UNIVERSIDAD DEL BÍO-BÍO

Estructura de Datos

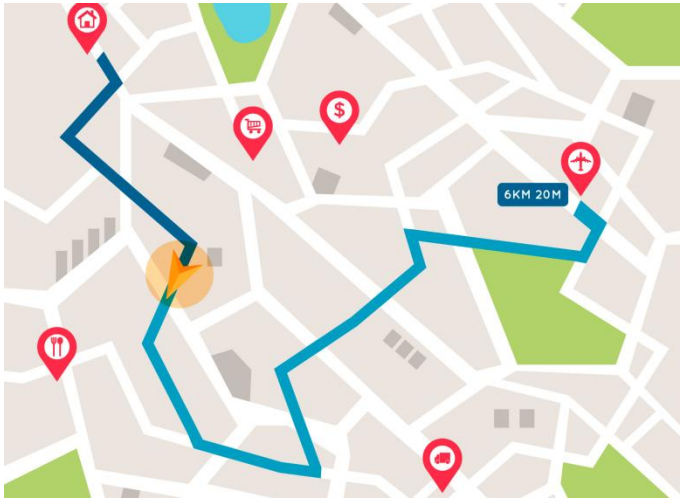
Grafos

Introducción

Los grafos se usan para modelar problemas definidos en términos de relaciones o conexiones entre objetos.

Tienen un amplio uso en ingeniería para representar redes de todo tipo:

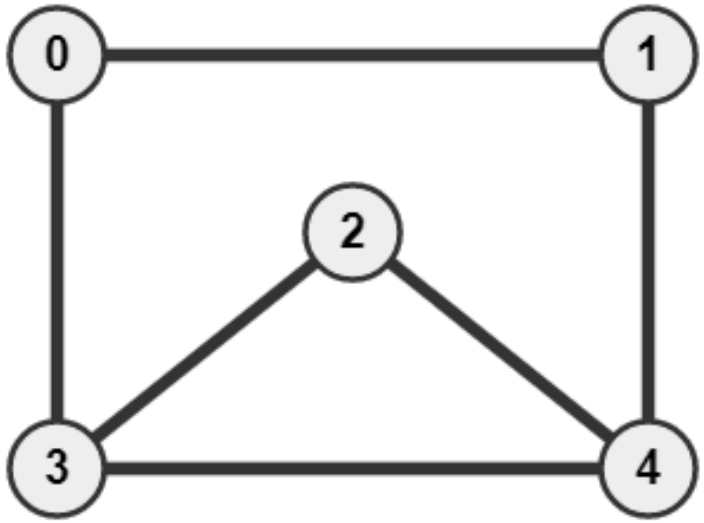
1. Transporte (tren, carretera, avión),
2. Servicios (Comunicación, Eléctrica, Gas, Agua),
3. Actividades en el planeamiento de proyectos.
4. Inteligencia Artificial (Redes Neuronales).



Designed by Freepik

¿Qué es un Grafo?

Un grafo es un conjunto de vértices (Nodos) conectados por medio de aristas o arcos.



Por lo tanto, podemos decir:

Un grafo G = (V, E).
V: Conjunto de vértices.
E: Conjunto de aristas.

V = { 0, 1, 2, 3, 4 }
E = { (0↔1), (0↔3), (1↔4), (4↔2), (4↔3)}

Definiciones

Arista dirigida: Par ordenado (u, v)

Arista **no** dirigida: Par no ordenado (u, v)

Grafo dirigido o digrafo: Grafo cuyas aristas son todas dirigidas.
Grafo no dirigido o grafo: grafo cuyas aristas son todas no dirigidas.

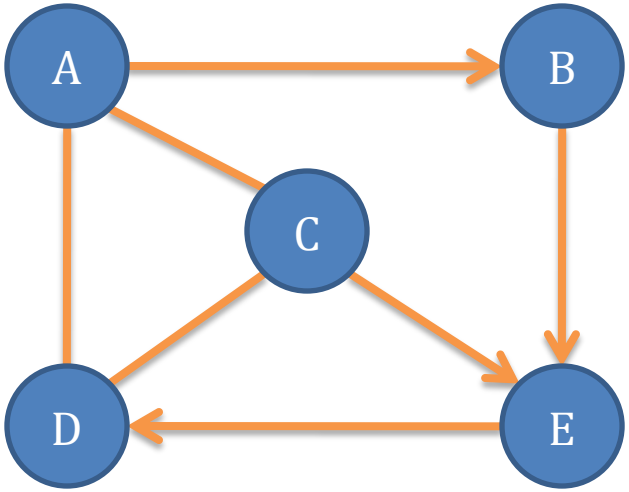
Definiciones

1. **Vértices finales o extremos de la arista:** Vértices unidos por una arista.
2. **Vértice origen:** Primer vértice de una arista dirigida.
3. **Vértice destino:** segundo vértice de una arista dirigida.
4. **Arista incidente en un vértice:** Si el vértice es uno de los vértices de la arista.
5. **Aristas salientes de un vértice:** Aristas dirigidas cuyo origen es ese vértice.
6. **Aristas entrantes de un vértice:** Aristas dirigidas cuyo destino es ese vértice.

Definiciones

Vértices adyacentes: vértices finales de una arista.

1. Un vértice w es adyacente a v sí y sólo si (v, w) (ó (w, v)) pertenece a un conjunto de arista.
2. En grafos no dirigidos la relación de adyacencia es simétrica.
3. En grafos dirigidos la relación de adyacencia no es simétrica.

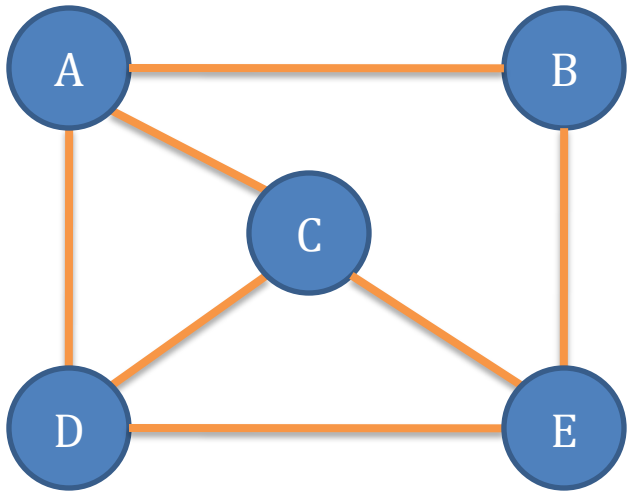


Vértices adyacentes:
A = { B, C, D }
B = { E }
C = { A, D, E }
D = { A, C }
E = { D }

Grado de un Nodo

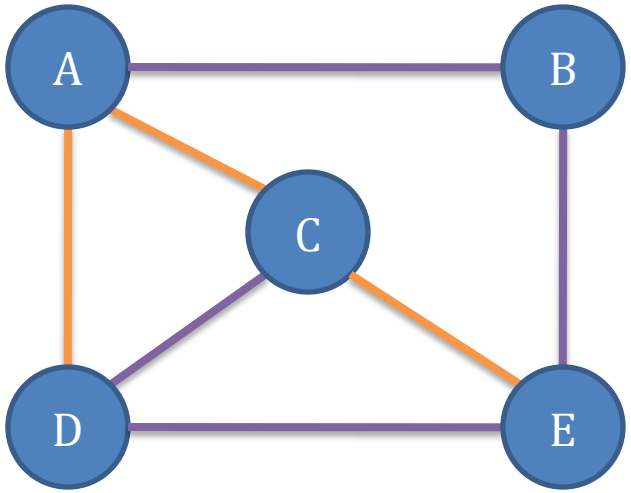
Gr(v): Total de aristas que inciden en v.

Gr(A) = 3
Gr(B) = 2
Gr(C) = 3
....

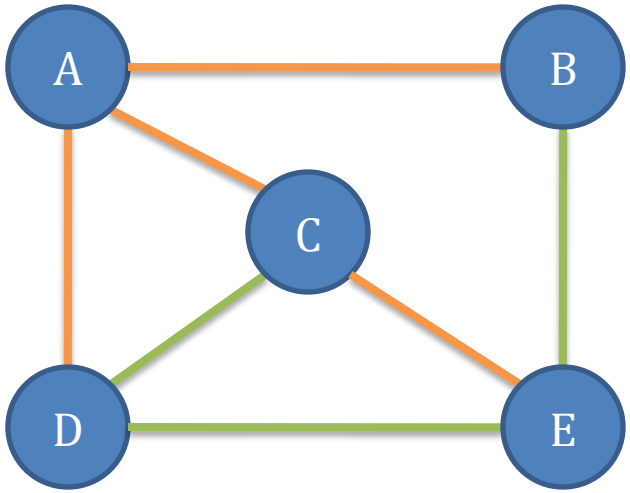


Definiciones

Camino: Secuencia de vértices <v(1), v(2), ..., v(i)> tal que (v(i), v(i+1)) son adyacentes.



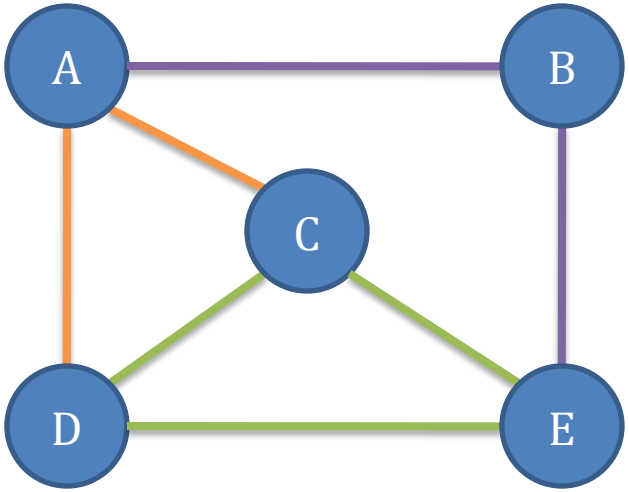
Camino 1 = { A, B, E, D, C}



Camino 2 = { B, E, D, C}

Definiciones

Camino simple: Todos los vértices son distintos.
Longitud de un camino: Número de aristas del camino = n – 1.
Ciclo: camino simple que tiene el mismo vértice inicial y final.



Camino Simple = { A, B, E }
Ciclo = { C, E, D, C}

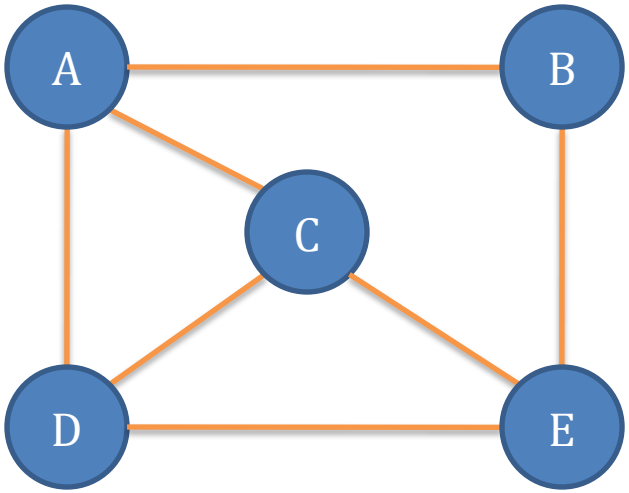
Observación 1: A menudo sólo se consideran ciclos para n ≥ 3.
Observación 2: Un grafo se dice cíclico cuando contiene algún ciclo como subgrafo

Definiciones

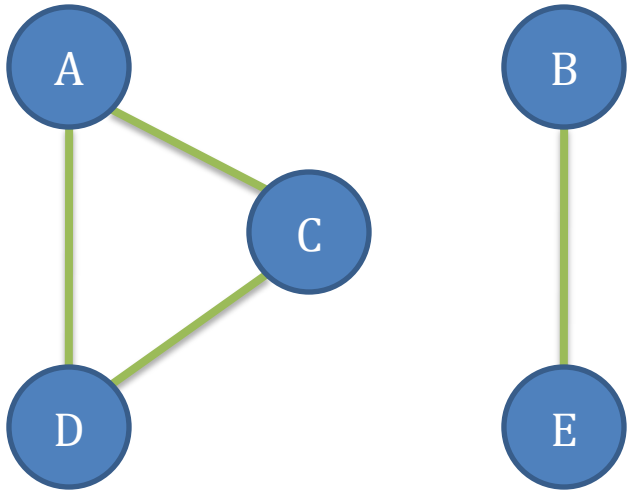
Dos vértices v, w están **conectados** si existe un camino de v a w.

Grafo conectado (conexo): Si hay un camino entre cualquier par de vértices.

Si es un grafo dirigido se llama fuertemente conexo.



Conectado



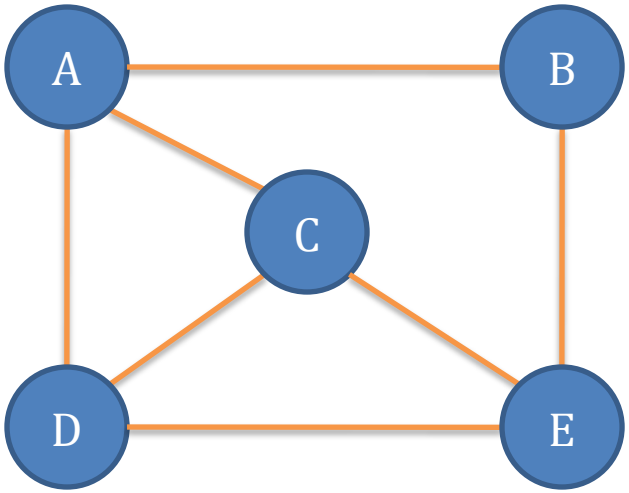
No conectado

Sub-Grafos

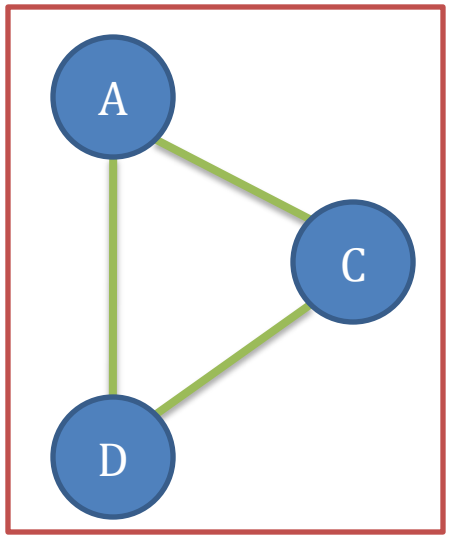
Sea G=(V,A). G'=(V',A') se dice sub-grafo de G si:

1. V' ⊆ V
2. A' ⊆ A
3. (V',A') es un grafo

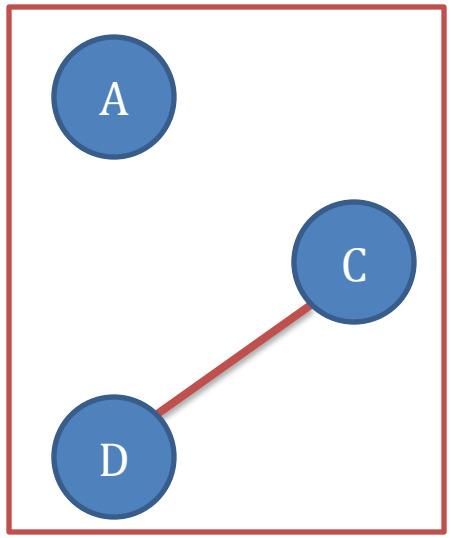
G1 y G2 son sub-grafo de G.



