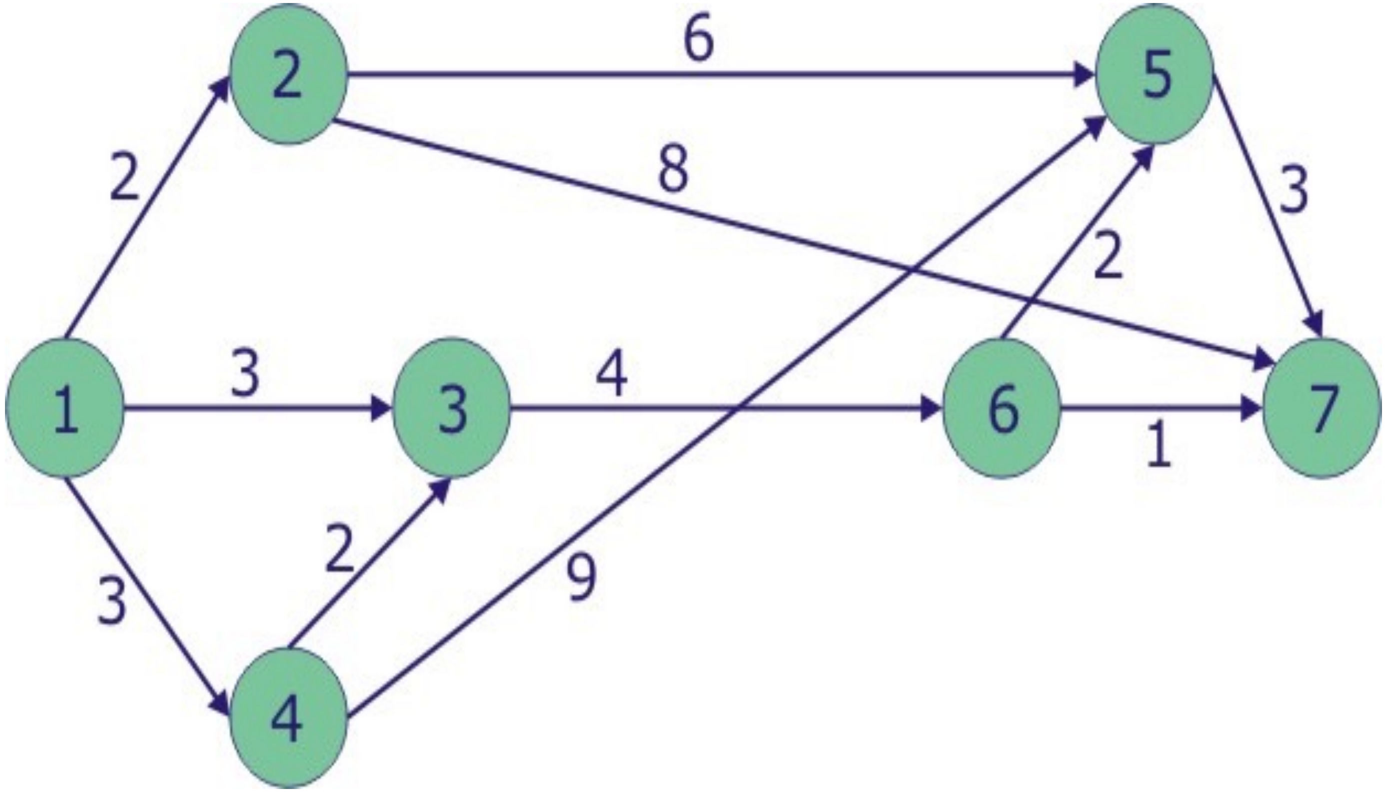
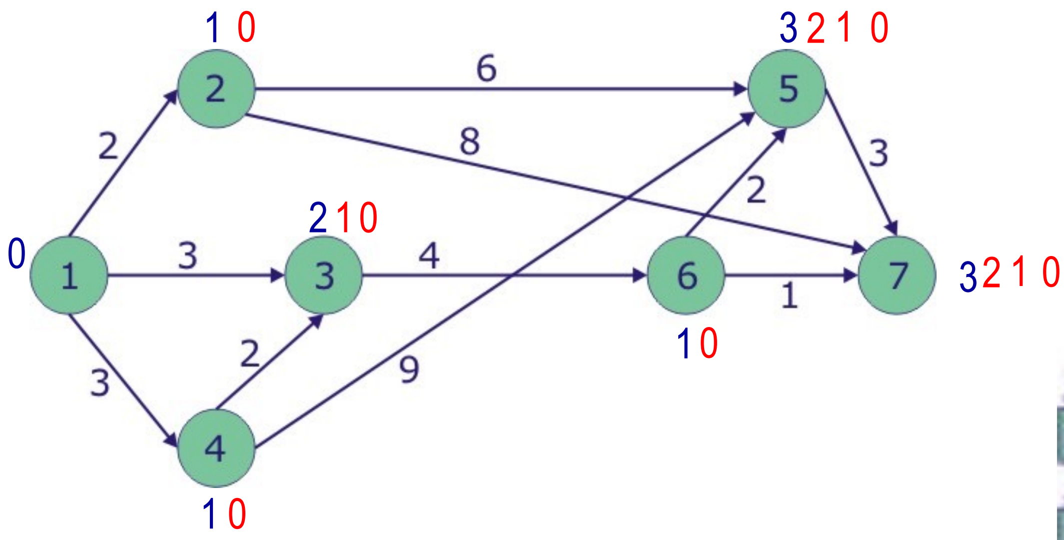
Ejercicios

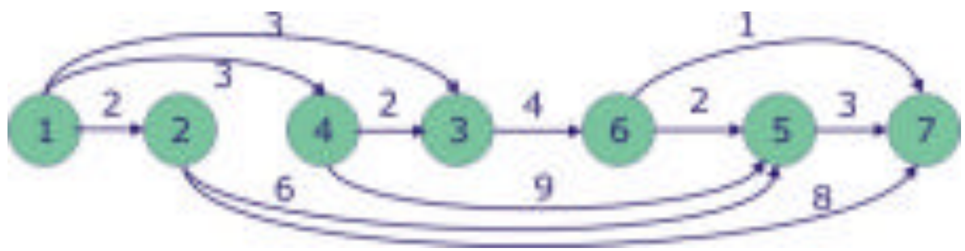
1. Dada la siguiente red, encuentra una ordenación topológica, explicando el proceso paso a paso.



