Estructura de Datos

Sesión 13

Grafos No Dirigidos

Gustavo Arredondo C.

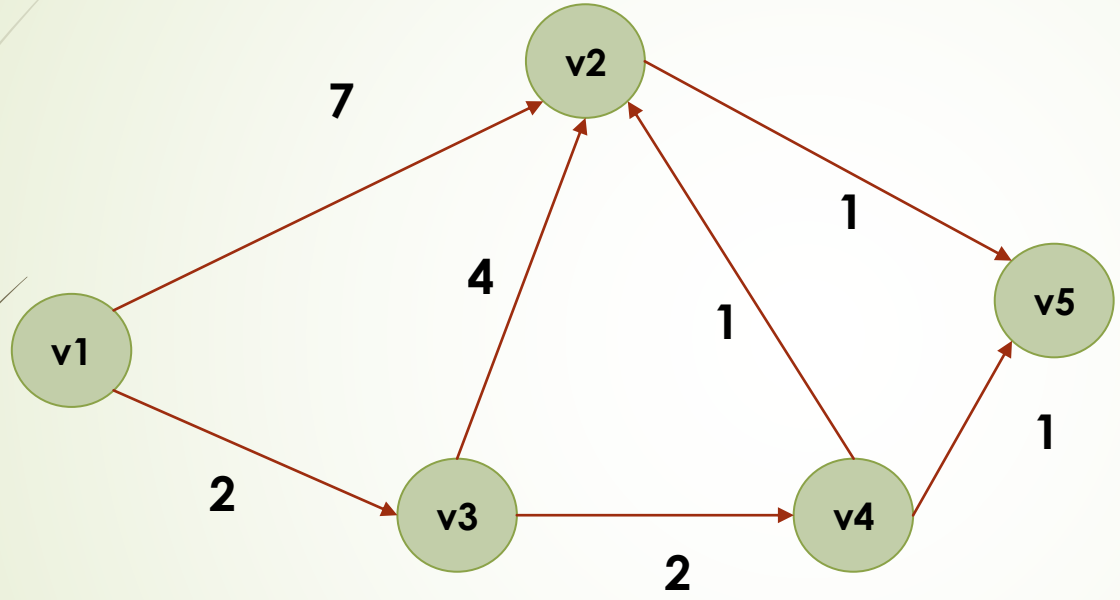
GRAFOS Algoritmos

- Grafos Dirigidos
- Algoritmo de Dijkstra
- Algoritmo de Floyd – Warshall
- Grafos No Dirigidos
- Algoritmo de Dijkstra
- Algoritmo de Kruskal
- Algoritmo de Prim

Algoritmo Floyd Wharshall

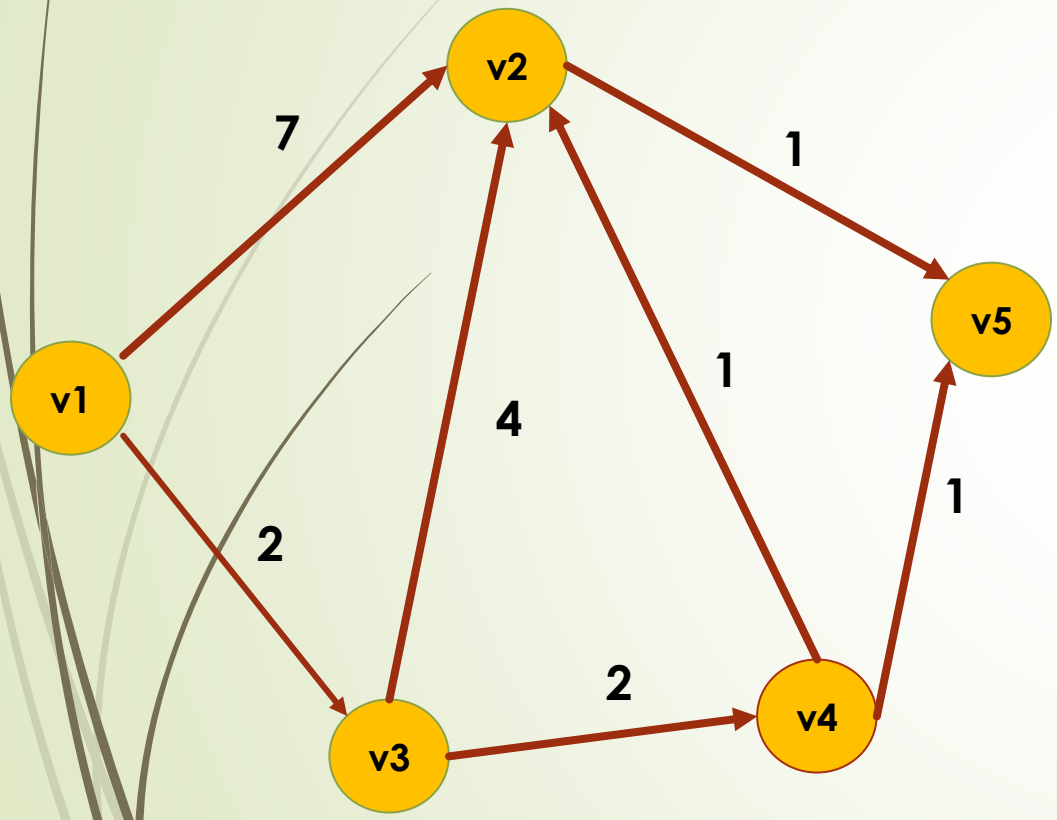
Matriz de Pesos

Matriz de Pesos
Algoritmo Floyd Wharshall



	v1	v2	v3	v4	v5
v1	0	7	2	--	--
v2	--	0	--	--	1
v3	--	4	0	2	--
v4	--	1	--	0	1
v5	--	--	--	--	0

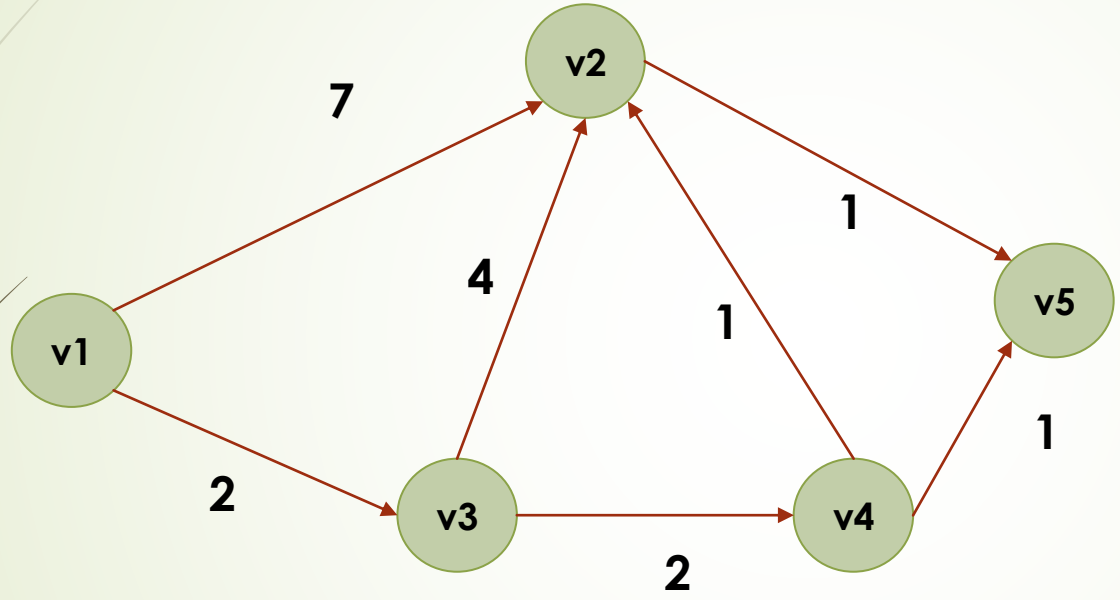
Matriz de Pesos
Algoritmo Floyd Wharshall



	v1	v2	v3	v4	v5
v1	0	**5**	**2**	**4**	**5**
v2	--	0	--	--	1
v3	--	**3**	0	2	**3**
v4	--	1	--	0	1
v5	--	--	--	--	0

Algoritmo Floyd Wharshall

USANDO v1 COMO VERTICE INTERMEDIO



	v1	v2	v3	v4	v5
v1	0	7	2	--	--
v2	--	0	--	--	1
v3	--	4	0	2	--
v4	--	1	--	0	1
v5	--	--	--	--	0

De v1 a vi (2,3,4,5) No lo consideramos (v1 es vértice origen)

De v2a vi (3,4,5) No lo consideramos (vértice v2 a v1 es --)

De v3 a vi (2,4,5) No lo consideramos (vértice v3 a v1 es --)

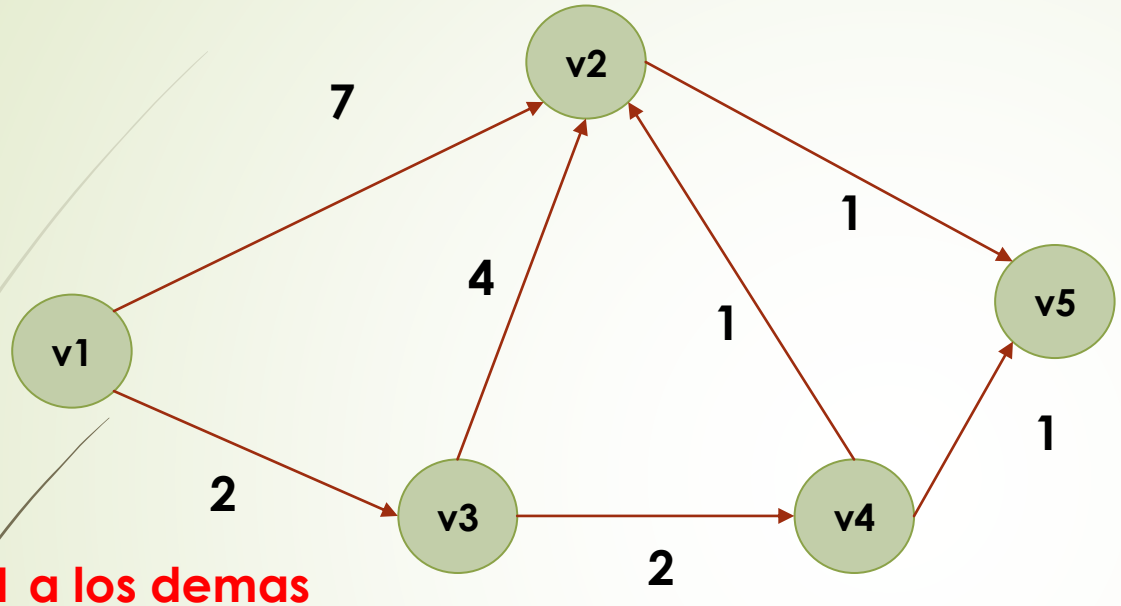
De v4 a vi (2,3,5) No lo consideramos (vértice v4 a v1 es --)

No sea mejorado nada

De v5 a vi (2,3,4) No lo consideramos (vértice v5 a v1 es --)

Algoritmo Floyd Wharshall

USANDO v2 COMO VERTICE INTERMEDIO



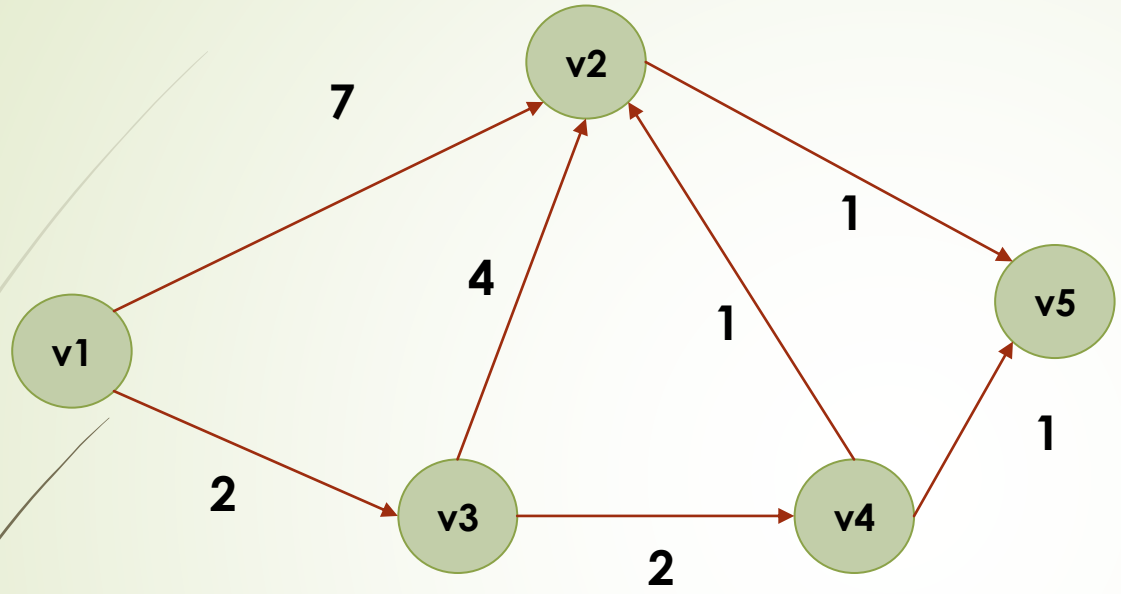
	v1	v2	v3	v4	v5
v1	0	7	2	--	**8**
v2	--	0	--	--	1
v3	--	4	0	2	--
v4	--	1	--	0	1
v5	--	--	--	--	0

Estudiando V1 a los demas

Ver si se puede rebajar el precio de 1 a 3, de y a 4 y de 1 a 5
Analizar las rutas de v1 a v3 usando v2 como intermedio = --
Analizar las rutas de v1 a v4 usando v2 como intermedio = --
Analizar las rutas de v1 a v5 usando v2 como intermedio = 7+1=8

Algoritmo Floyd Wharshall

USANDO v2 COMO VERTICE INTERMEDIO



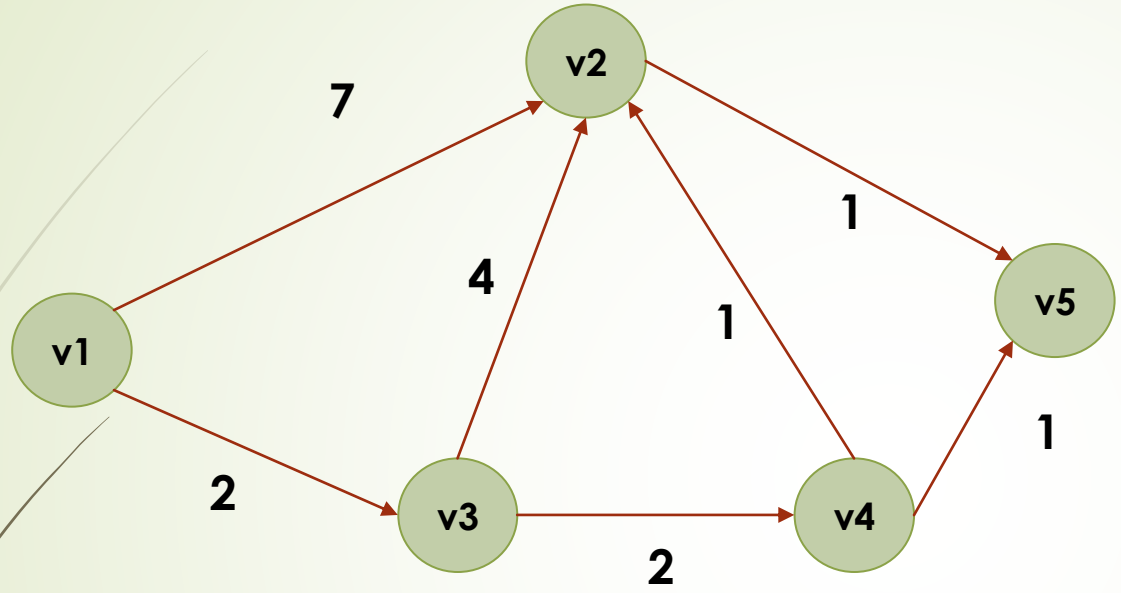
	v1	v2	v3	v4	v5
v1	0	7	2	--	**8**
v2	--	0	--	--	1
v3	--	4	0	2	--
v4	--	1	--	0	1
v5	--	--	--	--	0

Estudiando V2 a los demás (v1, v3,v4,v5)

Por ser el intermedio no se considera nada

Algoritmo Floyd Wharshall

USANDO v2 COMO VERTICE INTERMEDIO



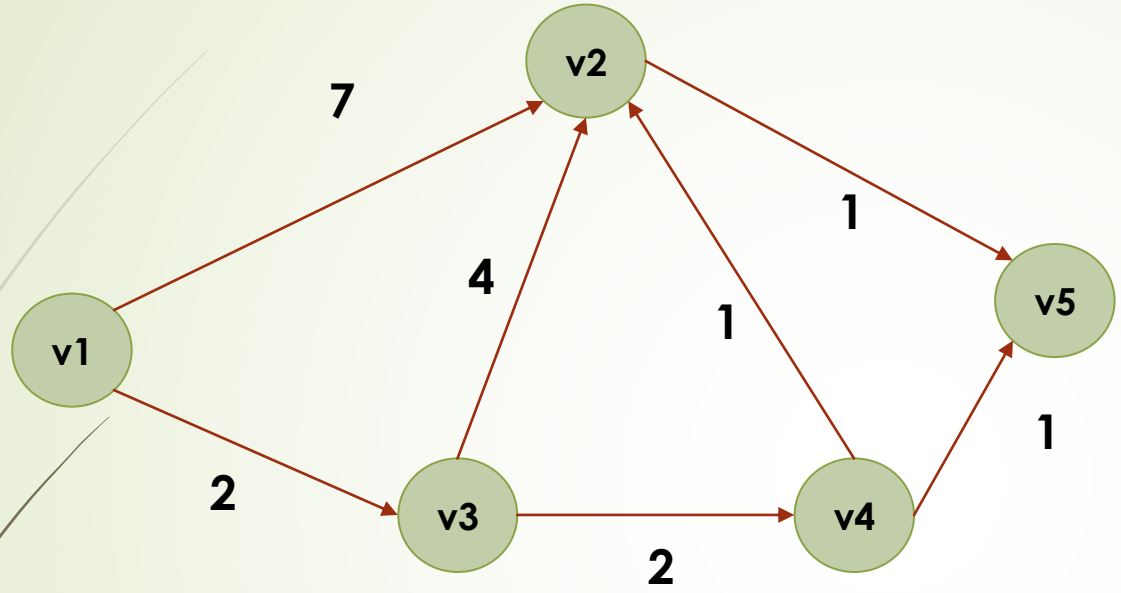
	v1	v2	v3	v4	v5
v1	0	7	2	--	8
v2	--	0	--	--	1
v3	--	4	0	2	**5**
v4	--	1	--	0	1
v5	--	--	--	--	0

Estudiando V3 a los demás (v1, v4,v5)

Analizar las rutas de v3 a v1 usando v2 como intermedio = --
Analizar las rutas de v3 a v4 usando v2 como intermedio = --
Analizar las rutas de v3 a v5 usando v2 como intermedio = 4+1=5

Algoritmo Floyd Wharshall

USANDO v2 COMO VERTICE INTERMEDIO



	v1	v2	v3	v4	v5
v1	0	7	2	--	8
v2	--	0	--	--	1
v3	--	4	0	2	**5**
v4	--	1	--	0	1
v5	--	--	--	--	0

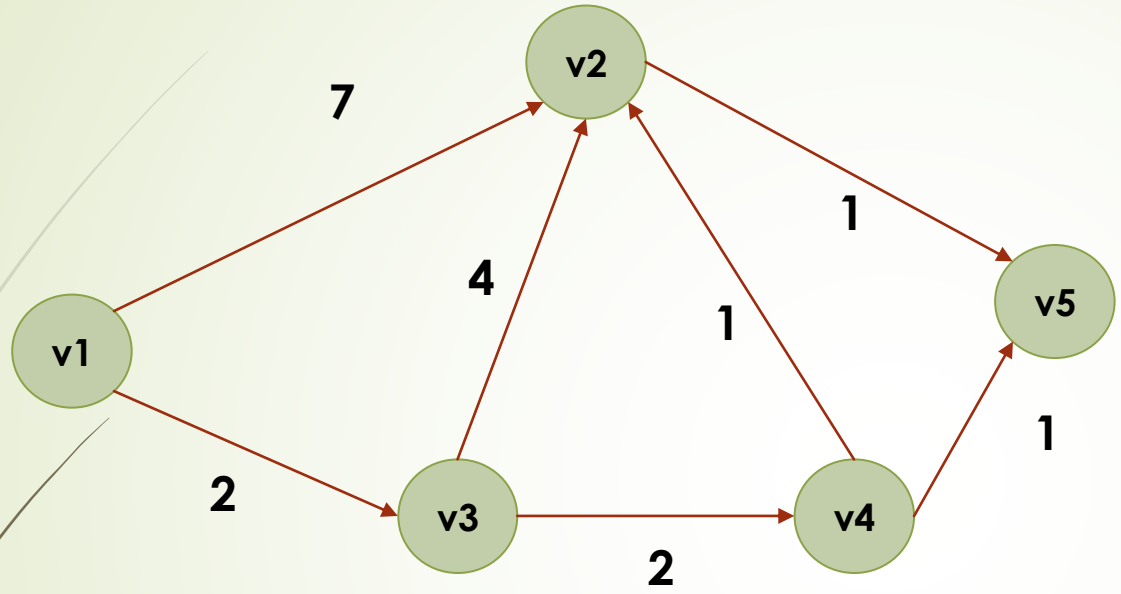
Estudiando V4 a los demás (v1, v3,v5)

Analizar las rutas de v4 a v1 usando v2 como intermedio = --
Analizar las rutas de v4 a v3 usando v2 como intermedio = --
Analizar las rutas de v4 a v5 usando v2 como intermedio = 1+1=2

No se cambia por ser mayor

Algoritmo Floyd Wharshall

USANDO v2 COMO VERTICE INTERMEDIO



	v1	v2	v3	v4	v5
v1	0	7	2	--	8
v2	--	0	--	--	1
v3	--	4	0	2	**5**
v4	--	1	--	0	1
v5	--	--	--	--	0

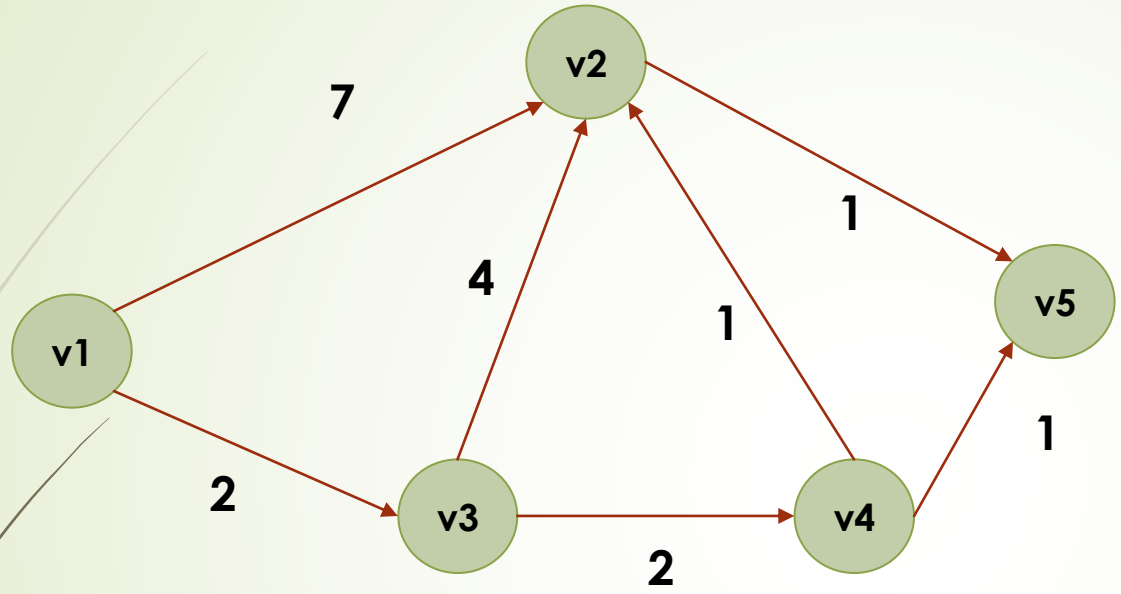
Estudiando V5 a los demás (v1, v3,v4)

Analizar las rutas de v5 a v1 usando v2 como intermedio = --
Analizar las rutas de v5 a v3 usando v2 como intermedio = --
Analizar las rutas de v5 a v4 usando v2 como intermedio = --

No se cambia no existe camino

Algoritmo Floyd Wharshall

USANDO v3 COMO VERTICE INTERMEDIO



	v1	v2	v3	v4	v5
v1	0	**6**	2	**4**	**5**
v2	--	0	--	--	1
v3	--	4	0	2	5
v4	--	1	--	0	1
v5	--	--	--	--	0

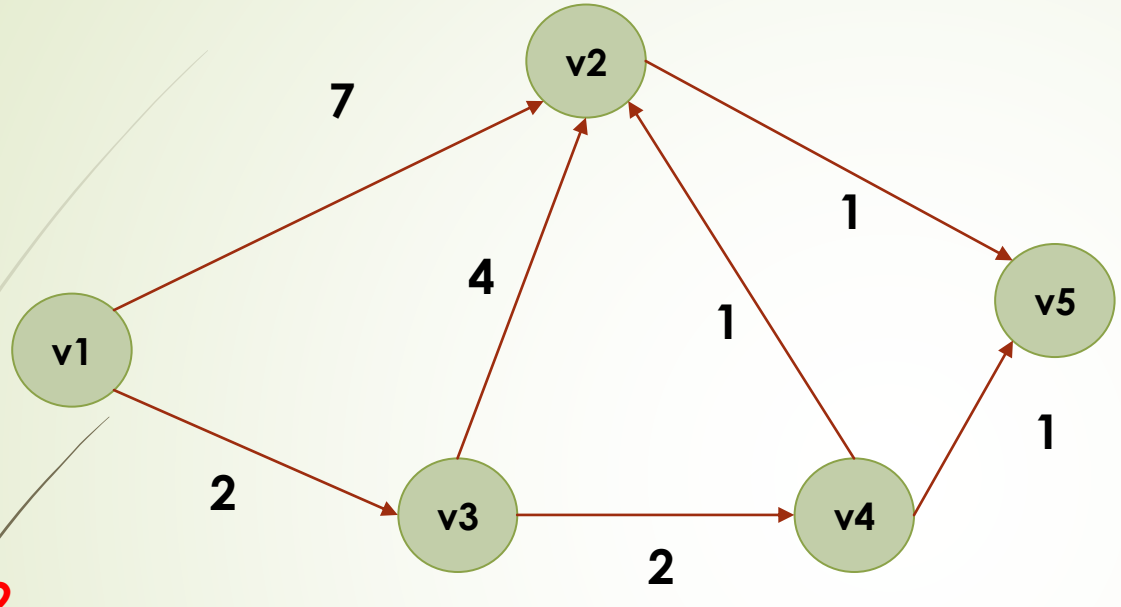
Desde v1

Se analizaran v2, v4, v5

Analizar las rutas de v1 a v2 usando v3 como intermedio = 6
Analizar las rutas de v1 a v4 usando v3 como intermedio = 4
Analizar las rutas de v1 a v5 usando v3 como intermedio = 5

Algoritmo Floyd Wharshall

USANDO v3 COMO VERTICE INTERMEDIO



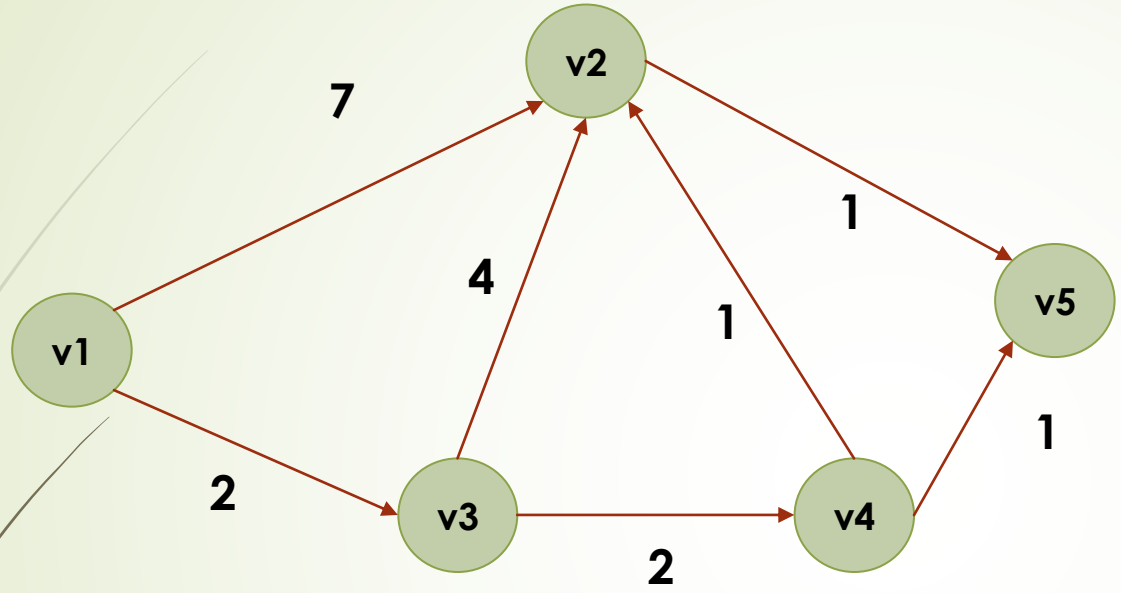
	v1	v2	v3	v4	v5
v1	0	**6**	2	**4**	**5**
v2	--	0	--	--	1
v3	--	4	0	2	5
v4	--	1	--	0	1
v5	--	--	--	--	0

Desde v2

No se considera, no existe conexion

Algoritmo Floyd Wharshall

USANDO v3 COMO VERTICE INTERMEDIO



	v1	v2	v3	v4	v5
v1	0	**5**	2	**4**	**5**
v2	--	0	--	--	1
v3	--	4	0	2	5
v4	--	1	--	0	1
v5	--	--	--	--	0

