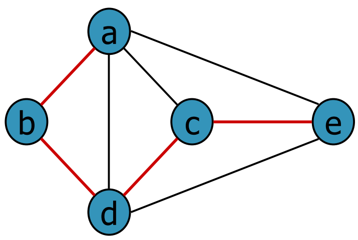
# UNIDAD 3 GRAFOS (PARTE 2)

Arboles de expansión mínima



## ÁRBOL DE EXPANSIÓN MÍNIMA

- Definición: Sea G = (V,A) un grafo no dirigido conexo ponderado, donde V representa el conjunto de vértices V y A el conjunto de aristas.
- Un subgrafo de G, es un grafo G' = (V', A') donde:
  - V' es un subconjunto de V
  - A' consta de las aristas (v,w) en A, tal que v y w están en V



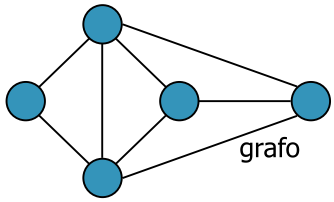

**V = { a, b, c, d, e }**

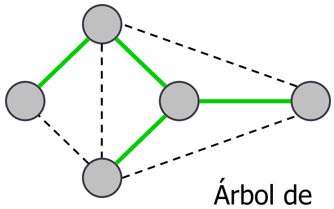
**A = { (a,b), (a, c), (a, d), (a, e), (b,d), (c, d), (c, e), (d, e) }**

**A' = { (a,b), (b,d), (c, d), (c, e) }**

## ÁRBOL DE EXPANSIÓN MÍNIMA

- Un árbol puede verse como un caso particular de un grafo: un grafo conexo acíclico
- Para obtener el árbol de expansión de un grafo hay que eliminar todas las aristas que producen ciclos, pero manteniéndolo conexo
  - Aplicación: encaminamiento en redes de comunicaciones
- No existe un único árbol de expansión de un grafo, pues dependerá del nodo de partida y de la forma de recorrerlo



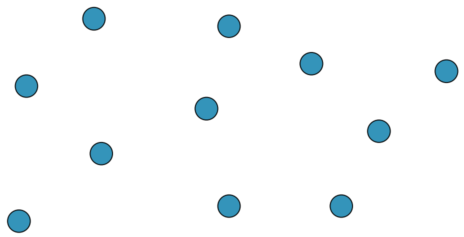



## ÁRBOL ABARCADOR/EXPANSIÓN MÍNIMA

- Definición: Un Árbol libre es un grafo conexo acíclico.
- Propiedades:
  - Todo árbol libre con n >= 1 vértices, contiene n-1 aristas
  - Si se agrega cualquier arista a un árbol libre, resulta un ciclo.
- Costo asociado: Supongamos que G = (V,A) es un grafo en donde cada arista (v,w) de A tiene un costo asociado c(v,u)
- El árbol de expansión: Es un árbol libre, que conecta todos los vértices de V. El costo del árbol de expansión es la suma de los costos de sus aristas.
- El árbol de expansión de costo mínimo: Es el árbol de expansión de un grafo G con costo mínimo.

# PROBLEMA DEL ÁRBOL DE EXPANSIÓN MÍNIMA

- Imagina que se tienen muchas ciudades, como se muestra. Supongamos que se quiere conectar todas las ciudades en un árbol de expansión de forma que se minimice la distancia total del árbol.
- ¿Qué algoritmo propondrías?



M.C. YALÚ GALICIA HDEZ. (FCC/BUAP) 5

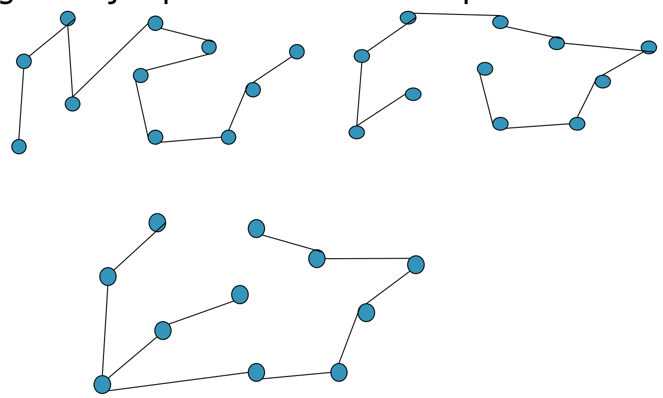
# PROBLEMA DEL ÁRBOL DE EXPANSIÓN MÍNIMA

- Un algoritmo intuitivo para encontrar un árbol de expansión mínima consiste en:
  - Encontrar todos los árboles de expansión posibles.
  - Para cada árbol de expansión se determina su longitud total
  - Seleccionar el árbol con longitud total mínima
- Lo anterior es muy costoso, puede demostrarse que el número total de árboles de expansión posibles para n ciudades es $n^n$-2.
  - Para n= 100, es un número tan grande que no hay computadora que lo haga!