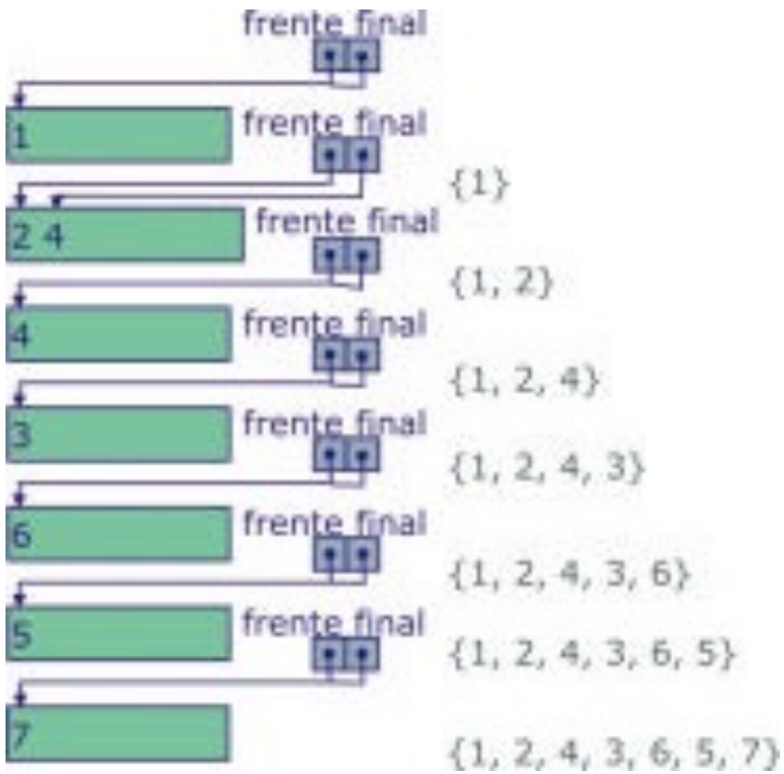
Ejercicios



1 0
3 2 1 0
6
8
2
3
2 1 0
2
0
3
4
3
2
9
1
3 2 1 0
1 0
1 0
1
2
3
4
5
6
7



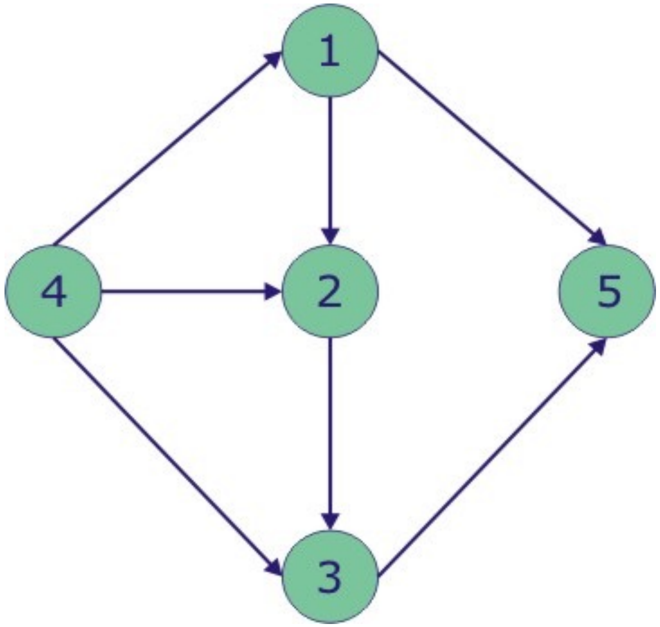
3
3
1
2
2
4
2
3
6
9
8
1
2
4
3
6
5
7



frente final
1
frente final
{1}
2 4
frente final
{1, 2}
4
frente final
{1, 2, 4}
3
frente final
{1, 2, 4, 3}
6
frente final
{1, 2, 4, 3, 6}
5
frente final
{1, 2, 4, 3, 6, 5}
7
{1, 2, 4, 3, 6, 5, 7}

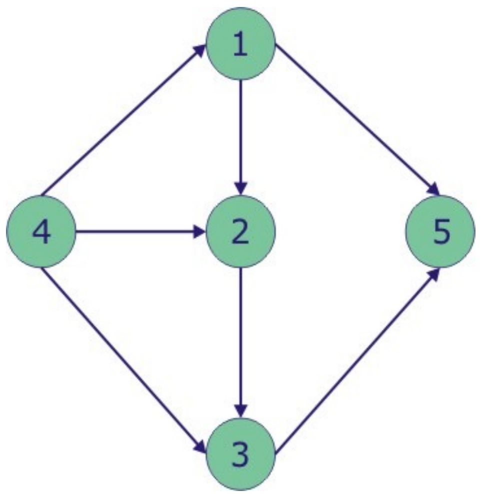
Ejercicios

2. Dado el siguiente grafo dirigido, escribe la matriz de caminos usando el algoritmo de Warshall.



Ejercicios

```
%Programa que calcula la matriz de caminos de un grafo usando el algoritmo
%de Warshall
clear all
n=5; %n es el número de vértices

%A es la matriz de adyacencia
A=[ 0 1 0 0 1;
    0 0 1 0 0;
    0 0 0 0 1;
    1 1 1 0 0;
    0 0 0 0 0];

%P0
P=A;
fprintf('Paso 0\n')
disp(P)

for k=1:n %k es el número de P's: de P1 a Pn
    %Recorro por filas y columnas calculando P(i,j)
    fprintf('Paso %d\n',k)
    for i=1:n
        for j=1:n
            if ~P(i,j) && P(i,k) && P(k,j)
                fprintf('P(%d,%d) Y P(%d,%d)\n',i,k,k,j)
            end
            P(i,j)=P(i,j) | (P(i,k) & P(k,j));
        end
    end
    disp(P)
end
```



Sólo imprimimos las novedades: Si P(i,j)=0 y existe camino de i a j a través de k