Desde v3
No se considera por ser intermedio

Desde v4

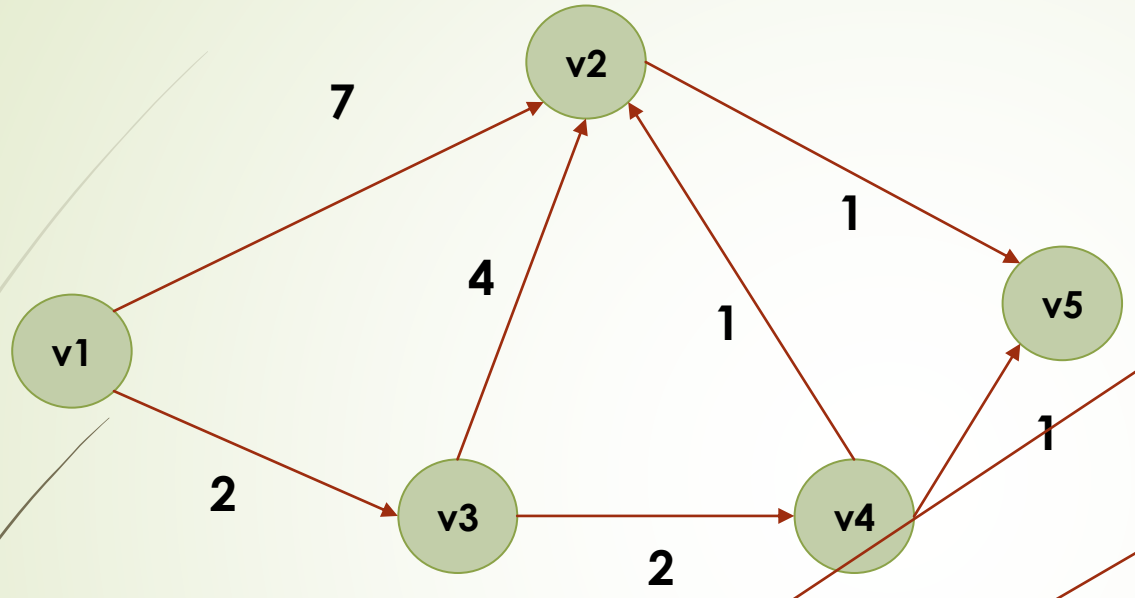
No se considera, no existe conexion

Desde v5

No se considera, no existe conexion

Algoritmo Floyd Wharshall

USANDO v4 COMO VERTICE INTERMEDIO



	v1	v2	v3	v4	v5
v1	0	**5**	2	**4**	**5**
v2	--	0	--	--	1
v3	--	4	0	2	5
v4	--	1	--	0	1
v5	--	--	--	--	0

Desde v1, pasando por v4, se analizaran v2, v3 y v5

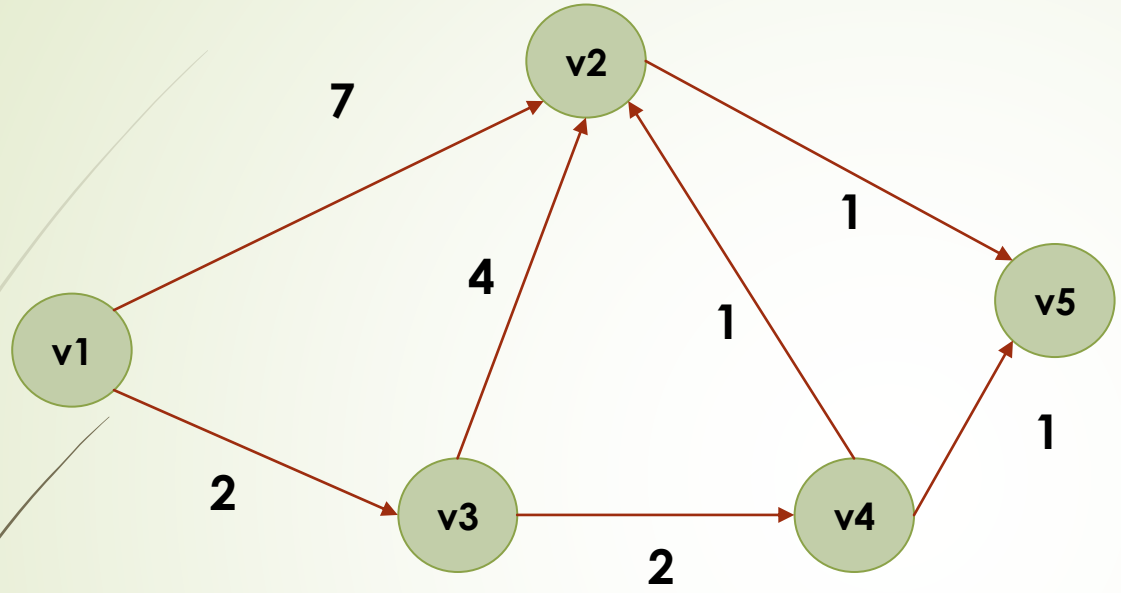
P(v1,v2) **2+2+1= 5** **Reemplaza por ser menor que 6**

P(v1,v3) **--**

P(v1,v5) **2+2+1= 5** **Reemplaza por ser menor que 7**

Algoritmo Floyd Wharshall

USANDO v4 COMO VERTICE INTERMEDIO



	v1	v2	v3	v4	v5
v1	0	**5**	2	**4**	**5**
v2	--	0	--	--	1
v3	--	4	0	2	5
v4	--	1	--	0	1
v5	--	--	--	--	0

Desde v2, pasando por v4, se analizaran v1, v3 y v5

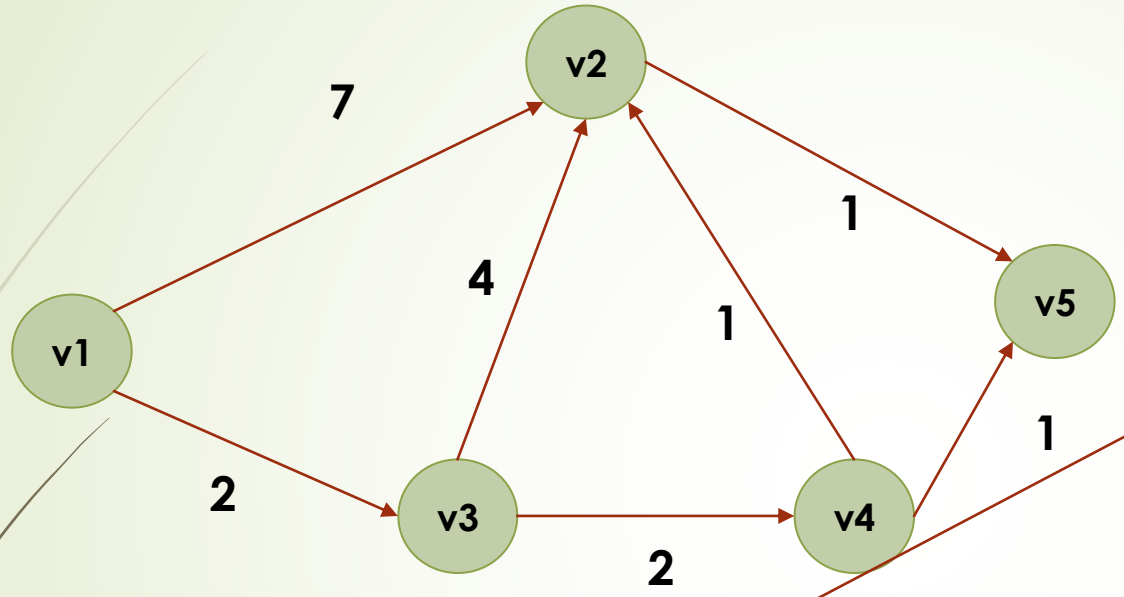
P(v2,v1) --

P(v2,v3) --

P(v2,v5) --

Algoritmo Floyd Wharshall

USANDO v4 COMO VERTICE INTERMEDIO



	v1	v2	v3	v4	v5
v1	0	**5**	2	**4**	**5**
v2	--	0	--	--	1
v3	--	**3**	0	2	**3**
v4	--	1	--	0	1
v5	--	--	--	--	0

Desde v3, pasando por v4, se analizaran v1, v2 y v5

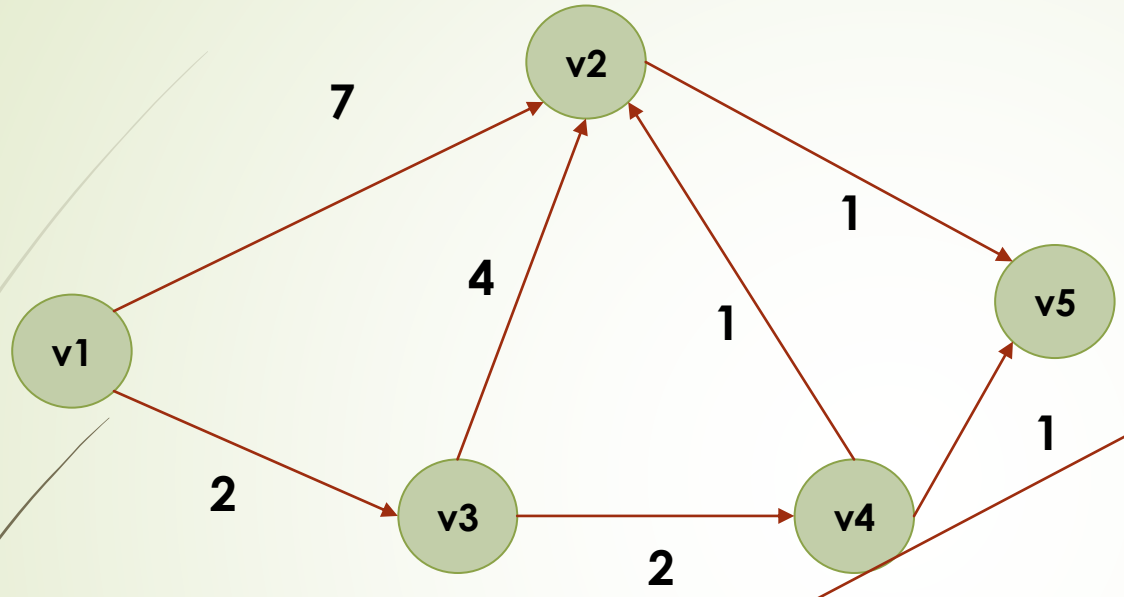
P(v3,v1) --

P(v3,v2) 3

P(v3,v5) 3

Algoritmo Floyd Wharshall

USANDO v4 COMO VERTICE INTERMEDIO



	v1	v2	v3	v4	v5
v1	0	**5**	2	**4**	**5**
v2	--	0	--	--	1
v3	--	**3**	0	2	**3**
v4	--	1	--	0	1
v5	--	--	--	--	0

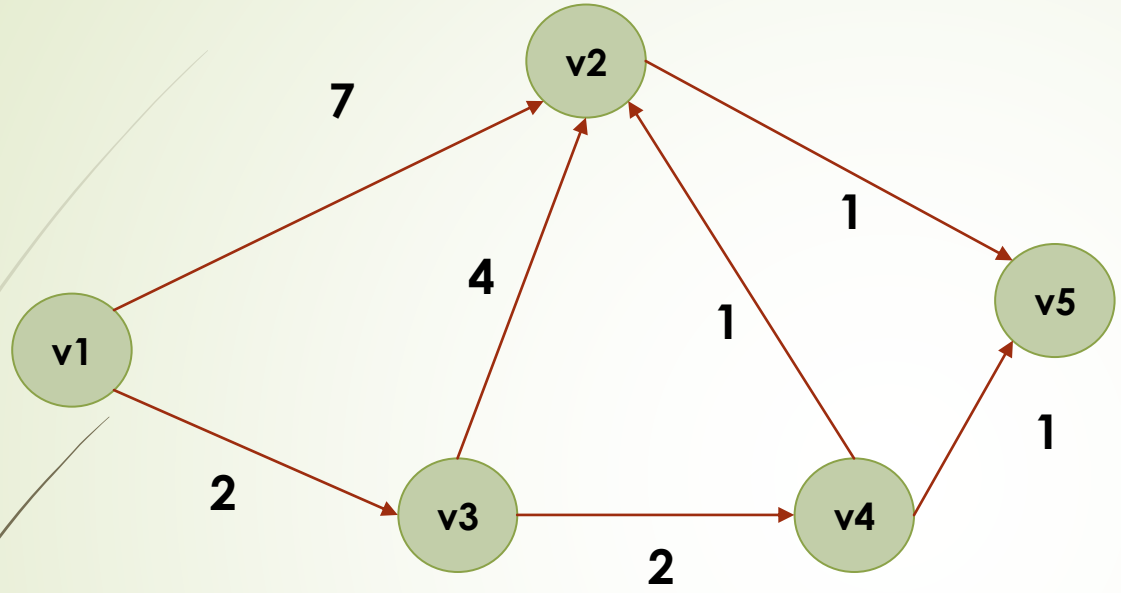
Desde v3, pasando por v4, se analizaran v1, v2 y v5

P(v3,v1) --
P(v3,v2) 3
P(v3,v5) 3

No se considera v4 por ser el intermedio
No se considera v5 por ser infinito

Algoritmo Floyd Wharshall

USANDO v5 COMO VERTICE INTERMEDIO

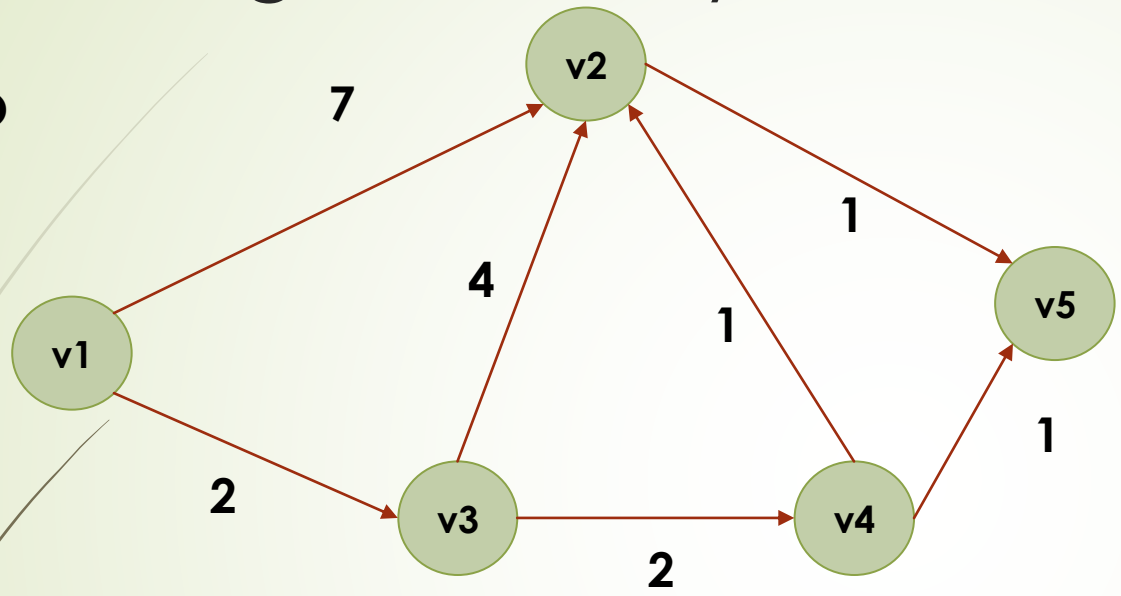


	v1	v2	v3	v4	v5
v1	0	**5**	2	**4**	**5**
v2	--	0	--	--	1
v3	--	**3**	0	2	**3**
v4	--	1	--	0	1
v5	--	--	--	--	0

V5 es infinito en todos los casos, no se considera ninguno

Matriz de Pesos
Algoritmo Floyd Wharshall

RESULTADO



	v1	v2	v3	v4	v5
v1	0	**5**	2	**4**	**5**
v2	--	0	--	--	1
v3	--	**3**	0	2	**3**
v4	--	1	--	0	1
v5	--	--	--	--	0

Matriz de camino mas corto

V5 es infinito en todos los casos, no se considera ninguno

Sesion 12

- Grafos (Parte II)

Grafos No Dirigidos

Algoritmo de Dijkstra

Algoritmo de Prim

Algoritmo de Kruskal

Grafos No Dirigidos

- Un gráfo no dirigida G=(V,A) consta de un numero finito de vértices V y un numero finito de Aristas A. Se diferencia de un grafo dirigido en que cada arista de A es un par no ordenado de vértices.
- Si (u,v) es una arista de un grafo no dirigido, entonces (u,v)=(v,u)

Edsger Dijkstra

Edsger Dijkstra en 2002.

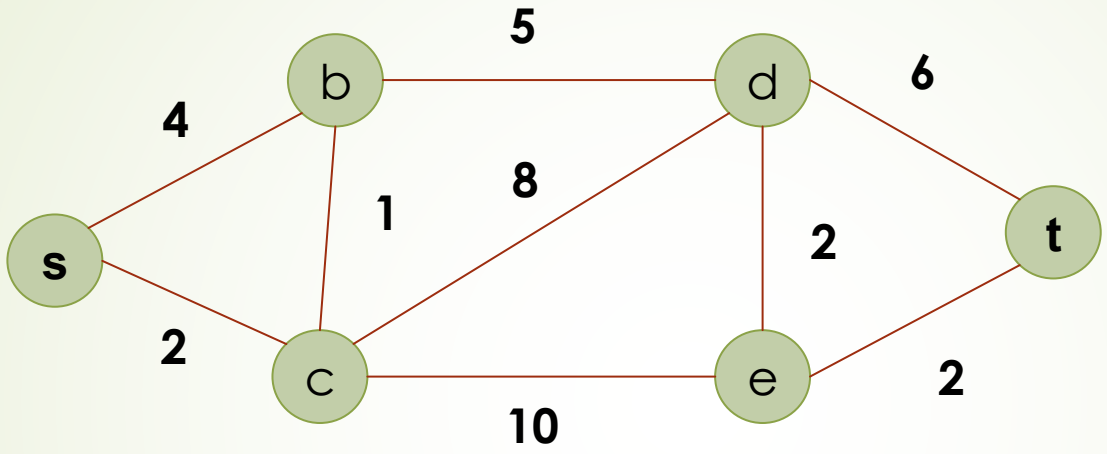
Nombre	Edsger Wybe Dijkstra
Nacimiento	11 de mayo de 1930 Róterdam, Países Bajos
Fallecimiento	6 de agosto de 2002 (72 años) Nuenen, Países Bajos
Causa de muerte	cáncer
Nacionalidad	holandés
Alma máter	Universidad de Leiden
Ocupación	Científico de la computación, físico
Premios	Premio Turing en 1972

Algoritmo de Dijkstra

Llamado también Algoritmo de Caminos minimos

Ejemplo:
Calcular el camino mas corto entre s y t

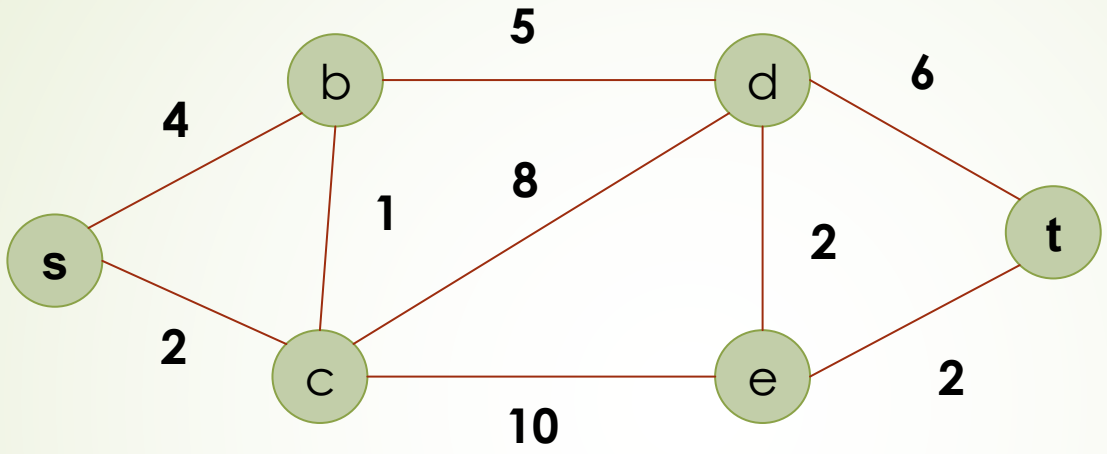
- Empieza por vértice s
- Distancia de s a s es cero
- Se coloca como definitivo



VERTICE	PASO 1	PASO 2	PASO 3	PASO 4	PASO 5	PASO 6
s	(0,s)					
b						
c						
d						
e						
t						

Ejemplo:
Calcular el camino mas corto entre s y t

- Distancia de b a s es cuatro
- Se coloca como definitivo

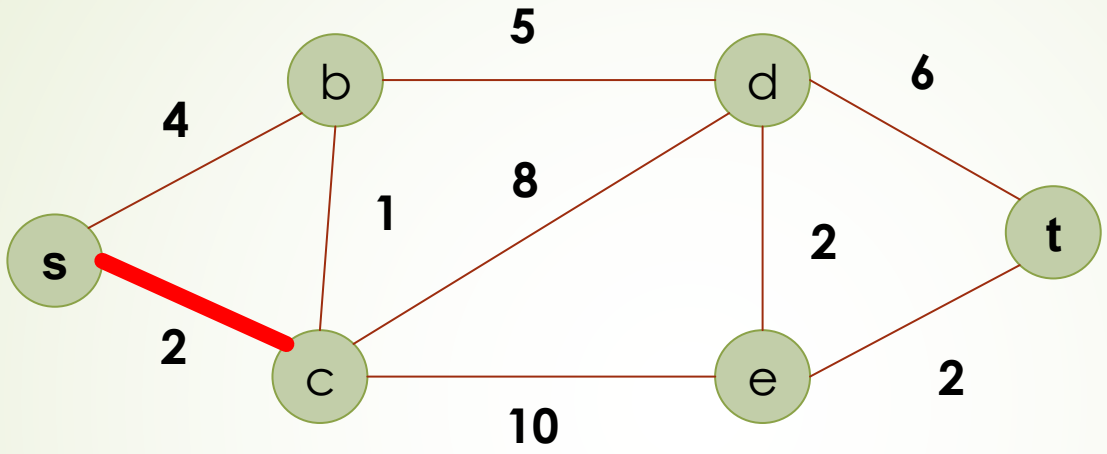


Vectores adyacentes a s: b y c

VERTICE	PASO 1	PASO 2	PASO 3	PASO 4	PASO 5	PASO 6
s	(0,s)	X	X	X	X	X
b	(4,s)					
c	(2,s)					
d	oo					
e	oo					
t	oo					

Ejemplo:
Calcular el camino mas corto entre s y t

- S a c pesa menos
- Se pasa a la sgte columna y se determina definitiva

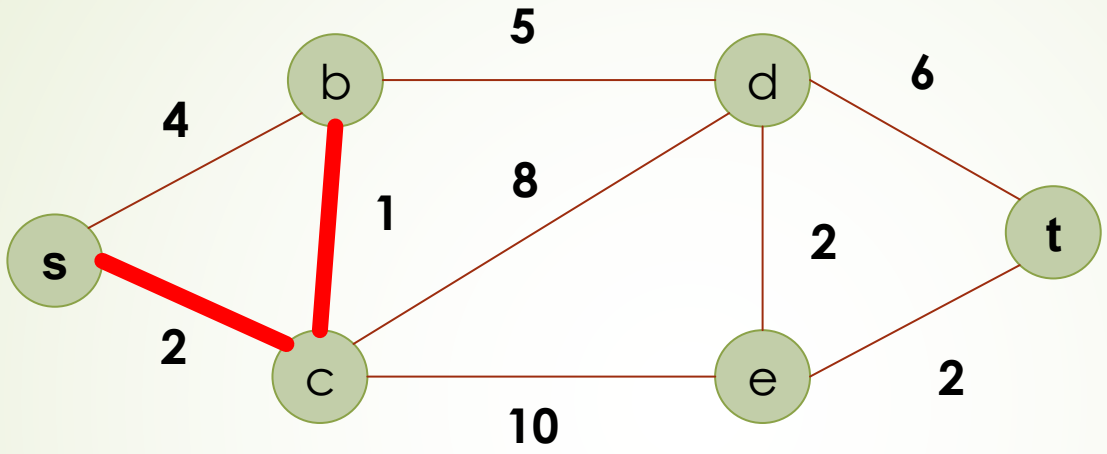


Vectores adyacentes a s: b y c

VERTICE	PASO 1	PASO 2	PASO 3	PASO 4	PASO 5	PASO 6
s	(0,s)	X	X	X	X	X
b	(4,s)					
c	(2,s)	(2,s)	X	X	X	X
d	oo					
e	oo					
t	oo					

Ejemplo:
Calcular el camino mas corto entre s y t

- Pesa 1+2=3
- Entre 3, 10 y 12, el 3 pesa menos
- Asigna etiqueta definitiva

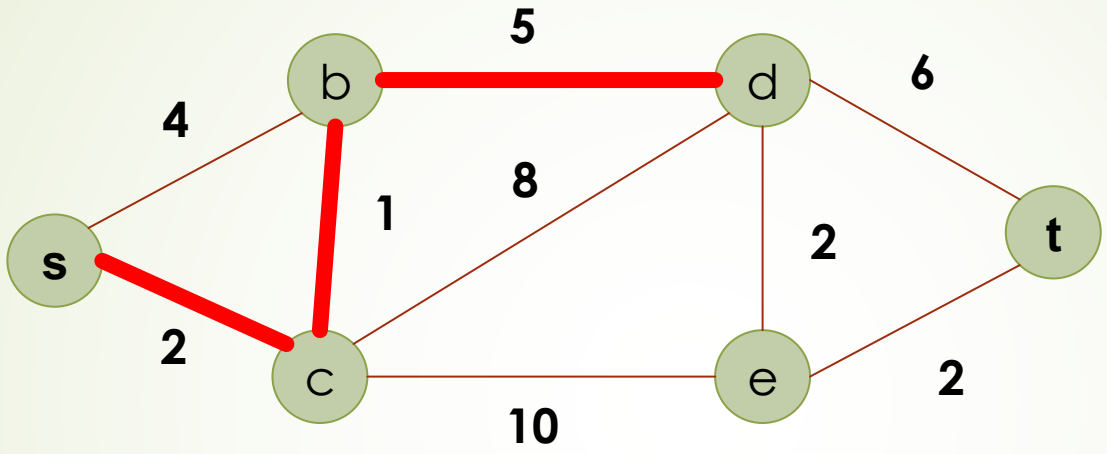


Vectores d y e son adyacentes a: b y c

VERTICE	PASO 1	PASO 2	PASO 3	PASO 4	PASO 5	PASO 6
s	(0,s)	X	X	X	X	X
b	(4,s)	(3,c)	(3,c)	X	X	X
c	(2,s)	(2,s)	X	X	X	X
d	oo	(10,c)				
e	oo	(12,c)				
t	oo	oo				

Ejemplo:
Calcular el camino mas corto entre s y t

- Entre 8 y 12 el menor es 8, se traslada y se y se fija

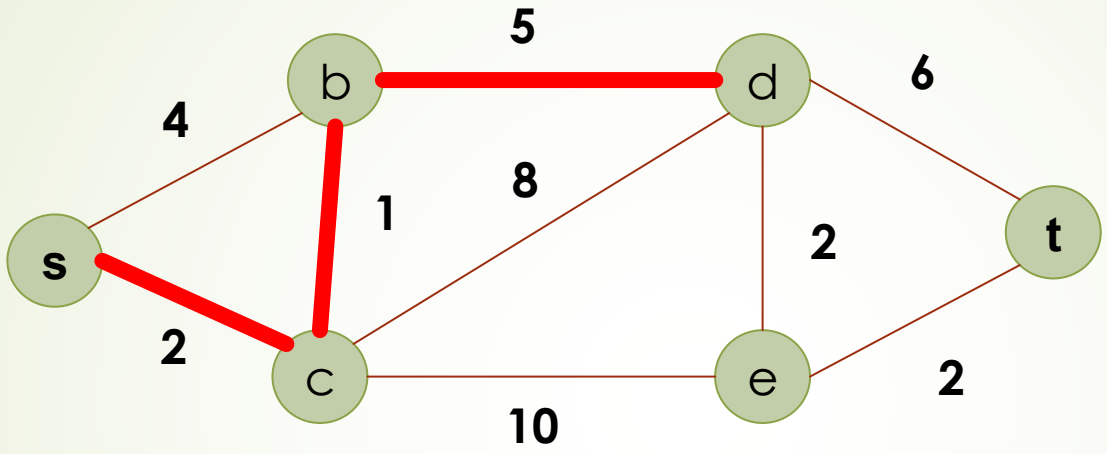


Vectores d y e son adyacentes a: b y c

VERTICE	PASO 1	PASO 2	PASO 3	PASO 4	PASO 5	PASO 6
s	(0,s)	X	X	X	X	X
b	(4,s)	(3,c)	(3,c)	X	X	X
c	(2,s)	(2,s)	X	X	X	X
d	oo	(10,c)	(8,b)	(8,b)	X	X
e	oo	(12,c)	(12,c)			
t	oo	oo	oo			

Ejemplo:
Calcular el camino mas corto entre s y t

- Vertice e referido por d para llegar a s 2+1+5+2=10 , por ser menor a 12, se registra

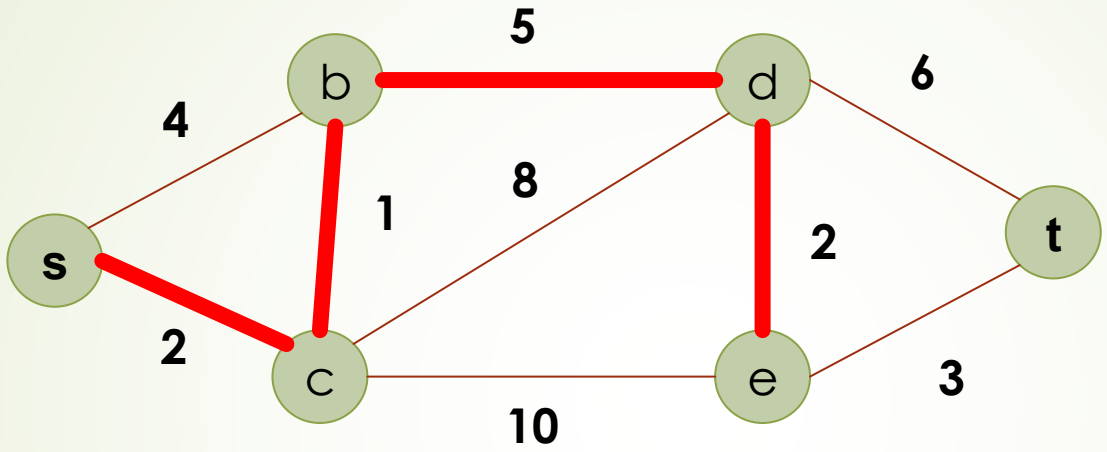


Vectores d y e son adyacentes a: b y c

VERTICE	PASO 1	PASO 2	PASO 3	PASO 4	PASO 5	PASO 6
s	(0,s)	X	X	X	X	X
b	(4,s)	(3,c)	(3,c)	X	X	X
c	(2,s)	(2,s)	X	X	X	X
d	oo	(10,c)	(8,b)	(8,b)	X	X
e	oo	(12,c)	(12,c)	(10,d)		
t	oo	oo	oo	(14,d)		