M.C. YALÚ GALICIA HDEZ. (FCC/BUAP) 6

## PROBLEMA DEL ÁRBOL DE EXPANSIÓN MÍNIMA

- Algunos ejemplos de arboles de expansión





## PROBLEMA DEL ÁRBOL DE EXPANSIÓN MÍNIMA

- En realidad para resolver el problema de árbol de expansión mínima existen un algoritmo excelente.
- Este algoritmo va como sigue:
  - Encontrar la menor de las distancias que comunican las ciudades. Sean 1 y 2, las ciudades conectadas con menor distancia.
  - Considerar {1, 2} como un conjunto y el resto de las demás ciudades como otro conjunto.
  - Encontrar la distancia mas corta entre estos dos conjuntos de ciudades.
  - Sea 2 y 3, las ciudades conectadas. Agregar 3 al primer conjunto y repetir el punto anterior.

## PROBLEMA DEL ÁRBOL DE EXPANSIÓN MÍNIMA

- Puede demostrarse que este sencillo algoritmo eficiente siempre produce una solución óptima.
  - Es decir, el árbol de expansión mínima final que produce siempre es un árbol de expansión mínima.
  - La demostración de que este algoritmo es correcto no es fácil, la estrategia detrás de este algoritmo se denomina método codicioso o greedy. Es una estrategia para resolver problemas de optimización.
  - Normalmente un algoritmo basado en el método codicioso es bastante eficiente.
- Desafortunadamente muchos problemas semejantes al problema del árbol de expansión mínima no pueden resolverse con el método codicioso: Un ejemplo es el problema del viajero.



## PROBLEMA DEL VIAJERO (TSP: *TRAVELLING SALESMAN PROBLEM)*

- Sean N ciudades de un territorio. El objetivo es encontrar una ruta que, comenzando y terminando en una ciudad concreta, pase una sola vez por cada una de las ciudades y minimice la distancia recorrida por el viajante.
- Es decir, encontrar una permutación $P = \{c_0, c_2, .., c_{n-1}\}$ tal que

$$dp = \sum_{i=0}^{n-1} d[c_i, c_{i+1 \mathrm{mod}(n)}]$$

sea mínimo.

- La distancia entre cada ciudad viene dada por la matriz D: NxN, donde d[x, y] representa la distancia que hay entre la ciudad X y la ciudad Y

## PROBLEMA DEL VIAJERO (TSP: TRAVELLING SALESMAN PROBLEM)

- La solución más directa es la que aplica la fuerza bruta: evaluar todas las posibles combinaciones de recorridos y quedarse con aquella cuyo trazado utiliza la menor distancia.
- El problema reside en el número de posibles combinaciones que viene dado por el factorial del número de ciudades (N!) y esto hace que la solución por fuerza bruta sea impracticable para valores de N incluso moderados con los medios computacionales actualmente a nuestro alcance.
- Por ejemplo, si una computadora fuese capaz de calcular la longitud de cada combinación en un microsegundo, tardaría algo más 3 segundos en resolver el problema para 10 ciudades, algo más de medio minuto en resolver el problema para 11 ciudades y 77.146 años en resolver el problema para sólo 20 ciudades