G



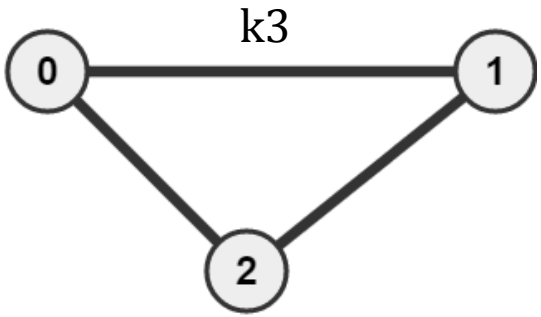
G1

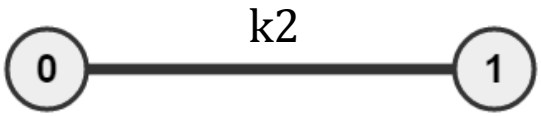


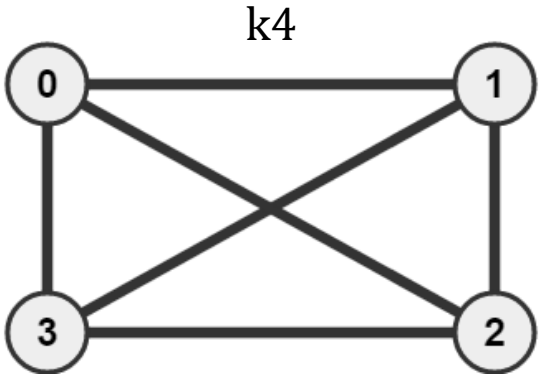
G2

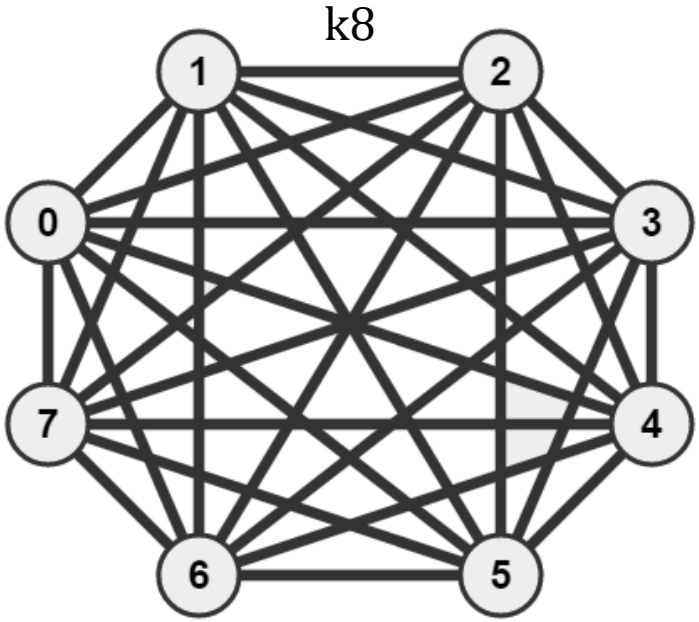
Kn

Para cada n ≥ 1 se llama grafo completo de orden n, y se representa por **Kn**, al grafo de n vértices en donde todos sus vértices están conectados entre sí.









Grafo Complementario

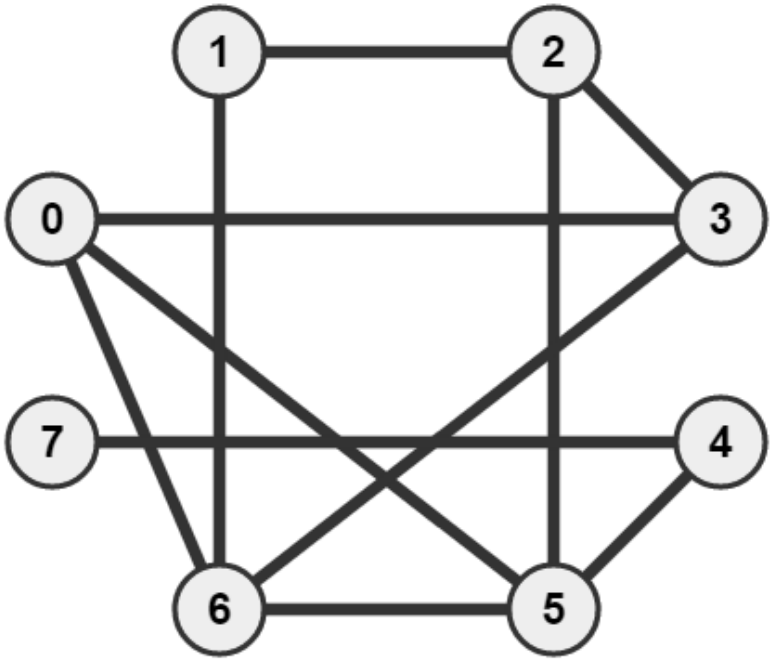
El complementario G' de un grafo G=(V, A) tiene:

1. Los mismos vértices que G
2. Si $\{u,v\} \in G$, entonces $\{u,v\} \notin G'$
3. Si $\{u,v\} \notin G$, entonces $\{u,v\} \in G'$

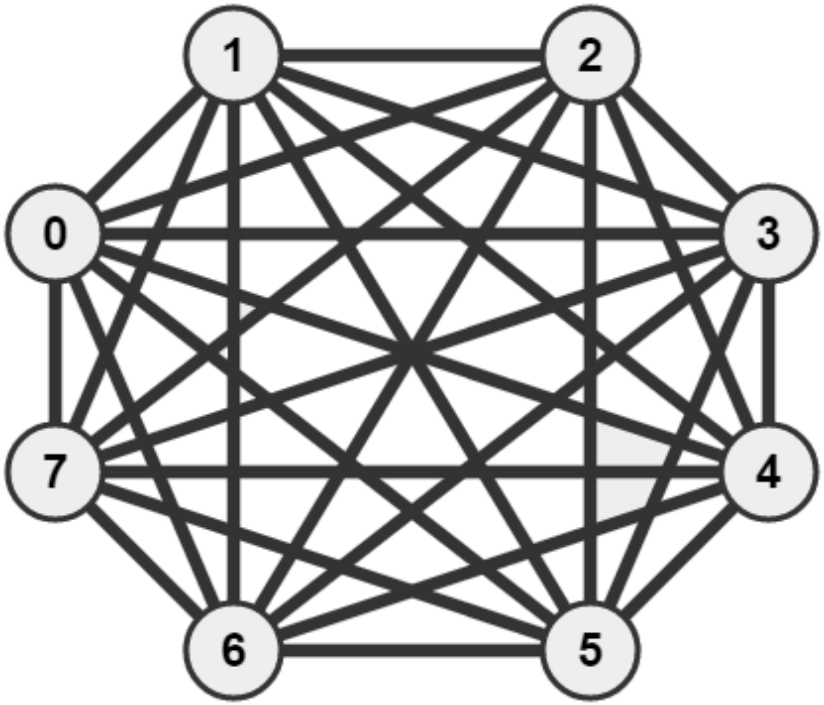
Una forma de construirlo:

1. Dibujar el correspondiente grafo completo Kn, con n=|V|
2. Eliminar de Kn las aristas $\{u,v\} \in G'$

Grafo Complementario

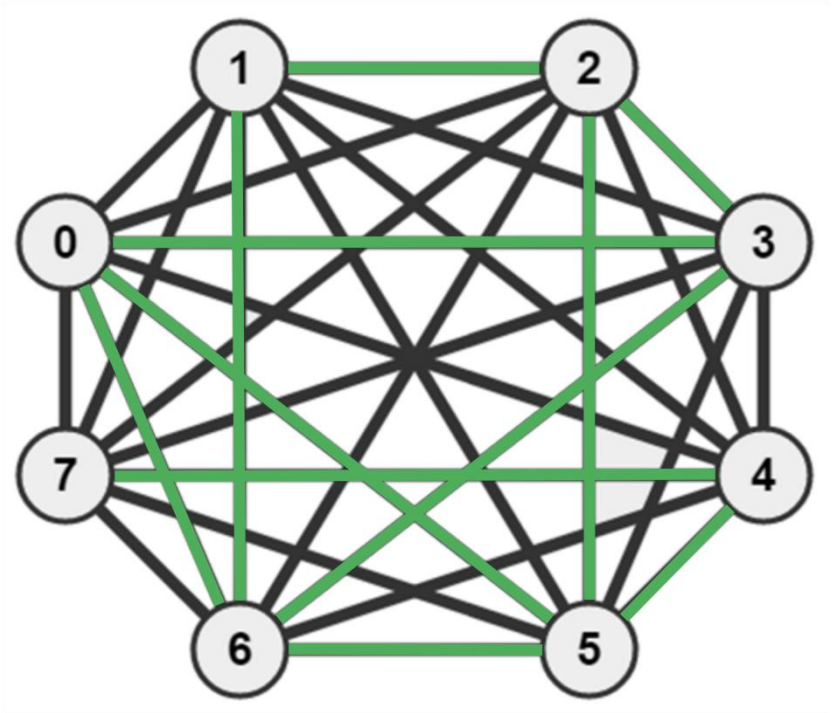


G



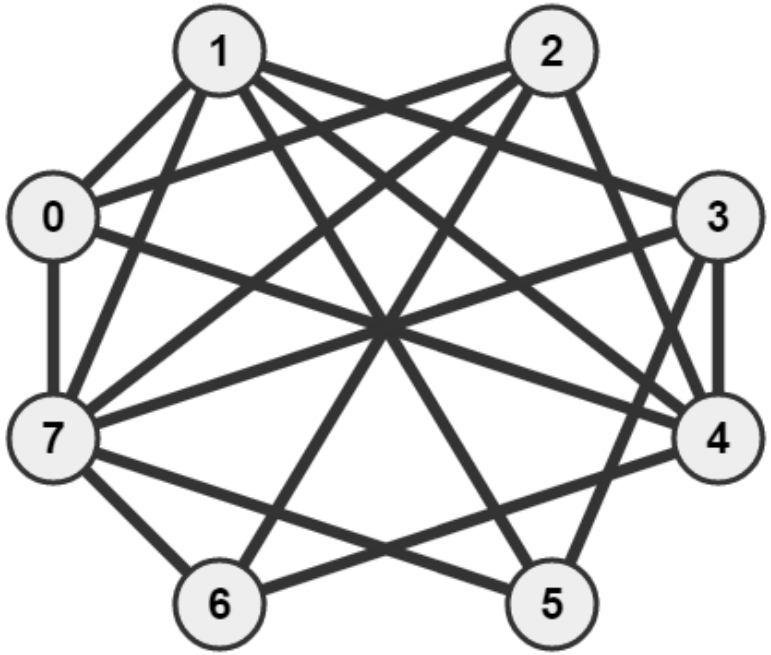
Generar K8

Grafo Complementario



Marcar las aristas
de G

Complemento(G)

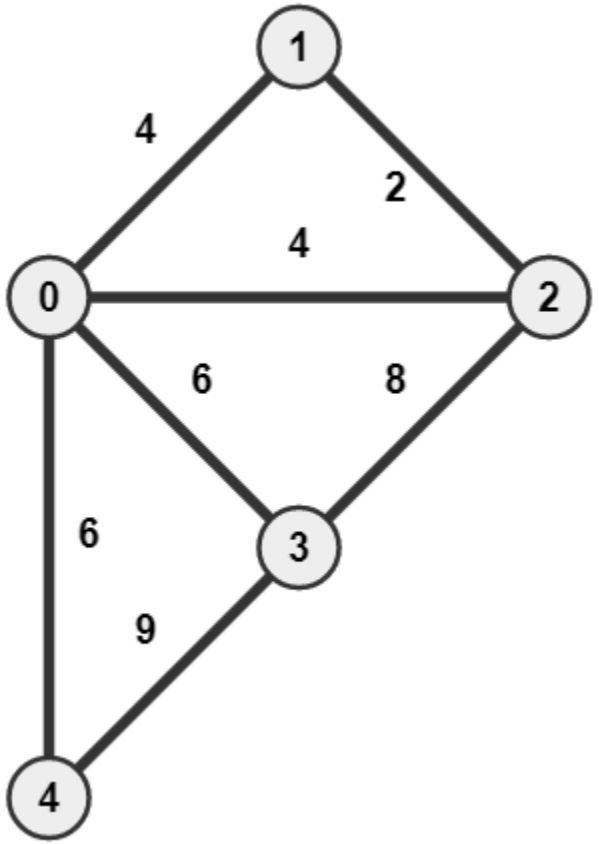


Eliminar aristas
marcadas.

Grafo Etiquetado

Un grafo está etiquetado si asociamos a cada arista un peso o valor.

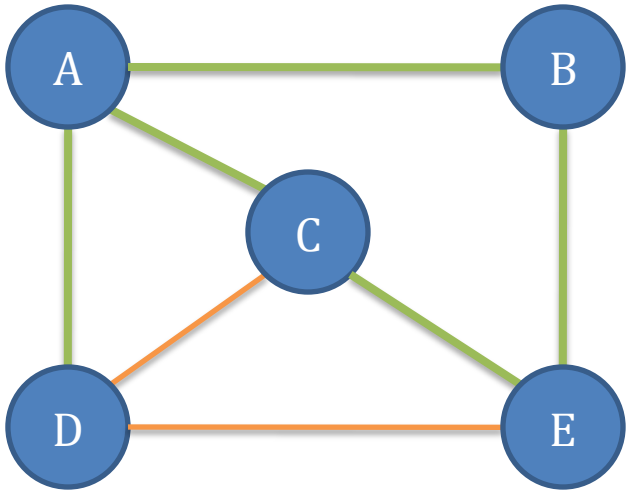
Grafo con pesos: grafo etiquetado con valores numéricos.