Ejemplo:
Calcular el camino mas corto entre s y t

- Entre 10 y 14 el qu3e menos pesa es 10, se traslada y se marcan los demás
- Se coloca como definitiva

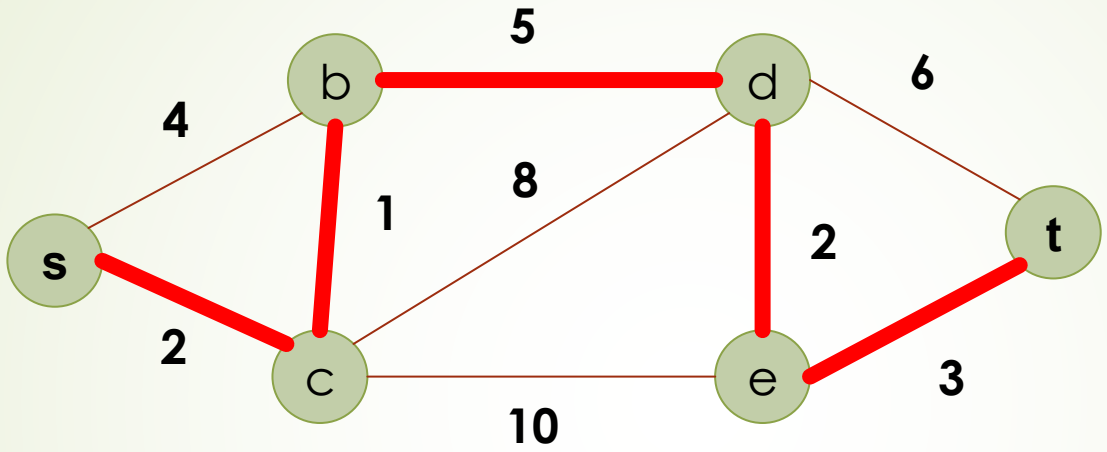


Vectores d y e son adyacentes
a: b y c

VERTICE	PASO 1	PASO 2	PASO 3	PASO 4	PASO 5	PASO 6
s	(0,s)	X	X	X	X	X
b	(4,s)	(3,c)	(3,c)	X	X	X
c	(2,s)	(2,s)	X	X	X	X
d	oo	(10,c)	(8,b)	(8,b)	X	X
e	oo	(12,c)	(12,c)	(10,d)	(10,d)	X
t	oo	oo	oo	(14,d)	(13,e)	

Ejemplo:
Calcular el camino mas corto entre s y t

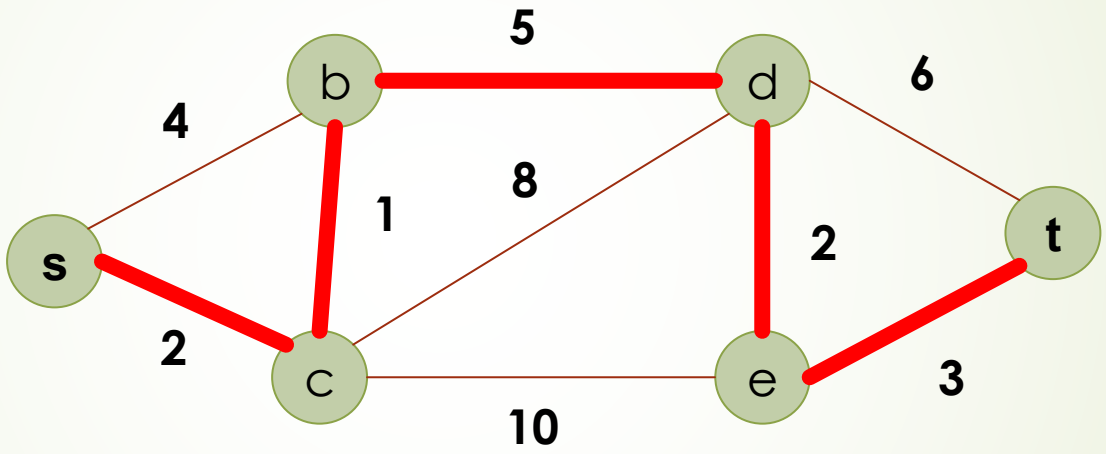
- Entre 10 y 14 el que menos pesa es 10, se traslada y se marcan los demás
- Se coloca como definitiva



Vectores d y e son adyacentes
a: b y c

VERTICE	PASO 1	PASO 2	PASO 3	PASO 4	PASO 5	PASO 6
s	(0,s)	X	X	X	X	X
b	(4,s)	(3,c)	(3,c)	X	X	X
c	(2,s)	(2,s)	X	X	X	X
d	oo	(10,c)	(8,b)	(8,b)	X	X
e	oo	(12,c)	(12,c)	(10,d)	(10,d)	X
t	oo	oo	oo	(14,d)	(13,e)	(13,e)

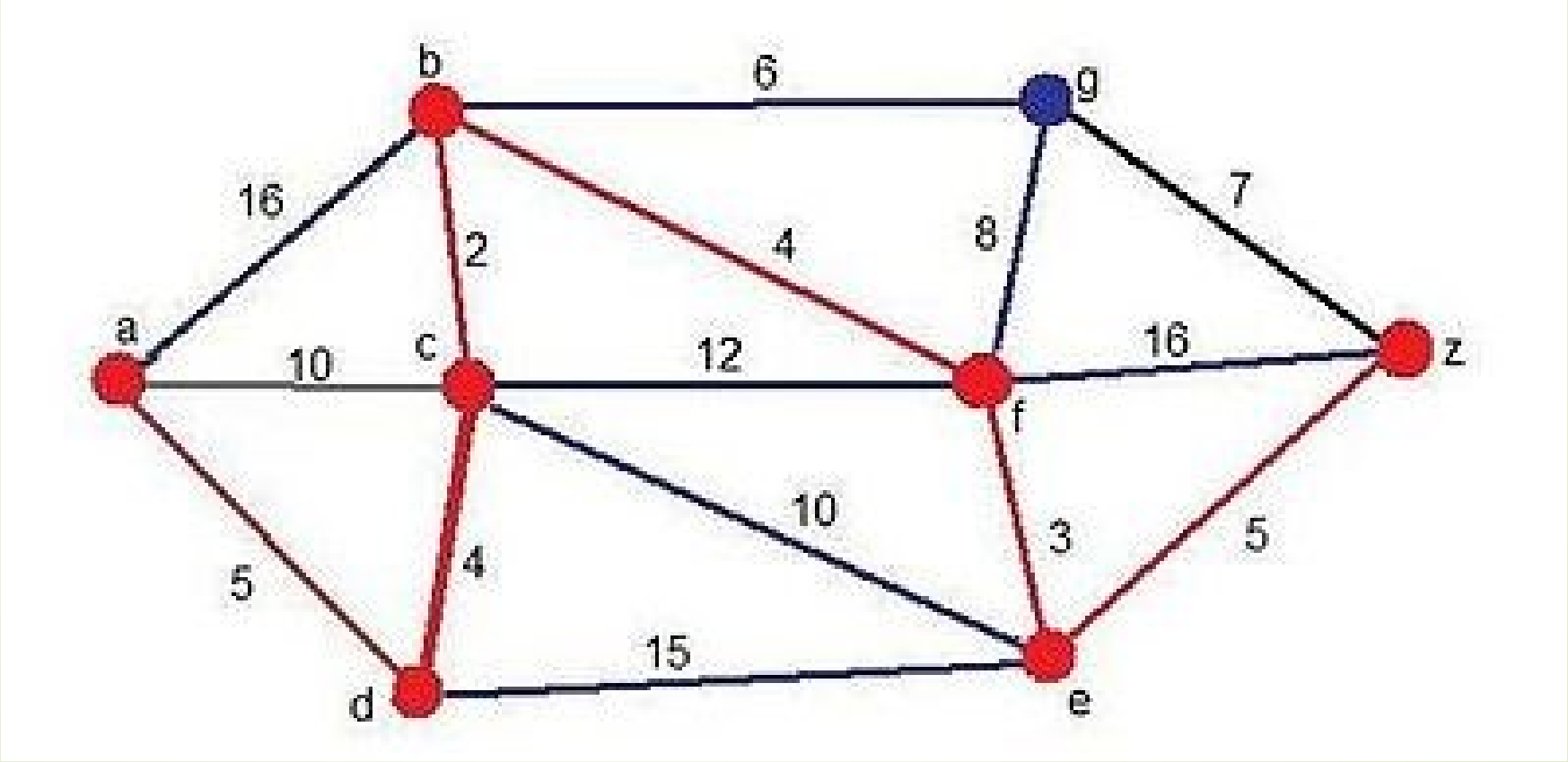
Resultado



- El peso minimo es 13, camino minimo de s a t

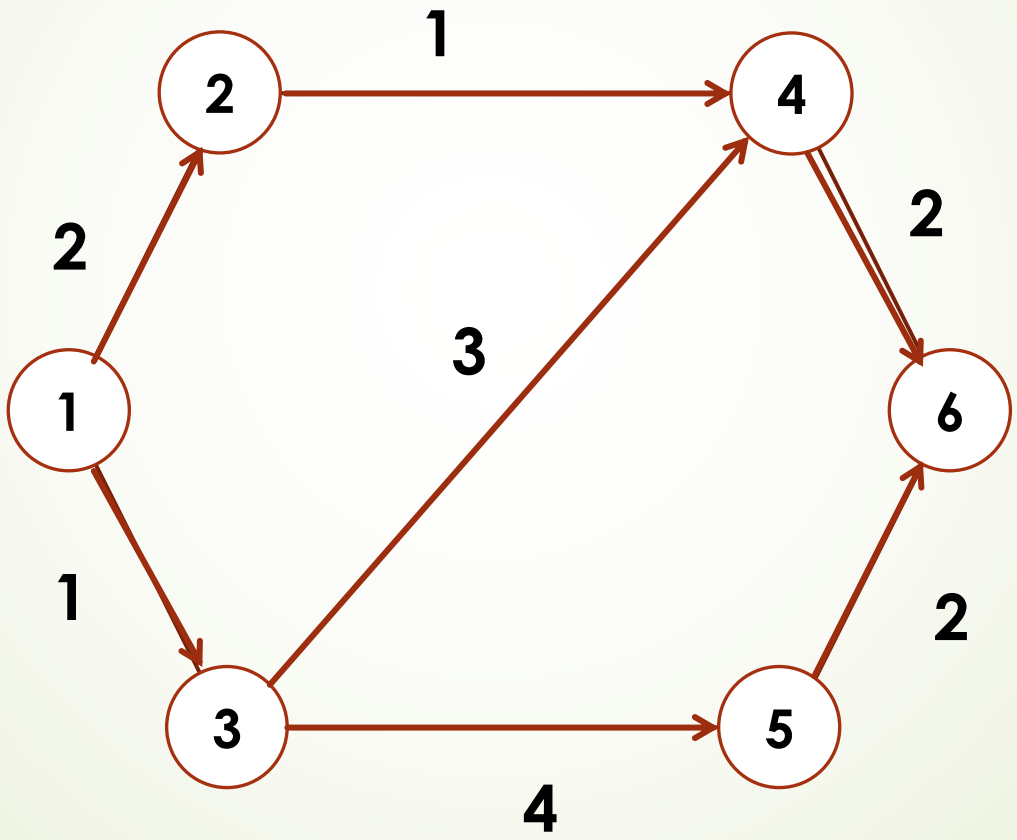
Ejercicio
Aplicando el algoritmo de Dijkstra

- Obtener el camino y distancia mínima entre el vértice a y el vértice z



Algoritmo de Dijstra con grafos Dirigidos - Ejercicio

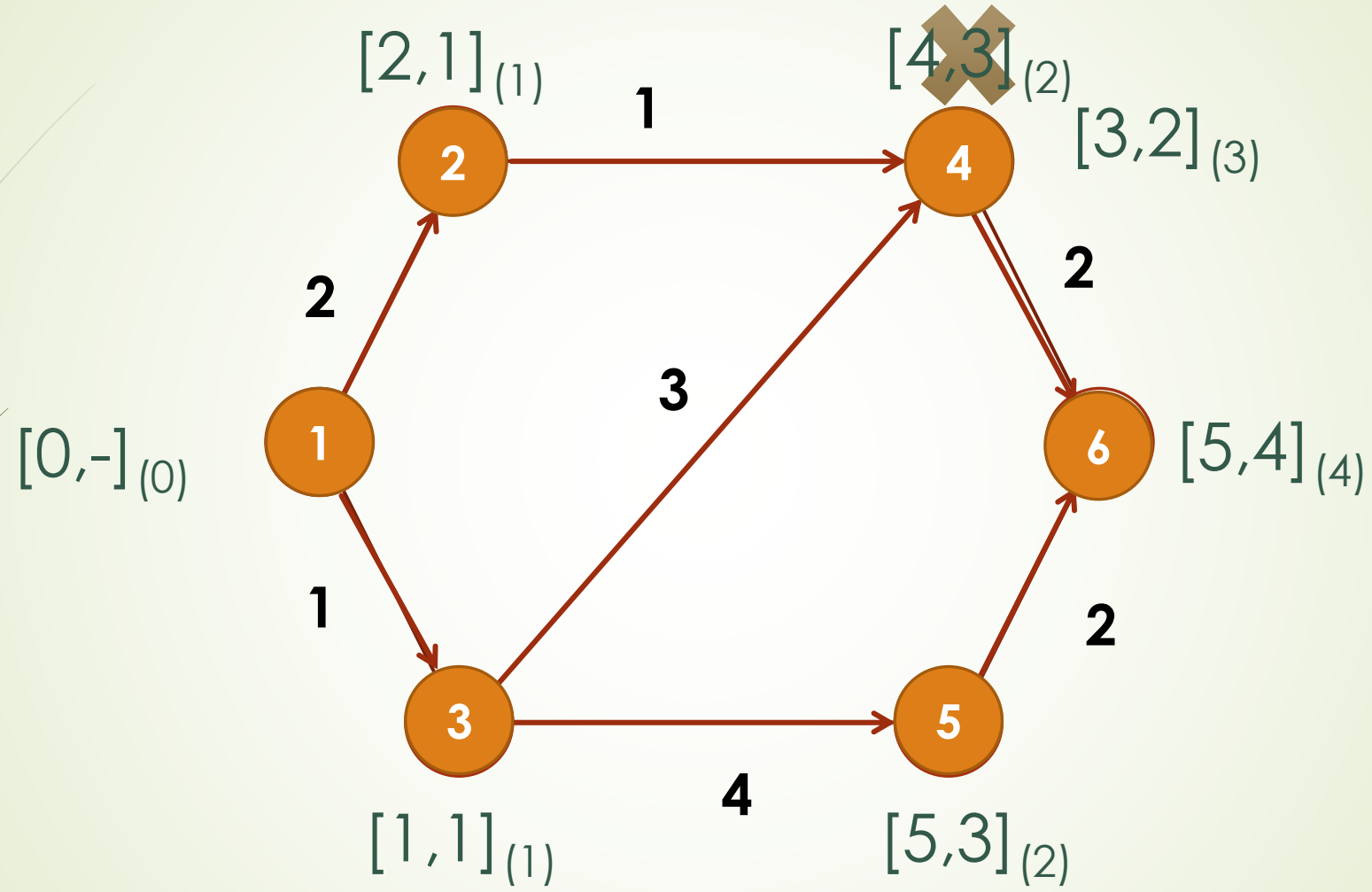
- Calcular la ruta y la distancia mas corta desde el vertice 1 al vertice 6 aplicando el método de Dijkstra



Etiquetado

$$[8,B]_{(2)}$$

Algoritmo de Dijkstra

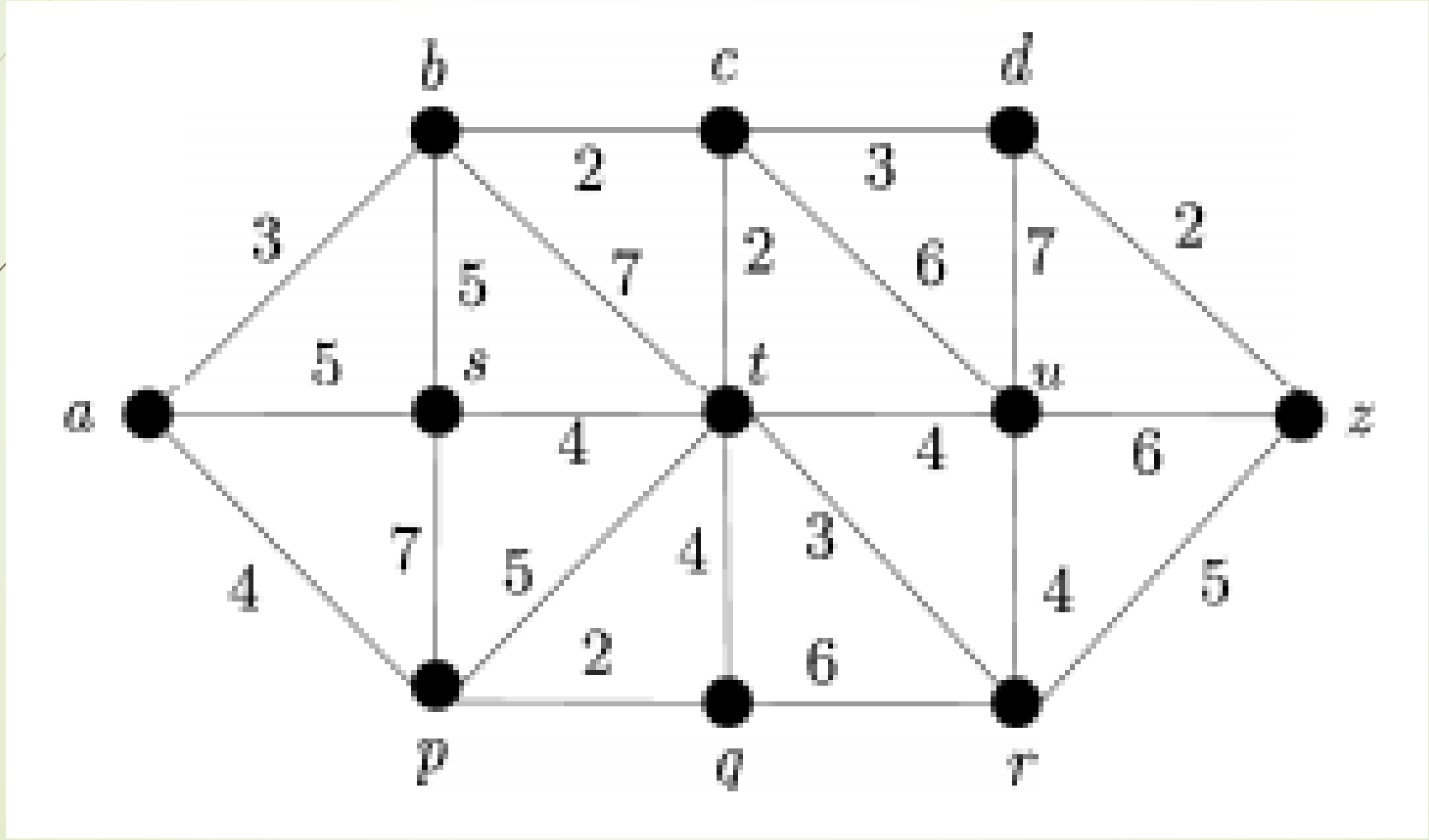


RUTA: **1 2 4 6**

Distancia: **5**

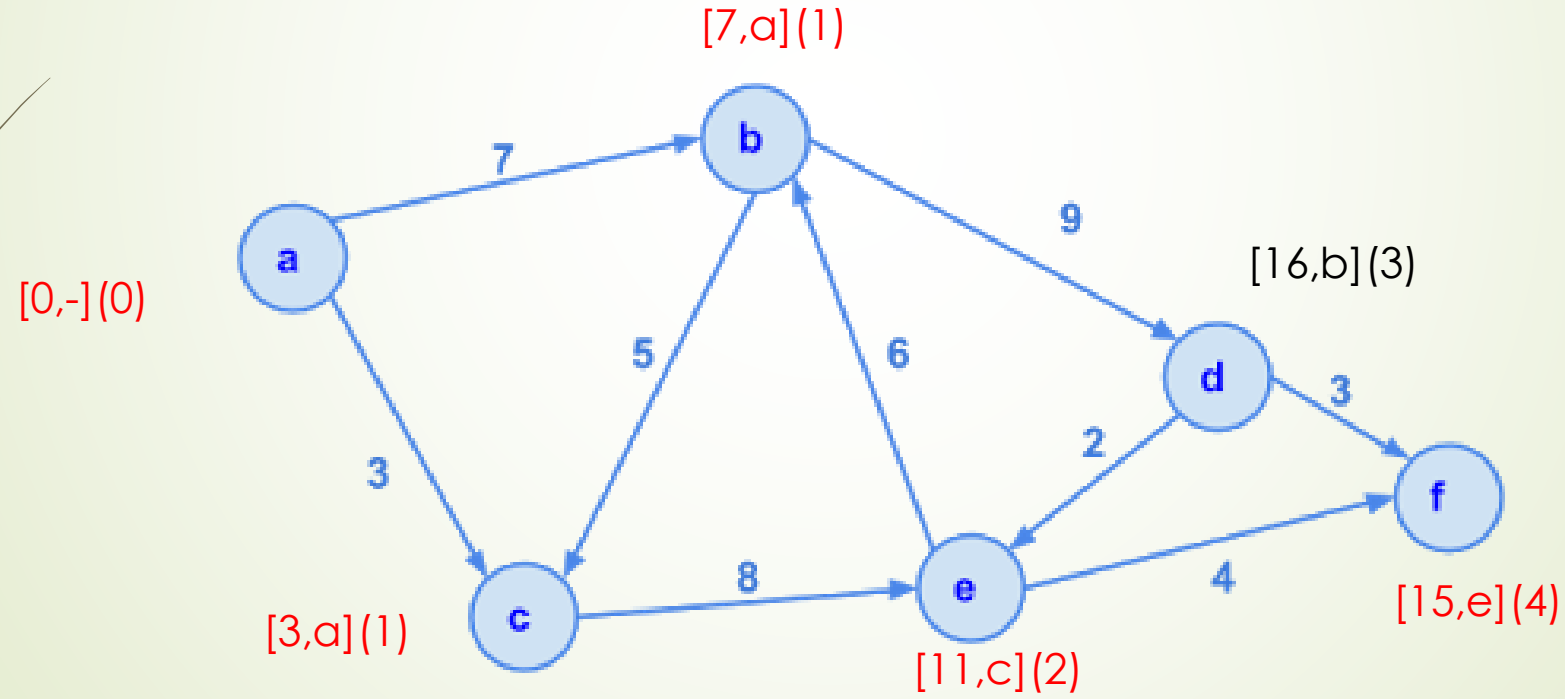
Ejercicio 1
Aplicando el algoritmo de Dijkstra

Hallar un camino de coste mínimo de a a z.



Ejercicio 2
Aplicando el algoritmo de Dijkstra

- Usando etiquetas, obtener el camino y distancia mínima entre el vértice **a** y el vértice **f**



Robert C. Prim

Matemático

Robert C. Prim es un matemático y científico de la computación Wikipedia

Fecha de nacimiento: 1921, Sweetwater, Texas, Estados Unidos

Educación: Universidad de Princeton

Vojtěch Jarník

Matemático

Vojtěch Jarník fue un matemático checo. Su principal área de trabajo fue en la teoría de los números y el análisis matemático, demostró una serie de resultados en problemas de punto de celosía. Wikipedia

Fecha de nacimiento: 22 de diciembre de 1897, Praga, República Checa

Fecha de la muerte: 22 de septiembre de 1970, Praga, República Checa

Algoritmo de Prim

algoritmo DJP o algoritmo de Jarnik

Algoritmo de Prim

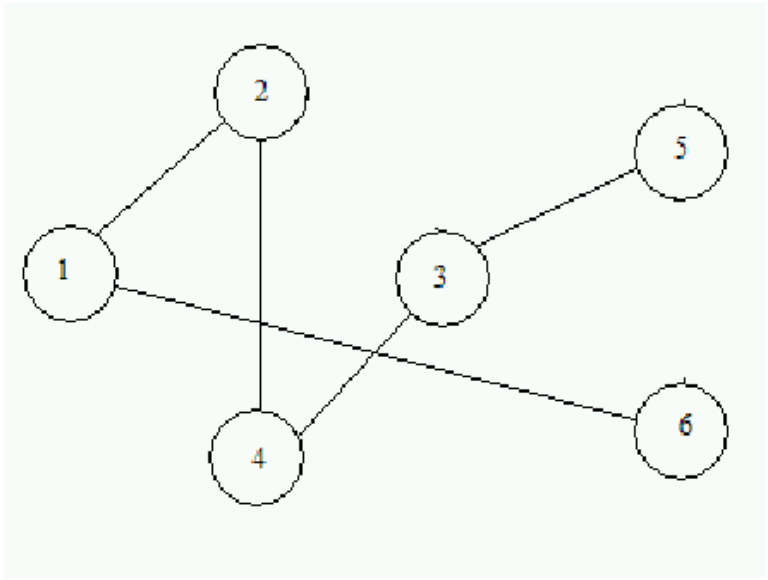
- El algoritmo encuentra un subconjunto de aristas que forman un árbol con todos los vértices
- Permite encontrar un árbol de cubrimiento mínimo
- Arbol conexo
- Grafo no Dirigido
- Aristas se encuentran etiquetadas con números enteros positivos

Arbol de Cubrimiento Minimo

- Sea G=(V,A) un grafo conexo con una función de costos definida sobre las aristas. Un árbol de cubrimiento para G es un árbol libre que conecta todos los vértices en V

Grafo Conexo

En teoría de grafos, un grafo se dice **conexo** si, para cualquier par de vértices *a* y *b* en *G*, existe al menos una trayectoria (una sucesión de vértices adyacentes que no repita vértices) de *a* a *b*.



Grafo conexo

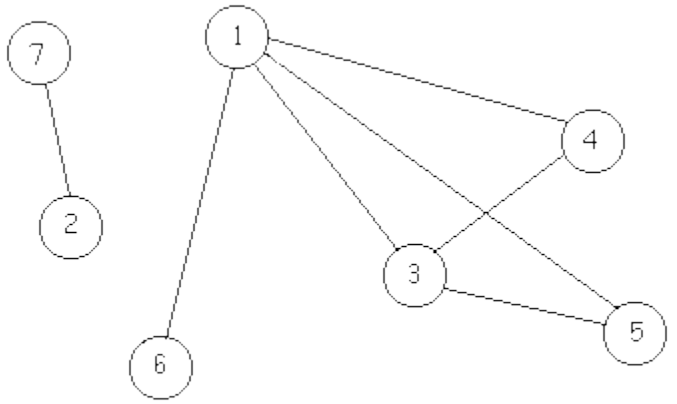
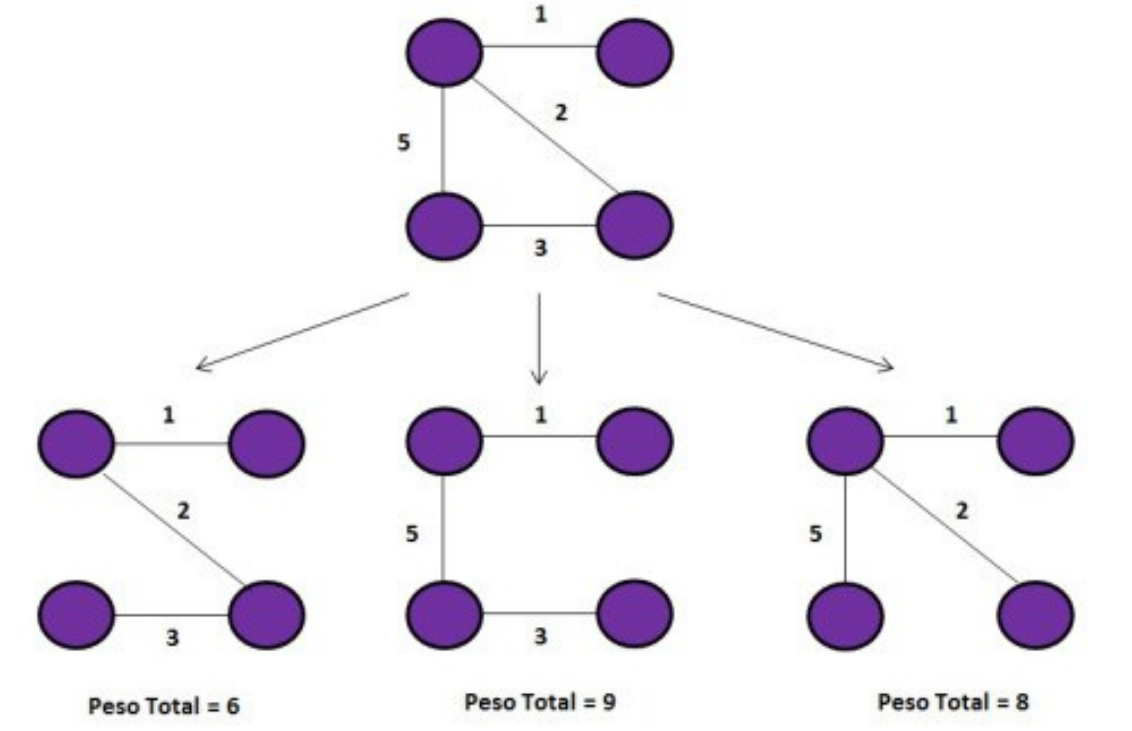


figura 2

Grafo no conexo

Árbol de Expansión Mínima

- Dado un grafo conexo, no dirigido y con pesos en las aristas, un árbol de expansión mínima es un árbol compuesto por todos los vértices y cuya suma de sus aristas es la de menor peso. Al siguiente ejemplo le agregamos pesos a sus aristas y obtenemos los arboles de expansiones siguientes:



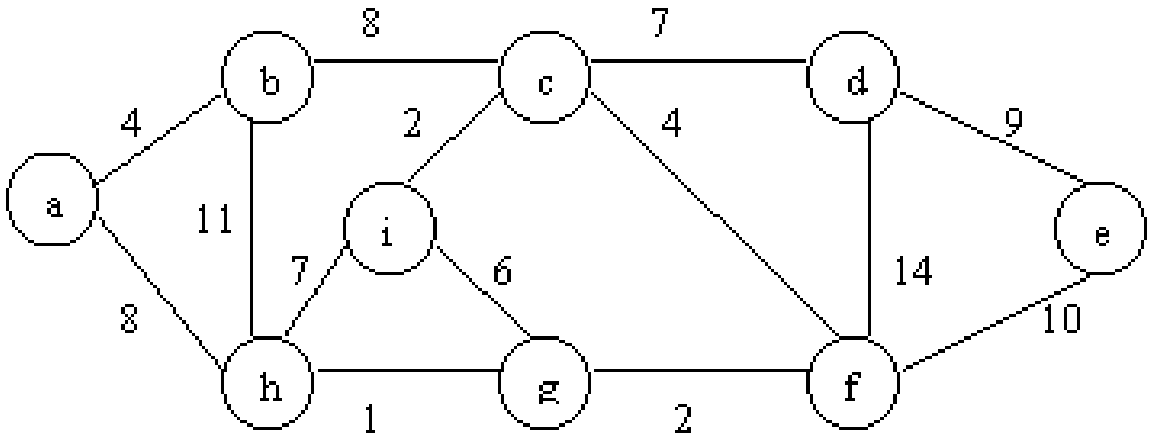
El árbol de expansión mínima seria el primer árbol de expansión cuyo peso total es 6.