Si ya había camino de i a j O si existe camino de i a j a través de k

Ejercicios

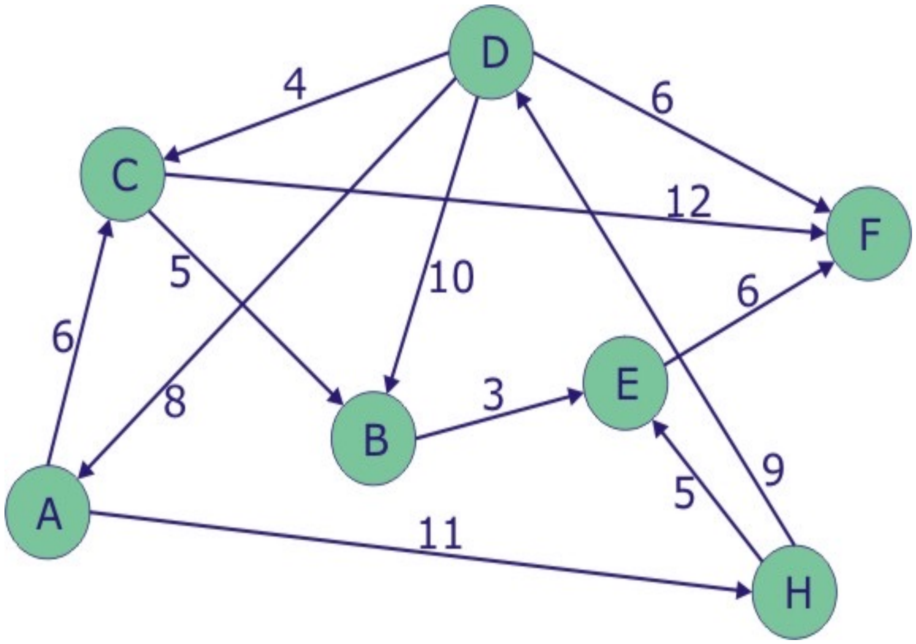
```
>> ejer2warshall
Paso 0
     0     1     0     0     1
     0     0     1     0     0
     0     0     0     0     1
     1     1     1     0     0
     0     0     0     0     0

Paso 1
   P(4,1) Y P(1,5)
     0     1     0     0     1
     0     0     1     0     0
     0     0     0     0     1
     1     1     1     0     1
     0     0     0     0     0

Paso 2
   P(1,2) Y P(2,3)
     0     1     1     0     1
     0     0     1     0     0
     0     0     0     0     1
     1     1     1     0     1
     0     0     0     0     0
```

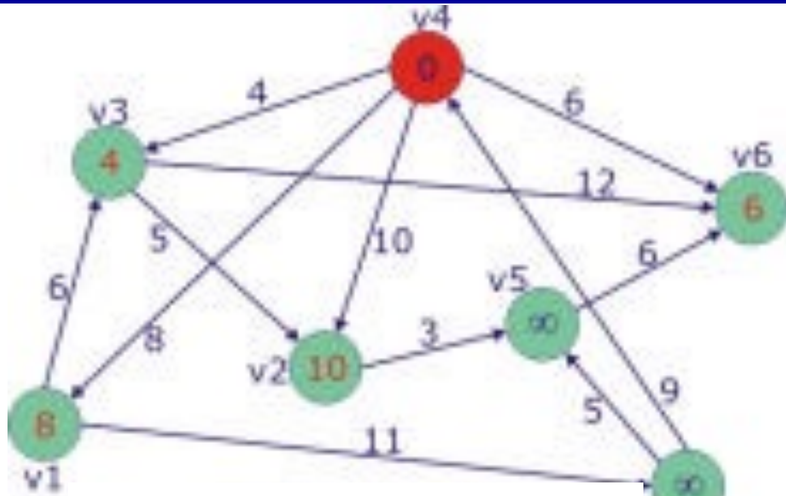
```
Paso 3
   P(2,3) Y P(3,5)
     0     1     1     0     1
     0     0     1     0     1
     0     0     0     0     1
     1     1     1     0     1
     0     0     0     0     0

Paso 4
     0     1     1     0     1
     0     0     1     0     1
     0     0     0     0     1
     1     1     1     0     1
     0     0     0     0     0

Paso 5
     0     1     1     0     1
     0     0     1     0     1
     0     0     0     0     1
     1     1     1     0     1
     0     0     0     0     0
```

No hay cambios porque a 4 no llega ningún arco

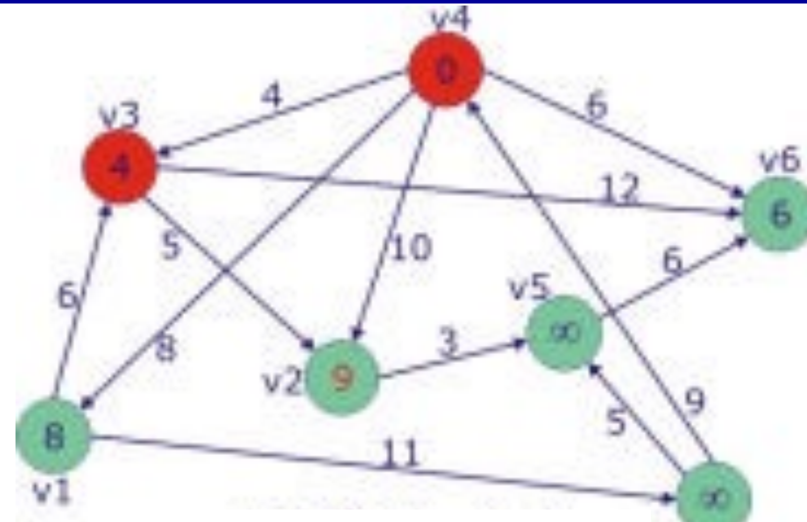
No hay cambios porque de 5 no sale ningún arco

3. Dado el grafo dirigido con factor de peso de la figura:
   a) Encuentra el camino más corto desde el vértice D a todos los demás vértices usando el algoritmo de Dijkstra. Describe el proceso paso a paso.
   b) Encuentra los caminos más cortos entre todos los pares de vértices siguiendo el algoritmo de Floyd.