Caminos y Conectividad

1. Un recorrido en un grafo G = (V, A) es una sucesión de vértices v(0), v(1), …, v(k) tal que {v(i), v(i+1)} ∈ A para todo $0 \leq i < k$.
2. La longitud de un recorrido v(0), v(1), …, v(k) es k

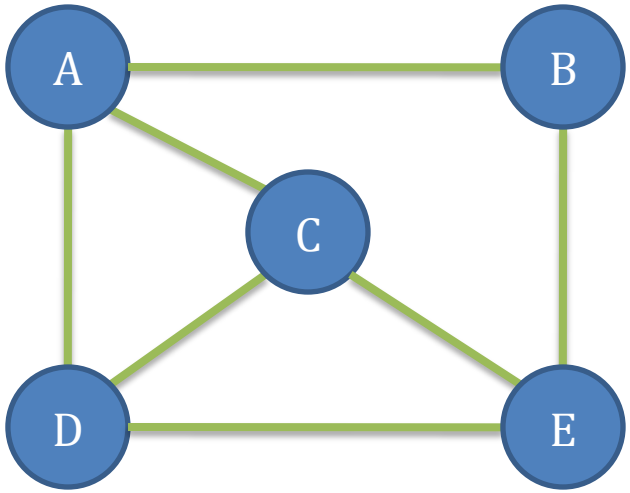
Ejemplo: A, B, E, C, A, D
Recorrido Longitud : 5



Caminos y Conectividad

1. Un recorrido v(0), v(1), ...,v(k) tal que v(0) = v(k) es un circuito.
2. Un camino v(0), v(1), ..., v(k) tal que v(0) = v(k) es un ciclo.

Circuito : A, B, E, C, A, D, C, A.
Ciclo: A, B, C, E, A.



Caminos y Conectividad

Observación: Un recorrido puede repetir vértices, y puede comenzar y acabar en vértices diferentes.

Un camino es un recorrido en donde todos los vértices son distintos entre sí, exceptuando la posibilidad en que el último vértice sea el primero.

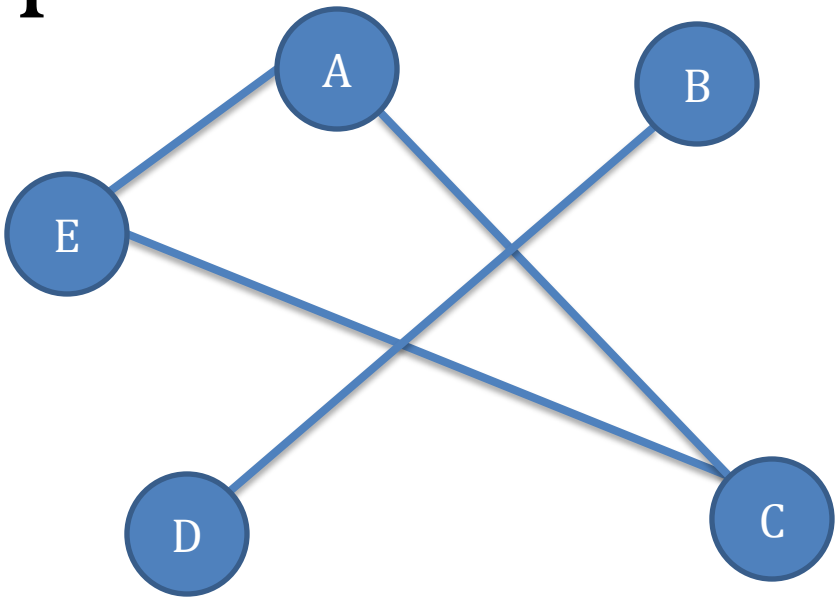
Si existe un camino entre dos vértices se dice que están conectados.

Si para todo par de vértices de un grafo están conectados se dice que el grafo es conexo g.

Las **componentes conexas** de un grafo G son los mayores sub-grafos conexos de G.

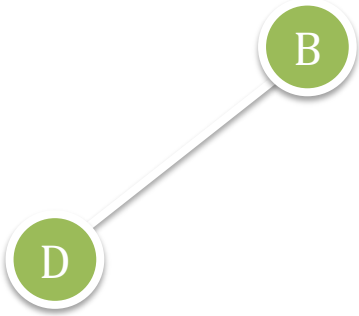
Ejemplo

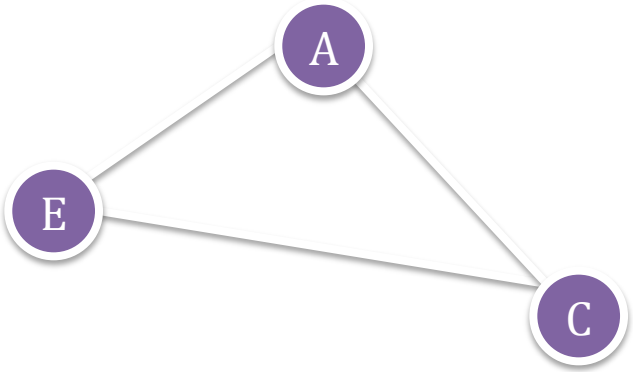
Considerando el siguiente grafo G:



Se tiene que:
G no es conexo, no hay camino entre a y b, por ejemplo.

1. [a] = {a, c, e}
2. [c] = {a, c, e}
3. [e] = {a, c, e}
4. [b] = {b, d}
5. [d] = {b, d}
6. G/R = {[a],[b]}
7. G tiene dos componentes conexas C1 y C2:

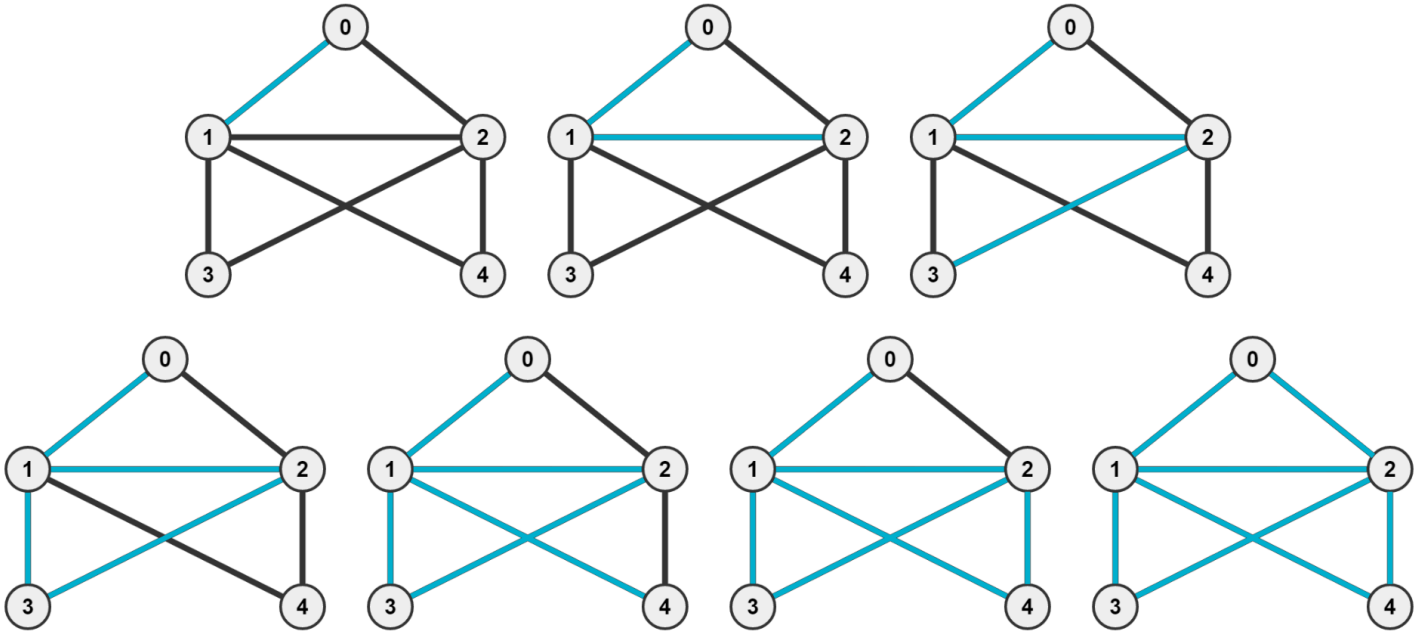




Recorridos Eulerianos
Grafo o multi-grafo euleriano

1. Admite un recorrido que pasa por todas las aristas una sola vez, empezando y terminando en el mismo vértice.
2. Los vértices sí se pueden repetir.

Circuito euleriano: 0, 1, 2, 3, 1, 4, 2, 0.

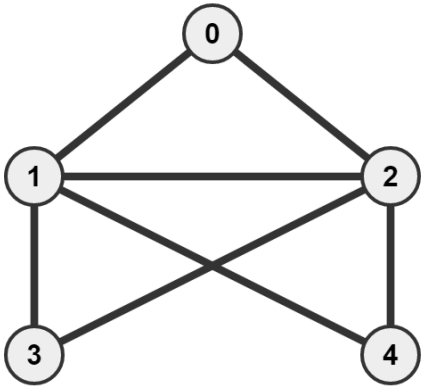


Recorridos Eulerianos

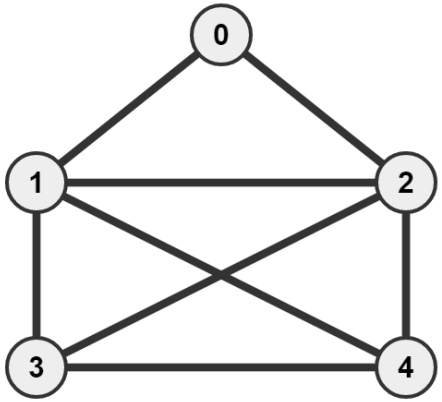
¿Cómo saber si un grafo (o multigrafo) es euleriano?

Teorema de Euler: Un grafo conexo es euleriano , si no tiene vértices de grado impar.

Ejemplo:
G1 tiene grados 2 y 4, es euleriano.
G2 tiene grados 2, **3**, 4, **no** es euleriano.



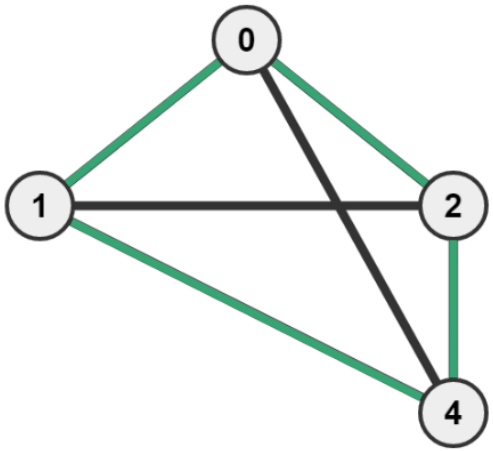
G1

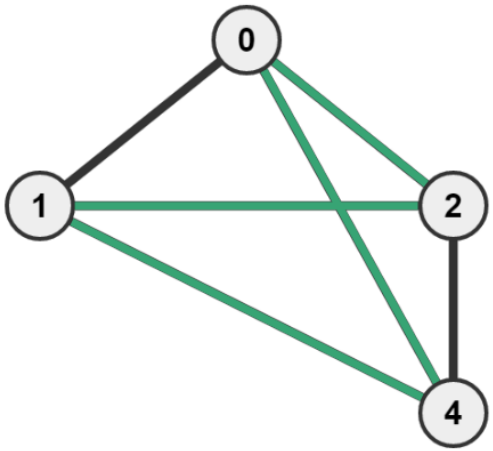


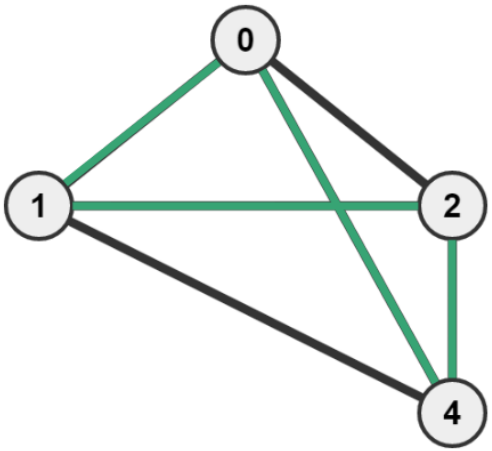
G2

Recorridos Halmiltonianos

Un grafo se dice hamiltoniano si existe un ciclo que recorre todos sus vértices. Al ciclo se le llama ciclo hamiltoniano.







Estructuras de Datos para Grafo

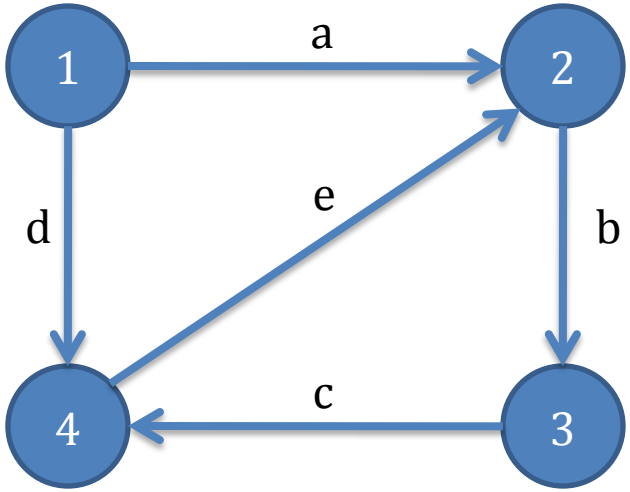
Se necesita almacenar los vértices y las aristas del grafo y realizar eficientemente las operaciones del TDA Grafo.