El problema de hallar el Árbol de Expansión Mínima (MST) puede ser resuelto con varios algoritmos, los mas conocidos con Prim y Kruskal

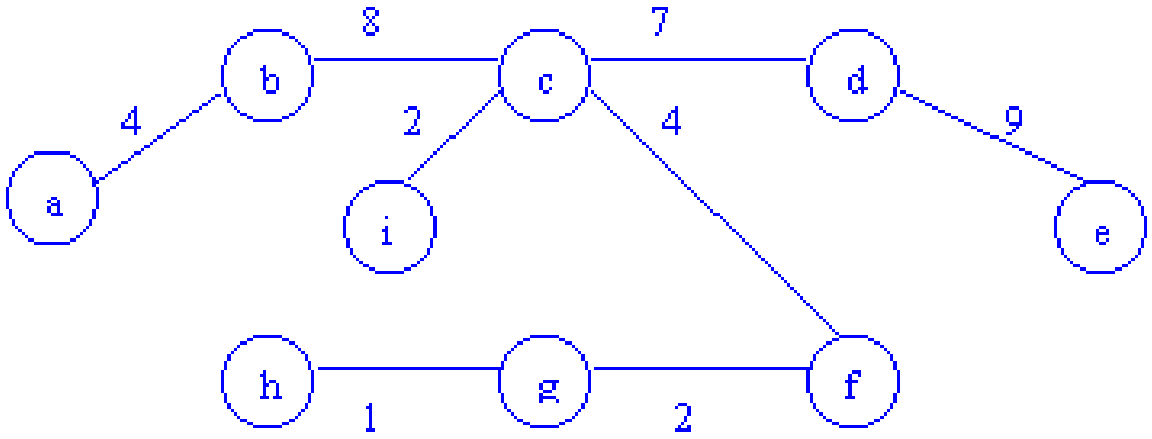
Arbol de Cubrimiento Minimo ¿Como funciona?

- El algoritmo incrementa continuamente el tamaño del árbol, comenzando por un vértice inicial al que se le va agregando vértices, cuya distancia a los anteriores debe ser mínima

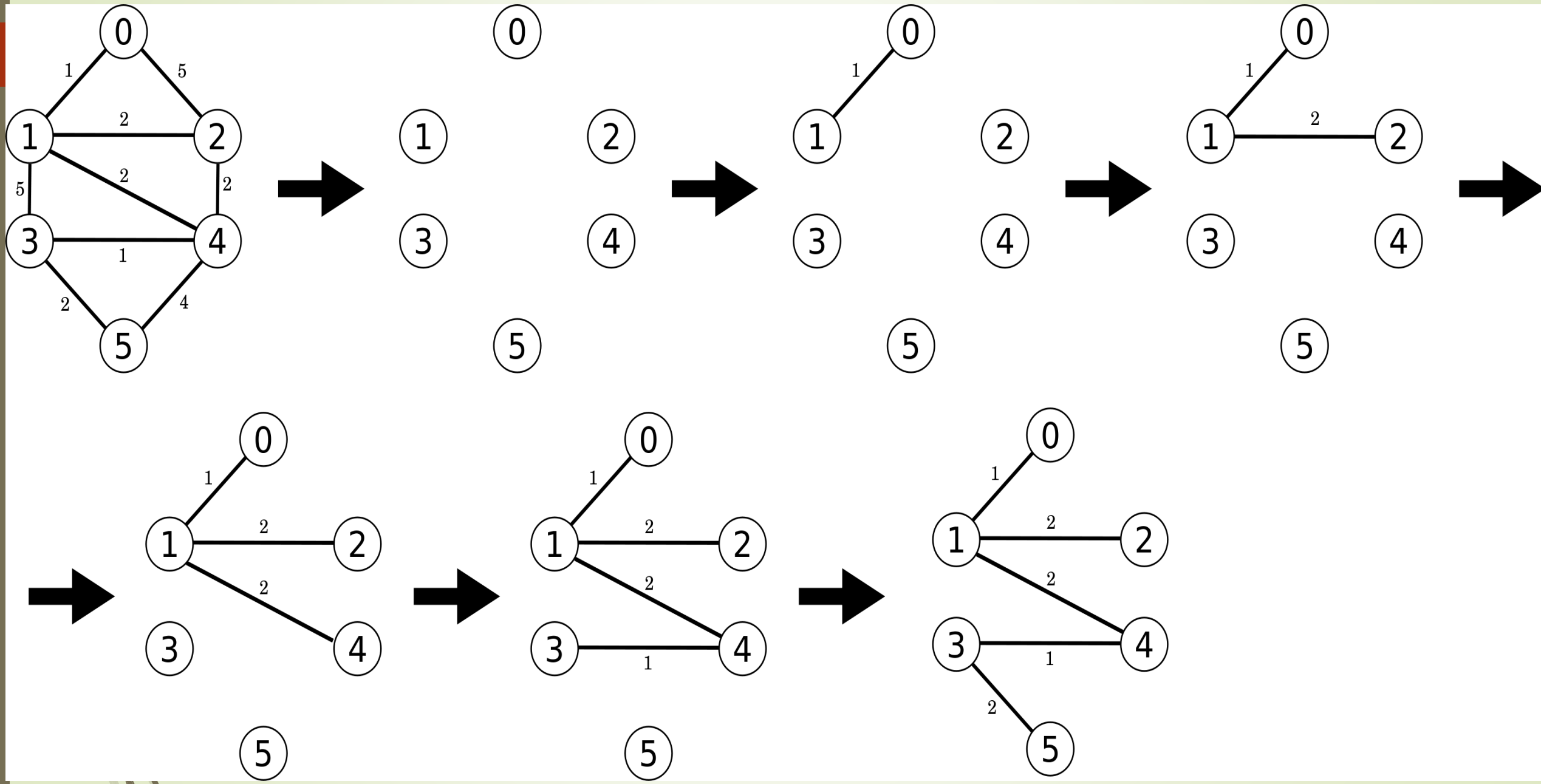
Caso Ejemplo – Algoritmo de Prim



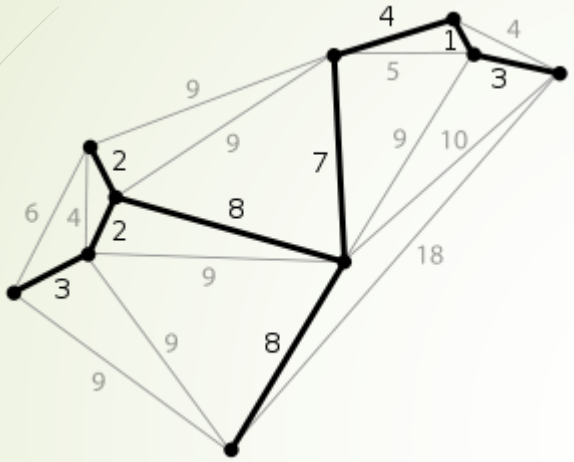
Prim



CT= 37


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

Joseph B. Kruskal (29 de enero de 1928 – Maplewood, Nueva Jersey, 19 de septiembre de 2010) fue un matemático y estadístico estadounidense.



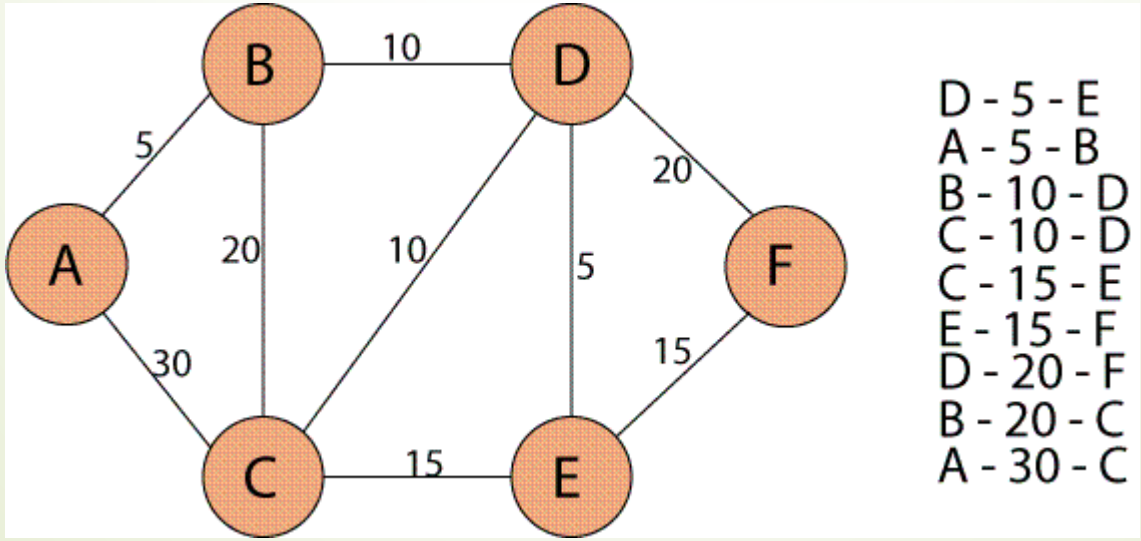
Algoritmo de Kruskal

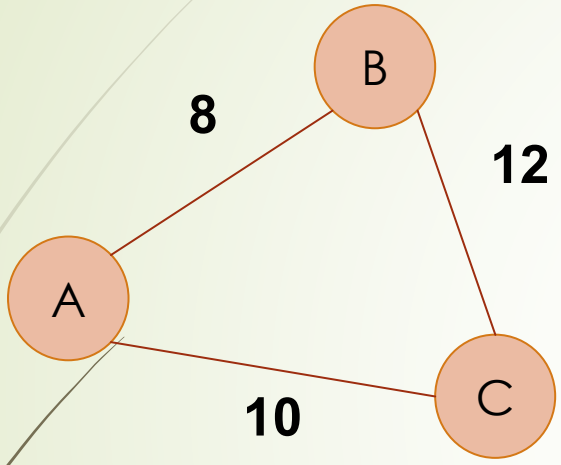
Algoritmo de Kruskal

- Este algoritmo resuelve la misma clase de problema que el algoritmo de Prim, con la única diferencia que en este caso no partimos desde ningún nodo elegido al azar

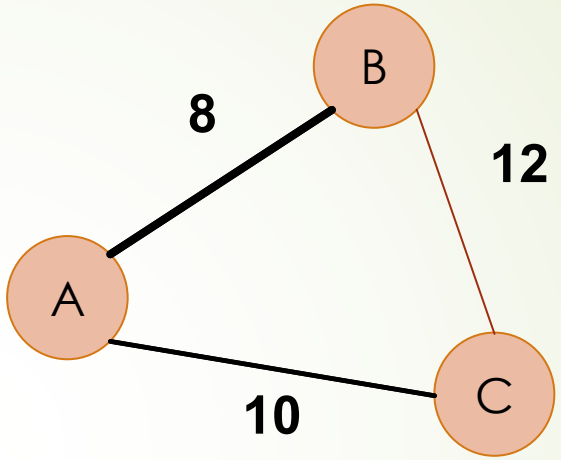
Como trabaja:

- Primeramente ordenaremos las aristas del grafo por su peso de menor a mayor. Mediante la técnica greedy. Kruskal intentará unir cada arista siempre y cuando no se forme un ciclo.
- Como hemos ordenado las aristas por peso comenzaremos con la arista de menor peso, si los vértices que contienen dicha arista no están en la misma componente conexa entonces los unimos para formar una sola componente,

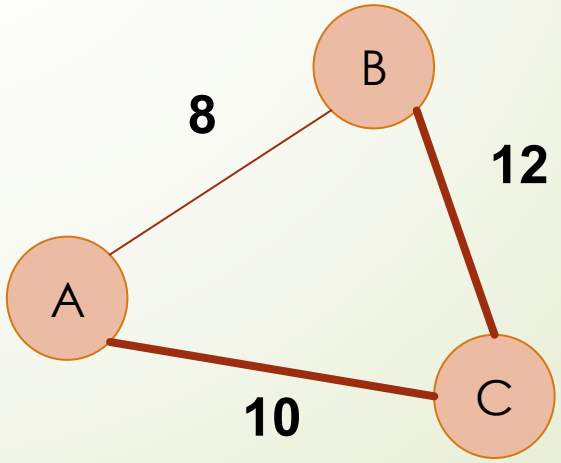




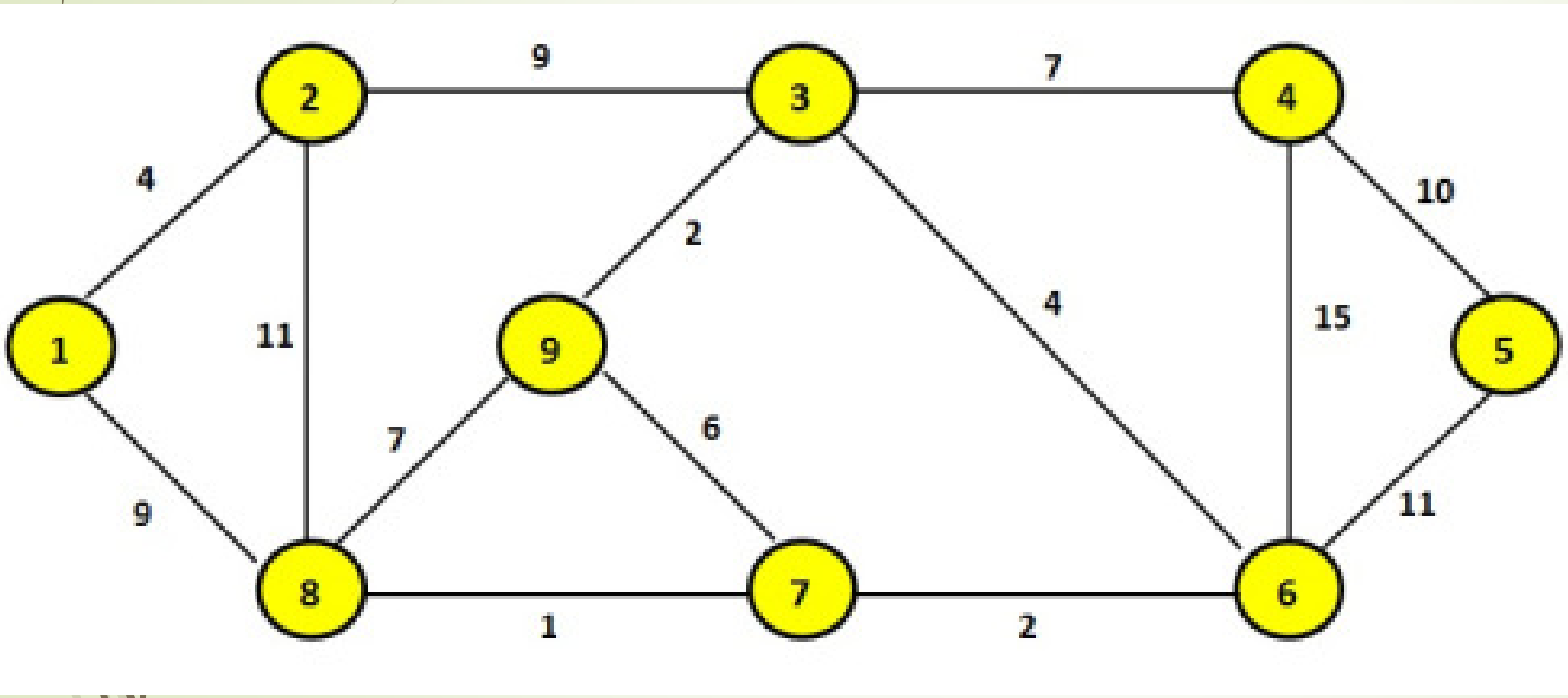
ARBOL DE MINIMO COSTE



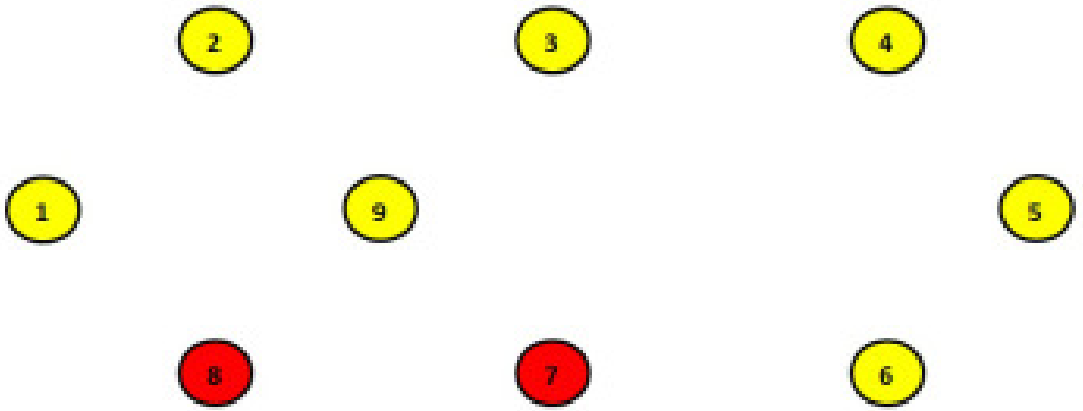
ARBOL DE MAXIMO COSTE



Arbol generador de minimo coste Algoritmo de Kruskal

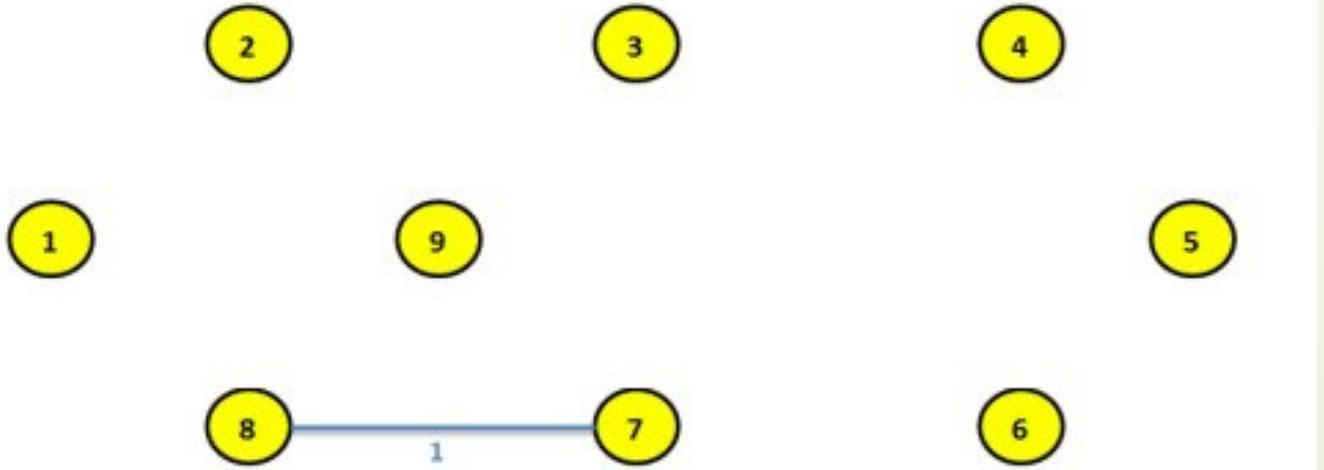


Vértices de las Aristas	Peso de la Arista
8 – 7	1
3 – 9	2
6 – 7	2
1 – 2	4
3 – 6	4
7 – 9	6
3 – 4	7
8 – 9	7
1 – 8	9
2 – 3	9
4 – 5	10
2 – 8	11
5 – 6	11
4 – 6	15

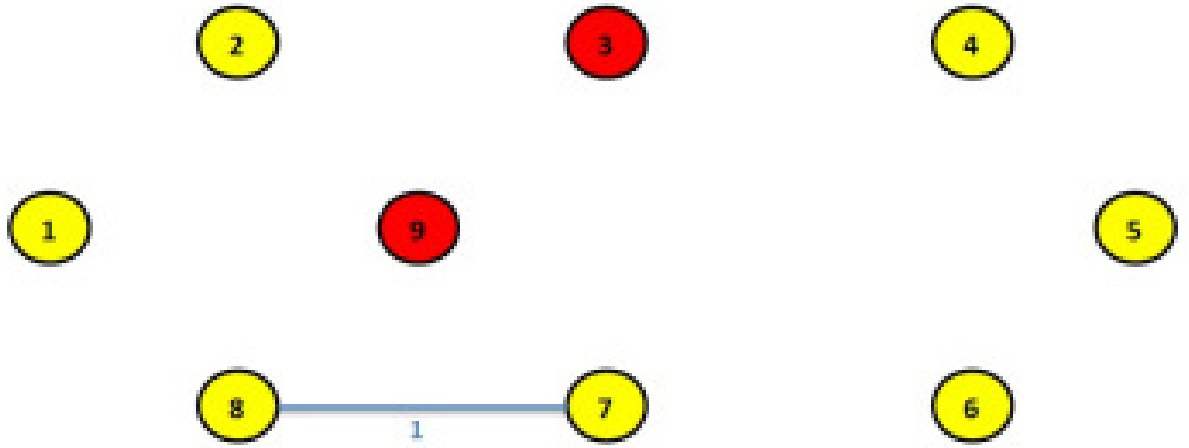


Vértices	1	2	3	4	5	6	7	8	9
Raíz	1	2	3	4	5	6	7	8	9

Vértices de las Aristas	Peso de la Arista
8 – 7	1
3 – 9	2
6 – 7	2
1 – 2	4
3 – 6	4
7 – 9	6
3 – 4	7
8 – 9	7
1 – 8	9
2 – 3	9
4 – 5	10
2 – 8	11
5 – 6	11
4 – 6	15

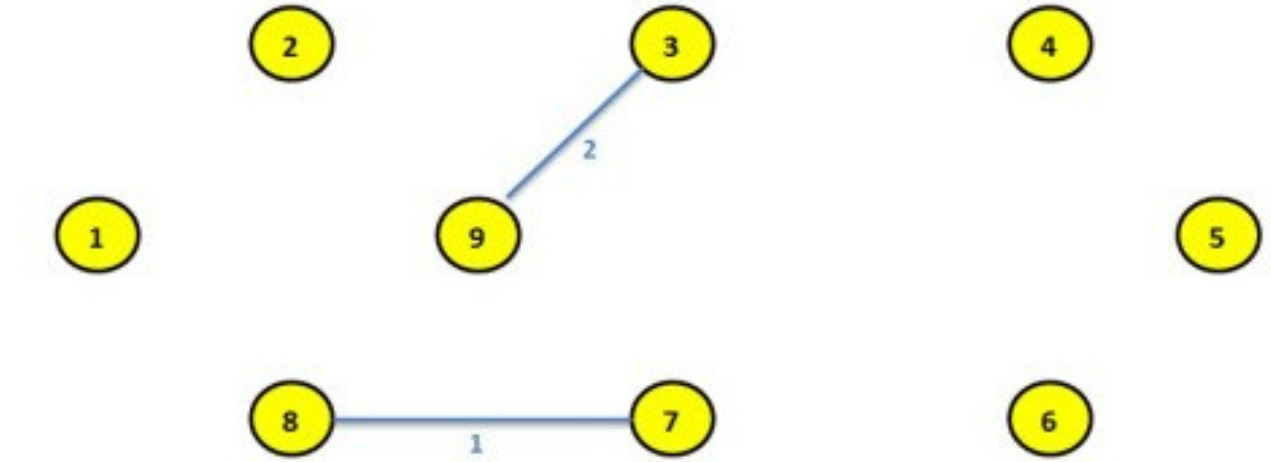


Vértices	1	2	3	4	5	6	7	8	9
Raíz	1	2	3	4	5	6	7	7	9

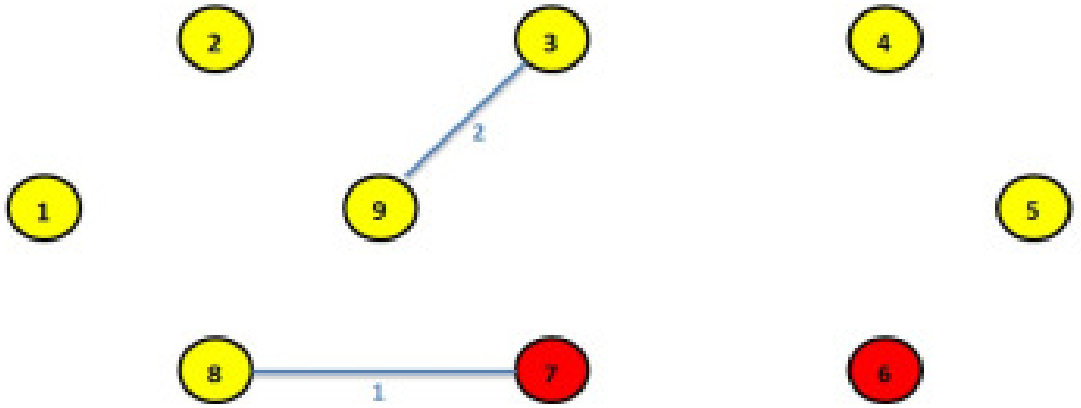


Vértices de las Aristas	Peso de la Arista
8 – 7	1
3 – 9	2
6 – 7	2
1 – 2	4
3 – 6	4
7 – 9	6
3 – 4	7
8 – 9	7
1 – 8	9
2 – 3	9
4 – 5	10
2 – 8	11
5 – 6	11
4 – 6	15

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

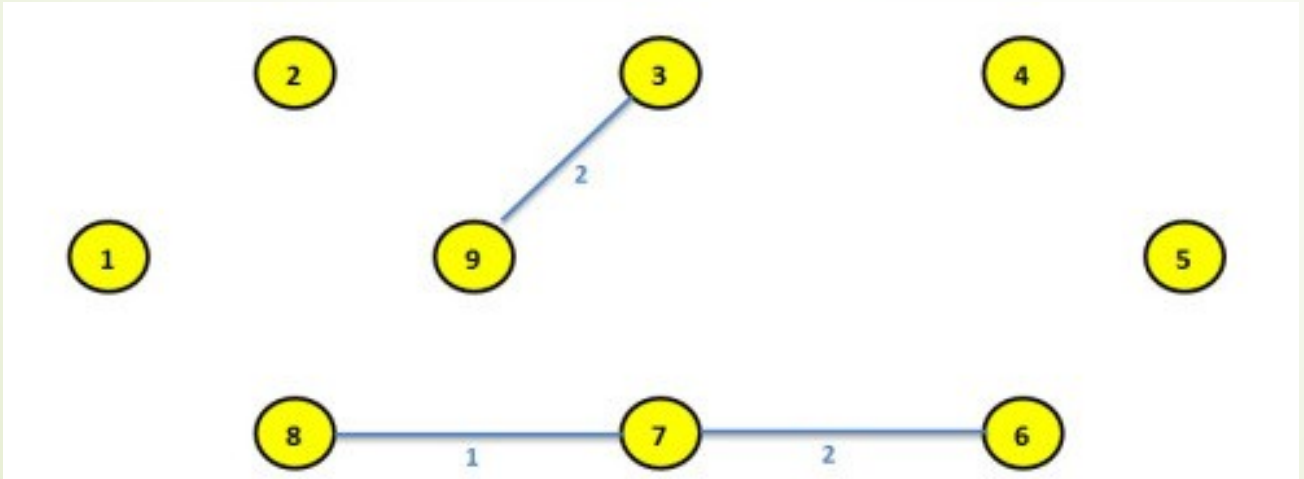


Vértices	1	2	3	4	5	6	7	8	9
Raíz	1	2	9	4	5	6	7	7	9

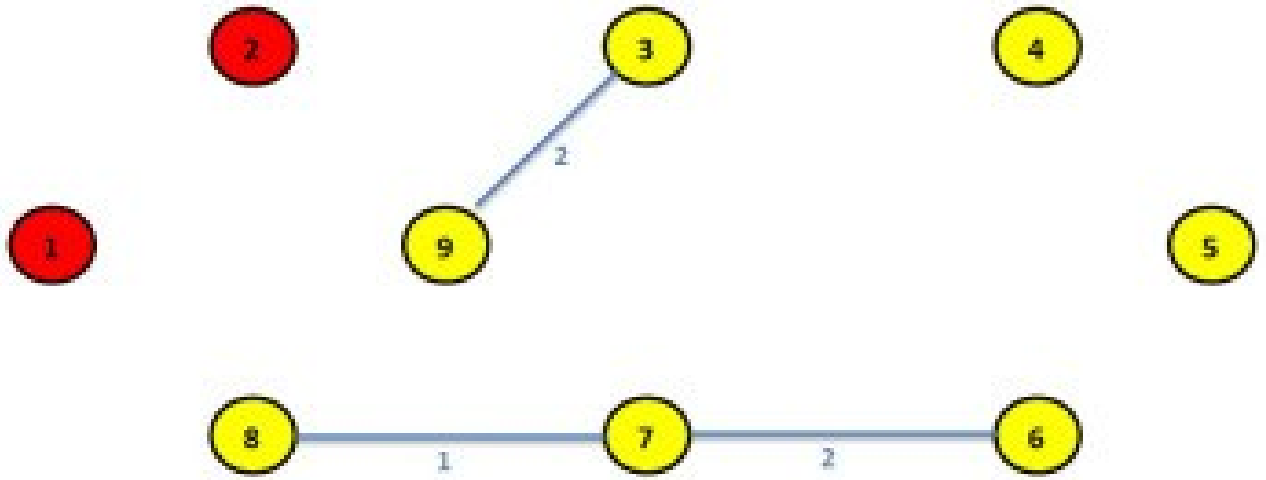


Vértices	1	2	3	4	5	6	7	8	9
Raíz	1	2	9	4	5	6	7	7	9

Vértices de las Aristas	Peso de la Arista
8 – 7	1
3 – 9	2
6 – 7	2
1 – 2	4
3 – 6	4
7 – 9	6
3 – 4	7
8 – 9	7
1 – 8	9
2 – 3	9
4 – 5	10
2 – 8	11
5 – 6	11
4 – 6	15