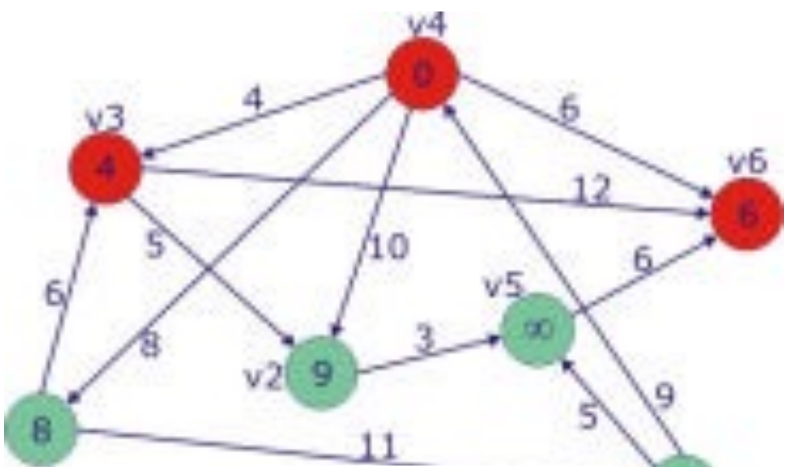




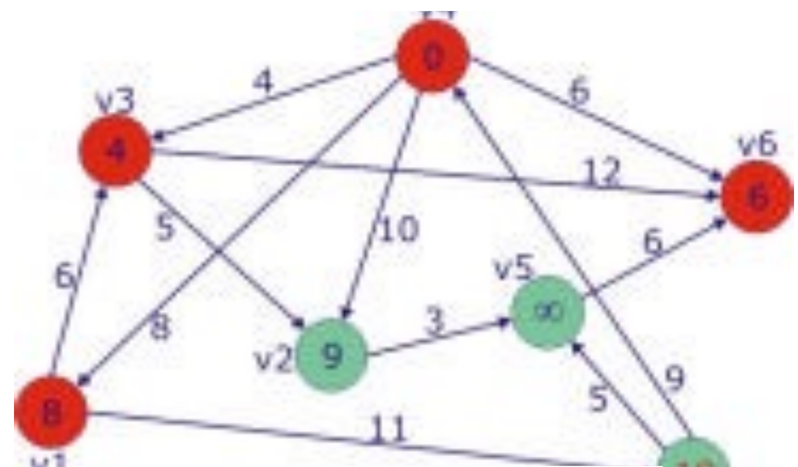
P=(4, 4,4,0,_,4,_)
D=(8,10,4,0,∞,6,∞)



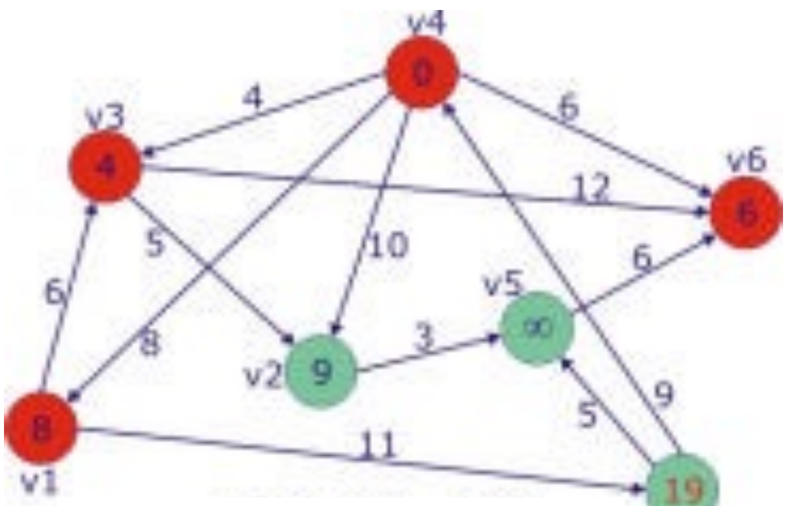
P=(4,3,4,0,_,4,_)
D=(8,9,4,0,∞,6,∞)



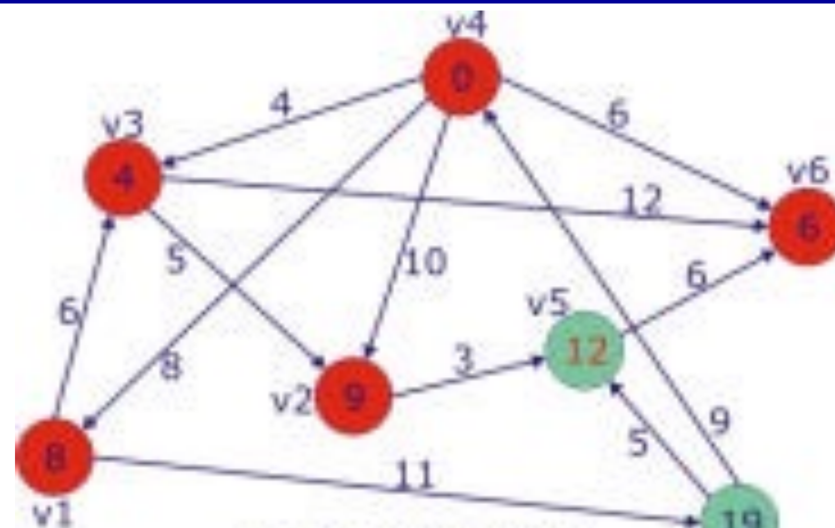
P=(4,3,4,0,_,4,_)
D=(8,9,4,0,∞,6,∞)



P=(4,3,4,0,_,4,1)
D=(8,9,4,0,∞,6,19)



P=(4,3,4,0,_,4, 1)
D=(8,9,4,0,∞,6,19)



P=(4,3,4,0, 2,4, 1)
D=(8,9,4,0,12,6,19)

Camino v4->v1: longitud=8, P(1)=4: v4→v1

Camino v4->v2: longitud=9, P(2)=3: v4→…→v3→v2, P(3)=4: v4→v3→v2

Camino v4->v3: longitud=4, P(3)=4: v4→v3

Camino v4->v5: longitud=12, P(5)=2: v4→…→v2→v5
P(2)=3: v4→…→v3→v2→v5, P(3)=4: v4→v3→v2→v5

Camino v4->v6: longitud=6, P(6)=4: v4→v6

Camino v4->v7: longitud=19, P(7)=1: v4→…→v1→v7, P(1)=4: v4→v1→v7

Ejercicios

```
%Programa que calcula los caminos más cortos entre todos los pares de
%vértices siguiendo el algoritmo de Floyd
clear all
n=7; %n es el número de vértices

%D es la matriz de distancias
D=[ 0 inf   6 inf inf inf  11;
    inf 0 inf inf   5 inf inf;
    inf   5 0 inf  inf 12 inf;
      8  10   4 0 inf   6 inf;
    inf inf inf inf 0   6 inf;
    inf inf inf inf inf 0 inf;
    inf inf inf   9   5 inf 0];
%P es la matriz de vértice previo
%Inicializamos al número de fila, vértice que estamos analizando, si hay arco
P=zeros(n);
for i=1:n
  for j=1:n
    if D(i,j)>0 && D(i,j)<inf
      P(i,j)=i;
    end
  end
end

fprintf('Paso 0\n')
fprintf('Matriz D\n'),disp(D)
fprintf('Matriz P\n'),disp(P)
```

$$D(i,j) = \begin{cases} 0, si\ i = j \\ A(i,j), si\ existe\ arco\ i \rightarrow j \\ \infty, si\ no\ existe\ arco\ i \rightarrow j \end{cases}$$

$$P(i,j) = \begin{cases} i\ si\ existe\ arco\ i \rightarrow j \\ 0\ en\ otro\ caso \end{cases}$$