## ÁRBOL DE EXPANSIÓN MÍNIMA

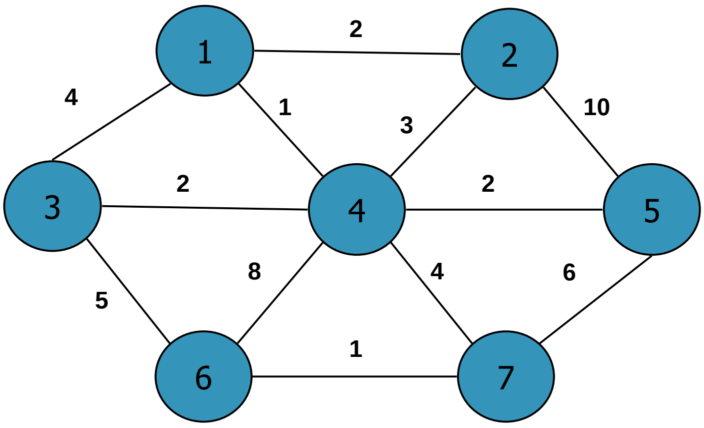
- Existen dos algoritmos basados en el método codicioso, para encontrar un árbol de expansión mínima:
  - Algoritmo de PRIM
  - Algoritmo de Kruskal

# ALGORITMO DE PRIM

- En este algoritmo se hace crecer el árbol en etapas sucesivas.
- En cada etapa se elige un vértice y se analizan sus vértices adyacentes para elegir las aristas de menor costo
- Se emplea Dv y pV como en Dijkstra, pero aquí la regla de actualización es más sencilla.
  - Después de elegir el vértice Vk para cada V desconocido (o no visitado) adyacente a Vk,
  - Dv = min (Dv, C(vk, v))



# ARBOL DE COSTO MÍNIMO RESULTANTE




Costo del arbol de expansión de costo mínimo: 12

## ALGORITMO DE PRIM

- Se asume que se cuenta con:
  - Una tabla con la siguiente configuración inicial.
  - La lista o matriz de adyacencia del grafo
  - c: Costo de la arista al vértice anterior
  - P: vértice anterior

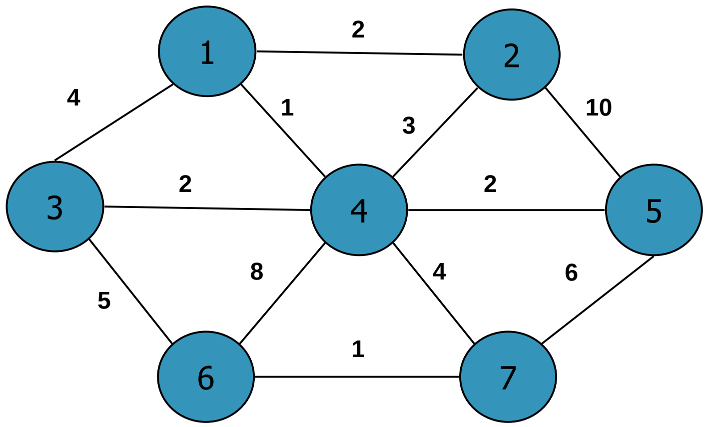
| V | visitado | ca | p |
|---|---|---|---|
| 1 | Falso | 0 | 0 |
| 2 | F | ∞ | 0 |
| 3 | F | ∞ | 0 |
| 4 | F | ∞ | 0 |
| 5 | F | ∞ | 0 |
| 6 | F | ∞ | 0 |
| 7 | F | ∞ | 0 |



## ALGORITMO DE PRIM

```
Inicio
        Para todo vi no visitado hacer
                sea vi el vértice con ca mínimo
                poner vi como visitado
                para cada vk adyacente a vi hacer
                    si vk no está visitado entonces
                        si c(vk)  > Costo(vi, vk) entonces
                                c(vk) = Costo(vi , vk)
                                 p(vk) = vi
                        fin_si
                    fin_si
                fin_para
        fin_para
Fin
```

# EJEMPLO USANDO ALGORITMO PRIM






| V | vis | c | p |
|---|---|---|---|
| 1 | T | **0** | 0 |
| 2 | F | **2** | **1** |
| 3 | F | **4** | **1** |
| 4 | F | **1** | **1** |
| 5 | F | ∞ | 0 |
| 6 | F | ∞ | 0 |
| 7 | F | ∞ | 0 |

| V | vis | c | p |
|---|---|---|---|
| 1 | T | 0 | 0 |
| 2 | F | 2 | 1 |
| 3 | F | **2** | **4** |
| 4 | T | 1 | 1 |
| 5 | F | **2** | **4** |
| 6 | F | **8** | **4** |
| 7 | F | **4** | **4** |