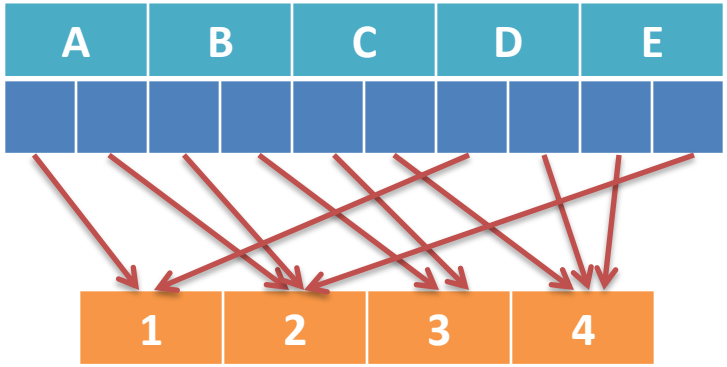
Las estructuras de datos usuales son:

1. Lista de aristas.
2. Lista de adyacencia.
3. Matriz de adyacencia.

Lista de aristas

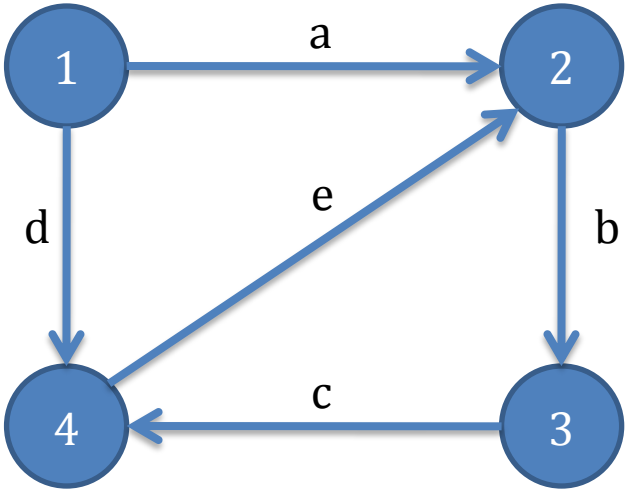
1. La estructura almacena los vértices y las aristas en secuencias sin ordenar.
2. Hallar las aristas incidentes sobre un determinado vértice es ineficiente porque requiere el examen entero de la estructura que almacena las aristas.

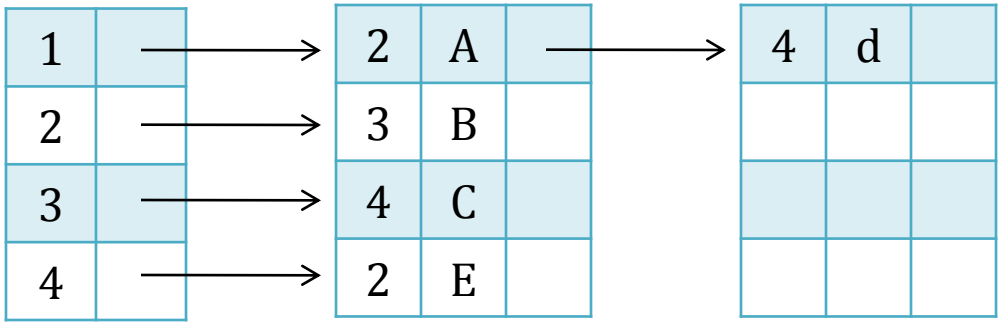




Lista de adyacencia

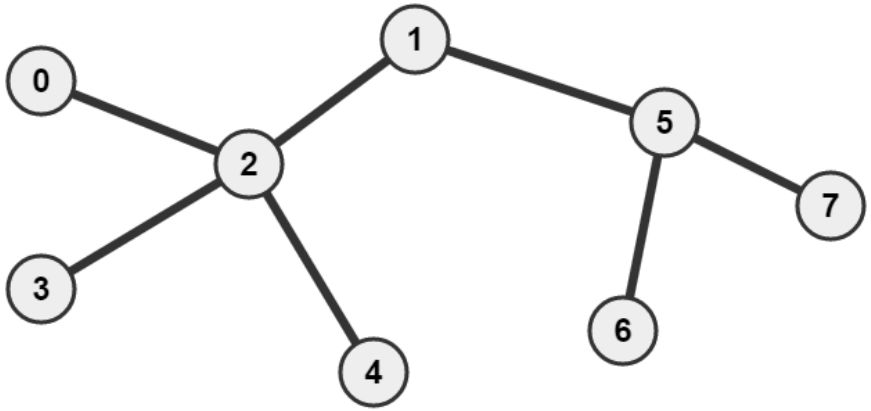
1. Lista de adyacencia del vértice v: Secuencia de vértices adyacentes a v.
2. Representa el grafo por las listas de adyacencia de todos los vértices.
3. Es la estructura más usada para representar grafos con pocas aristas (dispersos).

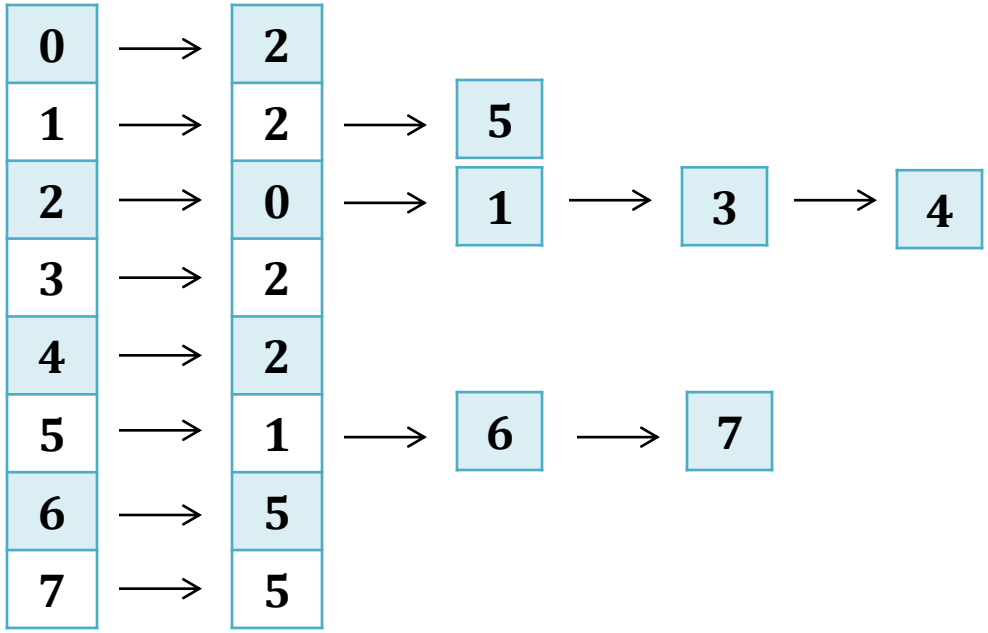




Lista de adyacencia

A cada vértice le corresponde una lista con sus adyacentes:

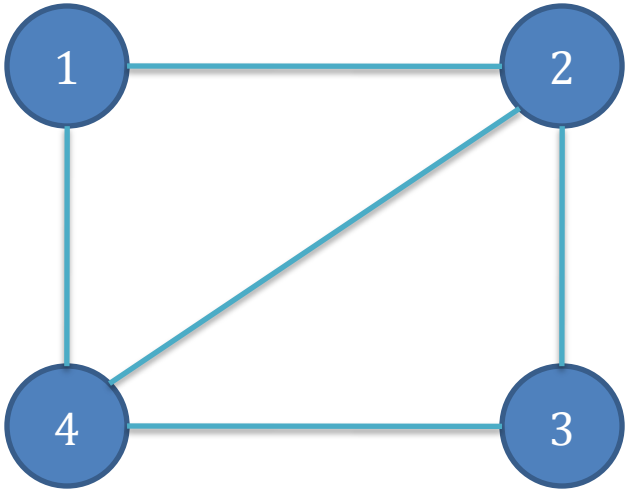




Matriz de adyacencia

Matriz M[i][j] con entradas para todos los pares de vértices.

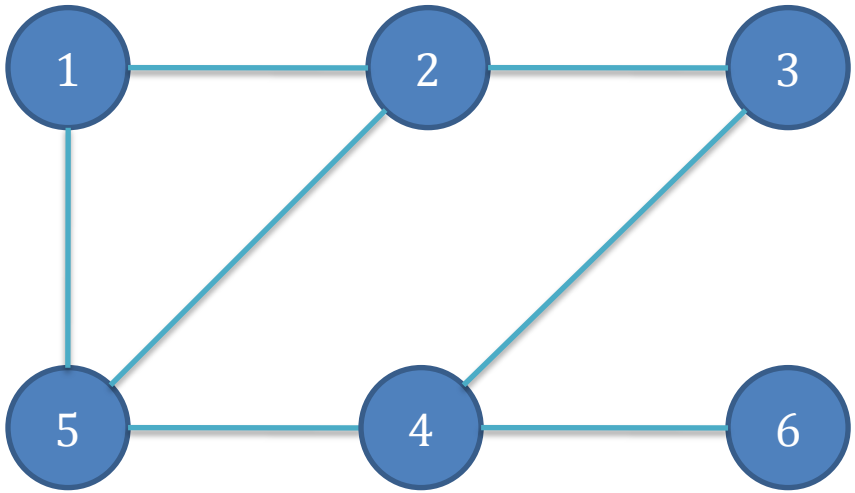
1. En grafos no etiquetados:
   a. M[i][j] = verdadero, si hay una arista (i, j) en el grafo.
   b. M[i][j] = falso, si no hay una arista (i, j) en el grafo.
2. En grafos no dirigidos: M[i][j] = M[j][i]. La matriz es simétrica.



	1	2	3	4
1	**0**	**1**	**0**	**1**
2	**1**	**0**	**1**	**1**
3	**0**	**1**	**0**	**1**
4	**1**	**1**	**1**	**0**

Matriz de adyacencia

Se construye imaginando que en las filas y las columnas corresponden a los vértices. Se pone un 0 para indicar que 2 vértices no son adyacentes, y un 1 para indicar que sí lo son:



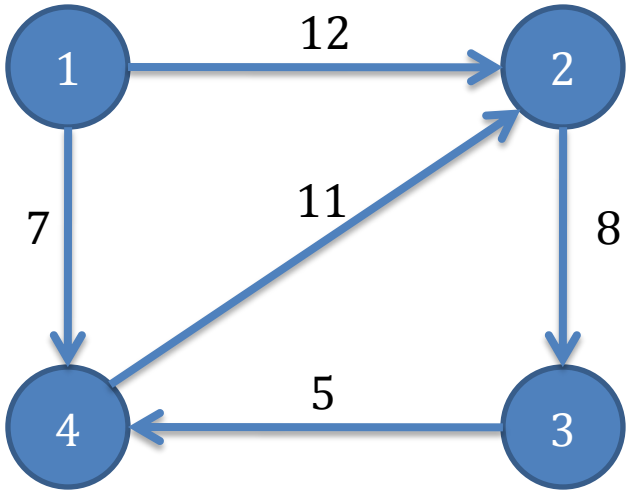
	1	2	3	4	5	6
1	**0**	**1**	**0**	**0**	**1**	**0**
2	**1**	**0**	**1**	**0**	**1**	**0**
3	**0**	**1**	**0**	**1**	**0**	**0**
4	**0**	**0**	**1**	**0**	**1**	**1**
5	**1**	**1**	**0**	**1**	**0**	**0**
6	**0**	**0**	**0**	**1**	**0**	**0**

Matriz de adyacencia

En grafos etiquetados:

M[i][j] = atributo de la arista (i, j) en el grafo, indicador especial si no hay una arista (i, j).

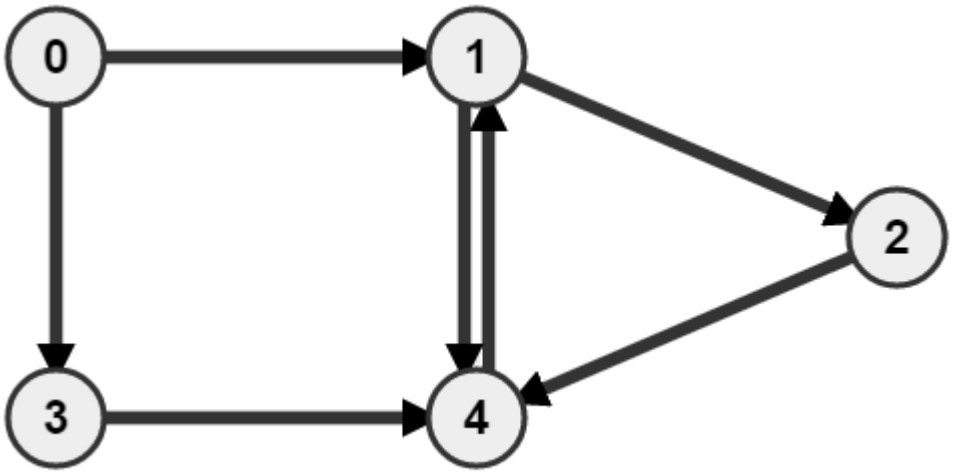
Es la estructura más usada para representar grafos con muchas aristas (densos).



	1	2	3	4
1	-	**12**	-	**7**
2	-	-	**8**	-
3	-	-	-	**5**
4	-	**11**	-	-

Representación de Grafos

En el caso de un grafo no dirigido la matriz será simétrica. No ocurre lo mismo para grafos dirigidos:

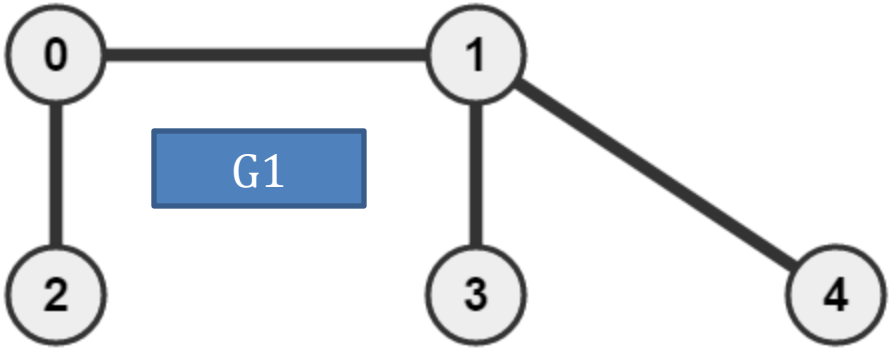


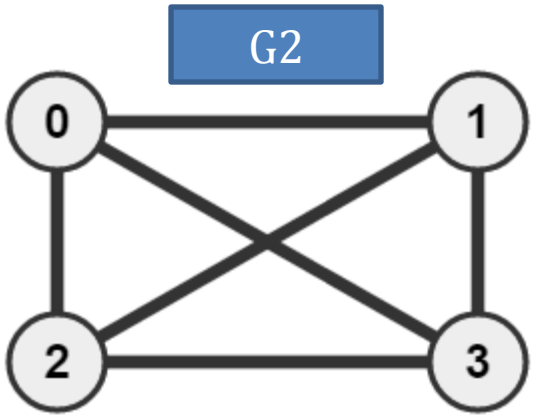
	0	1	2	3	4
0	**0**	**1**	**0**	**1**	**0**
1	**0**	**0**	**1**	**0**	**1**
2	**0**	**0**	**0**	**0**	**1**
3	**0**	**0**	**0**	**0**	**1**
4	**0**	**1**	**0**	**0**	**0**

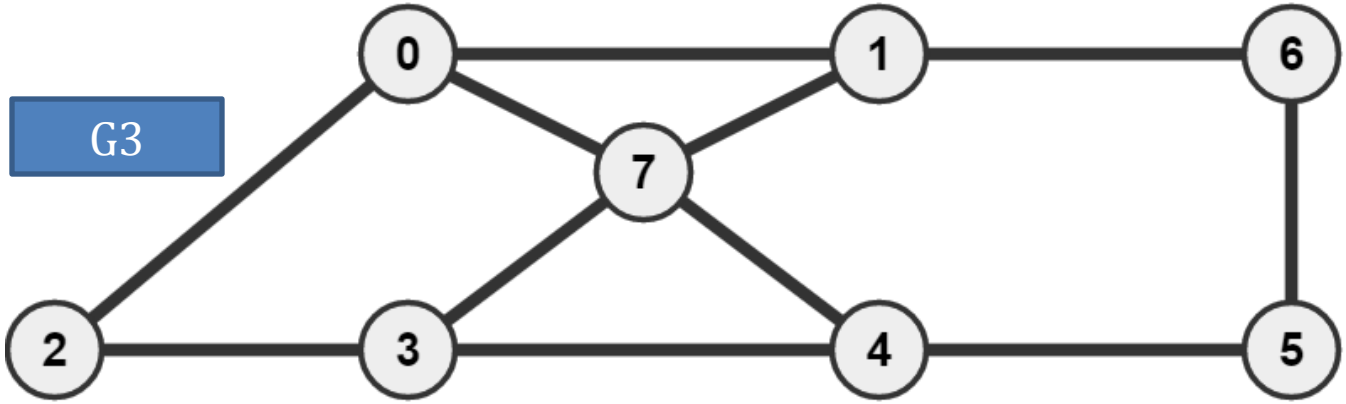
Se supone que la fila representa el vértice origen, y la columna el vértice destino del arco.