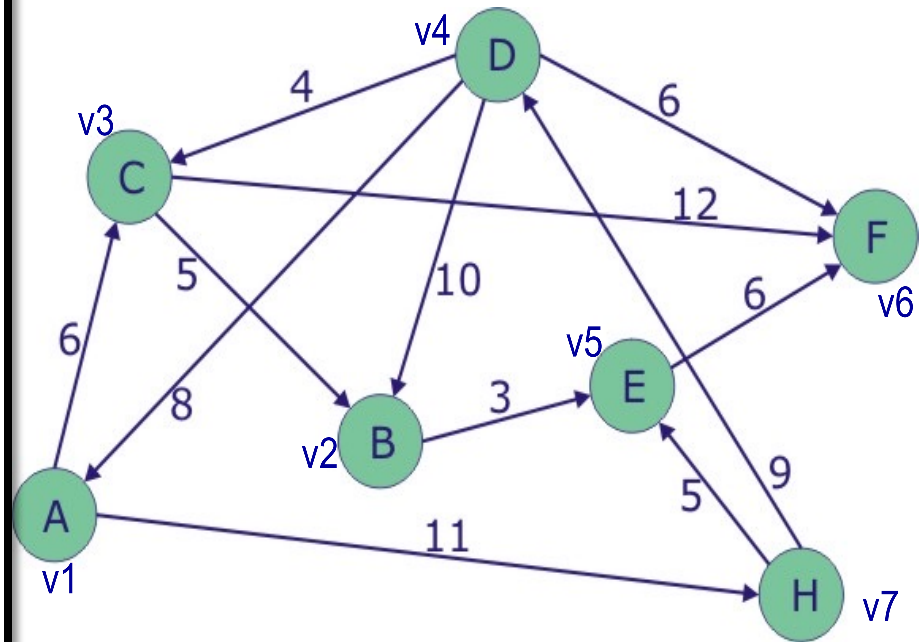
```
Paso 0
Matriz D
     0   Inf     6   Inf   Inf   Inf    11
   Inf     0   Inf   Inf     5   Inf   Inf
   Inf     5     0   Inf   Inf    12   Inf
     8    10     4     0   Inf     6   Inf
   Inf   Inf   Inf   Inf     0     6   Inf
   Inf   Inf   Inf   Inf   Inf     0   Inf
   Inf   Inf   Inf     9     5   Inf     0
Matriz P
   0   0   1   0   0   0   1
   0   0   0   0   2   0   0
   0   3   0   0   0   3   0
   4   4   4   0   0   4   0
   0   0   0   0   0   5   0
   0   0   0   0   0   0   0
   0   0   0   7   7   0   0
```

```
for k=1:n %k es el número de vértices y el número de matrices D: D1 a Dn
    fprintf('Paso %d\n',k)
    for i=1:n
        for j=1:n
            if D(i,j)>D(i,k)+D(k,j)
                P(i,j)=P(k,j);
                fprintf('D(%d,%d) Y D(%d,%d)\n',i,k,k,j)
            end
            D(i,j)=min(D(i,j),D(i,k)+D(k,j));
        end
    end
    fprintf('Matriz D\n'),disp(D)
    fprintf('Matriz P\n'),disp(P)
end
```

```
Paso 0
Matriz D
     0   Inf     6   Inf   Inf   Inf    11
   Inf     0   Inf   Inf     5   Inf   Inf
   Inf     5     0   Inf   Inf    12   Inf
     8    10     4     0   Inf     6   Inf
   Inf   Inf   Inf   Inf     0     6   Inf
   Inf   Inf   Inf   Inf   Inf     0   Inf
   Inf   Inf   Inf     9     5   Inf     0
Matriz P
     0     0     1     0     0     0     1
     0     0     0     0     2     0     0
     0     3     0     0     0     3     0
     4     4     4     0     0     4     0
     0     0     0     0     0     5     0
     0     0     0     0     0     0     0
     0     0     0     7     7     0     0
Paso 1
D(4,1) Y D(1,7)
Matriz D
     0   Inf     6   Inf   Inf   Inf    11
   Inf     0   Inf   Inf     5   Inf   Inf
   Inf     5     0   Inf   Inf    12   Inf
     8    10     4     0   Inf     6    19
   Inf   Inf   Inf   Inf     0     6   Inf
   Inf   Inf   Inf   Inf   Inf     0   Inf
   Inf   Inf   Inf     9     5   Inf     0
Matriz P
     0     0     1     0     0     0     1
     0     0     0     0     2     0     0
     0     3     0     0     0     3     0
     4     4     4     0     0     4     1
     0     0     0     0     0     5     0
     0     0     0     0     0     0     0
     0     0     0     7     7     0     0
```