| V | vis | C | p |
|---|---|---|---|
| 1 | T | 0 | 0 |
| 2 | T | 2 | 1 |
| 3 | T | 2 | 4 |
| 4 | T | 1 | 1 |
| 5 | F | 2 | 4 |
| 6 | F | **5** | **3** |
| 7 | F | 4 | 4 |

| V | vis | c | p |
|---|---|---|---|
| 1 | T | 0 | 0 |
| 2 | T | 2 | 1 |
| 3 | T | 2 | 4 |
| 4 | T | 1 | 1 |
| 5 | T | 2 | 4 |
| 6 | F | 5 | 3 |
| 7 | F | 4 | 4 |

| V | vis | c | p |
|---|---|---|---|
| 1 | T | 0 | 0 |
| 2 | T | 2 | 1 |
| 3 | T | 2 | 4 |
| 4 | T | 1 | 1 |
| 5 | T | 2 | 4 |
| 6 | F | **1** | **7** |
| 7 | T | 4 | 4 |

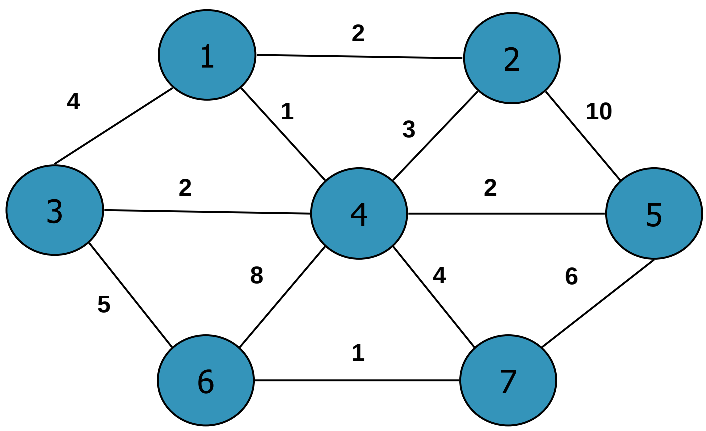
| V | vis | c | p |
|---|---|---|---|
| 1 | T | 0 | 0 |
| 2 | T | 2 | 1 |
| 3 | T | 2 | 4 |
| 4 | T | 1 | 1 |
| 5 | T | 2 | 4 |
| 6 | T | 1 | 7 |
| 7 | T | 4 | 4 |

