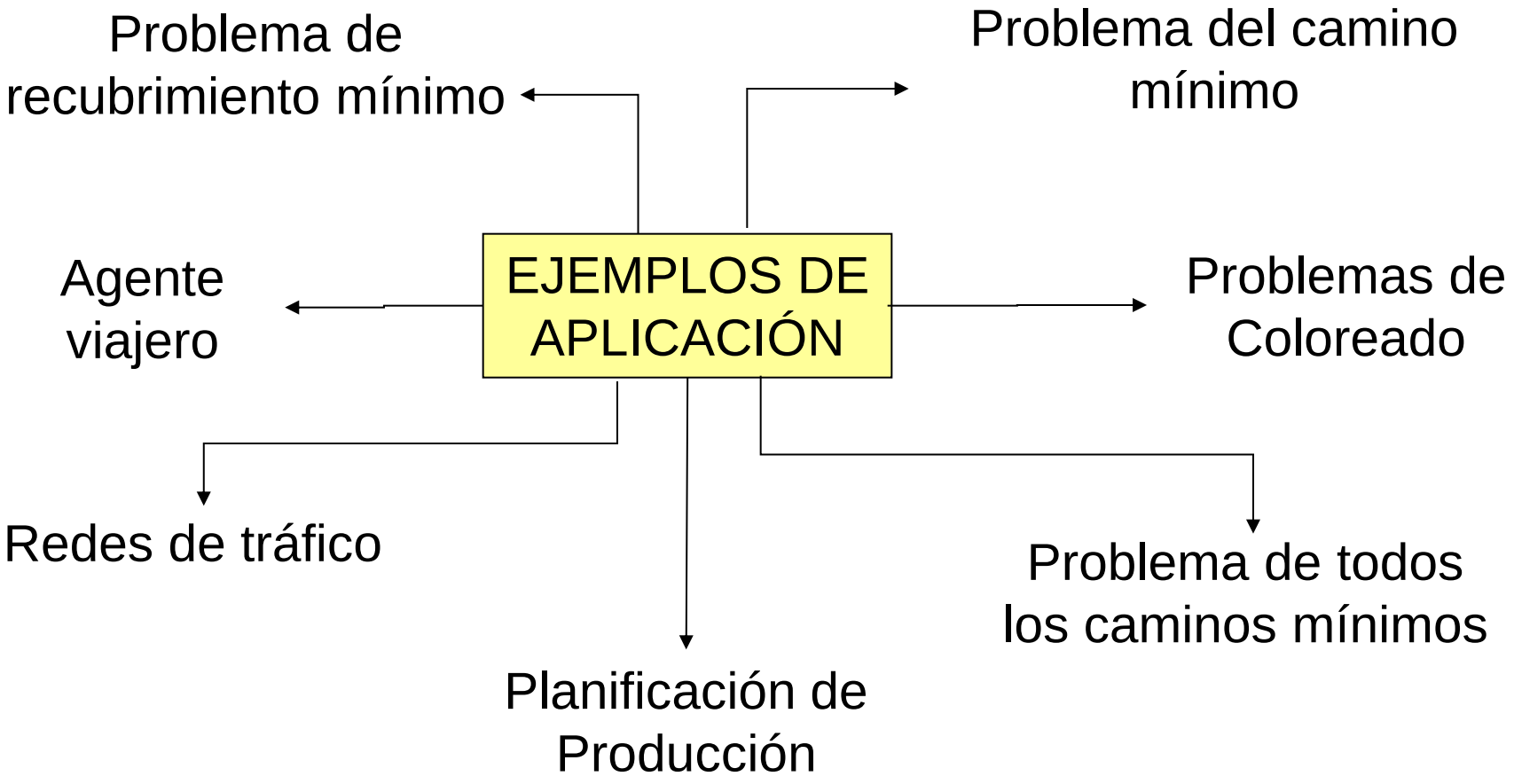
TPN°10: Grafos

Algoritmos y Estructuras de Datos II

GRAFOS

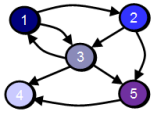
GRAFOS


Problema de recubrimiento mínimo
Problema del camino mínimo
Agente viajero
EJEMPLOS DE APLICACIÓN
Problemas de Coloreado
Redes de tráfico
Problema de todos los caminos mínimos
Planificación de Producción