Ejercicios

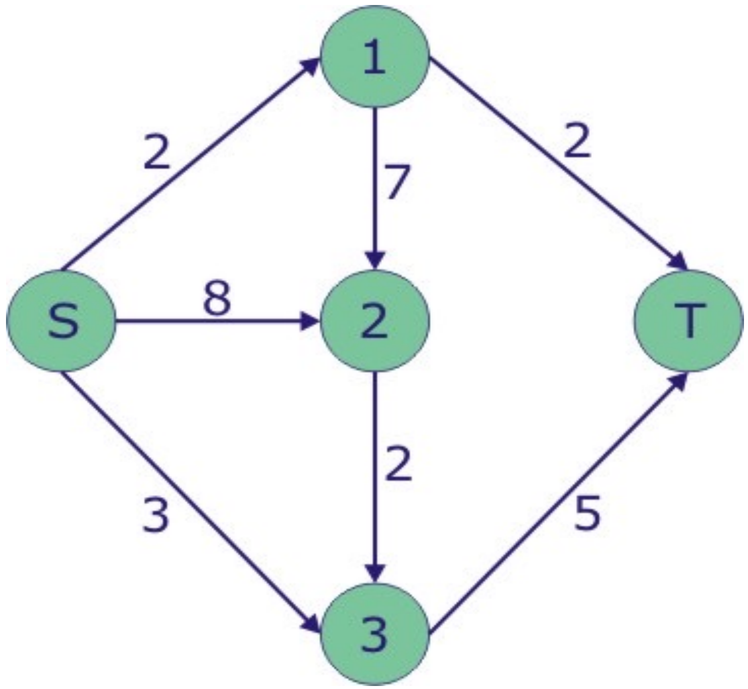
```
Paso 7
D(1,7) Y D(7,4)
Matriz D
      0     11      6     20     16     18     11
    Inf      0    Inf    Inf      5     11    Inf
    Inf      5      0    Inf     10     12    Inf
      8      9      4      0     14      6     19
    Inf    Inf    Inf    Inf      0      6    Inf
    Inf    Inf    Inf    Inf    Inf      0    Inf
     17     18     13      9      5     11      0
Matriz P
    0    3    1    7    2    3    1
    0    0    0    0    2    5    0
    0    3    0    0    2    3    0
    4    3    4    0    2    4    1
    0    0    0    0    0    5    0
    0    0    0    0    0    0    0
    4    3    4    7    7    5    0
>> 
```

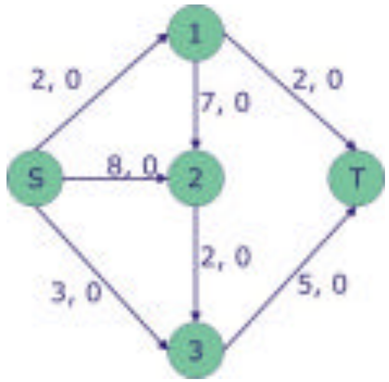


Camino v1->v6: longitud=18; P(1,6)=3: v1→…v3→v6;
P(1,3)=1: v1→v3→v6

Camino v7->v2: longitud=18; P(7,2)=3: v7→…v3→v2;
P(7,3)=4: v7→…v4→v3→v2;
P(7,4)=7: v7→v4→v3→v2

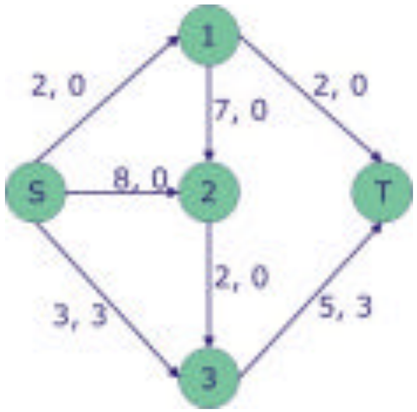
4. Dado el siguiente grafo, calcula el aumento máximo de flujo





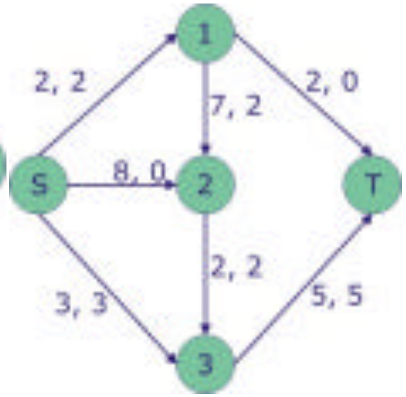
Cadena S-3-T

Arcos incrementables:
min(Uij-Fij)=min(3-0,5-0)=3



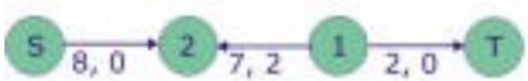
Cadena S-1-2-3-T

Arcos incrementables:
mín(Uij-Fij)=mín(2-0,7-0, 2-0, 5-3)=2



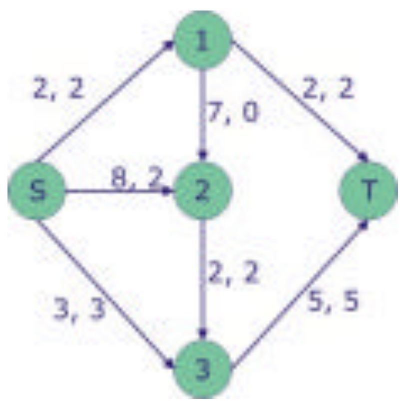
Cadena S-1-T: mín(Uij-Fij)=0
Cadena S-2-3-T: mín(Uij-Fij)=0

Cadena S-2-1-T



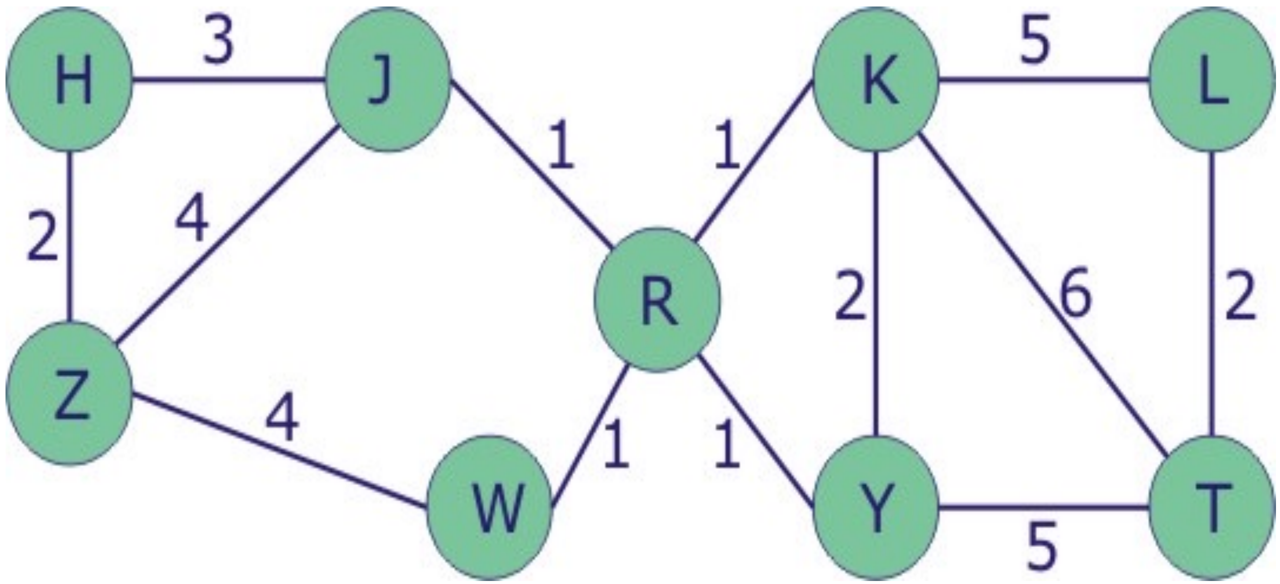
Arcos incrementables:
mín(Uij-Fij)=mín(8-0,2-0)=2
Arcos reducibles:
mín(Fij)=mín(2)=2