Ejercicios

Construya las matrices de adyacencia basado en los siguientes grafos:







Soluciones

G1

	0	1	2	3	4
0	**0**	**1**	**1**	**0**	**0**
1	**1**	**0**	**0**	**1**	**1**
2	**1**	**0**	**0**	**0**	**0**
3	**0**	**1**	**0**	**0**	**0**
4	**0**	**1**	**0**	**0**	**0**

G2

	0	1	2	3
0	**0**	**1**	**1**	**1**
1	**1**	**0**	**1**	**1**
2	**1**	**1**	**0**	**1**
3	**1**	**1**	**1**	**0**

G3

	0	1	2	3	4	5	6	7
0	**0**	**1**	**1**	**0**	**0**	**0**	**0**	**1**
1	**1**	**0**	**0**	**0**	**0**	**0**	**1**	**1**
2	**1**	**0**	**0**	**1**	**0**	**0**	**0**	**0**
3	**0**	**0**	**1**	**0**	**1**	**0**	**0**	**1**
4	**0**	**0**	**0**	**1**	**0**	**1**	**0**	**1**
5	**0**	**0**	**0**	**0**	**1**	**0**	**1**	**0**
6	**0**	**1**	**0**	**0**	**0**	**1**	**0**	**0**
7	**1**	**1**	**0**	**1**	**1**	**0**	**0**	**0**

Ejercicio

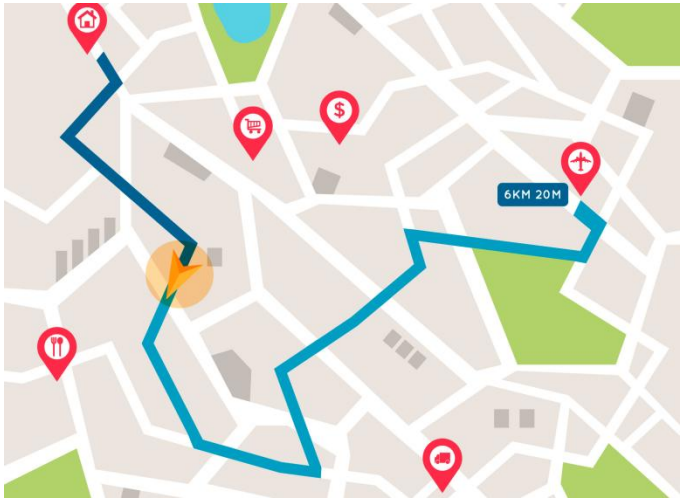
	1	2	3	4	5	6
1	**0**	**0**	**1**	**1**	**0**	**1**
2	**0**	**0**	**1**	**1**	**1**	**0**
3	**1**	**1**	**0**	**0**	**1**	**1**
4	**1**	**1**	**0**	**0**	**1**	**0**
5	**0**	**1**	**1**	**1**	**0**	**0**
6	**1**	**0**	**1**	**0**	**0**	**0**

1. Dibuje el grafo asociado.
2. ¿Es un grafo conexo?
3. ¿Es un grafo completo(Kn)?
4. ¿Cuál es el nodo con mayor grado?

UNIVERSIDAD DEL BÍO-BÍO

Dijkstra

Algoritmo de ruta más corta



Dijkstra

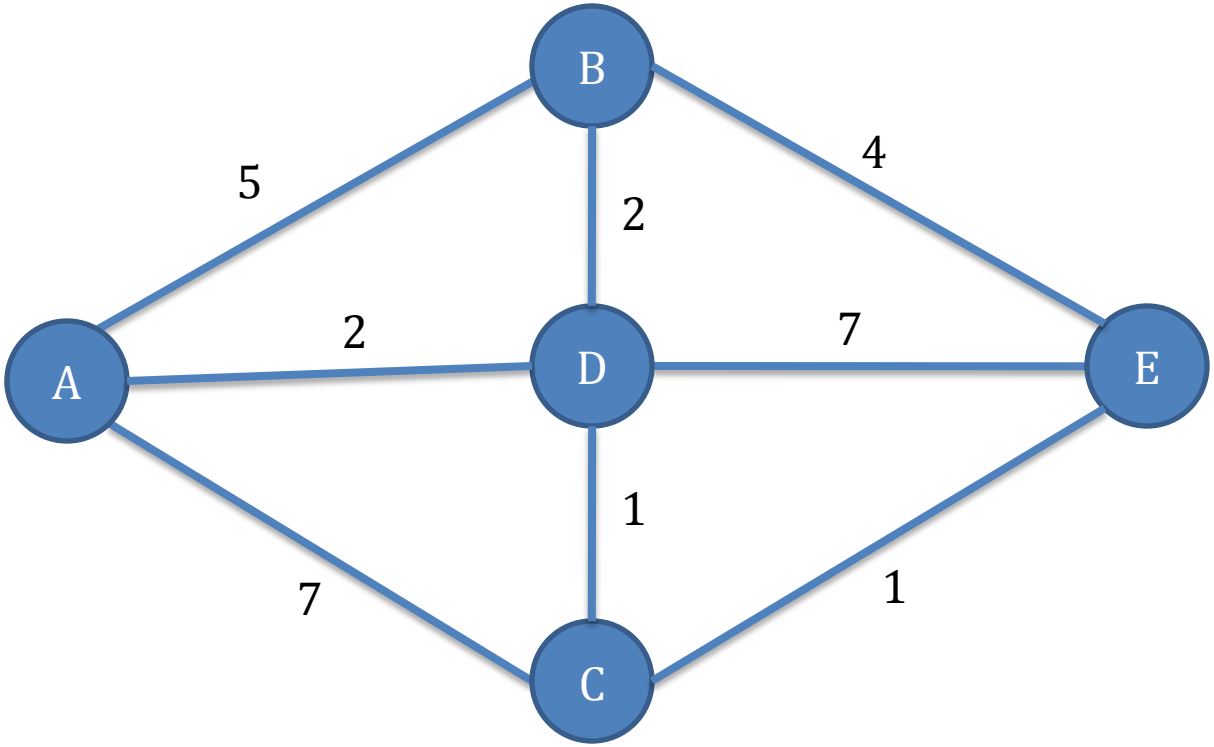
Es un algoritmo para encontrar la ruta más corta desde un nodo de **inicio** a un nodo de **destino** en un grafo **ponderado**. El algoritmo crea un árbol de rutas más cortas desde el vértice inicial, la fuente, a todos los otros puntos.

Observación: El grafo puede ser dirigido o no dirigido. Un pre-requisito para usar el algoritmo es que deberá tener un peso **no negativo** en cada arista.

Dijkstra - Algoritmo

1. La idea del algoritmo es mantener un conjunto A de nodos "alcanzables" desde el nodo origen, e ir extendiendo este conjunto en cada iteración.
2. Los nodos alcanzables son aquellos para los cuales ya se ha encontrado su camino óptimo desde el nodo origen. Para esos nodos su distancia óptima al origen es conocida.

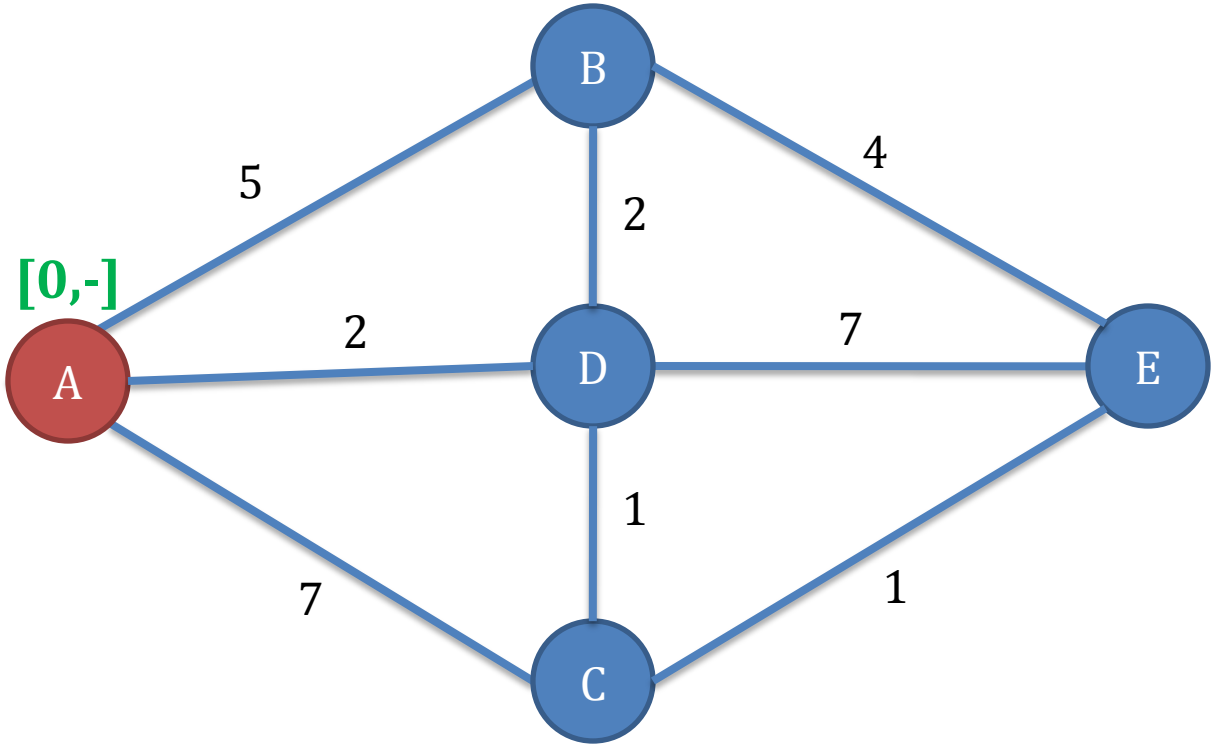
Ejemplo
Encontrar camino mínimo entre **A** y **E**



Ejemplo
Encontrar camino mínimo entre **A** y **E**

Cola de Prioridad – Menor a Mayor

A(0, -)									

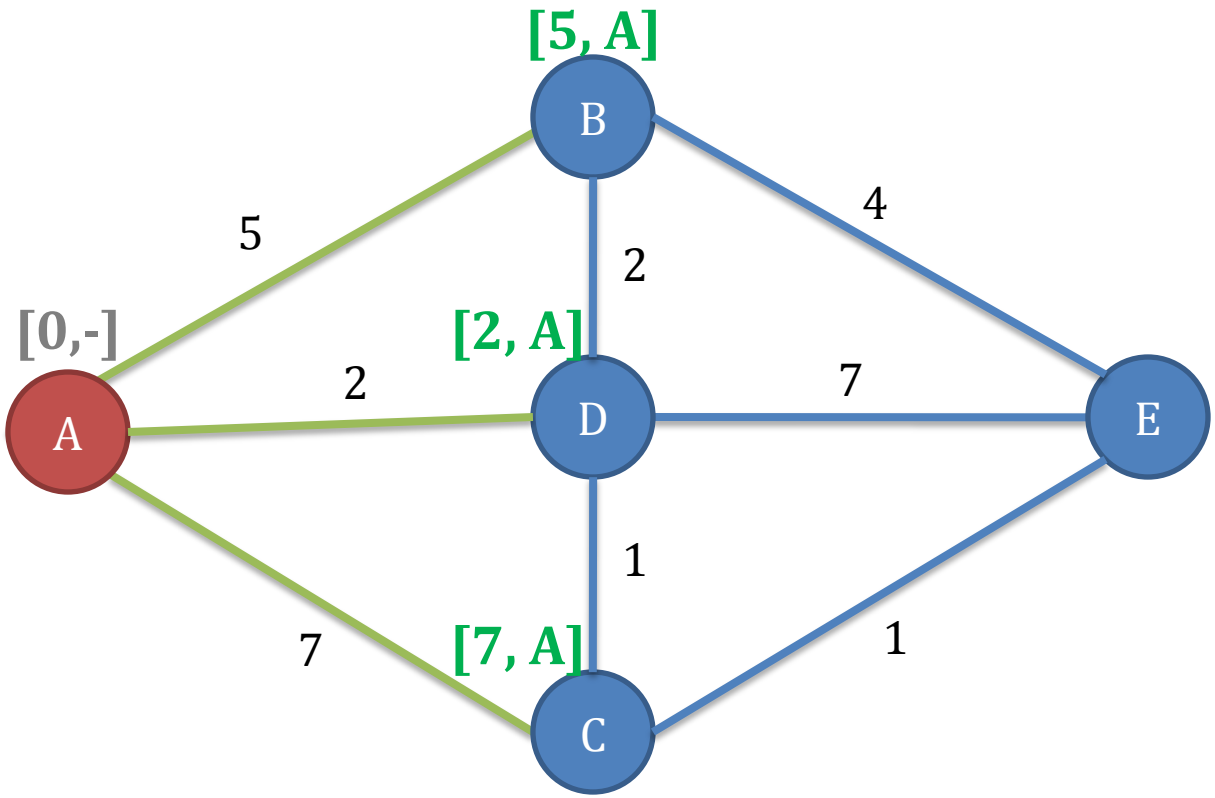


Se inicia desde el Nodo Origen, indicando con una etiqueta:
[Distancia Acumulada desde Nodo Origen, Nodo Predecesor].