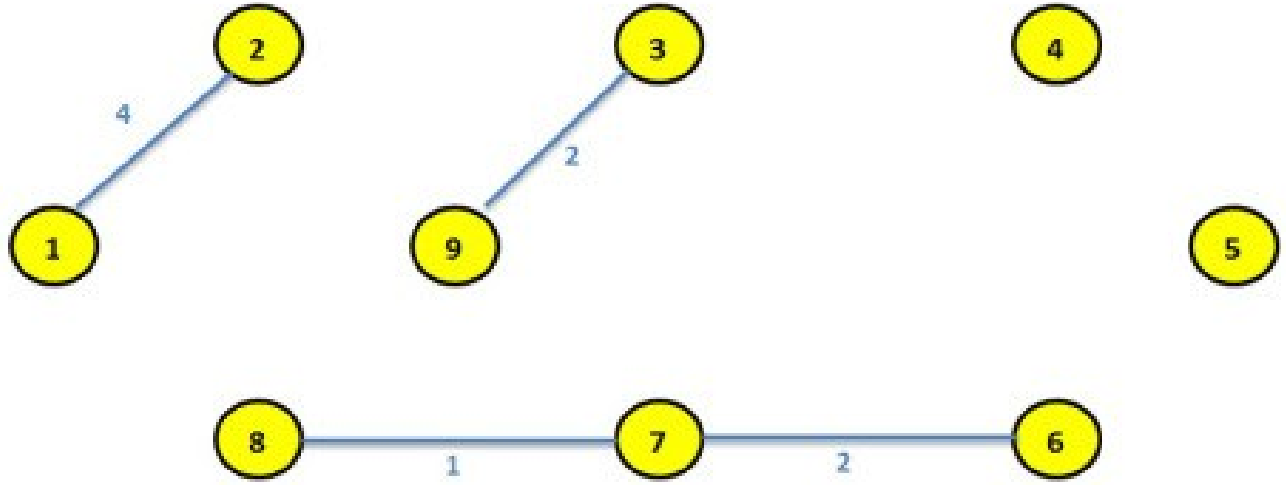


Vértices	1	2	3	4	5	6	7	8	9
Raíz	1	2	9	4	5	7	7	7	9

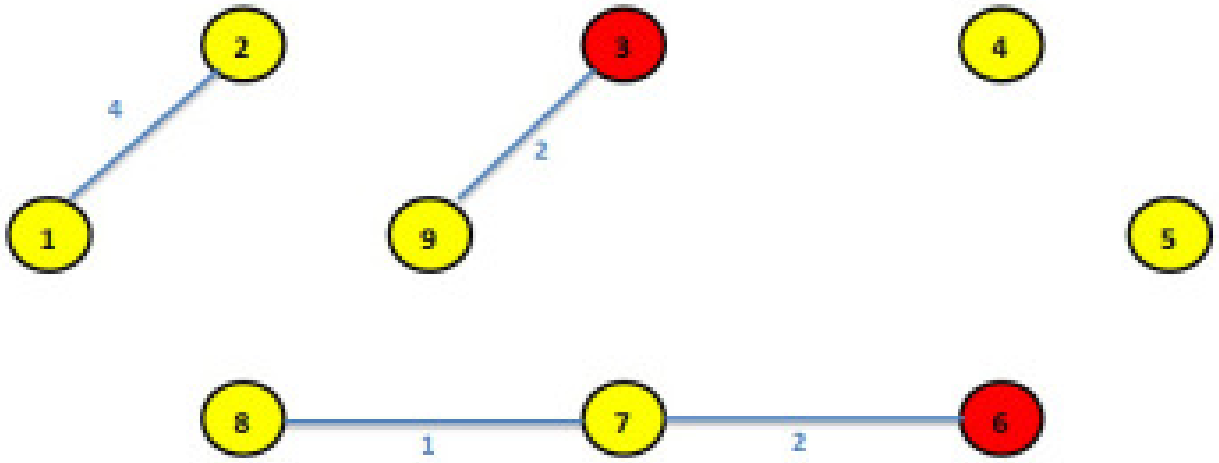


Vértices de las Aristas	Peso de la Arista
8 – 7	1
3 – 9	2
6 – 7	2
1 – 2	4
3 – 6	4
7 – 9	6
3 – 4	7
8 – 9	7
1 – 8	9
2 – 3	9
4 – 5	10
2 – 8	11
5 – 6	11
4 – 6	15

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

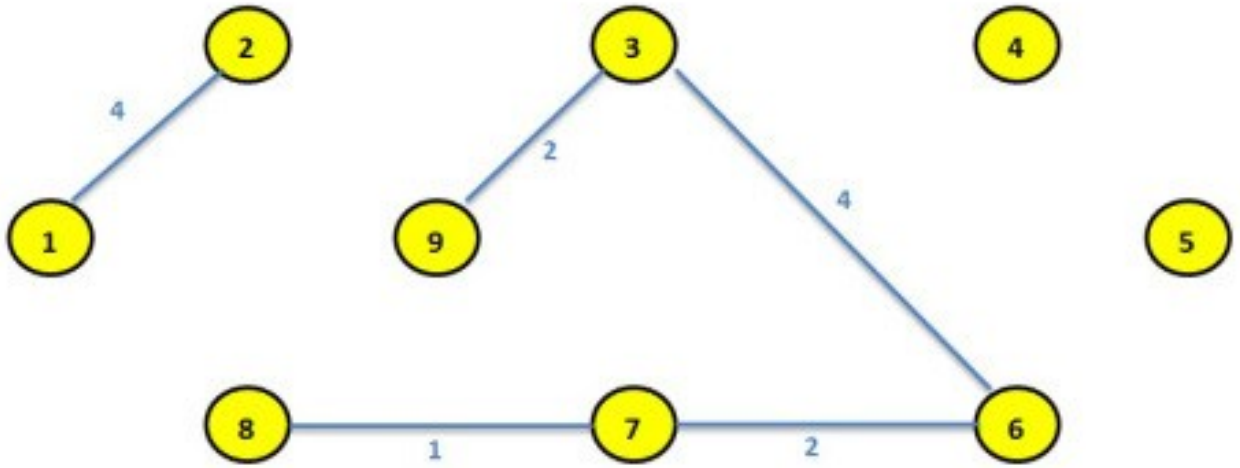


Vértices	1	2	3	4	5	6	7	8	9
Raíz	2	2	9	4	5	7	7	7	9

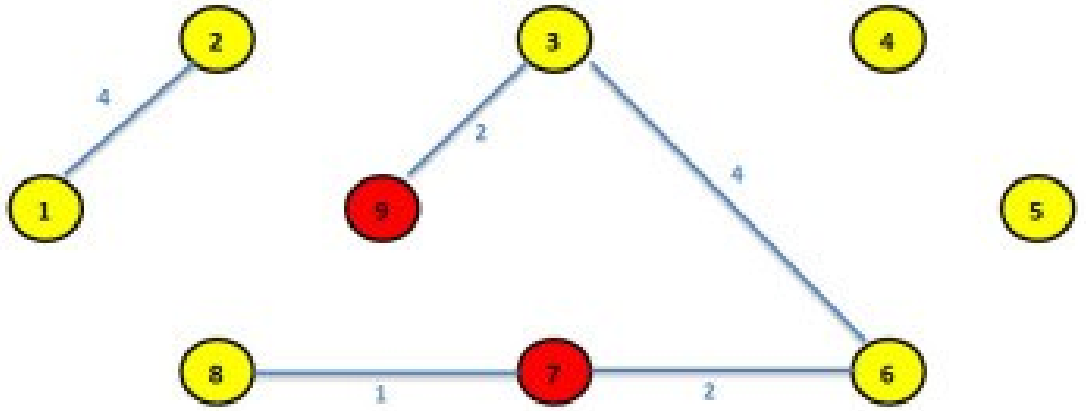


Vértices	1	2	3	4	5	6	7	8	9
Raíz	2	2	9	4	5	7	7	7	9

Vértices de las Aristas	Peso de la Arista
8 – 7	1
3 – 9	2
6 – 7	2
1 – 2	4
3 – 6	4
7 – 9	6
3 – 4	7
8 – 9	7
1 – 8	9
2 – 3	9
4 – 5	10
2 – 8	11
5 – 6	11
4 – 6	15

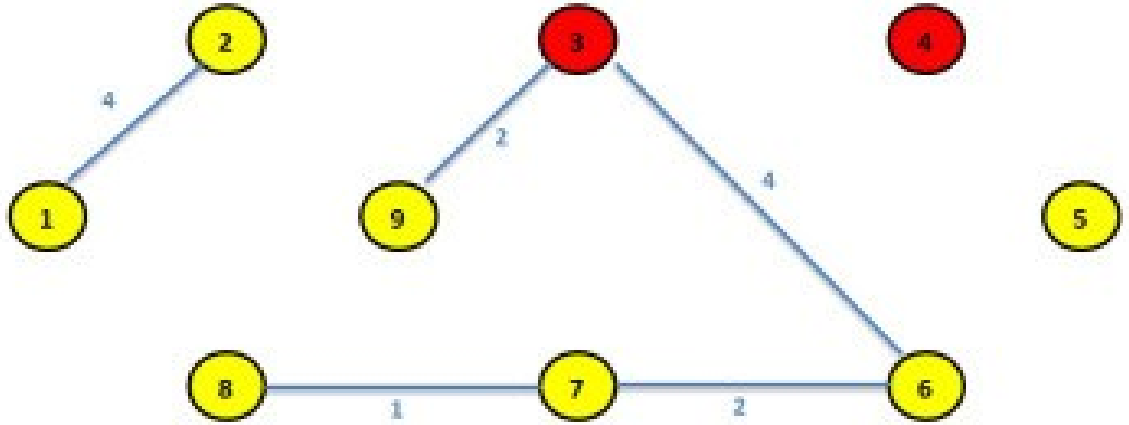


Vértices	1	2	3	4	5	6	7	8	9
Raíz	2	2	7	4	5	7	7	7	7



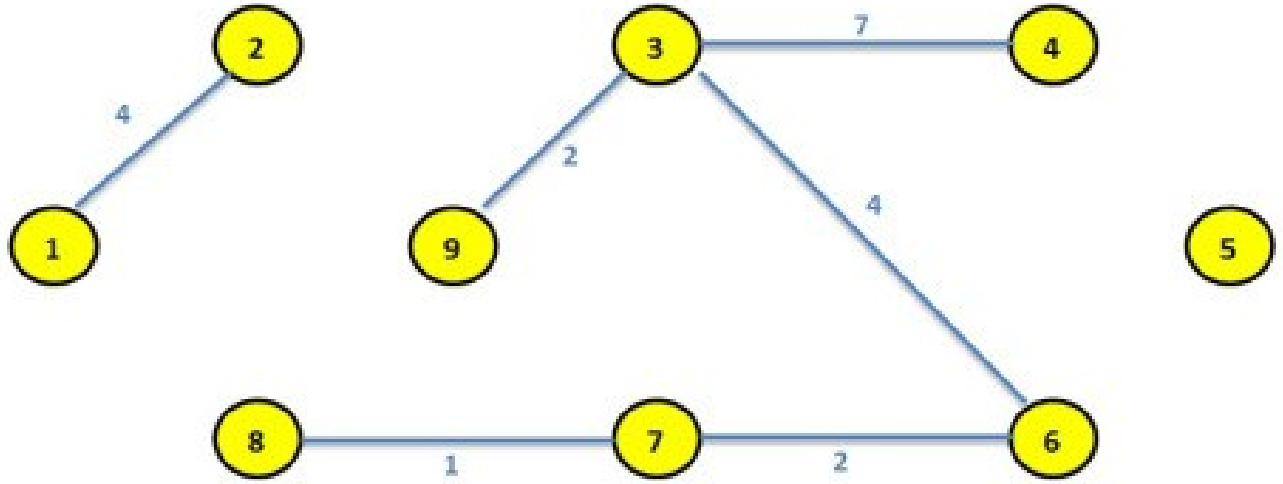
Vértices	1	2	3	4	5	6	7	8	9
Raiz	2	2	7	4	5	7	7	7	7

Vértices de las Aristas	Peso de la Arista
8 – 7	1
3 – 9	2
6 – 7	2
1 – 2	4
3 – 6	4
7 – 9	**6**
3 – 4	7
8 – 9	7
1 – 8	9
2 – 3	9
4 – 5	10
2 – 8	11
5 – 6	11
4 – 6	15

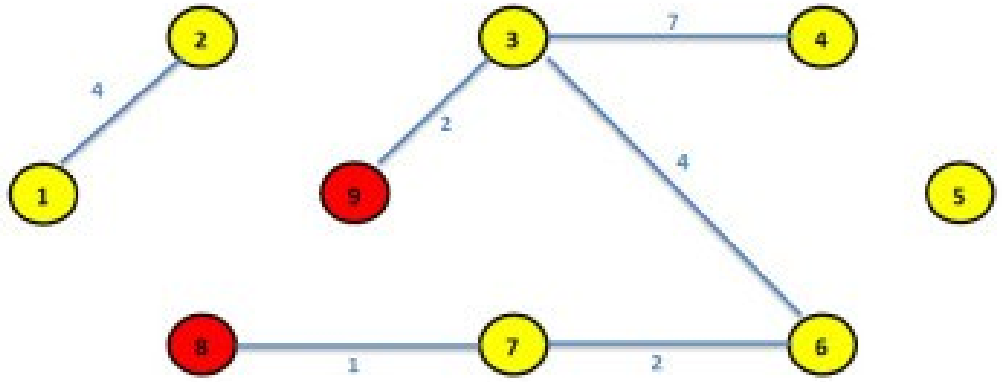


Vértices	1	2	3	4	5	6	7	8	9
Raiz	2	2	7	4	5	7	7	7	7

Vértices de las Aristas	Peso de la Arista
8 – 7	1
3 – 9	2
6 – 7	2
1 – 2	4
3 – 6	4
7 – 9	6
3 – 4	**7**
8 – 9	7
1 – 8	9
2 – 3	9
4 – 5	10
2 – 8	11
5 – 6	11
4 – 6	15

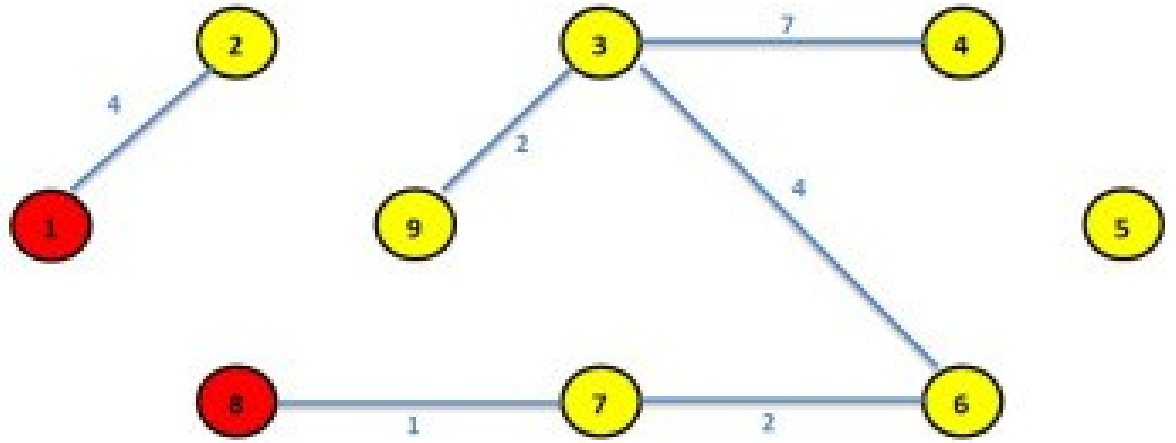


Vértices	1	2	3	4	5	6	7	8	9
Raíz	2	2	7	7	5	7	7	7	7



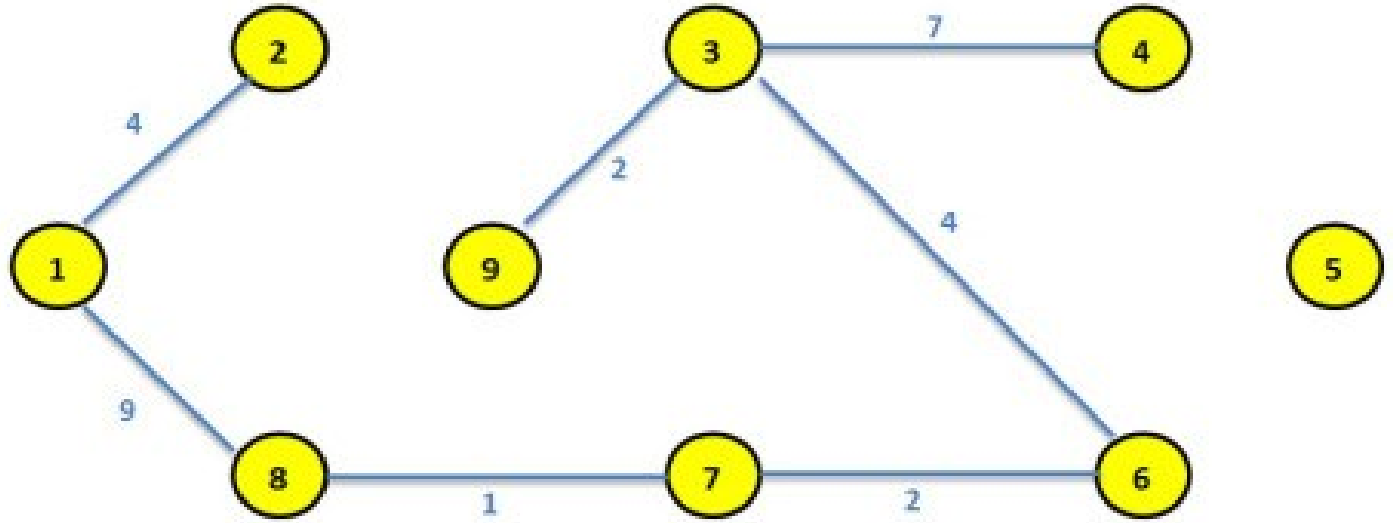
Vértices	1	2	3	4	5	6	7	8	9
Raíz	2	2	7	7	5	7	7	7	7

Vértices de las Aristas	Peso de la Arista
8 – 7	1
3 – 9	2
6 – 7	2
1 – 2	4
3 – 6	4
7 – 9	6
3 – 4	7
8 – 9	7
1 – 8	9
2 – 3	9
4 – 5	10
2 – 8	11
5 – 6	11
4 – 6	15

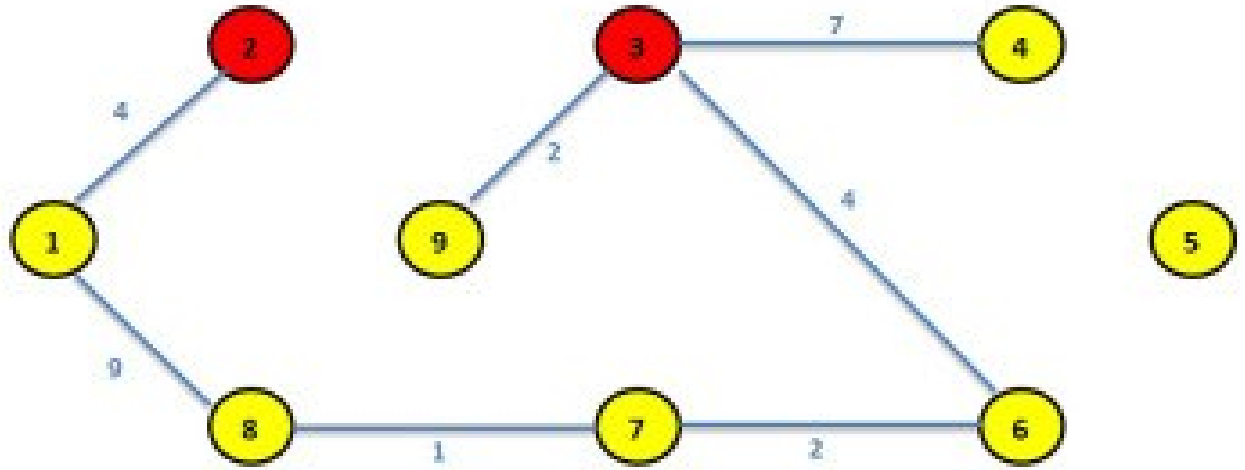


Vértices de las Aristas	Peso de la Arista
8 – 7	1
3 – 9	2
6 – 7	2
1 – 2	4
3 – 6	4
7 – 9	6
3 – 4	7
8 – 9	7
1 – 8	**9**
2 – 3	9
4 – 5	10
2 – 8	11
5 – 6	11
4 – 6	15

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

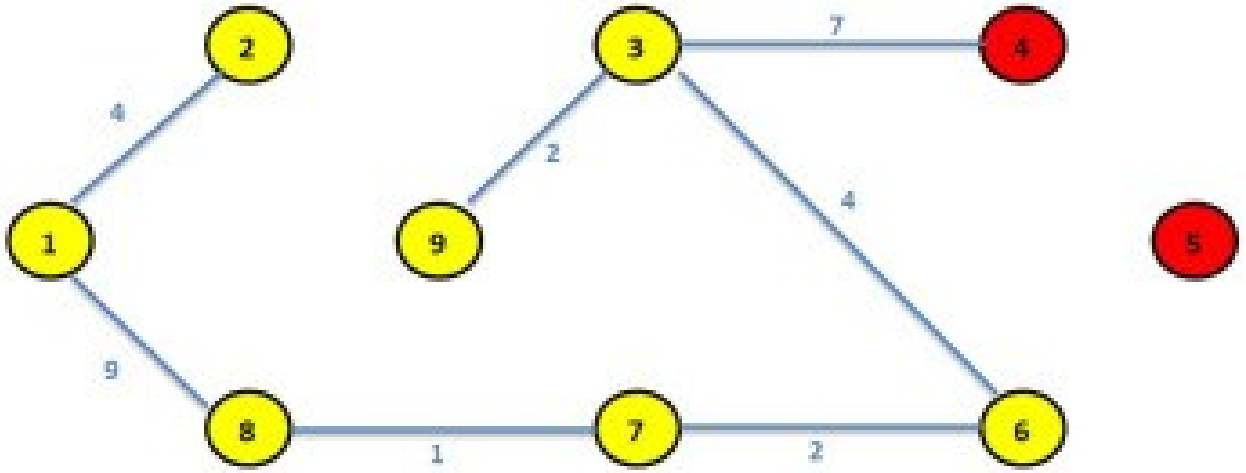


Vértices	1	2	3	4	5	6	7	8	9
Raíz	7	7	7	7	5	7	7	7	7



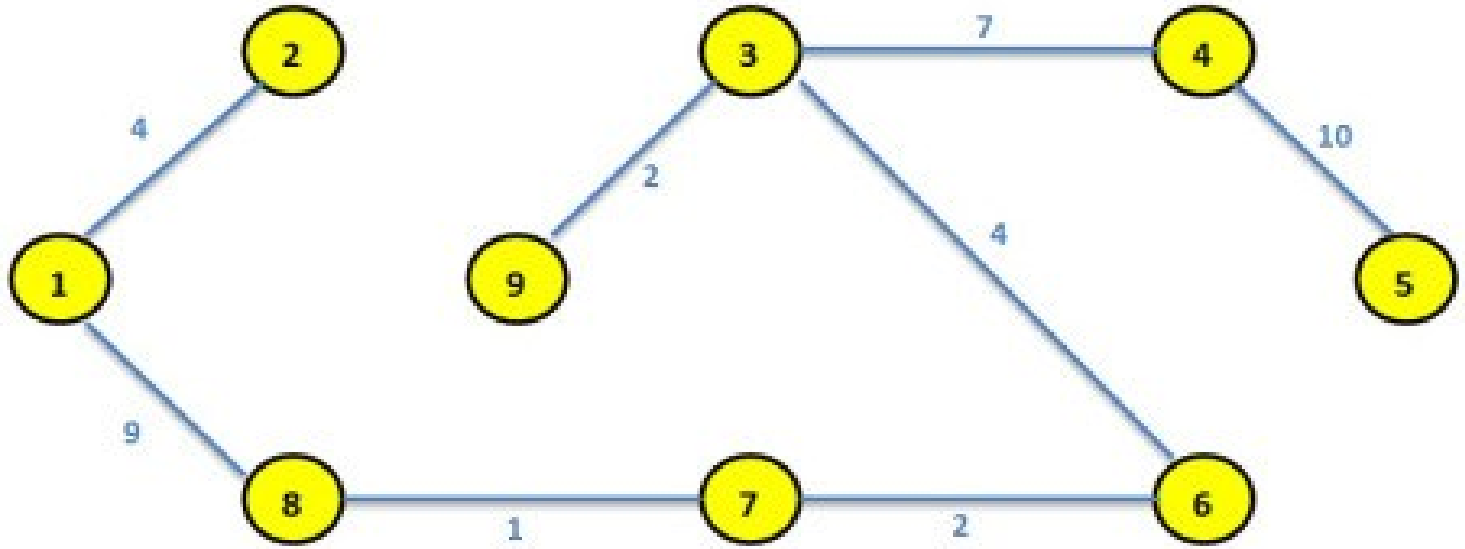
Vértices	1	2	3	4	5	6	7	8	9
Raíz	7	7	7	7	5	7	7	7	7

Vértices de las Aristas	Peso de la Arista
8 – 7	1
3 – 9	2
6 – 7	2
1 – 2	4
3 – 6	4
7 – 9	6
3 – 4	7
8 – 9	7
1 – 8	9
2 – 3	**9**
4 – 5	10
2 – 8	11
5 – 6	11
4 – 6	15

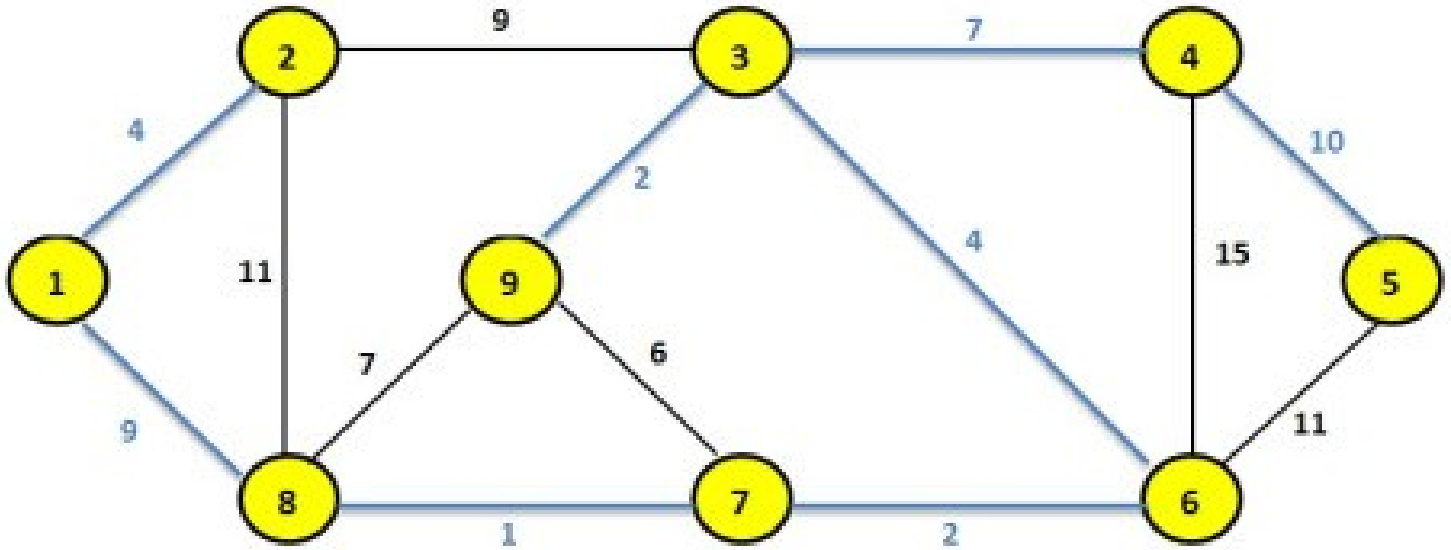


Vértices	1	2	3	4	5	6	7	8	9
Raíz	7	7	7	7	5	7	7	7	7

Vértices de las Aristas	Peso de la Arista
8 – 7	1
3 – 9	2
6 – 7	2
1 – 2	4
3 – 6	4
7 – 9	6
3 – 4	7
8 – 9	7
1 – 8	9
2 – 3	9
4 – 5	**10**
2 – 8	11
5 – 6	11
4 – 6	15