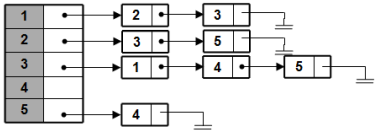
GRAFOS

GRAFOS → Colección de vértices unidos por un conjunto de arcos

Representación
- **Gráfica**



- **Lista de Adyacencia**



- **Matriz de Adyacencia**

	1	2	3	4	5
1	0	1	1	0	0
2	0	0	1	0	1
3	1	0	0	1	1
4	0	0	0	0	0
5	0	0	0	1	0

APLICACIONES
- **Árbol de Recubrimiento Mínimo**
  - **PRIM**
  - **KRUSKAL**
- **Problema Camino Mínimo**
  - **DIJKSTRA**
  - **BELLMAN-FORD**
  - **FLOYD**

GRAFOS

Algoritmo de Bellman-Ford

Problema del Camino Mínimo en un

grafo con ciclos no negativos

El ***problema de camino mínimo*** entre dos nodos *u* y *v* consiste en encontrar un camino entre esos nodos cuyo costo sea menor o igual que el costo de cualquier otro camino entre u y v.

GRAFOS

Algoritmo de Bellman-Ford

Algoritmo Bellman-Ford (G,A,n,D,C): grafo x matriz x ent → vector x bool

ENTRADA: G: grafo G(V,E)
n: número de vértices.
A: matriz de adyacencia con los costos.

SALIDA: D: vector de distancias especiales.
C: bool, true: indica que no existen ciclos de costo negativo

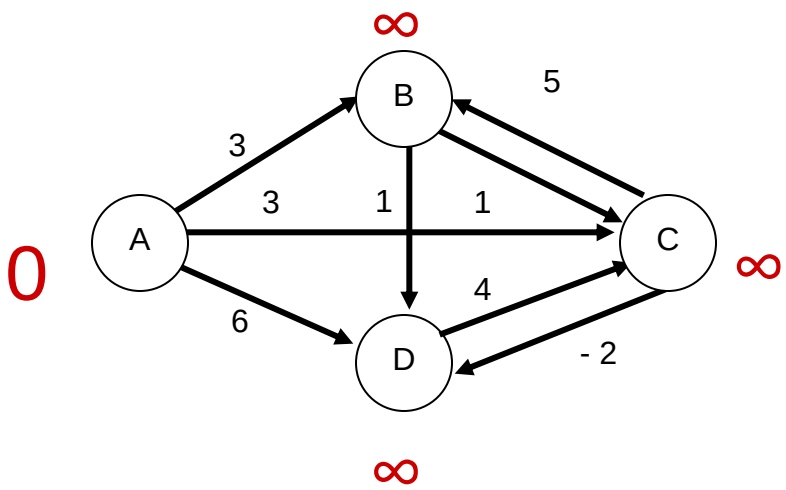
```
D(1) ← 0          // origen vértice 1
C ← True
Para i desde 2 hasta n hacer
    D( i ) ← ∞
Para i desde 1  hasta n-1 hacer
        Para cada arista (u,v) en E hacer      // relajar
                Si  D (v)  >  (D ( u ) + A ( u , v )) entonces
                        D (v) ←  D ( u ) + A ( u , v )
Para cada arista (u,v) en E hacer              // detectar ciclo de costo negativo
        Si  D (v)  >  (D ( u ) + A ( u , v )) entonces
                        C ←  False             // hay ciclo de costo negativo
Fin.
```

GRAFOS

Algoritmo de Bellman-Ford

1	2	3	4	5	6	7	8
(A,B)	(D,C)	(C,B)	(B,C)	(C,D)	(A,D)	(B,D)	(A,C)