Se busca todos los vecinos de A y se asigna la suma acumulada del camino.

Cola de Prioridad – Menor a Mayor

D(2, A)	B(5, A)	C(7, A)							

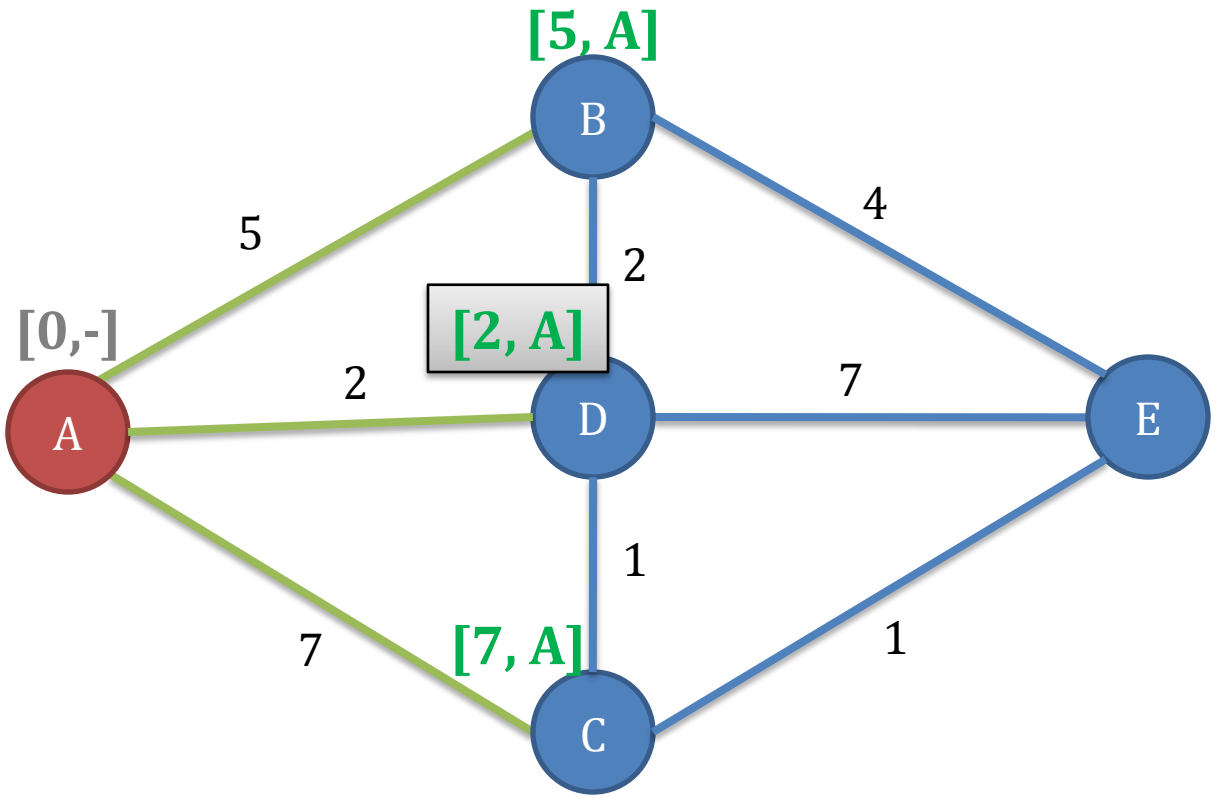


Para cada nodo alcanzado asignar etiqueta **[Distancia Acumulada, Nodo Predecesor]**

Seleccionar Nodo con menor valor recorrido.

Cola de Prioridad – Menor a Mayor

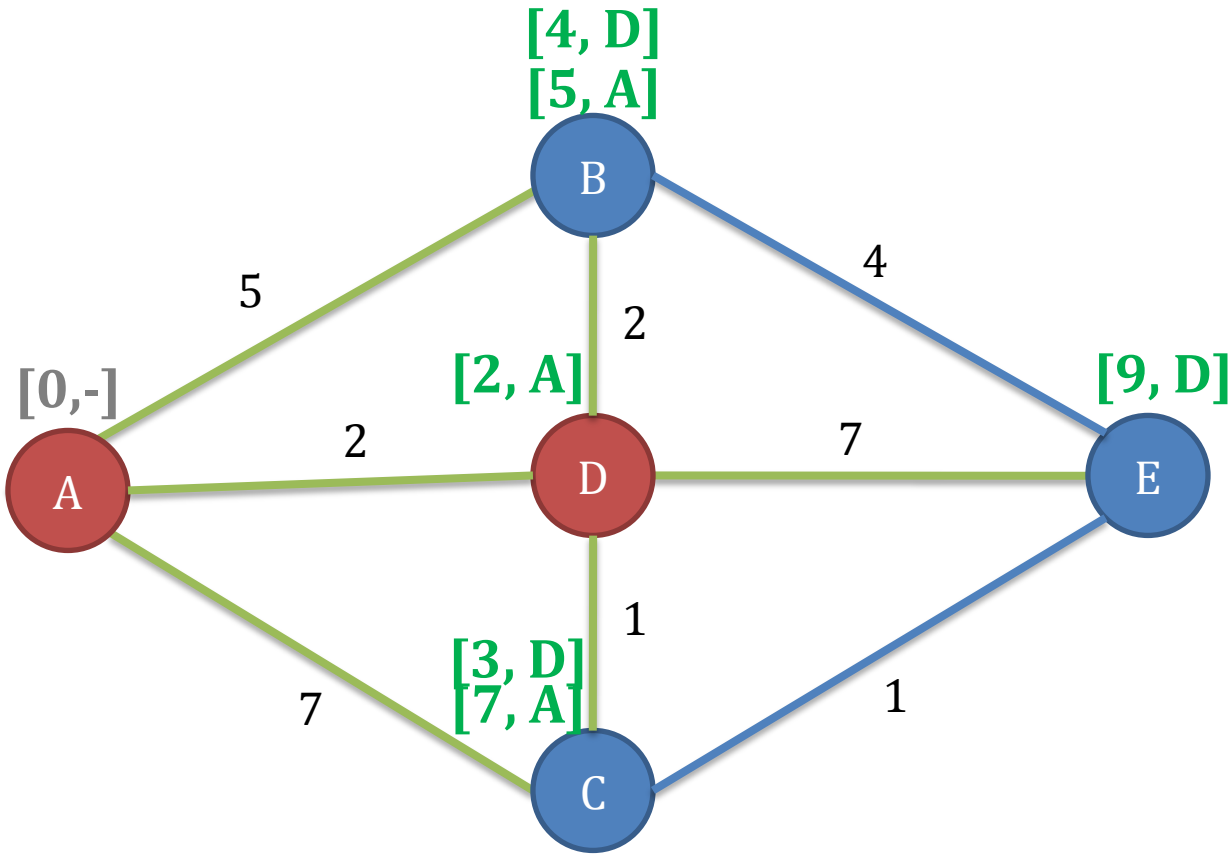
D(2, A)	B(5, A)	C(7, A)							



Expandir desde Nodo D a sus vecinos (Ignorar caminos ya recorridos)

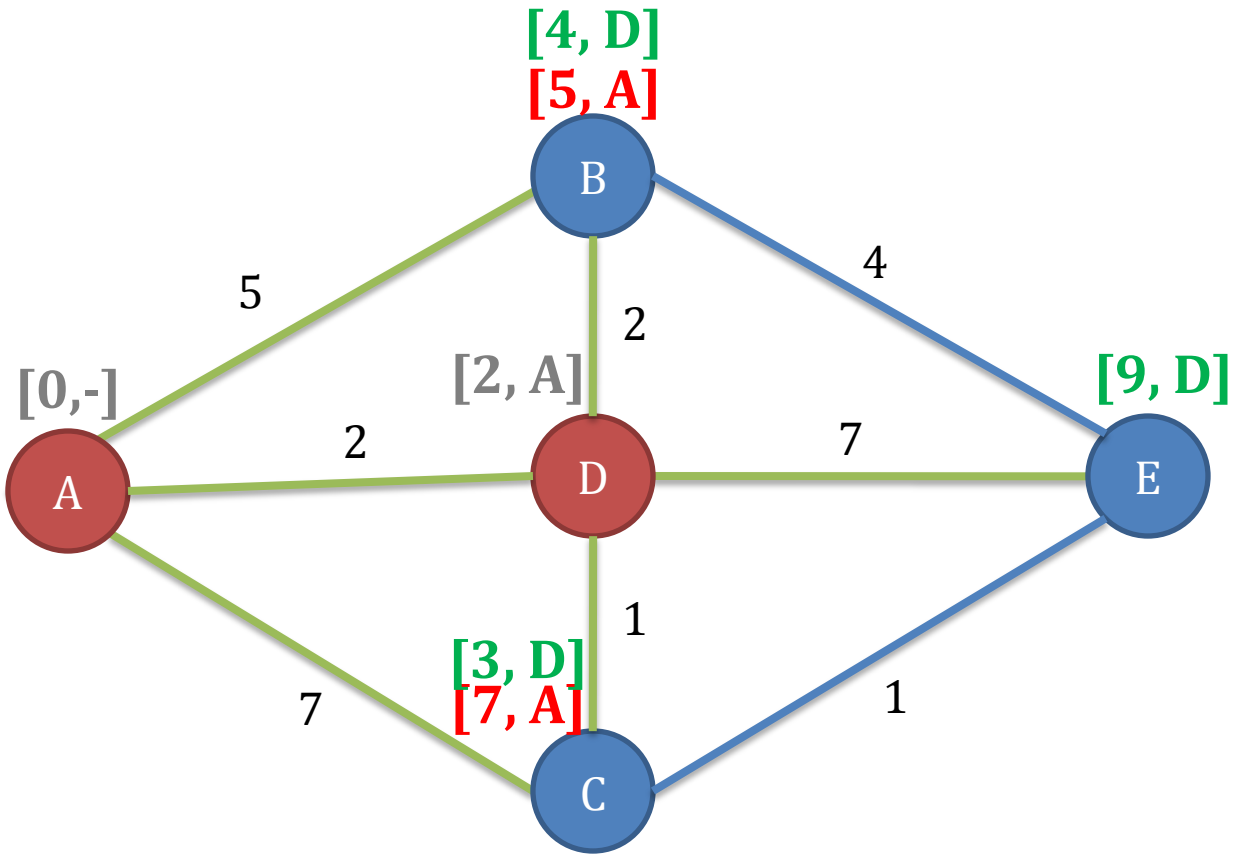
Cola de Prioridad – Menor a Mayor

C(3, D)	B(4, A)	B(5, A)	C(7, A)						



Para cada nodo alcanzado asignar etiqueta **[Distancia Acumulada, Nodo Predecesor]**

Descartar caminos con mayor costo (En los que haya más de una etiqueta por Nodo)

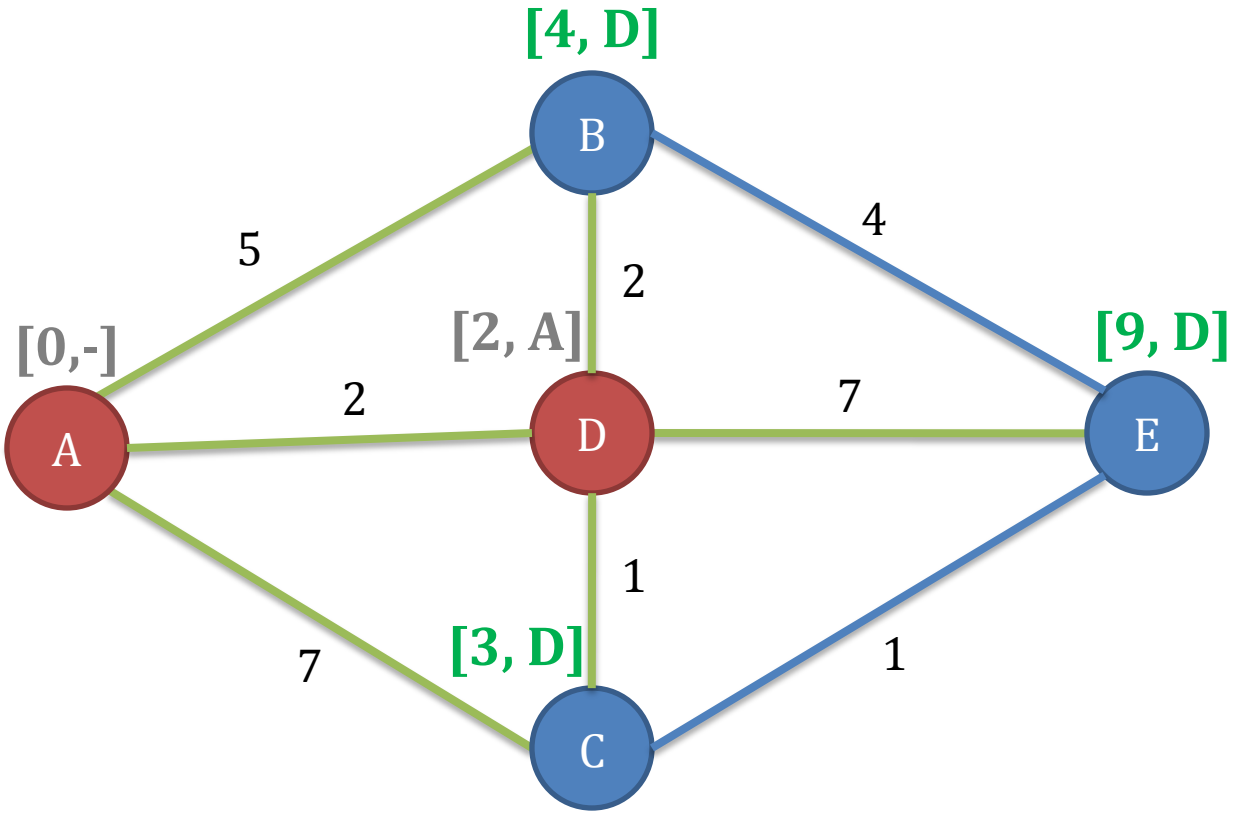


Para cada nodo alcanzado asignar etiqueta **[Distancia Acumulada, Nodo Predecesor]**