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

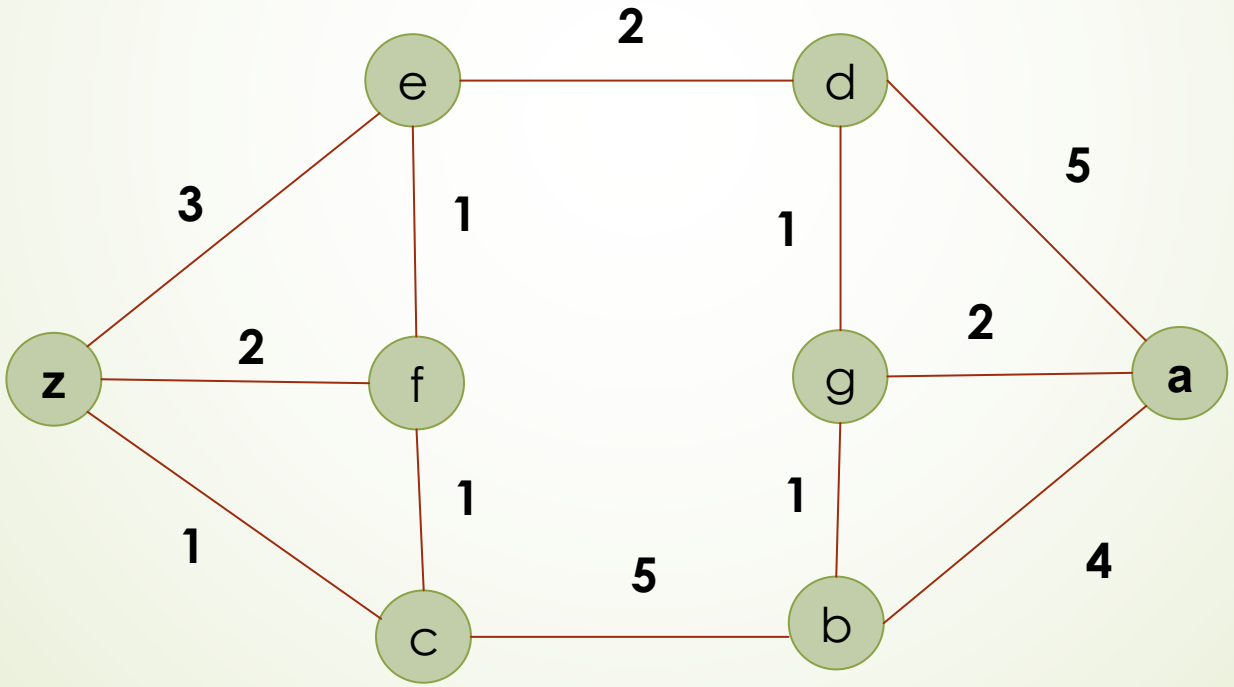


GRAFOS

Ejercicios

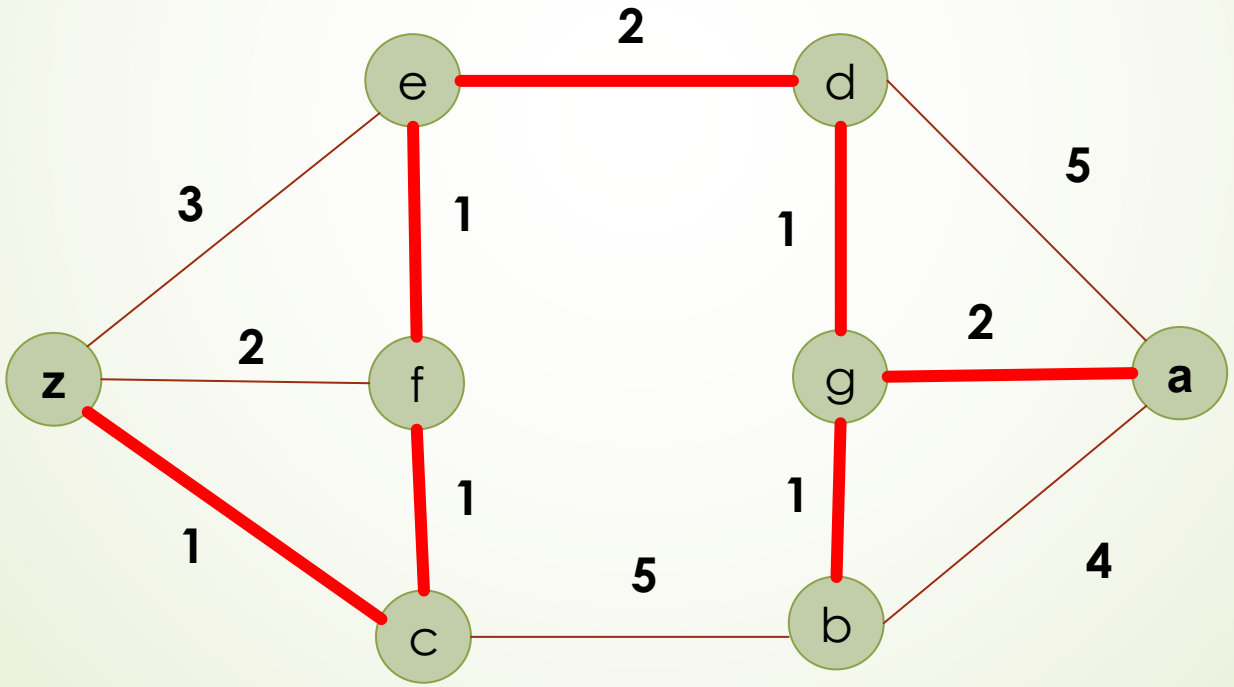
Ejercicio Practico

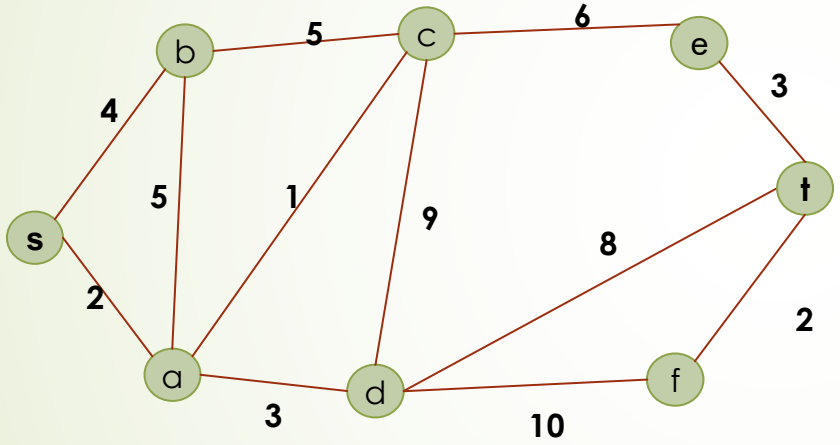
- Utilizando el algoritmo de Dijtstra, trace el camino minimo que hay entre a y z. Calcule su peso



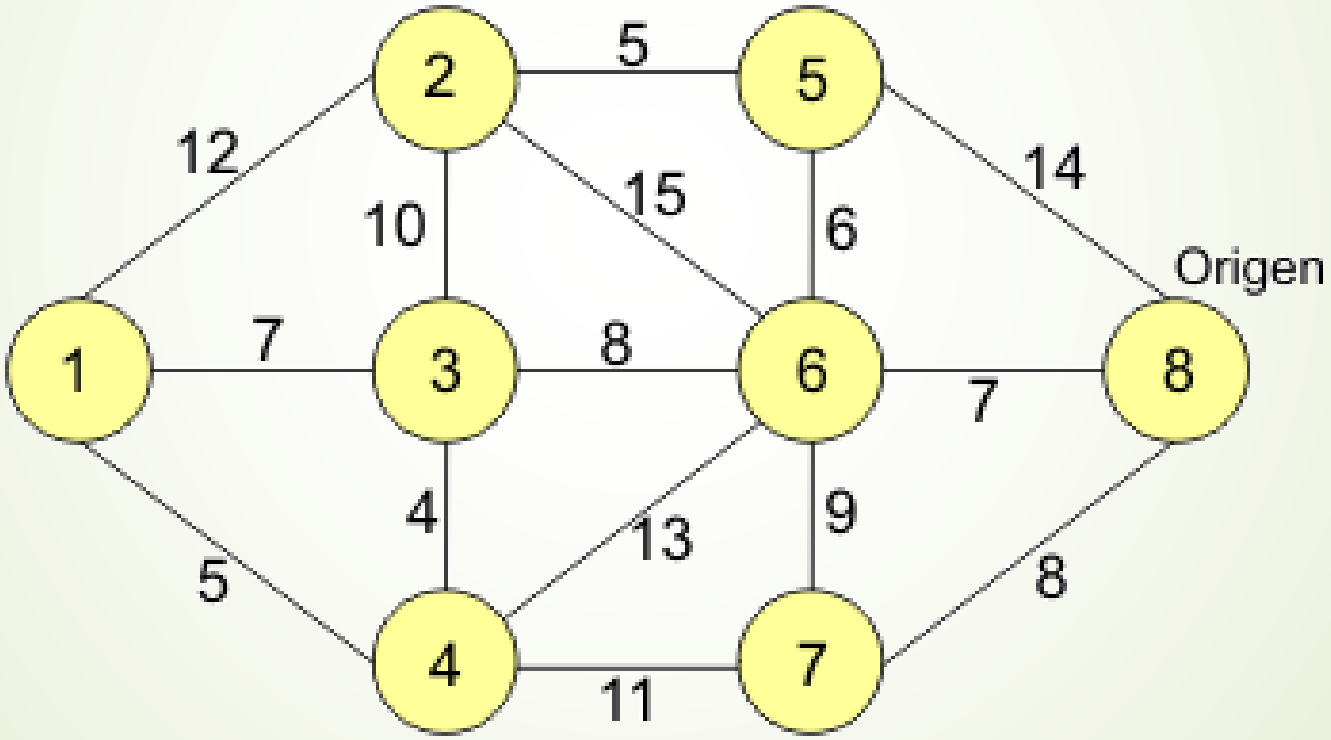
RESULTADO Ejercicio Practico

- Peso del camino es 9

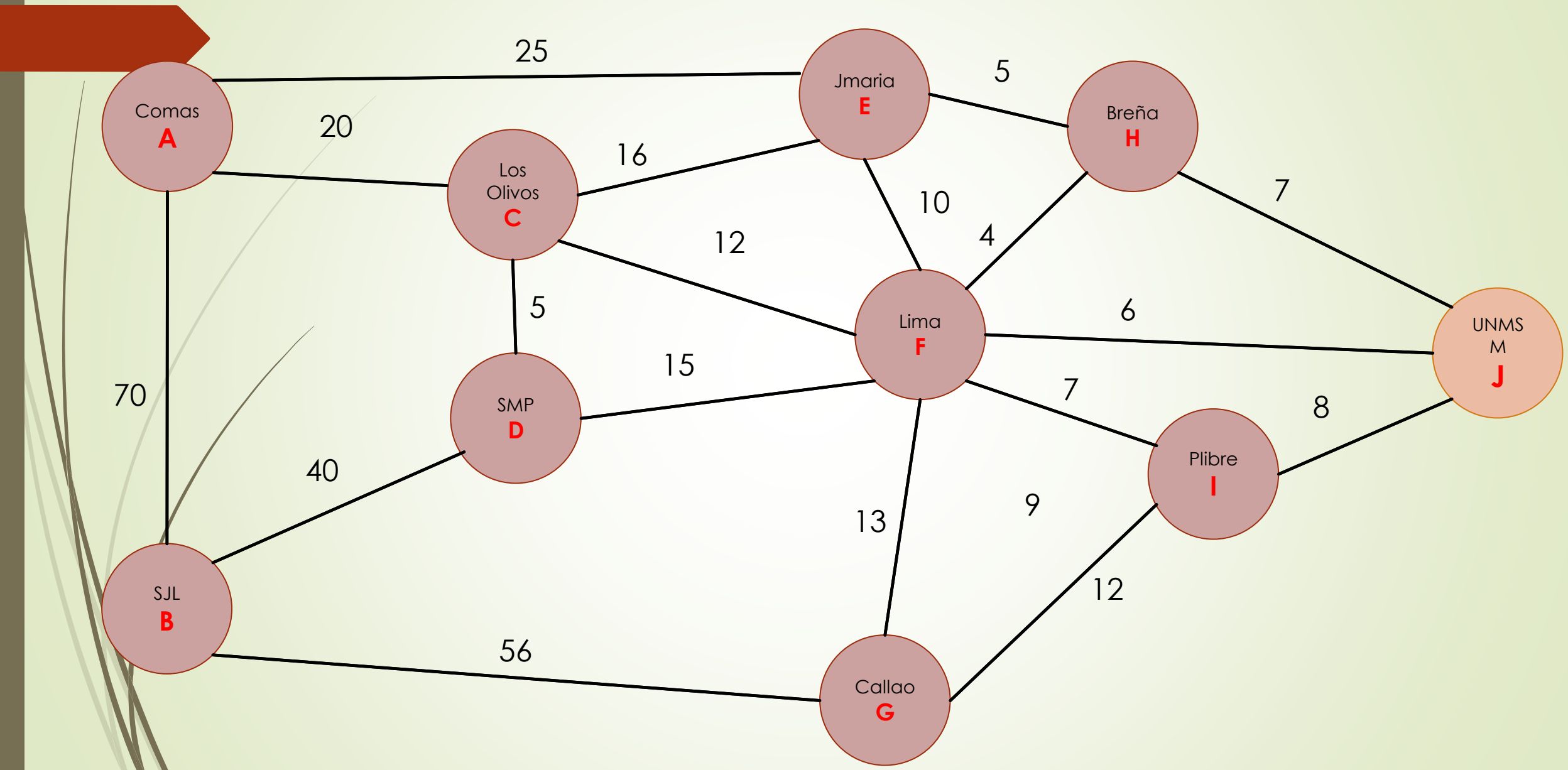



b
c
e
s
a
d
f
t
5
6
4
3
5
1
9
8
2
2
3
10

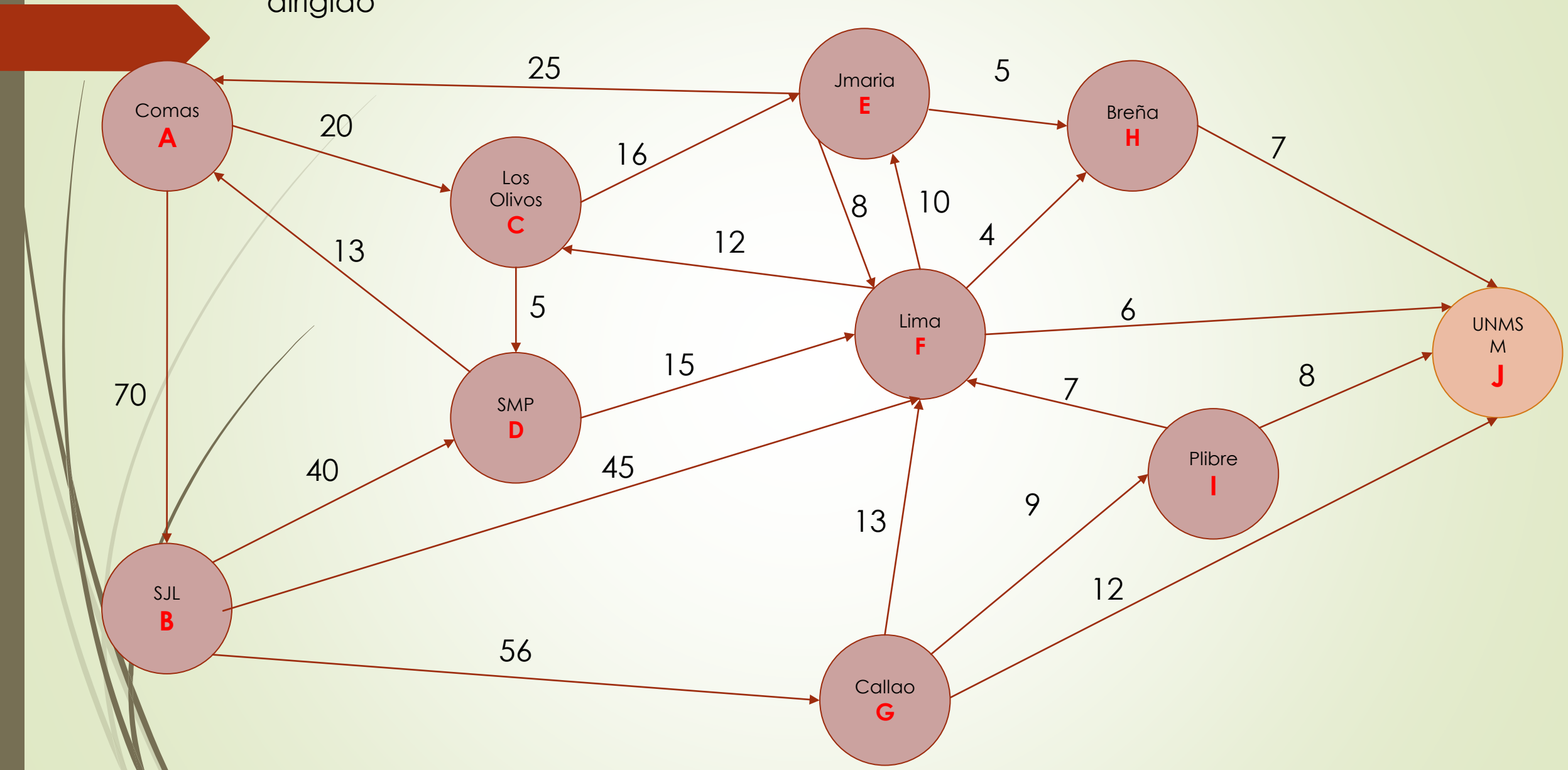
Aplicar el algoritmo de Kruskal para obtener el camino de ruta mínima



EJERCICIO: Obtener el árbol de cubrimiento mínimo

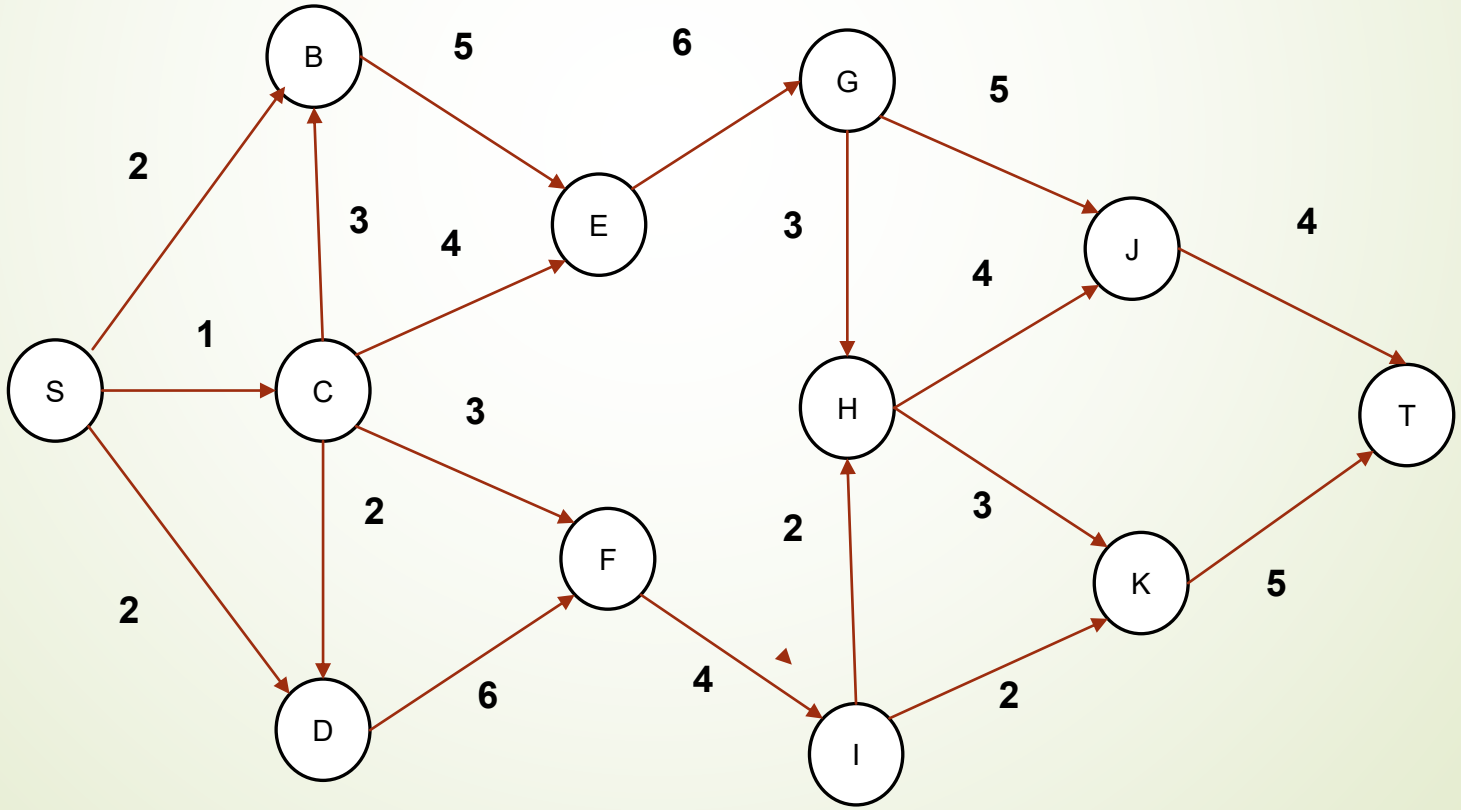


EJERCICIO: Obtener el camino mínimo entre SJL y La UNMSM en base al grafo dirigido



Encontrar el árbol de cobertura minima, utilizando el algoritmo de Kruskal.

Indicar el peso total



Grafos

Grafos Eulerianos

Grafos Hamiltonianos

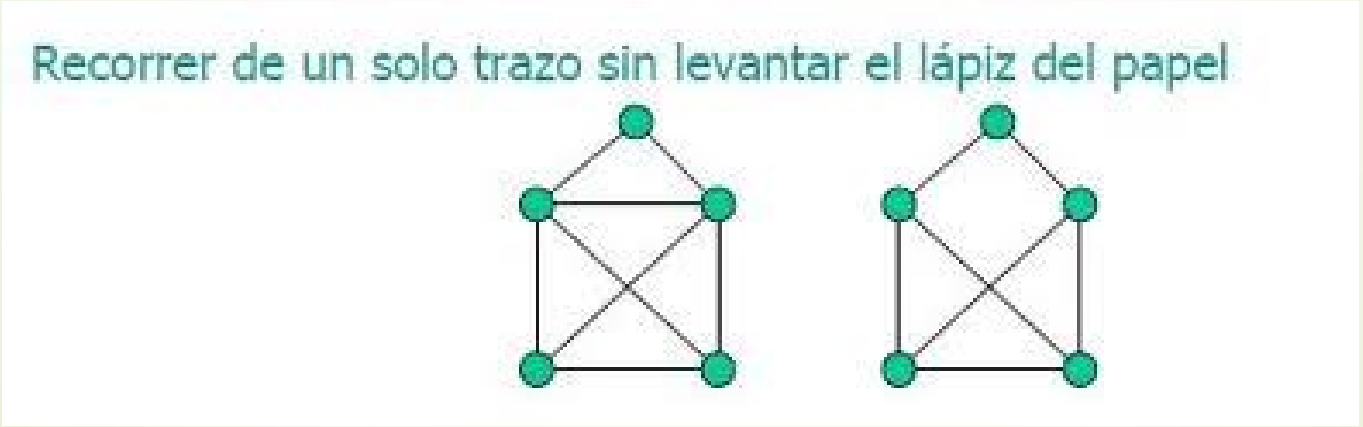
Sesión 14

Gustavo Arredondo C.

UNIVERSIDAD NACIONAL
MAYOR DE SAN MARCOS

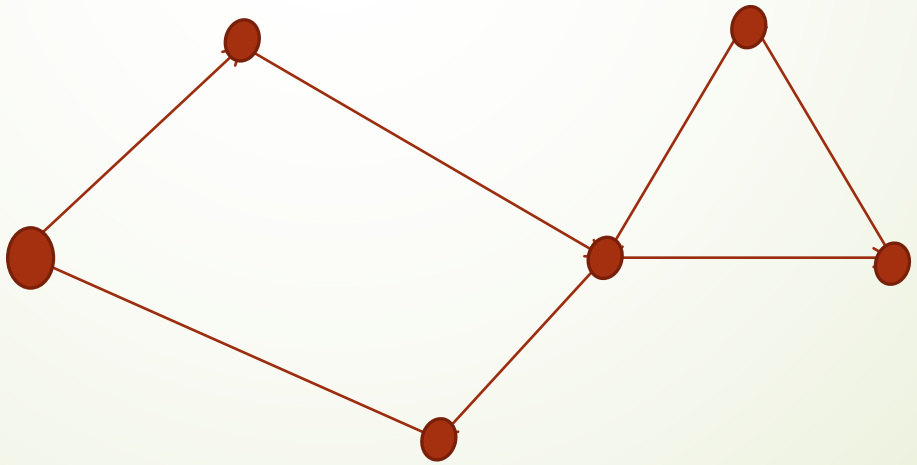
Grafo Euleriano

- **Un circuito Euleriano** en un grafo o multigrafo G es un circuito que recorre cada arista una y sólo una vez.
- Un grafo o multigrafo es Euleriano si tiene un circuito Euleriano.



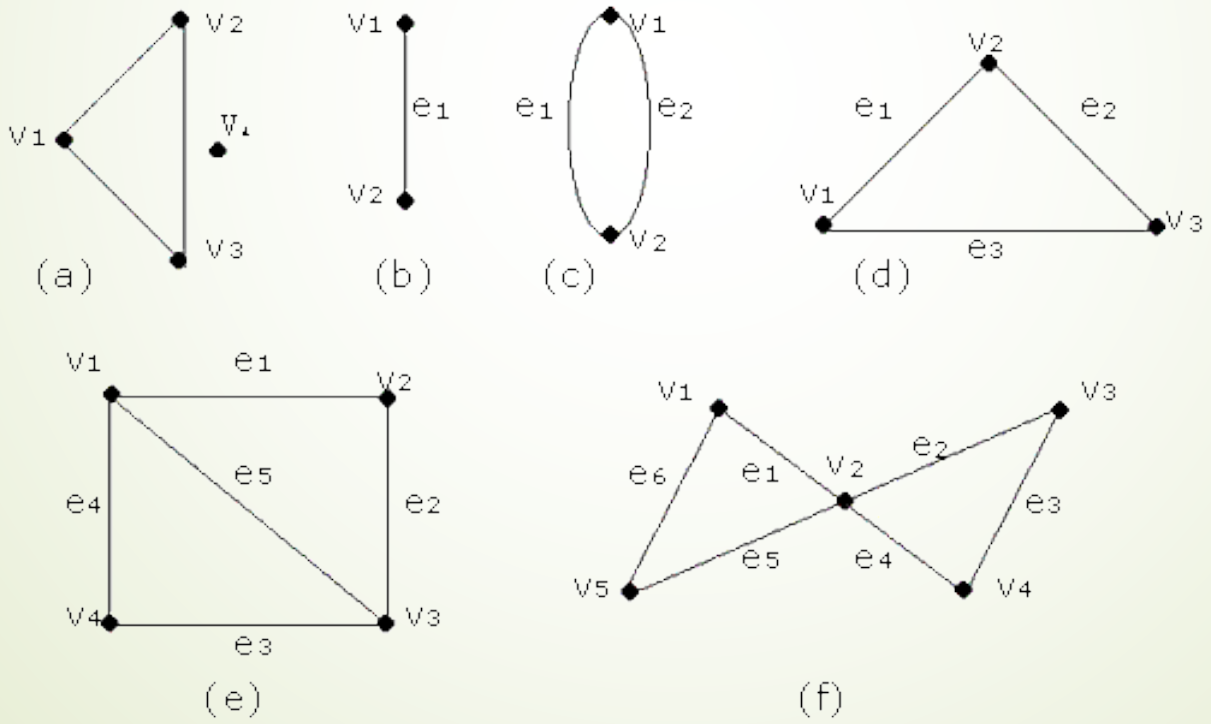
Ciclo Euleriano

- Un ciclo euleriano o circuito euleriano es aquel camino que recorre todas las aristas de un grafo tan solo una única vez, siendo condición necesaria que regrese al vértice inicial de salida (ciclo = camino en un grafo donde coinciden vértice inicial o de salida y vértice final o meta). Una definición más formal lo define como: "aquel ciclo que contiene todas las aristas de un grafo solamente una vez".
- Se debe tener en cuenta que no importa la repetición de vértices mientras no se repitan aristas

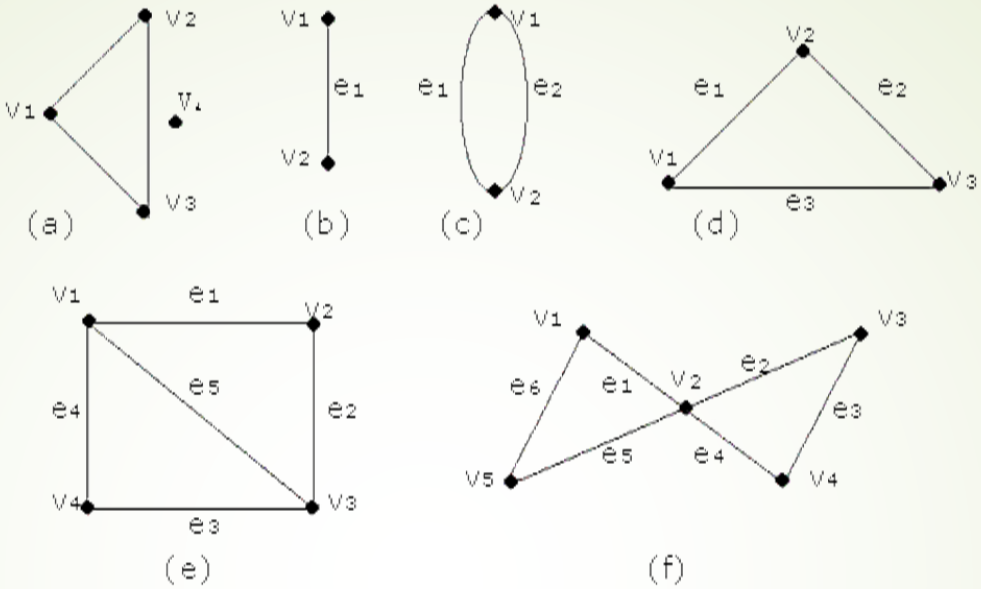


Grafo Euleriano

- En los grafos siguientes, cuáles admiten circuitos eulerianos?