ITERACIÓN 0



It_0 **D = {0, ∞, ∞ , ∞}**

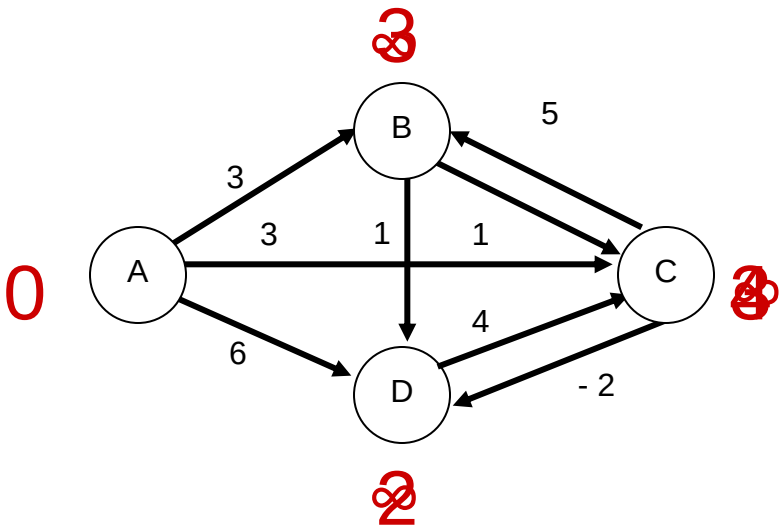
Para cada arista (u,v) en E hacer // relajar
Si D (v) > (D (u) + A (u , v)) entonces
D (v) ← D (u) + A (u , v)

GRAFOS

Algoritmo de Bellman-Ford

1	2	3	4	5	6	7	8
(A,B)	(D,C)	(C,B)	(B,C)	(C,D)	(A,D)	(B,D)	(A,C)

ITERACIÓN 1



It_0 **D = {0, ∞, ∞ , ∞}**

It_1 **D = {0, 3, 3 , 2}**

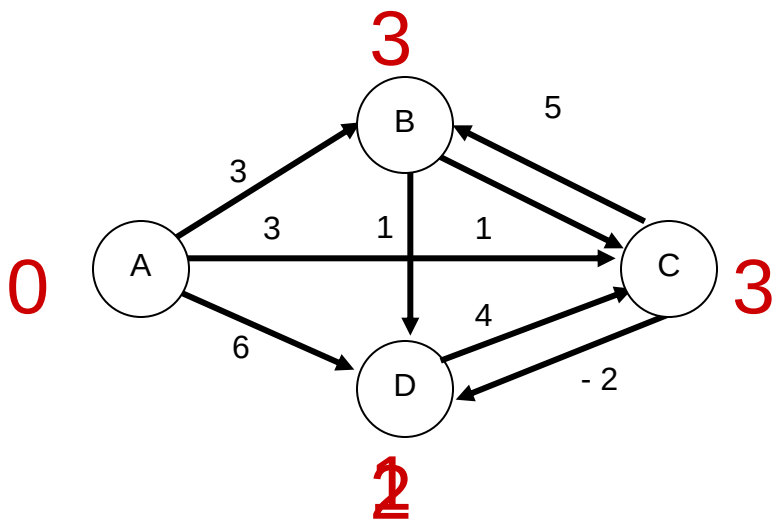
Para cada arista (u,v) en E hacer // relajar
Si D (v) > (D (u) + A (u , v)) entonces
D (v) ← D (u) + A (u , v)

GRAFOS

Algoritmo de Bellman-Ford

1	2	3	4	5	6	7	8
(A,B)	(D,C)	(C,B)	(B,C)	(C,D)	(A,D)	(B,D)	(A,C)

ITERACIÓN 2



It_0 **D = {0, ∞, ∞ , ∞}**

It_1 **D = {0, 3, 3 , 2}**

It_2 **D = {0, 3, 3 , 1}**

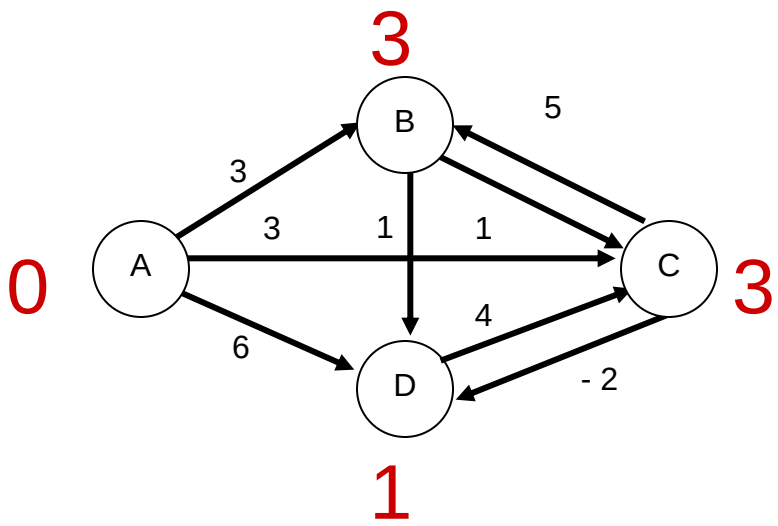
Para cada arista (u,v) en E hacer // relajar
Si D (v) > (D (u) + A (u , v)) entonces
D (v) ← D (u) + A (u , v)