Mínimo=2

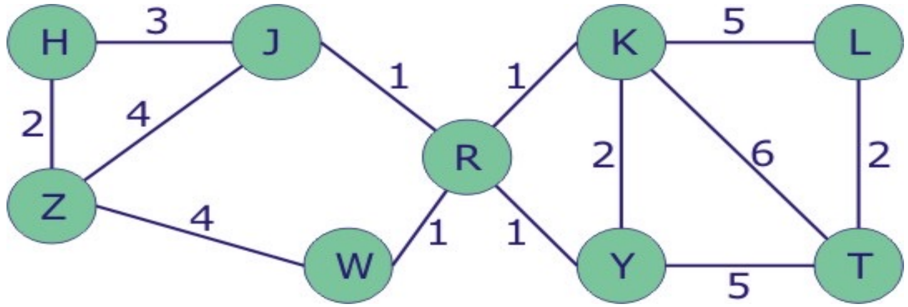


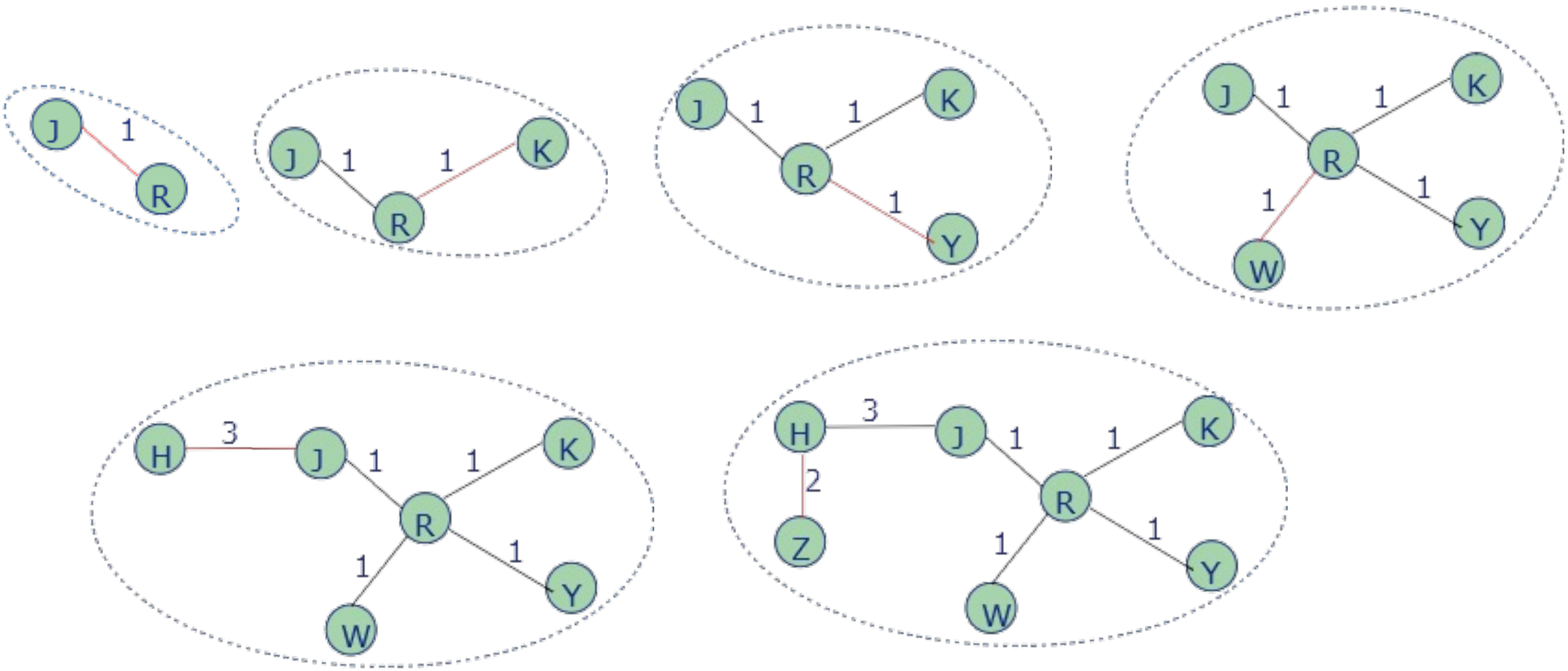
FLUJO MÁXIMO=7

5. Dado el grafo de la figura, encuentra un árbol de expansión de coste mínimo describiendo el proceso paso a paso mediante el algoritmo de Prim y mediante el algoritmo de Kruskal.

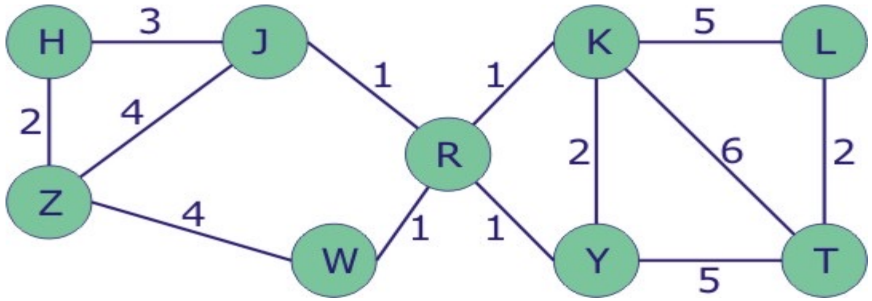


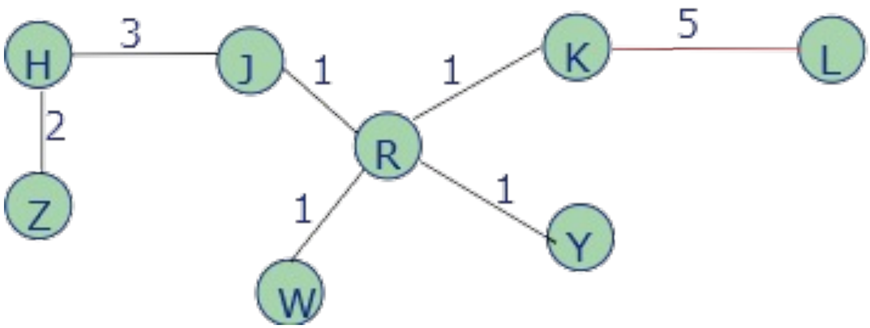
5. (CONT.) Algoritmo de Prim, empezamos por un vértice por ejemplo J y añadimos al conjunto el vértice cuya arista tenga menor coste.