Seleccionar Nodo con menor valor

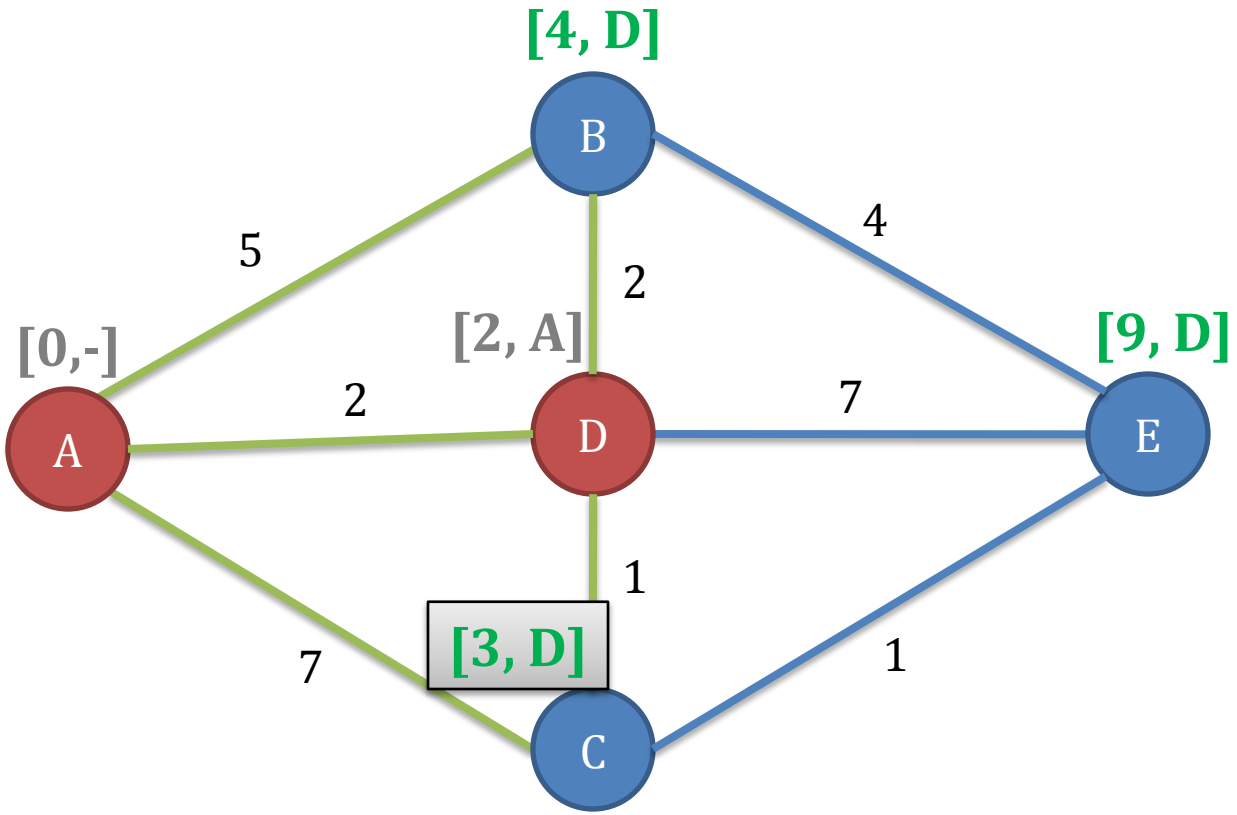
Cola de Prioridad – Menor a Mayor

C(3, D)	B(4, A)	B(5, A)	C(7, A)	E(9, D)					



Para cada nodo alcanzado asignar etiqueta **[Distancia Acumulada, Nodo Predecesor]**

Expandir desde Nodo C a sus vecinos (Ignorar caminos ya recorridos)

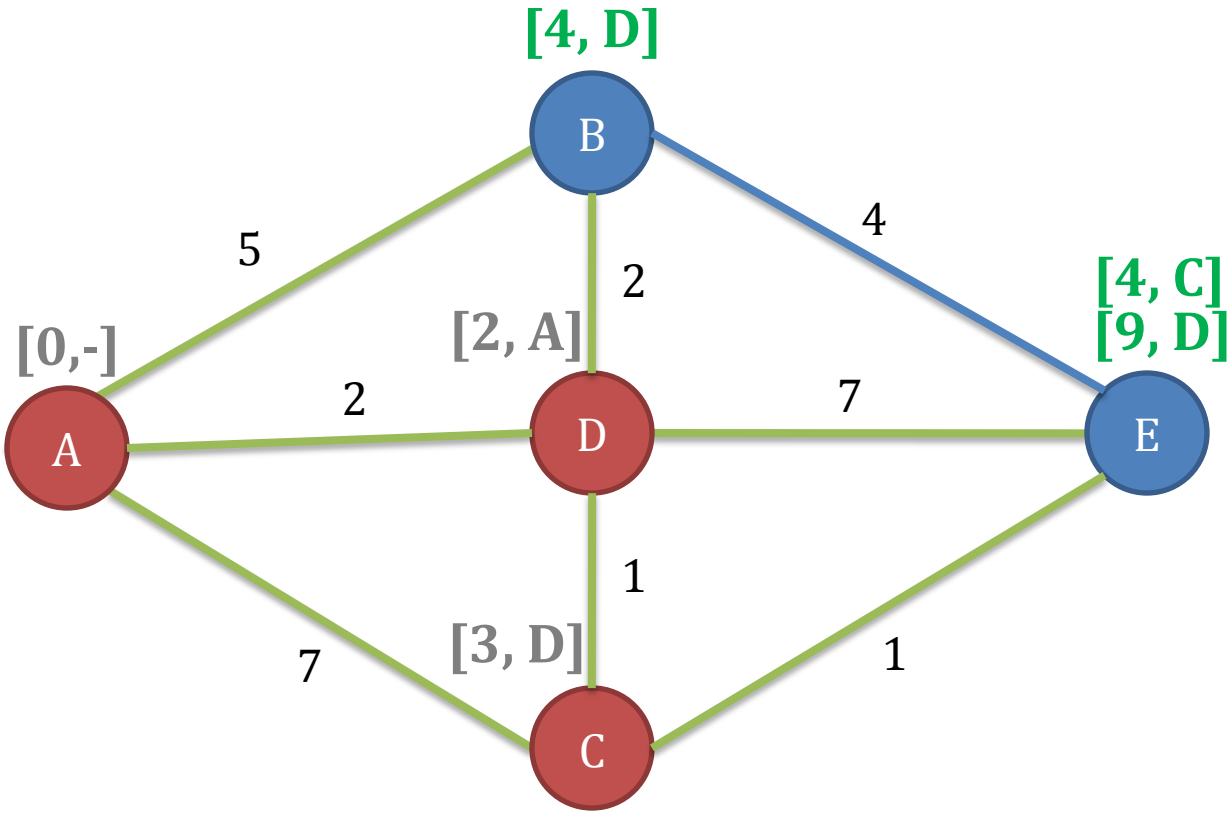


Para cada nodo alcanzado asignar etiqueta **[Distancia Acumulada, Nodo Predecesor]**

Expandir desde Nodo C a sus vecinos (Ignorar caminos ya recorridos)

Cola de Prioridad – Menor a Mayor

E(4, C)	B(4, A)	B(5, A)	C(7, A)	E(9, D)					

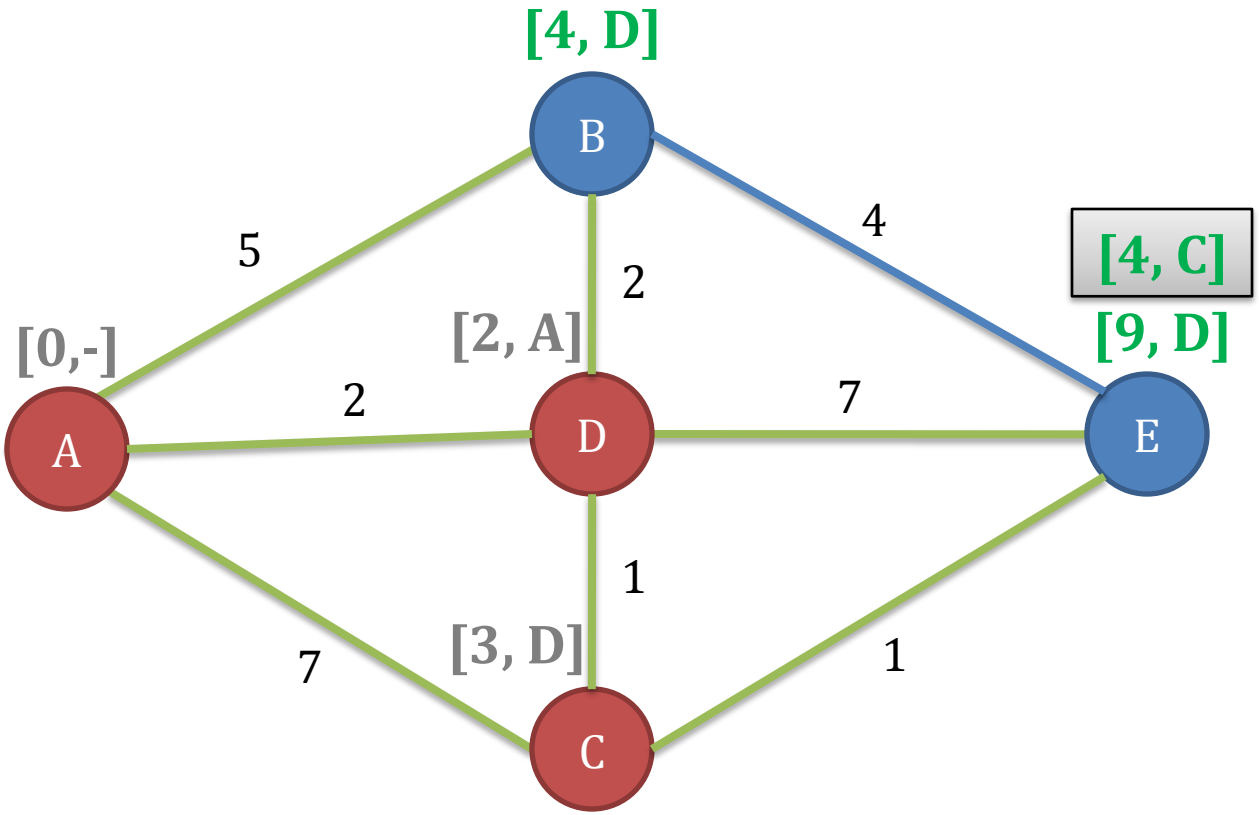


Para cada nodo alcanzado asignar etiqueta **[Distancia Acumulada, Nodo Predecesor]**

Seleccionar Nodo con menor valor

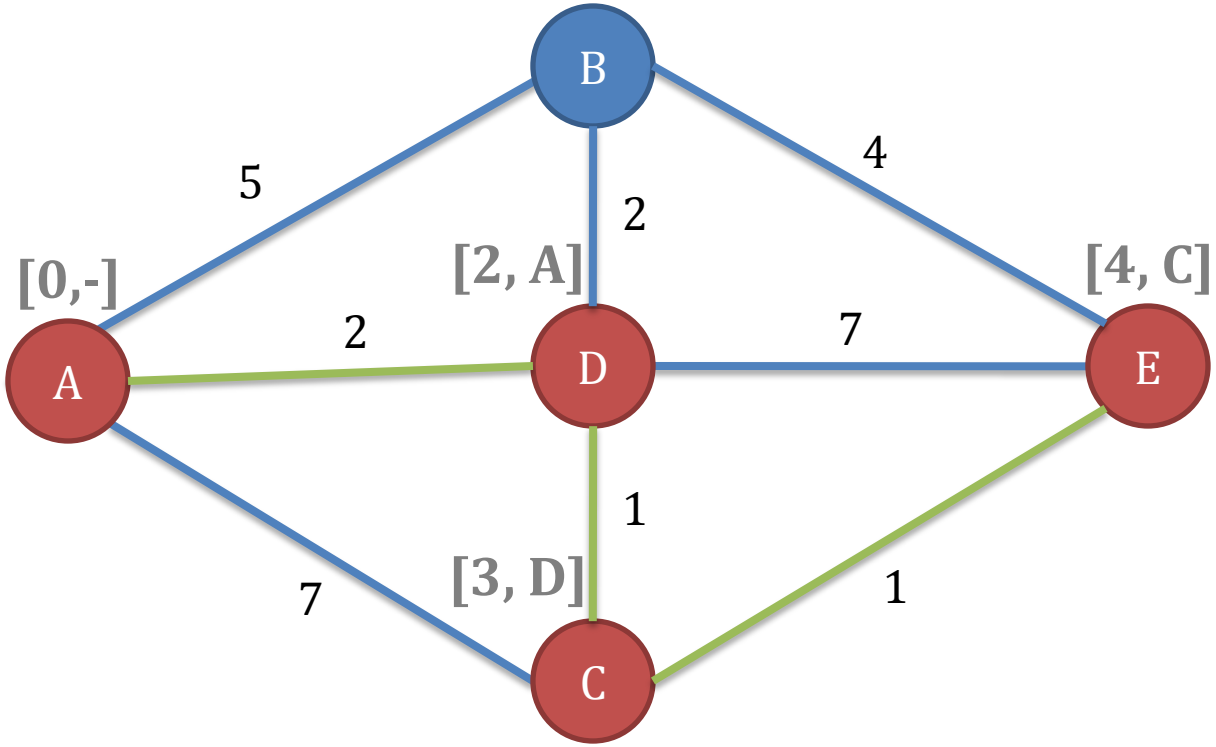
Cola de Prioridad – Menor a Mayor

E(4, C)	B(4, A)	B(5, A)	C(7, A)	E(9, D)					



Para cada nodo alcanzado asignar etiqueta **[Distancia Acumulada, Nodo Predecesor]**

Como el nodo obtenido es justamente el Nodo que se buscaba, se puede detener la ejecución en este punto o continuar buscando nuevos caminos de igual o mayor valor.