**Aristas del árbol de expansión mínima**

**A = { (1,2), (4, 3), (1, 4), (5, 4), (7,6), (4, 7) }** **COSTO: 12**

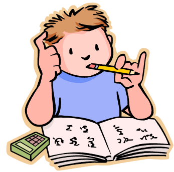
M.C. YALÚ GALICIA HDEZ. (FCC/BUAP) 19

## ARBOL DE COSTO MÍNIMO RESULTANTE



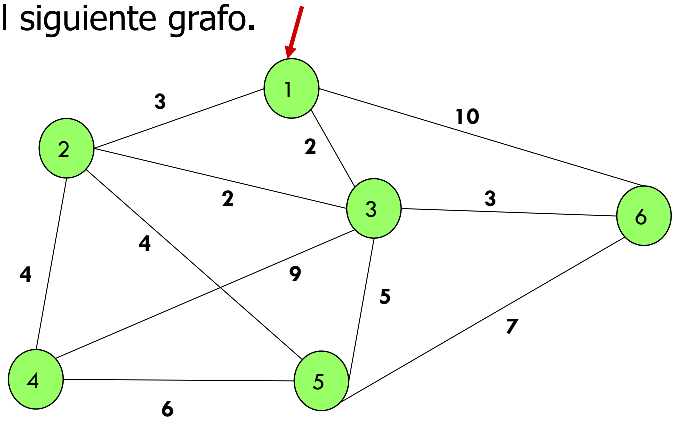

Costo del arbol de expansión de costo mínimo: 12

M.C. YALÚ GALICIA HDEZ. (FCC/BUAP) 20



## ACTIVIDAD INDIVIDUAL

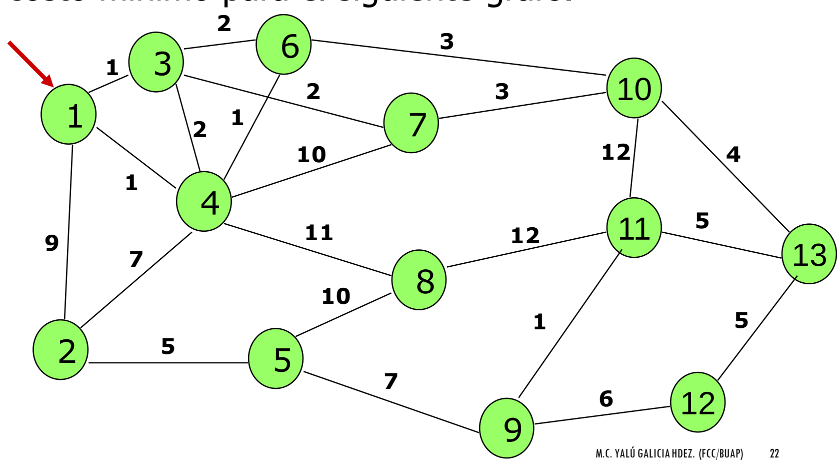
- Obtén el árbol de expansión mínima para el siguiente grafo.




M.C. YALÚ GALICIA HDEZ. (FCC/BUAP) 21

## ACTIVIDAD COLABORATIVA

- En binas, obtengan el árbol abarcador de costo mínimo para el siguiente grafo.




M.C. YALÚ GALICIA HDEZ. (FCC/BUAP) 22

# ACTIVIDAD COLABORATIVA

- Formar equipos de 4.
- A partir del grafo obtenido en la actividad colaborativa anterior de la Planta VW, aplica el algoritmo de Prim para resolver el problema de "Averiguar cuales serian las líneas de comunicación dentro de su red de distribución que tendría que mantener para dar servicio a todas las localidades con un costo total mínimo."
  - Obtengan la tabla correspondiente, dibuja el árbol de expansión mínima e indica el costo



# VARIACIÓN DEL ALGORITMO DE PRIM

- La idea básica consiste en añadir, en cada paso, una arista de peso mínimo a un árbol previamente construido. Más explícitamente:
- Paso 1.
  - Se elige un vértice u de G y se considera el árbol S={u}
- Paso 2.
  - Se considera la arista e de mínimo peso que une un vértice de S y un vértice que no es de S, y se hace S=S+e
- Paso 3.
  - Si el nº de aristas de T es n-1 el algoritmo termina. En caso contrario se vuelve al paso 2

## ALGORITMO DE KRUSKAL

- Este algoritmo consiste en ordenar las aristas por costos de forma ascendente e ir marcándolas como aceptadas, siempre y cuando no ocasionen un ciclo.
- Meta: obtener un árbol de expansión o abarcador de costo mínimo.



## ÁRBOL EXPANSIÓN MÍNIMA: KRUSKAL

- Algoritmo de Kruskal
  - Para grafos no dirigidos, valorados, de n vértices
  - El árbol reducido tiene n-1 aristas
- Obtención:
  - Partir de un grafo G sin aristas y añadir una cada vez, hasta tener n-1 aristas
  - Ir suprimiendo aristas del grafo de forma que no contenga ningún ciclo y siga siendo conexo
  - Seleccionar cada vez la de menor peso

```
Algoritmo:
 Inicializar arbol(A)
 para cada vi ∈ G
   Incluir vértice vi en A
 mientras Nº aristas(A) < n
   Seleccionar arista a de G con
   menos peso;
   Eliminar arista a de G;
   si a no forma ciclo en A
   entonces
         Incluir arista a en A;
```

# ALGORITMO DE KRUSKAL

| Arista | Costo | Aceptada |
|---|---|---|
| 1, 4 | 1 | ☑ |
| 6, 7 | 1 | ☑ |
| 1, 2 | 2 | ☑ |
| 3, 4 | 2 | ☑ |
| 4, 5 | 2 | ☑ |
| 2, 4 | 3 | x |
| 1, 3 | 4 | x |

1er. Paso: Ordenar las aristas en orden ascendente de costos, usando la tabla que se muestra.

2do. Paso: Ir recorriendo la tabla, marcando las aristas como aceptados o no, dependiendo si al incluirlas forman un ciclo.