GRAFOS

Algoritmo de Bellman-Ford

ITERACIÓN 3

1	2	3	4	5	6	7	8
(A,B)	(D,C)	(C,B)	(B,C)	(C,D)	(A,D)	(B,D)	(A,C)



It_0 **D = {0, ∞, ∞ , ∞}**

It_1 **D = {0, 3, 3 , 2}**

It_2 **D = {0, 3, 3 , 1}**

It_3 **D = {0, 3, 3 , 1}**

Para cada arista (u,v) en E hacer // relajar
Si D (v) > (D (u) + A (u , v)) entonces
D (v) ← D (u) + A (u , v)

COMPROBAMOS QUE NO TIENE CICLOS

CLIQUE MÁXIMA

CLIQUE

Subgrafo completo

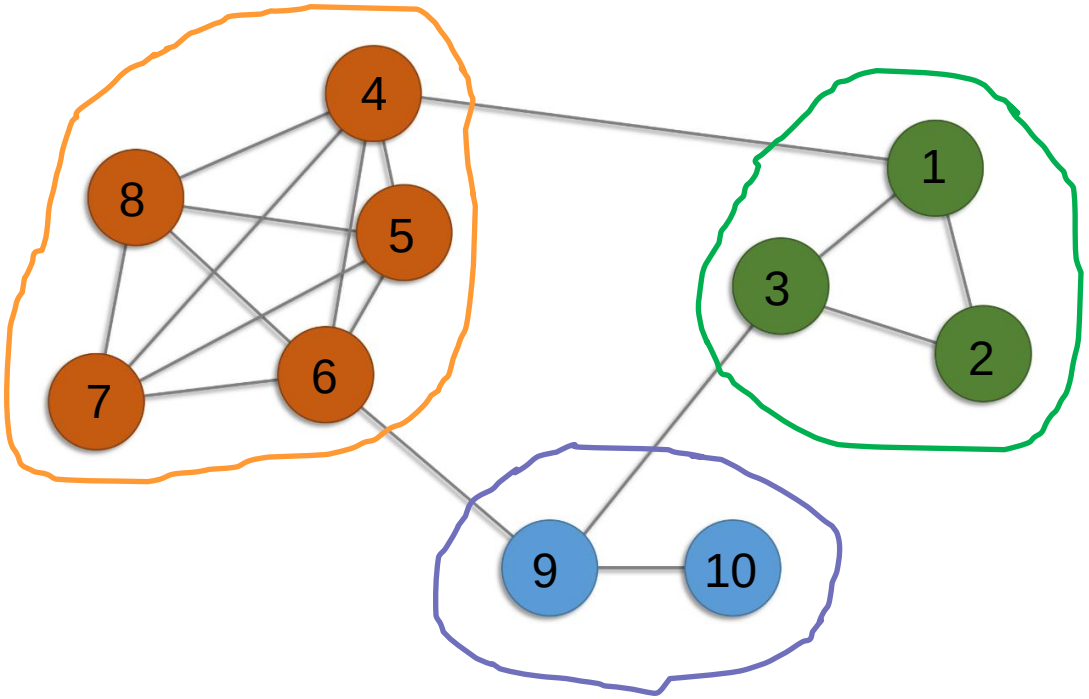
de un grafo dado

GRAFO COMPLETO

Todos los vértices del grafo son adyacentes entre sí, es decir hay una arista entre dos vértices cualesquiera del grafo