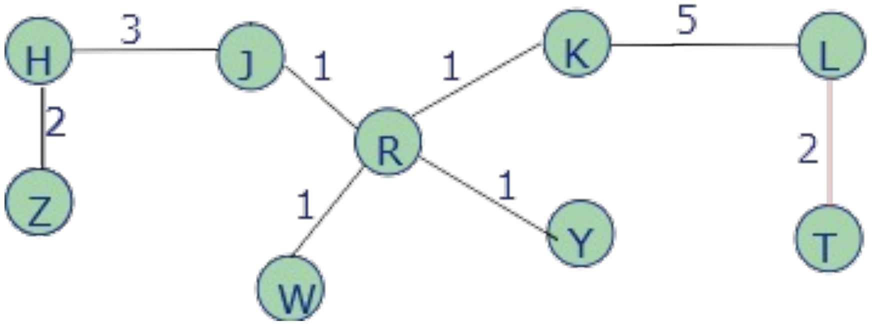




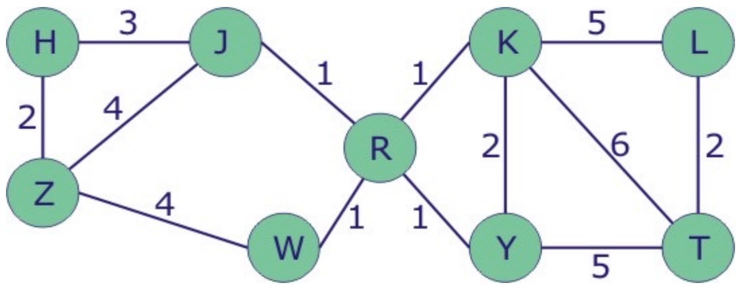
5. (CONT.) Algoritmo de Prim, empezamos por un vértice por ejemplo J y añadimos al conjunto el vértice cuya arista tenga menor coste.

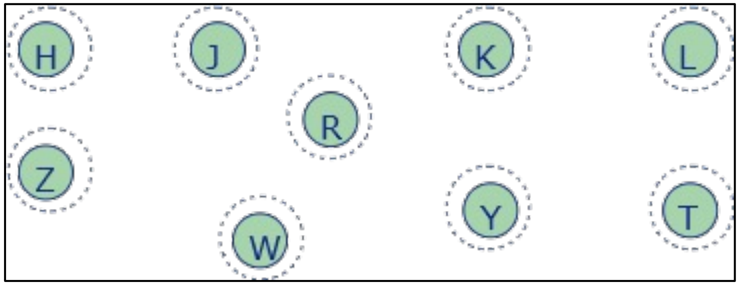


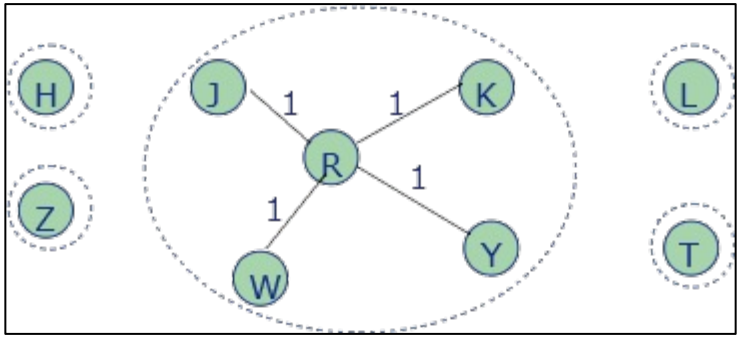


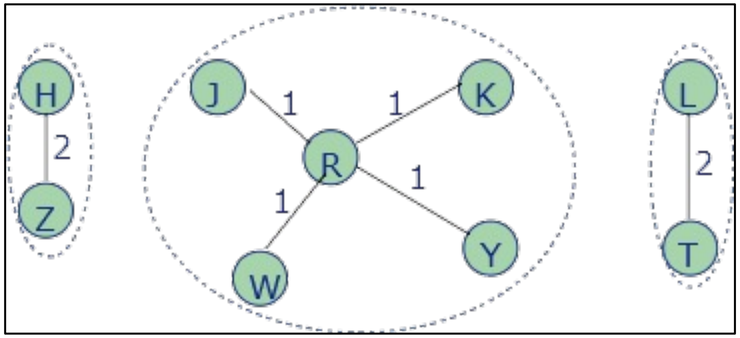


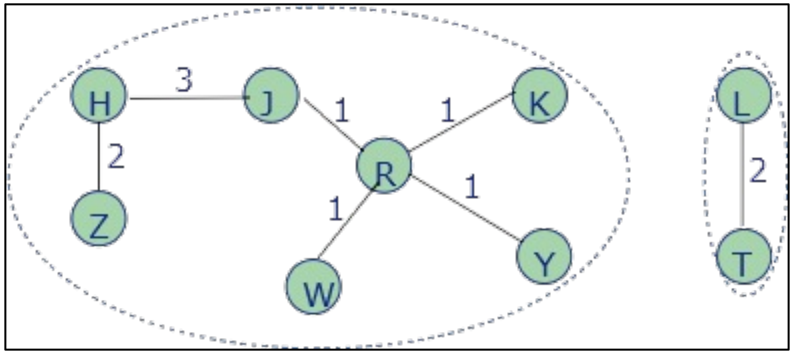
5. (cont.) Algoritmo de Kruskal: cada vértice es un conjunto independiente y se ordenan los arcos por coste de menor a mayor.











5. Algoritmo de Kruskal: cada vértice es un conjunto independiente y se ordenan los arcos por coste de menor a mayor.