# ALGORITMO DE KRUSKAL

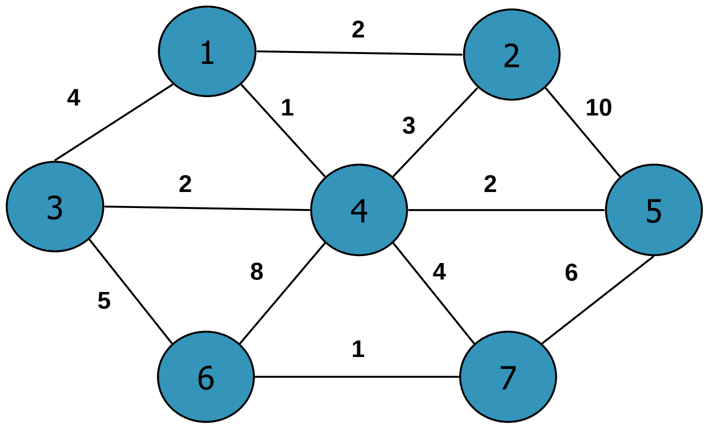
| Arista | Costo | Aceptada |
|---|---|---|
| 1, 4 | 1 | ☑ |
| 6, 7 | 1 | ☑ |
| 1, 2 | 2 | ☑ |
| 3, 4 | 2 | ☑ |
| 4, 5 | 2 | ☑ |
| 2, 4 | 3 | x |
| 1, 3 | 4 | x |

| Arista | Costo | Aceptada |
|---|---|---|
| 4, 7 | 4 | ☑ |
| 3, 6 | 5 | x |
| 5, 7 | 6 | x |
| 4, 6 | 8 | x |
| | | |

**Aristas del árbol de expansión mínima**

**A = { (1,4), (6, 7), (1, 2), (3, 4), (4,5) (4,7) }   COSTO: 12**

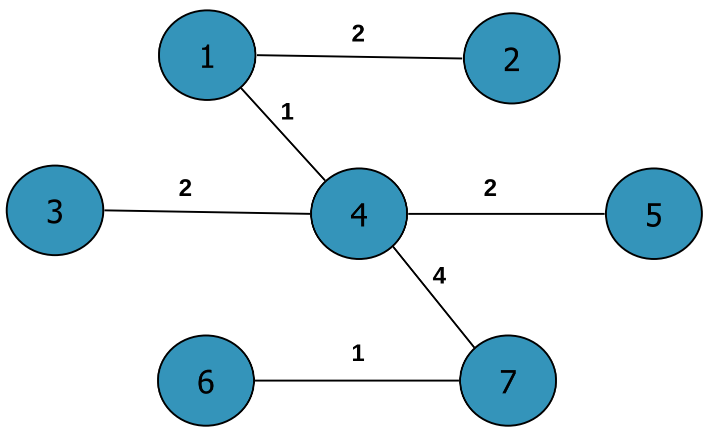
## ARBOL DE COSTO MÍNIMO RESULTANTE




Costo del arbol de expansión de costo mínimo: 12



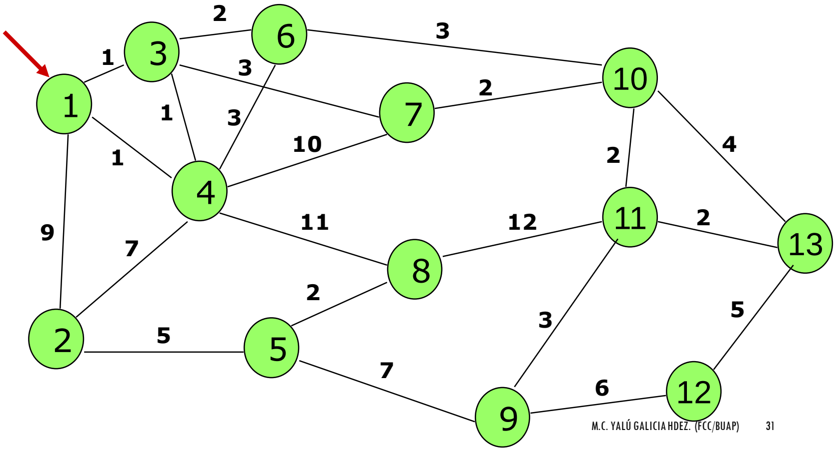
## EJEMPLO USANDO ALGORITMO KRUSKAL




Costo del arbol de expansión de costo mínimo: 12

## ACTIVIDAD INDIVIDUAL

- Obtén el árbol abarcador de costo mínimo para el siguiente grafo usando el algoritmo de Kruskal






## ¿QUÉ HEMOS APRENDIDO?