Solución



a) No lo admite porque v4 es un vértice aislado.

b) No lo admite porque cualquier ciclo utilizará la arista e1 dos veces.

c) El circuito v1 e1 v2 e2 v1 es euleriano.

d) El circuito v3 e3 v1 e1 v2 e2 v3 es euleriano.

e) No admite ningún circuito euleriano.

f) v1 e1 v2 e2 v3 e3 v4 e4 v2 e5 v5 e6 v1 es un circuito euleriano.

Teorema

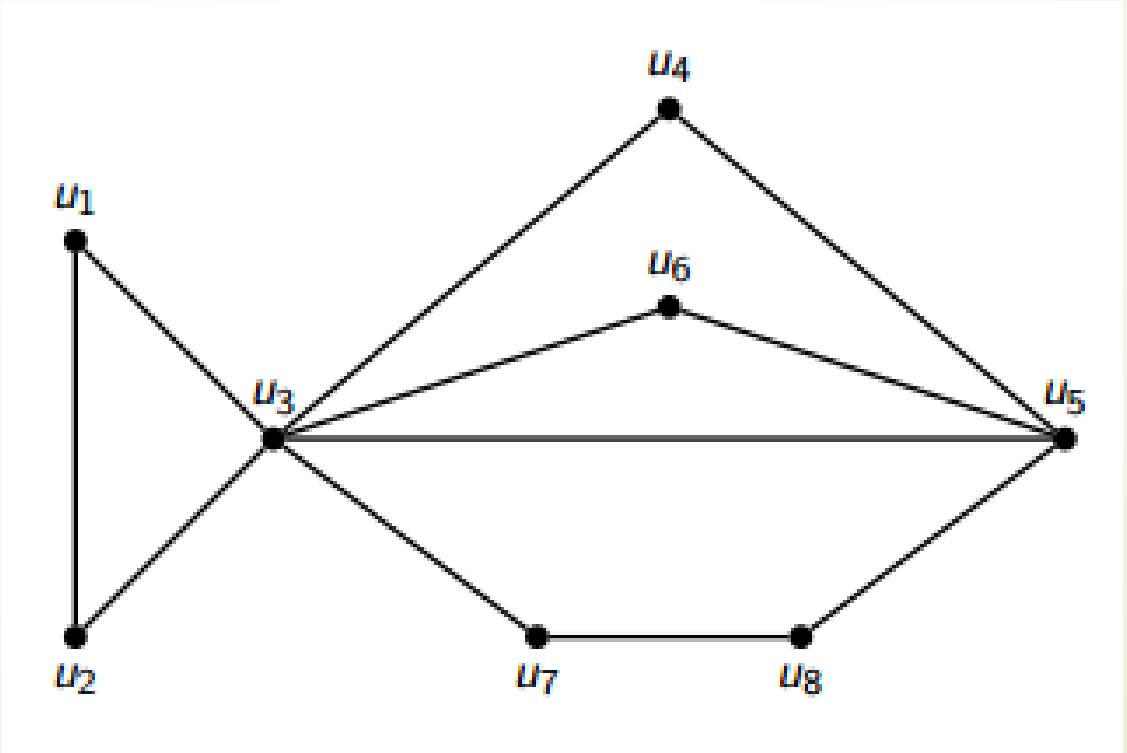
Existe un criterio preciso para saber cuando un grafo admite un circuito Euleriano. Este criterio lo proporciona el siguiente teorema.

Teorema. Sea G un grafo. G contiene un circuito euleriano sí y sólo sí:

- G es conexo.
- Cada vértice de G es de grado par.

Ejercicio

- En el siguiente grafo indique el ciclo euleriano



Grafo Hamiltoniano

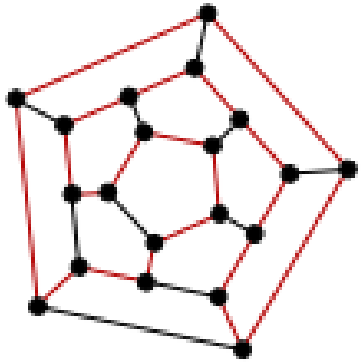
- Un camino hamiltoniano, es un camino de un grafo, una sucesión de aristas adyacentes, que visita todos los vértices del grafo una sola vez.
- Si además el último vértice visitado es adyacente al primero, el camino es un ciclo hamiltoniano.

Grafo Hamiltoniano

Definiciones

- Un camino Hamiltoniano en un grafo G es un camino que recorre cada vértice una y sólo una vez.
- Un circuito Hamiltoniano en un grafo G es un circuito que recorre cada vértice una y sólo una vez.
- Un grafo es Hamiltoniano si tiene un circuito Hamiltoniano.

Ej:



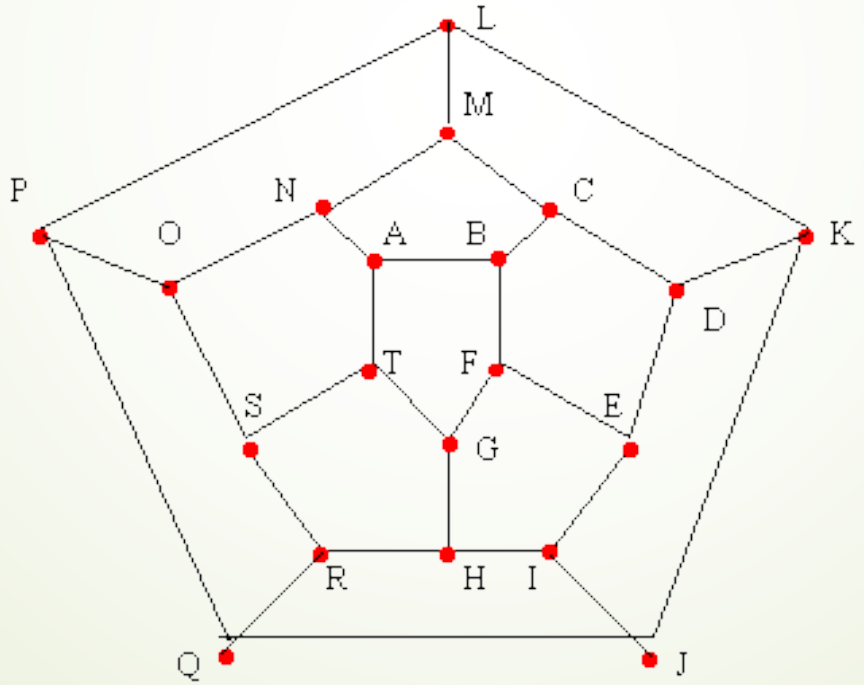
Obs: Si un grafo no es conexo, no puede tener camino ni circuito Hamiltoniano.

Aplicación en recorrido de ruta

Grafo Hamiltoniano

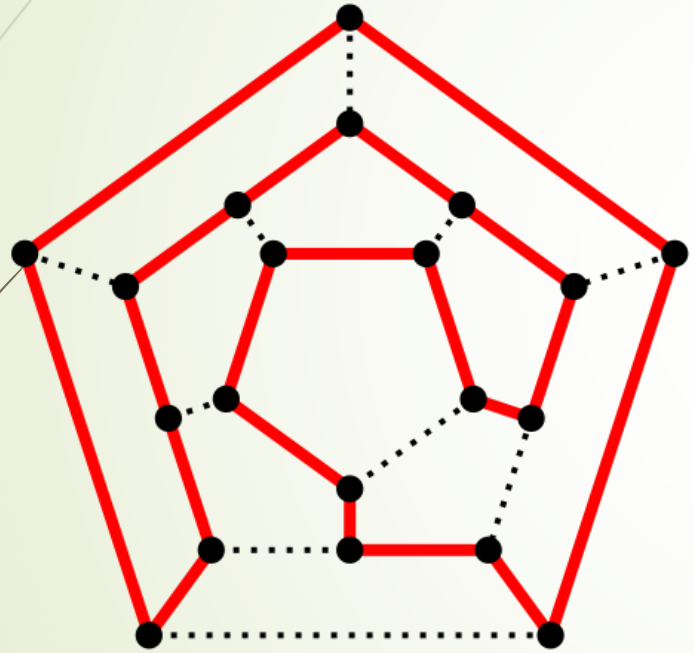
Ejercicio

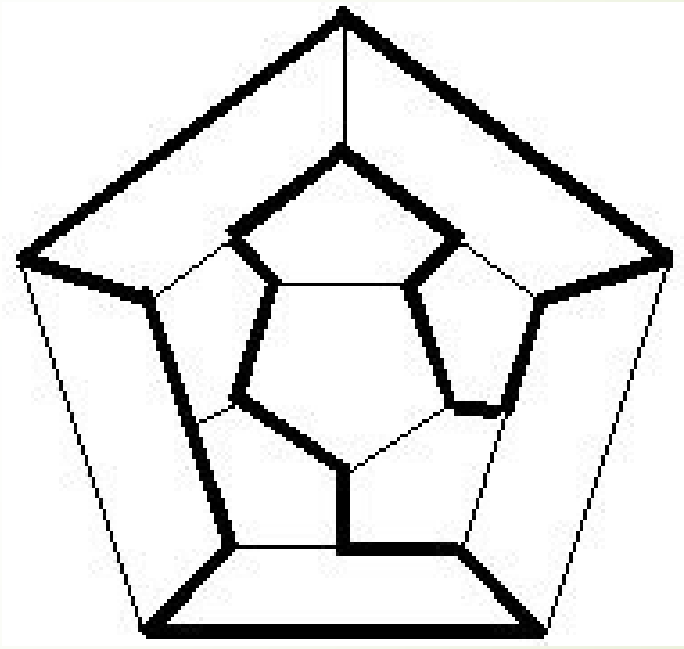
- Indique y demuestre si el siguiente grafo es hamiltoniano



Grafo Hamiltoniano

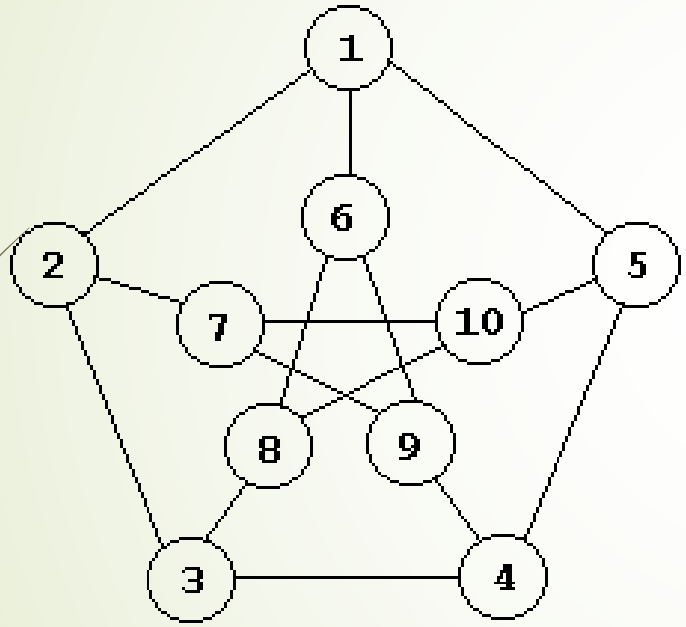
- Solución

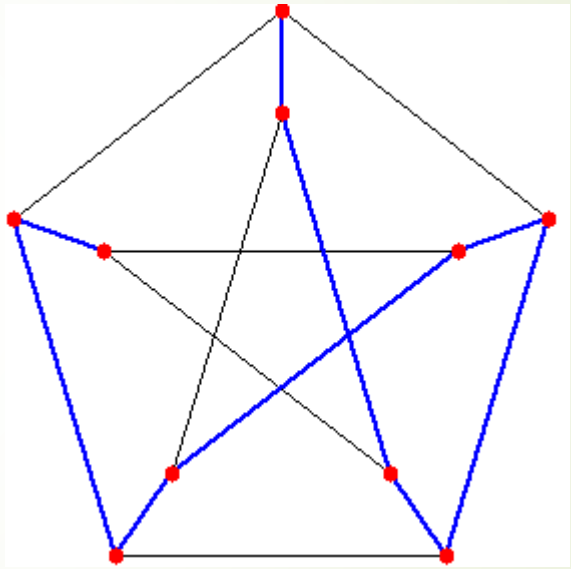




Grafo Hamiltoriano

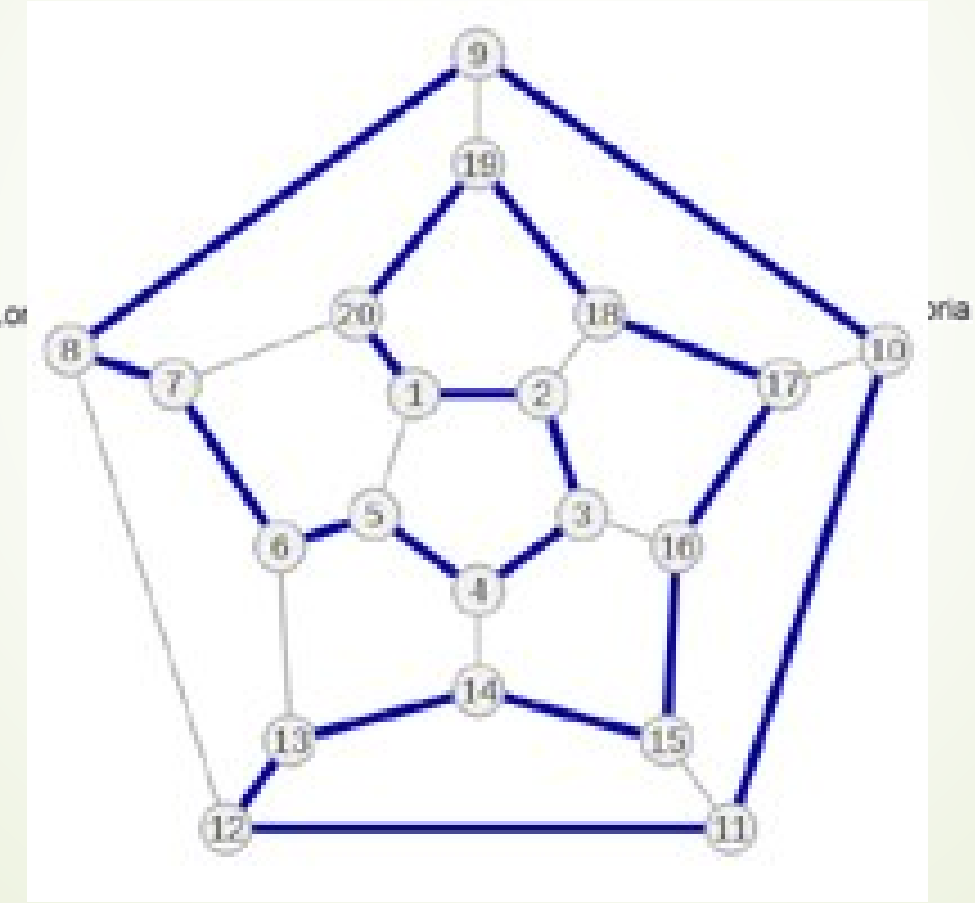
- Ejercicio 2





Grafo Hamiltoniano

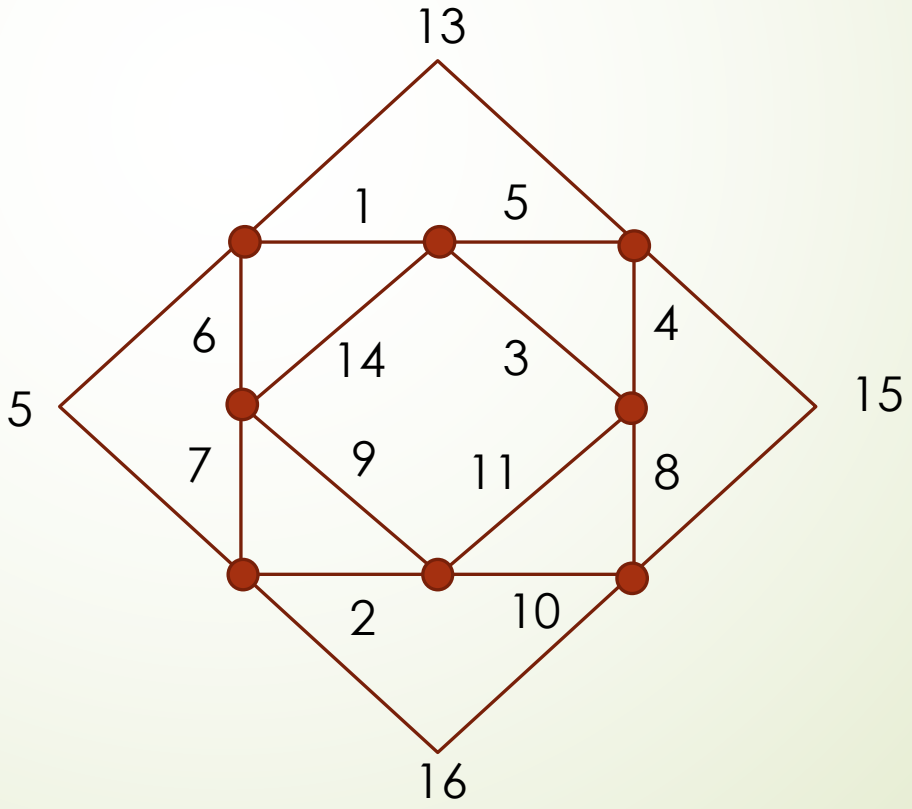
- Ejercicio 3



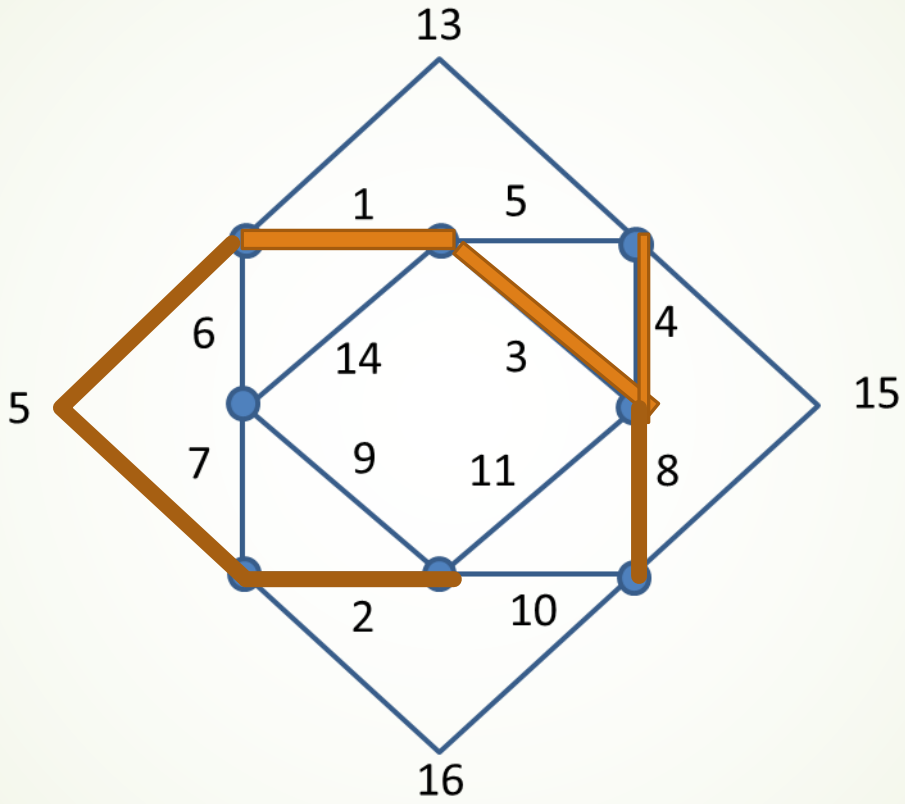
Ejercicios de Algoritmos de Grafos

Ejercicio 1

- Calcular mediante el algoritmo de Prim o Kruskal un arbol generador minimo del grafo

