponiendo que el bus tiene capacidad infinita; dados **G**, ***u*** y ***v***, proponga un algoritmo que partiendo del nodo ***u***, llegue a v con la máxima cantidad de pasajeros.

Para que el problema tenga sentido, el grafo debe ser dirigido. De otra forma, significa que hay un sólo camino desde u a v y por ende nada que optimizar.
El destino es único, por lo tanto estamos en un caso tipo Dijkstra

```
Dijkstra(G, w, s, v) {
        for (cada vértice v en V[G] ) {
                d [v] = -1
                p [v] = NIL;
        }
        d [s] = 0;
        S = {};
        Q = V [G];
        while (Q != {}) {
                u = Extract_Max(Q);
                if (u==v) return;  /* d[v] contiene la máxima cantidad de pasajeros */
                S = S U {u};
                for ( cada vértice v en adj[u] )
                        if (d[v] < d[u] + w(u,v) ){  /*si esta ruta tiene más personas, tómela*/
                                d[v] = d[u] + w(u,v);
                                p[v] = u;
                        }
        }
}
```

Con este algoritmo el problema se resuelve invocando Dijkstra(G, w, u, v); luego la ruta a seguir será aquella indicada por p[i] (en sentido inverso).

5.- Suponga usted que la universidad dispone regalar entradas al cine a un grupo de estudiantes. Los estudiantes seleccionan las películas de su gusto y se presenta el siguiente grafo bipartito donde hay un arco entre alumno Ai y la película Pj si Ai quiere ver Pj.



Al lado izquierdo tenemos a M estudiantes y en el lado derecho N películas. M>N.

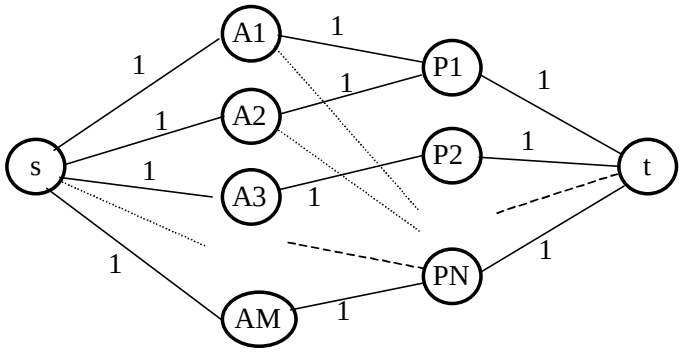
Consideraciones: A cada estudiante se le regala a lo más una entrada.

a) Indique los pasos a seguir y algoritmo a utilizar para lograr asignar la mayor cantidad de entradas según el gusto de los estudiantes. **En este caso la Universidad regala una entrada por película.**

b) Suponga ahora que la Universidad decide (~~entregar~~) **regalar** hasta dos entradas para cada función de modo que más estudiantes obtengan este beneficio. ¿Cuáles serían los pasos que ahora usted seguiría para resolver el problema?

Nota: Usted puede asumir que dispone de versiones ya programadas para todos los algoritmos vistos en clases. No los re-escriba. Puede usar pseudo-lenguaje.

a) Lo que corresponde es llevar el grafo a una situación donde podamos aplicar redes de flujo máximo. Para ello se incorporan dos nodos s y t, además, asociamos peso 1 a cada arco.



El segundo paso es resolver la red de flujo máxima de s a t y obtendremos la asociación bipartita máxima.

b) Si ahora hay dos entradas por película, ocurre que los recursos de la derecha se han duplicado.
Como antes tomamos grafo de gustos.
Luego agregamos los nodos s y t. Asociamos peso 1 a cada arco, excepto a aquellos de Pi a t, los cuales tienen peso 2 por haber dos entrada disponibles por película.
Resolvemos la red de flujo máxima y obtenemos la asignación máxima de entradas.