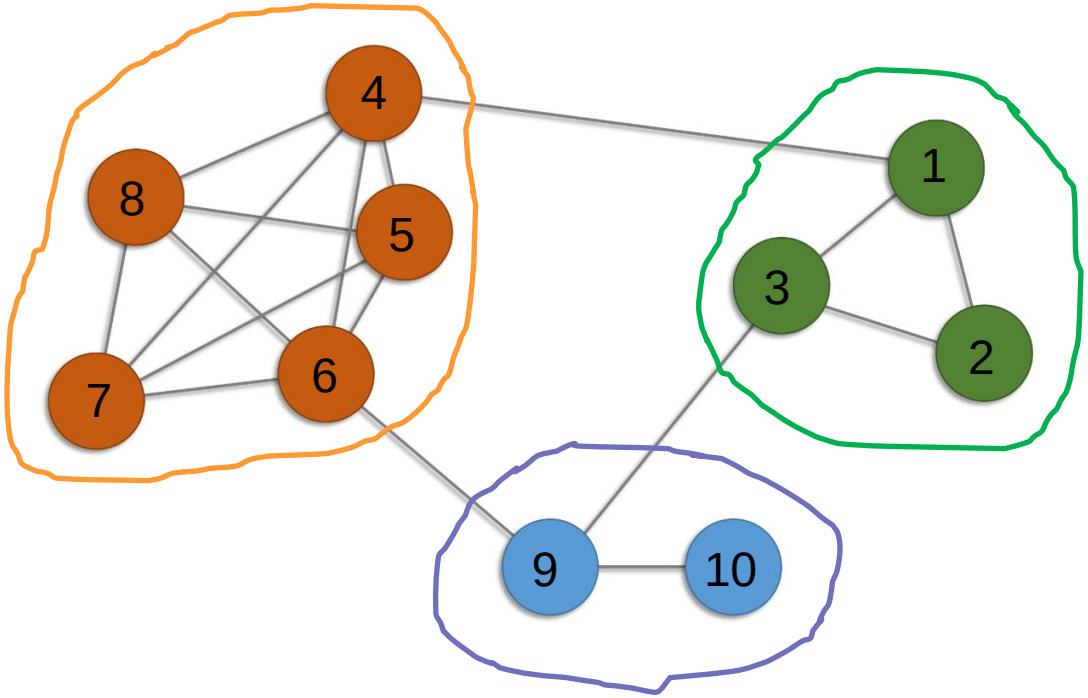
CLIQUE MÁXIMA

Clique con el máximo número de vértices



CLIQUE MÁXIMA

Diseñe un algoritmo, aplicando la técnica de **Backtracking**, que reciba un grafo G y retorne el tamaño que posee la clique máxima que se puede conseguir en ese grafo.



CLIQUE MÁXIMA

CM = {4,5,6,7,8}

Vértices = 5

CLIQUE MÁXIMA

PISTA

Modificar el algoritmo de Backtracking del problema de la Mochila 0/1

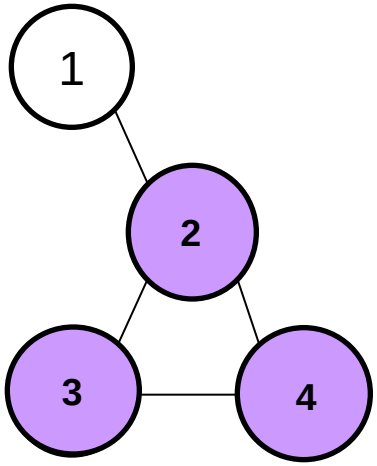
```
Función mochila (i, M): tipo x peso → beneficio
// globales: n, b y p
// entrada: elementos de tipos i a n y con peso máximo M.
// salida: bmax el beneficio de la mejor carga.

        bmax←0
        Para k=i hasta n hacer
                si p(k) ≤ M entonces
                        bmax← max (bmax, b(k)+mochila(k+1, M-p(k))
        retorna bmax
```

CLIQUE MÁXIMA

Ejemplo:

n=4



CLIQUE MÁXIMA
CM = {2, 3, 4}
 # Vértices = 3

CLIQUE MÁXIMA

PISTA

Modificar el algoritmo de Backtracking del problema de la Mochila 0/1

Vector que contiene los vértices agregados al clique

Cantidad de vértices agregados al vector S

```
FUNCIÓN maxClique(i, S, b): ent≥0 x Vector x ent≥0 → ent≥0
Vbles. Globales:
- n: cantidad de vértices del grafo
- A: matriz de  adyacencia del grafo

. . .
```

Invocación
`maxClique(1, S, 0)`

CLIQUE MÁXIMA

CLIQUE

Todos los vértices del clique son adyacentes entre sí

Vector que contiene los vértices agregados al clique

Cantidad de vértices agregados al vector S

```
FUNCION esClique(S, b): Vector x ent≥0 → bool
Vbles. Globales:
- n: cantidad de vértices del grafo
- A: matriz de  adyacencia del grafo

  PARA i=1,..b HACER
    u = S[i]
    PARA j=i+1,..b HACER
        v = S[j]
        SI A(u, v) = FALSE ENTONCES // u y v no son adyacentes
                RETORNA FALSE
  RETORNA TRUE
```

Preguntas