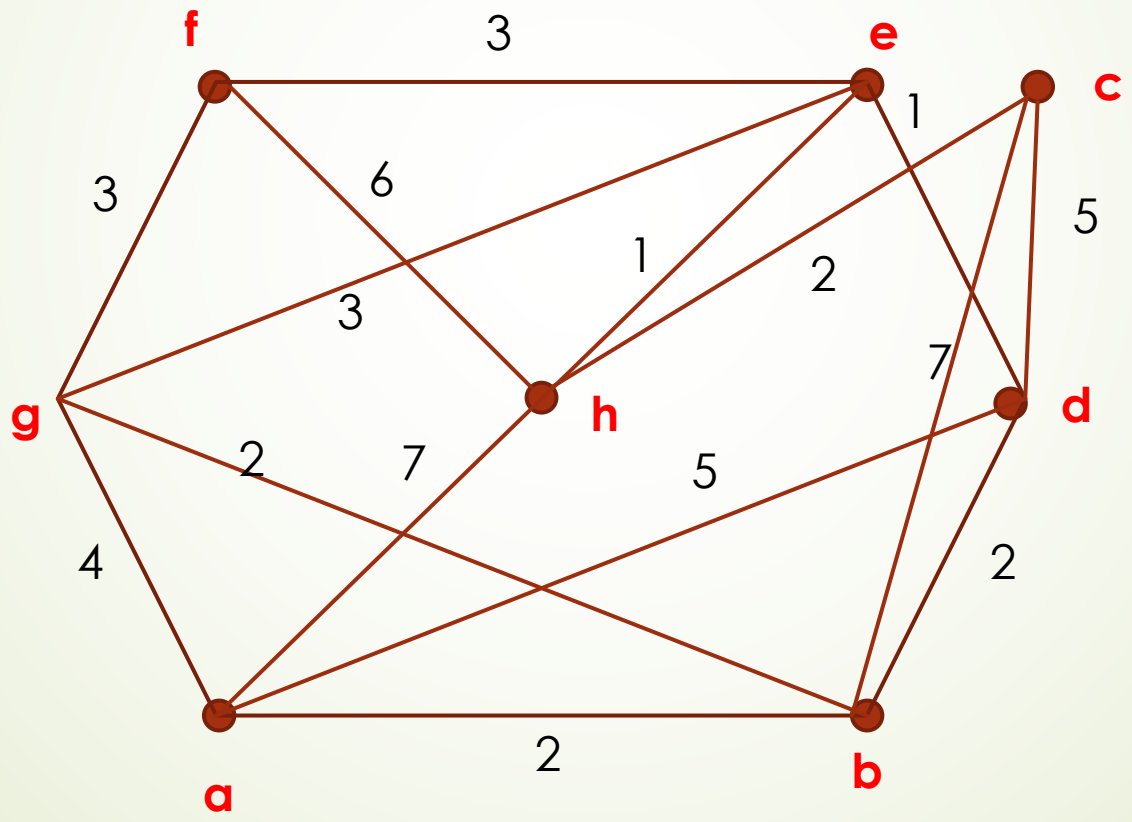


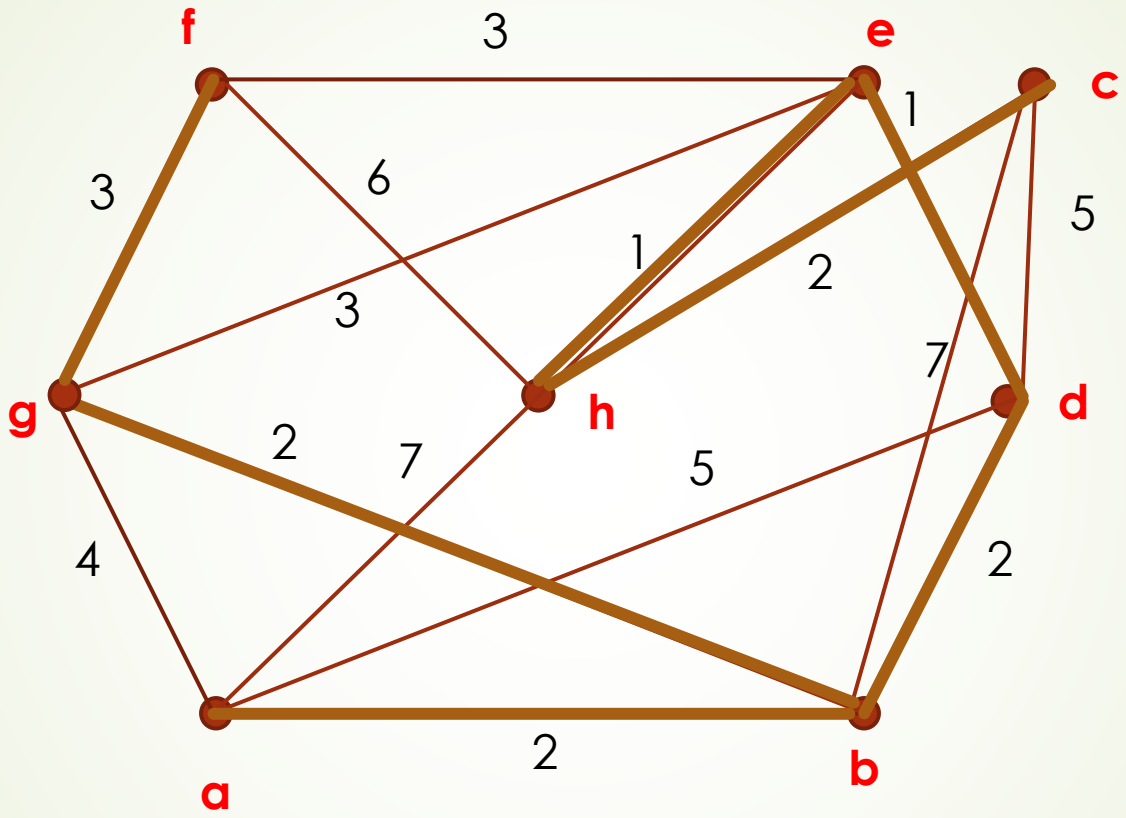
Aplicando Prim



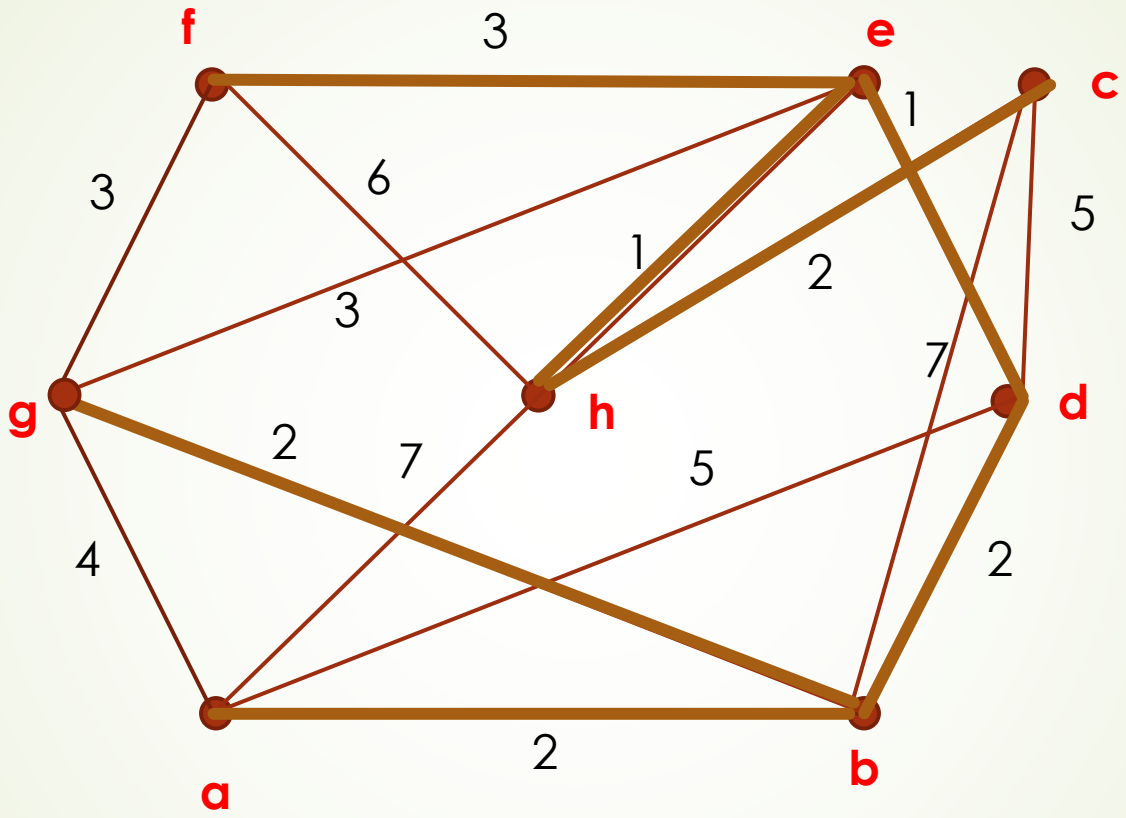
Ejercicio 2

- Calcular mediante el algoritmo de Kruskal un arbol generador minimo del grafo



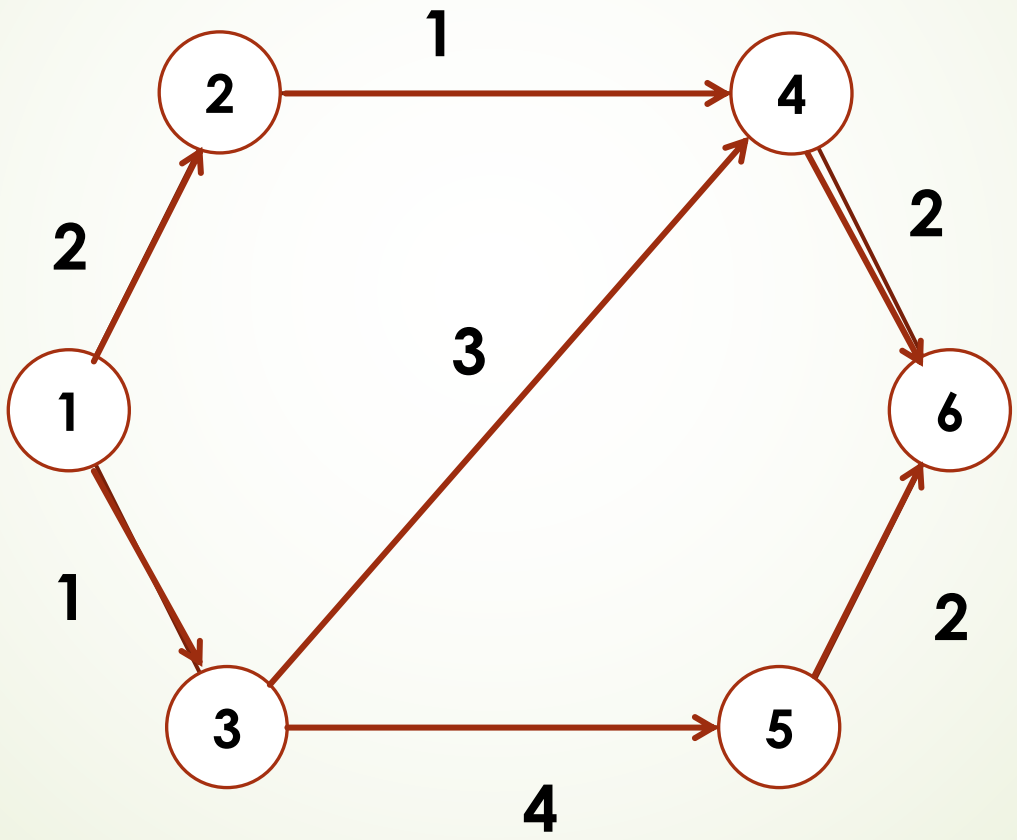


Peso: 1 + 1 + 2 + 2 + 2 + 2 + 3 = 13


f
3
e
c
1
3
6
5
1
2
3
7
g
h
d
2
7
5
4
2
2
a
b

Ejercicio 3

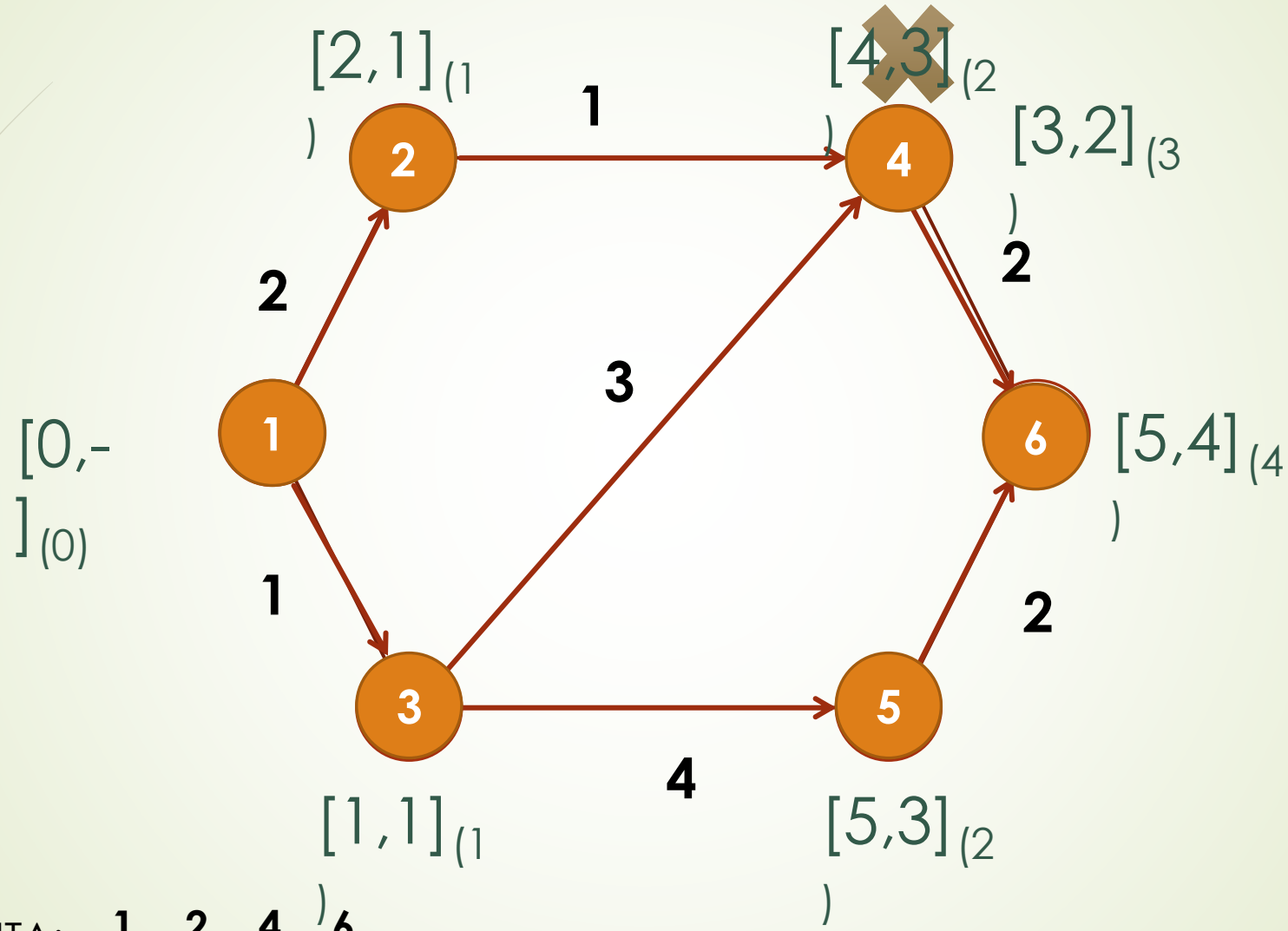
- Calcular la ruta y la distancia mas corta desde el vertice 1 al vertice 6 aplicando el método de Dijkstra



Etiquetado

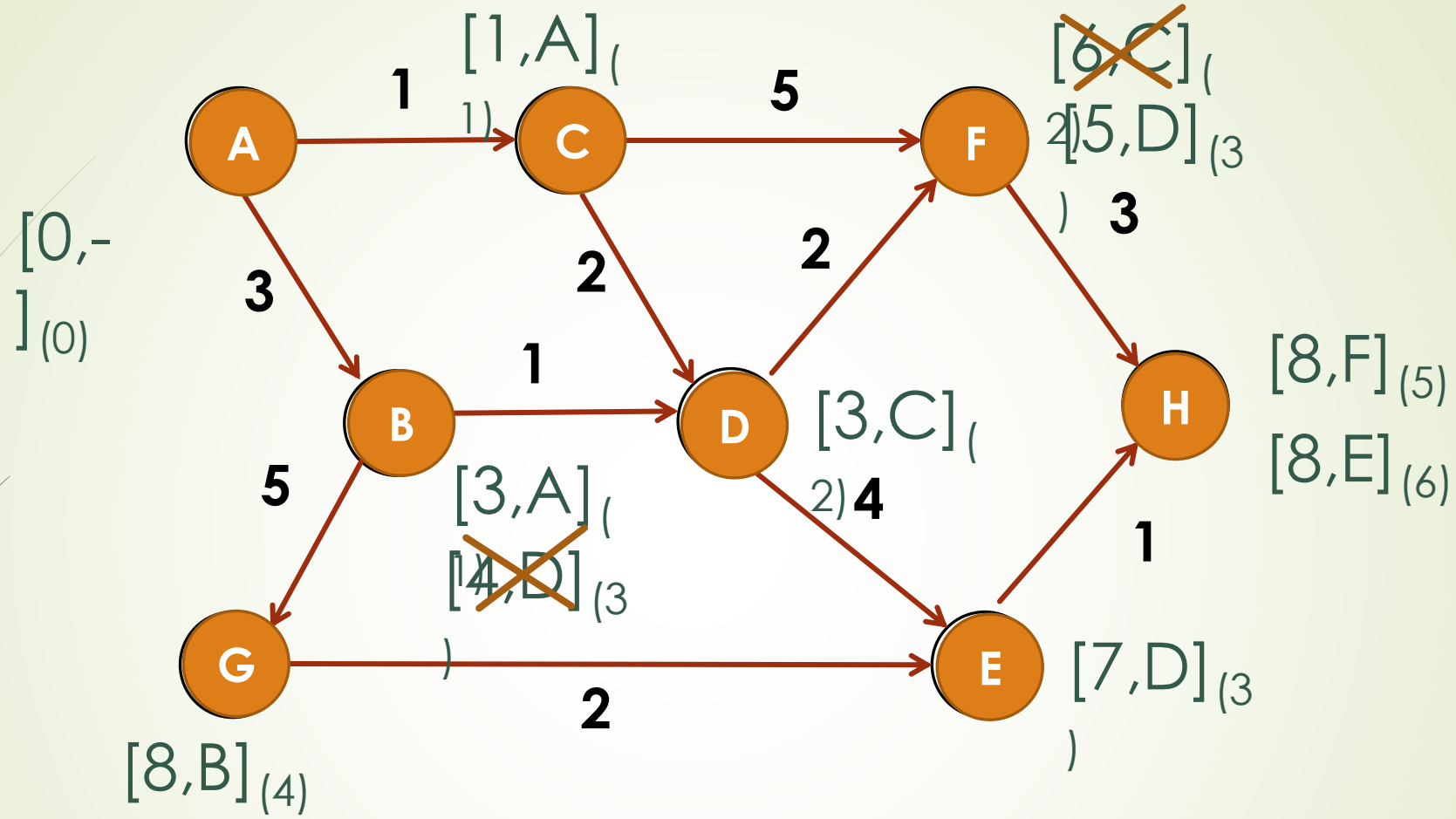
$$[8,B]_{(2)}$$

Algoritmo de Dijkstra



RUTA: 1 2 4 6

Distancia: 5



Ruta 1: **A - C - D - F - H**

Ruta 2: **A - C - D - E - H**

Distancia: 8

Ejercicios

Ejercicio 1

- Utilizando el algoritmo de Huffman, construir el árbol y la tabla de códigos correspondiente a cada letra de la palabra

MATEMATICAMENTE

MATEMATICAMENTE

- Letra - Frecuencia

M (3)

A (3)

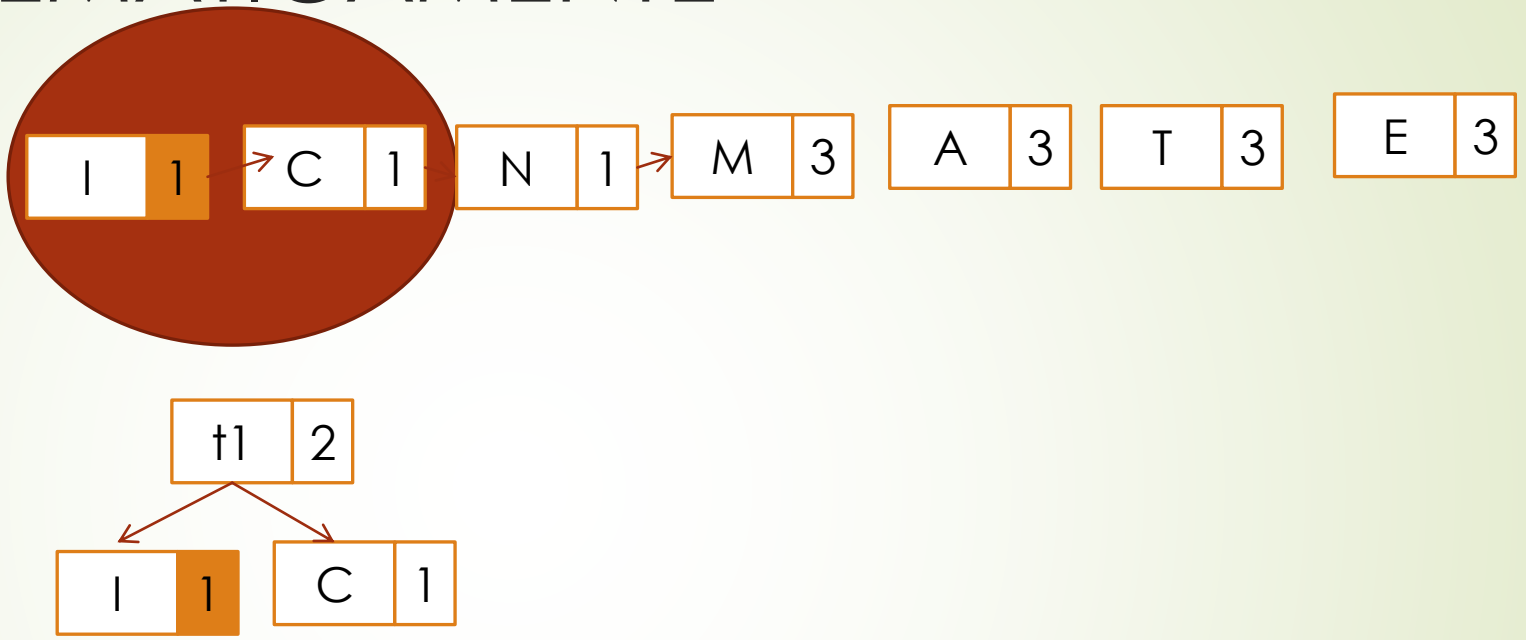
T (3)

E(3)

I (1)

C(1)

N (1)



MATEMATICAMENTE

- Letra - Frecuencia

M (3)

A (3)

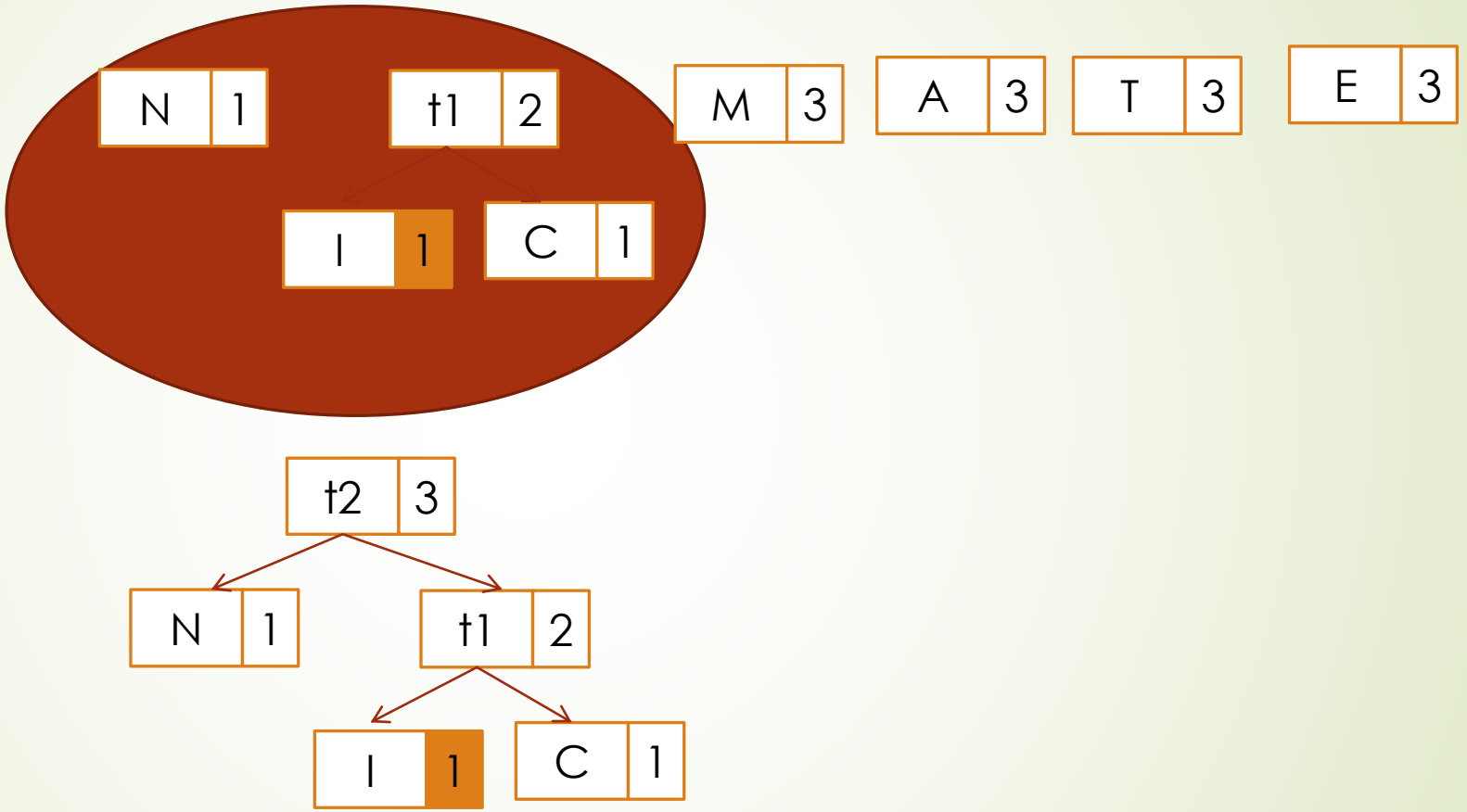
T (3)

E(3)

I (1)

C(1)

N (1)



MATEMATICAMENTE

- Letra - Frecuencia

M (3)

A (3)

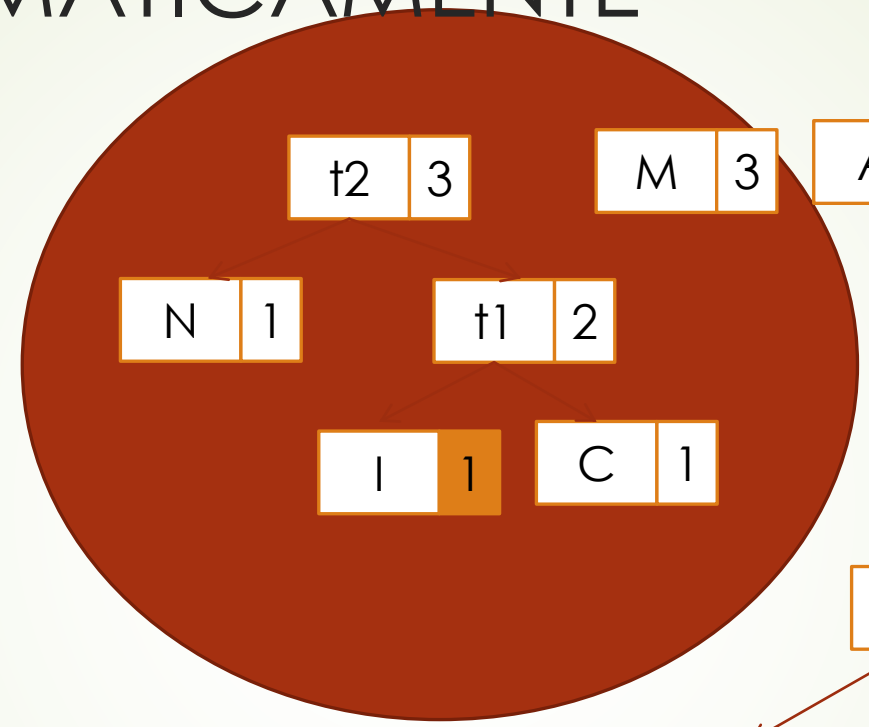
T (3)

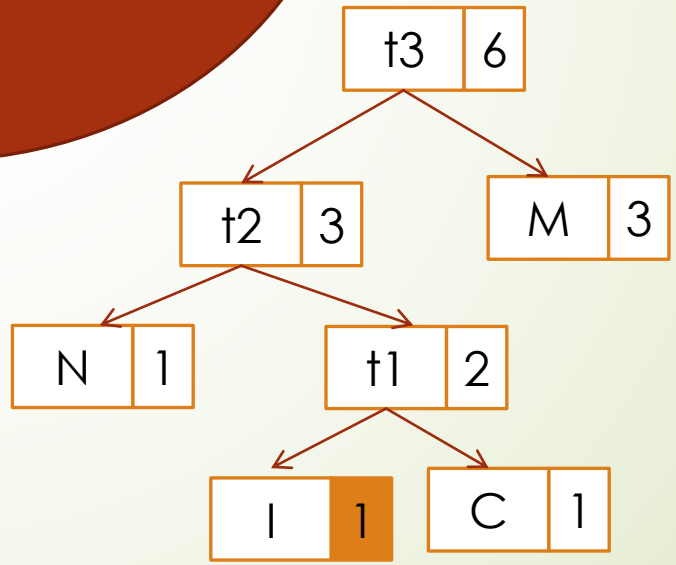
E(3)

I (1)

C(1)

N (1)





MATEMATICAMENTE

- Letra - Frecuencia

M (3)

A (3)

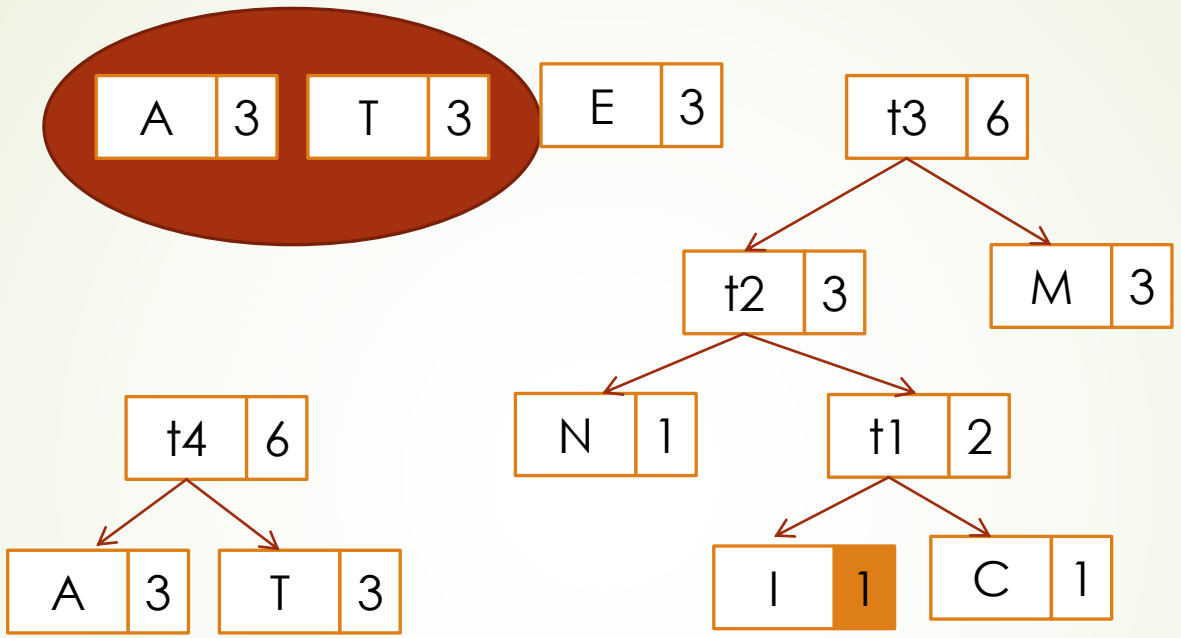
T (3)

E(3)

I (1)

C(1)

N (1)



MATEMATICAMENTE

- Letra - Frecuencia

M (3)

A (3)

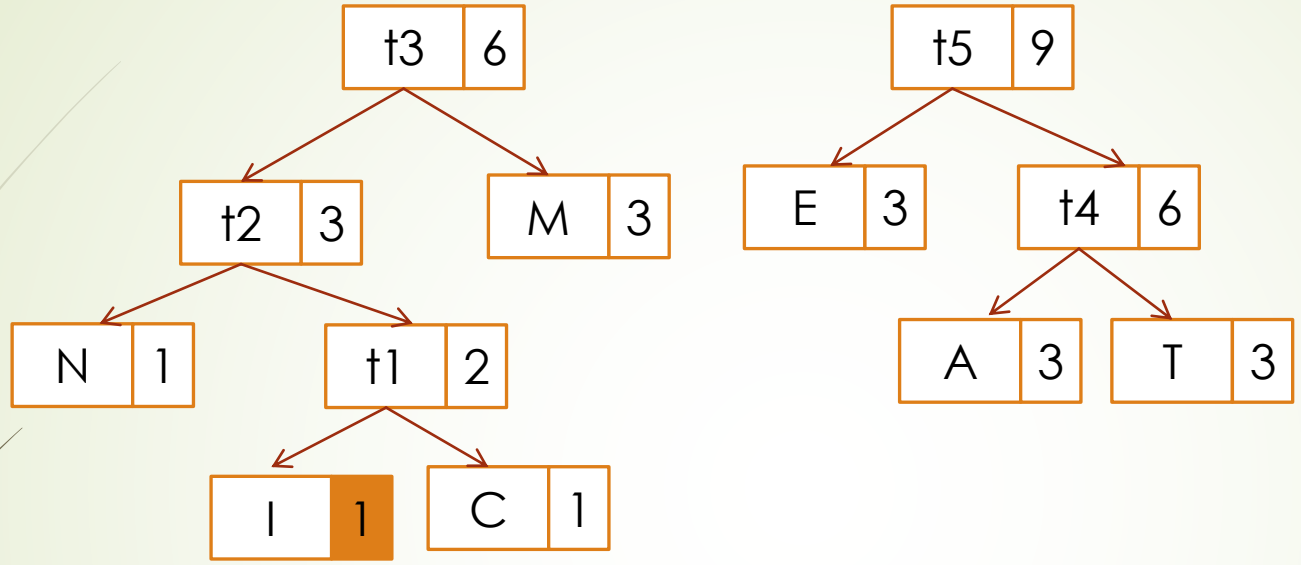
T (3)

E(3)

I (1)

C(1)

N (1)



MATEMATICAMENTE

- Letra - Frecuencia

M (3) = 01

A (3) = 110

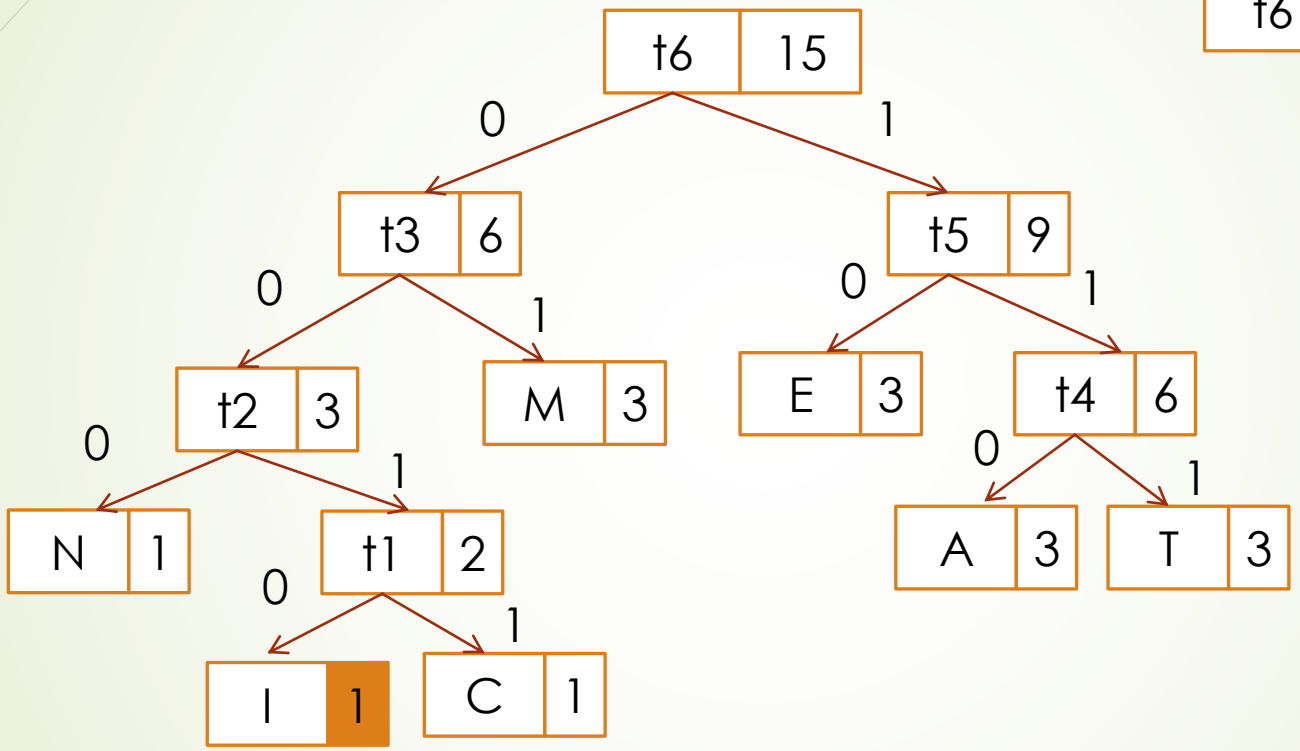
T (3) = 111

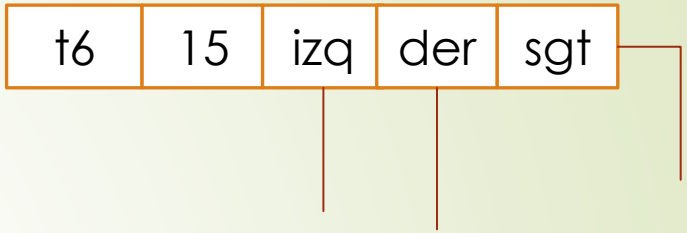
E(3) = 10

I (1)= 0010

C(1)=0011

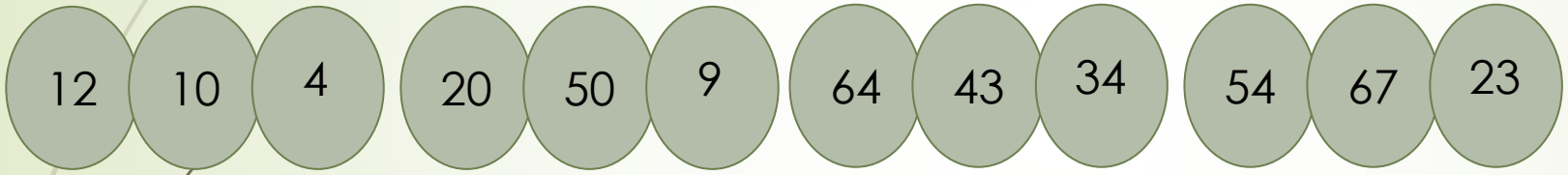
N (1)=000





Ejercicio 2

Dados los siguientes nodos, verificar si se puede construir un arbol tipo heap-min y un árbol tipo heap-max



Ejercicio: Arboles ABB

1. Construir un algoritmo que permita insertar nodos en un árbol ABB
   - Considerar si el árbol esta vacío
   - Dado un nodo, registrarlo en el lugar adecuado del árbol
   - No permitir ingresar un nodo que ya exista
   - Permita hacer el recorrido Pre, In y Post Orden. Mostrar los valores de los nodos según el tipo de recorrido
2. Construir un algoritmo para obtener la altura del árbol y el peso del árbol
3. Implementar la opción de eliminar un nodo en el árbol creado en el punto 1 que permita realizar lo siguiente.
   - Eliminar un nodo hoja
   - Eliminar un nodo con un hijo