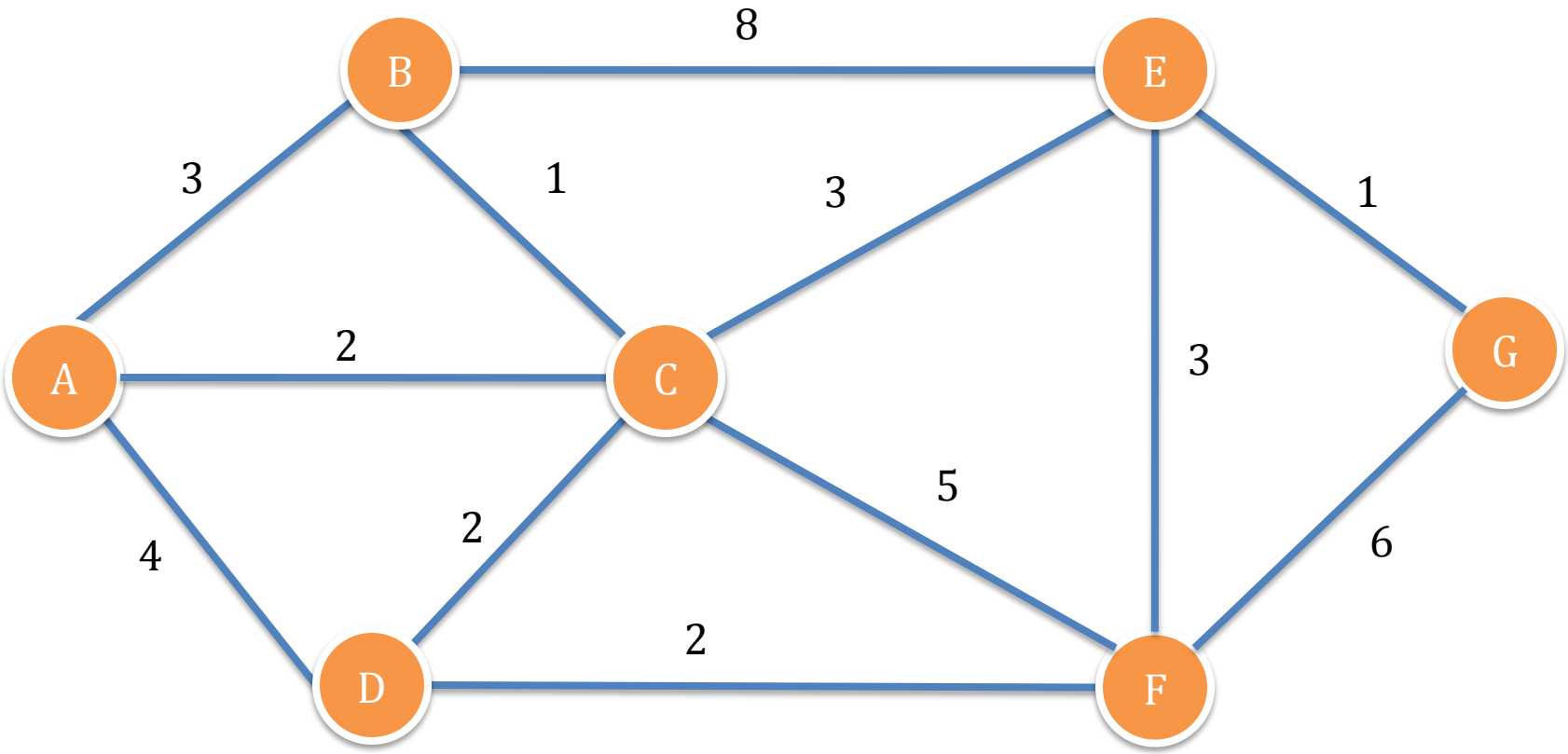


Ahora para obtener el camino, desde el Nodo **FINAL** se consulta la Etiqueta con la Letra del Nodo predecesor hasta llegar al Nodo Inicial.

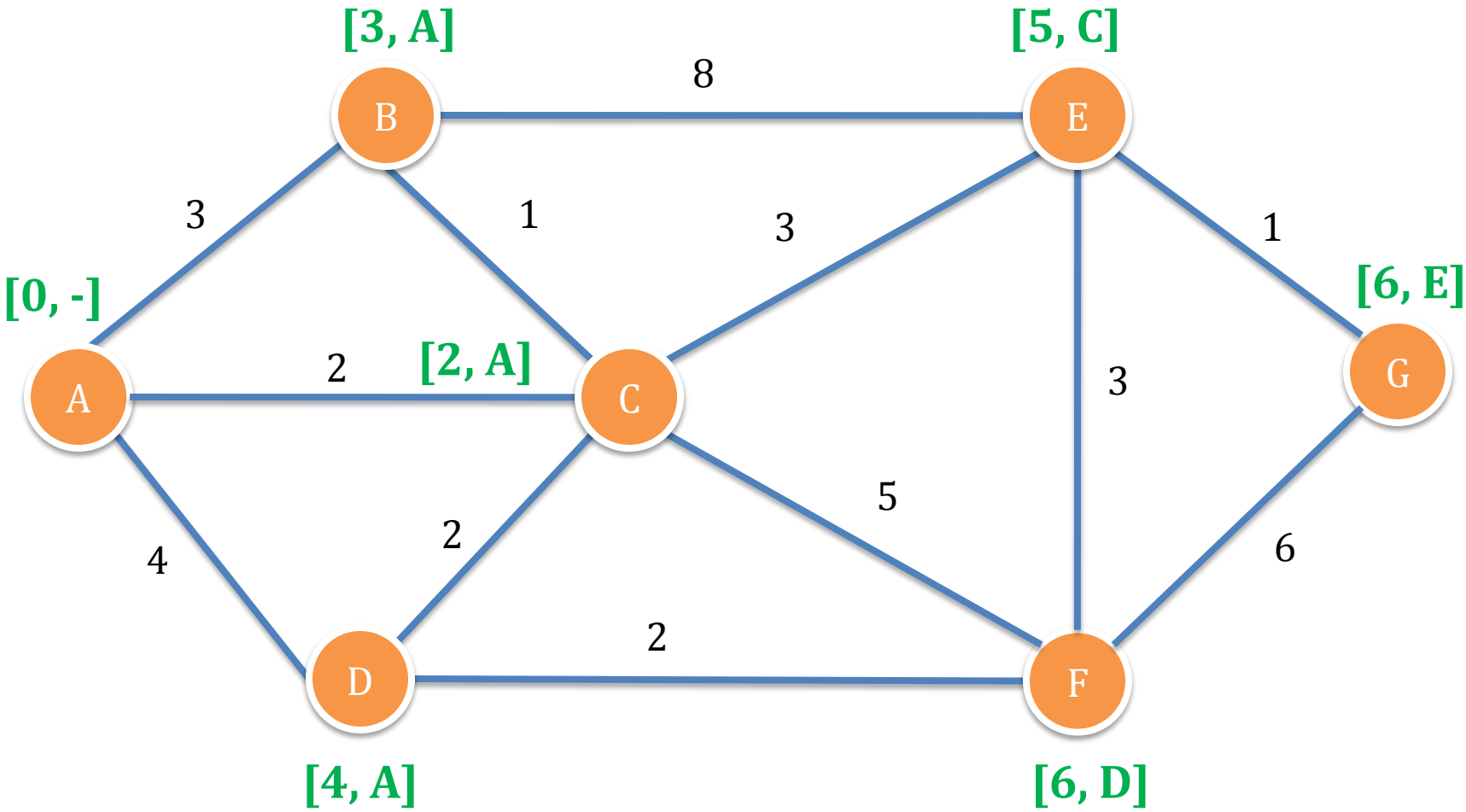
Resultado: E -> C-> D -> A
Costo: 4.

Ejercicio

Encuentre la ruta más corta desde el Nodo A a cualquier otro Nodo



Solución



UNIVERSIDAD DEL BÍO-BÍO

Solución

Origen	Destino	Camino	Costo
A	**B**	B -> A	**3**
A	**C**	C -> A	**2**
A	**D**	D -> A	**4**
A	**E**	E -> C -> A	**5**
A	**F**	F -> D -> A	**6**
A	**G**	G -> E -> C-> A	**6**

UNIVERSIDAD DEL BÍO-BÍO

Kruskal

Algoritmo de recubrimiento mínimo

UNIVERSIDAD DEL BÍO-BÍO

Kruskal

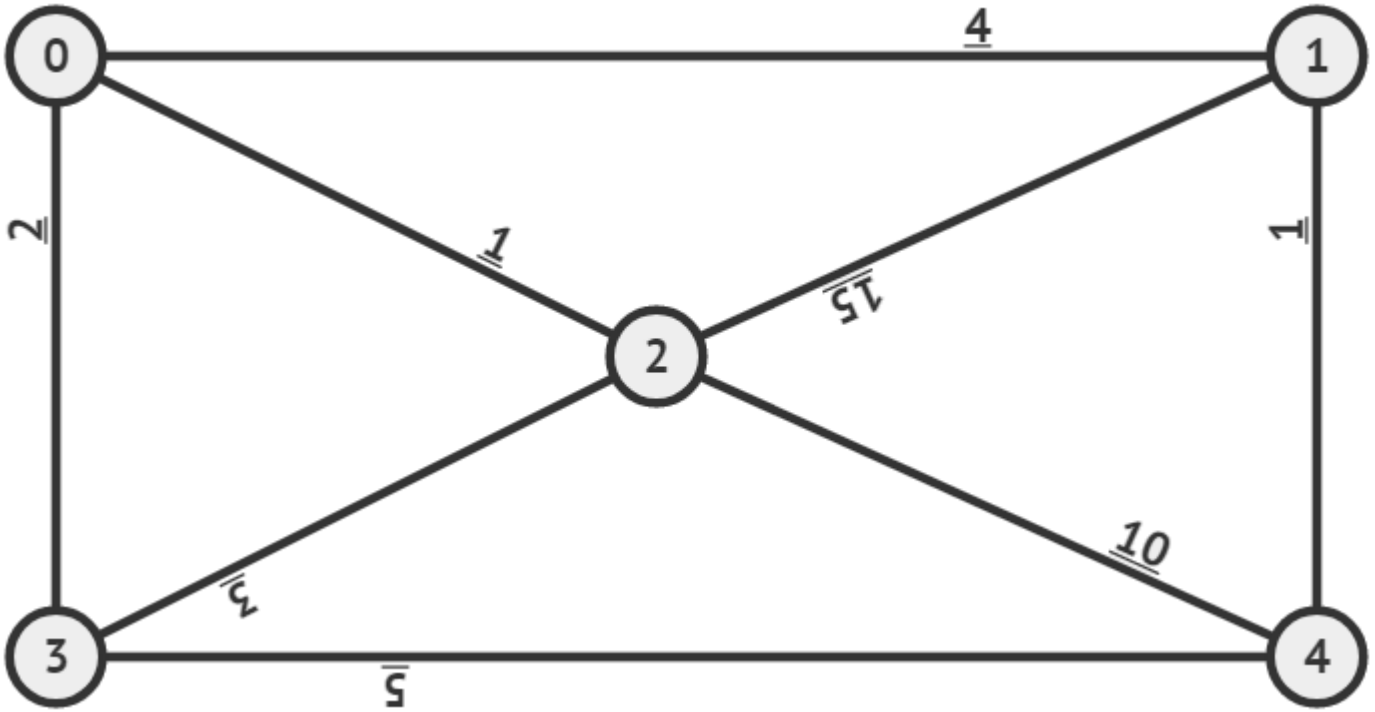
El algoritmo de Kruskal es un algoritmo de la teoría de grafos para encontrar un árbol recubridor mínimo en un grafo conexo y ponderado. Es decir, busca un subconjunto de aristas que, formando un árbol, incluyen todos los vértices y donde el valor de la suma de todas las aristas del árbol es el mínimo. Si el grafo no es conexo, entonces busca un bosque expandido mínimo (un árbol expandido mínimo para cada componente conexa).

Kruskal

Para encontrar el cubrimiento mínimo se debe seleccionar las aristas de menor valor sin formar un CICLO. El proceso se repite hasta que todos los vértices estén conectado al menos por una arista.

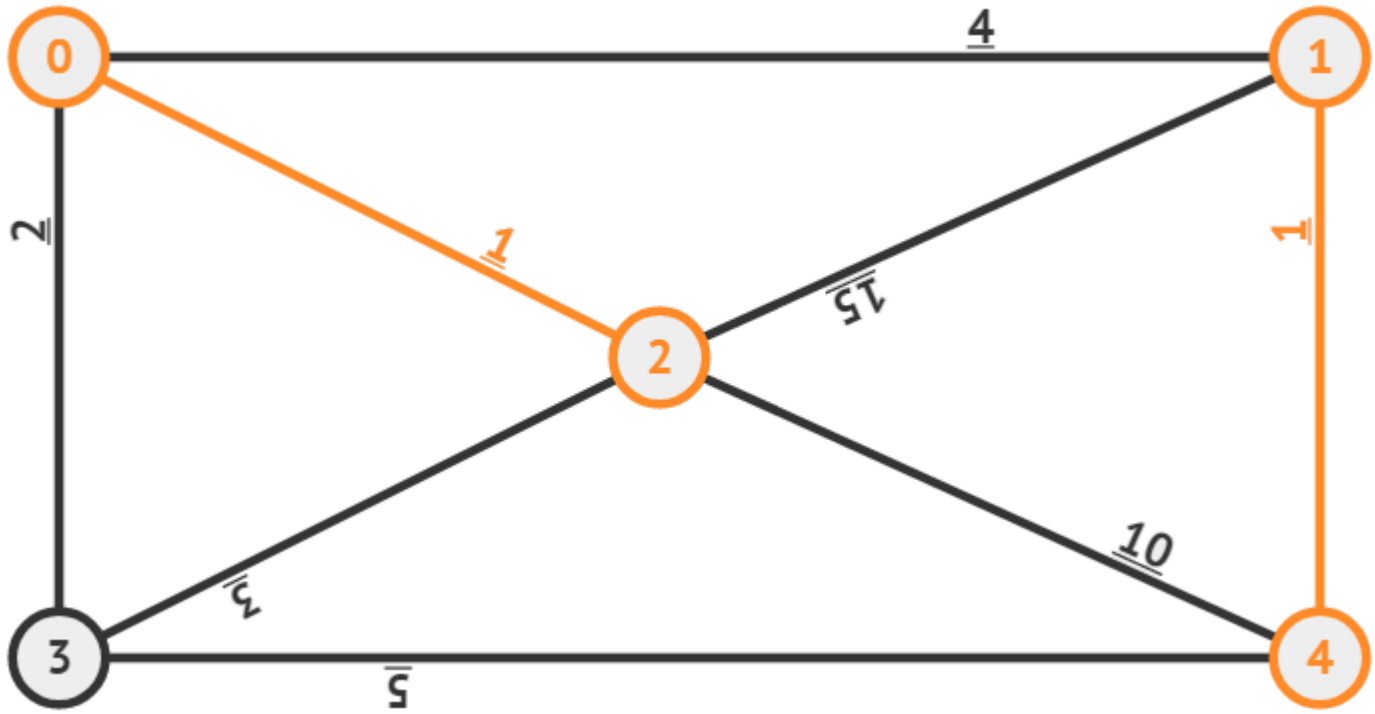
Ejercicio

Buscamos las aristas de Menor Valor, si alguna llega a formar un **CICLO** se descarta.
Seleccionamos aristas de Valor 1.



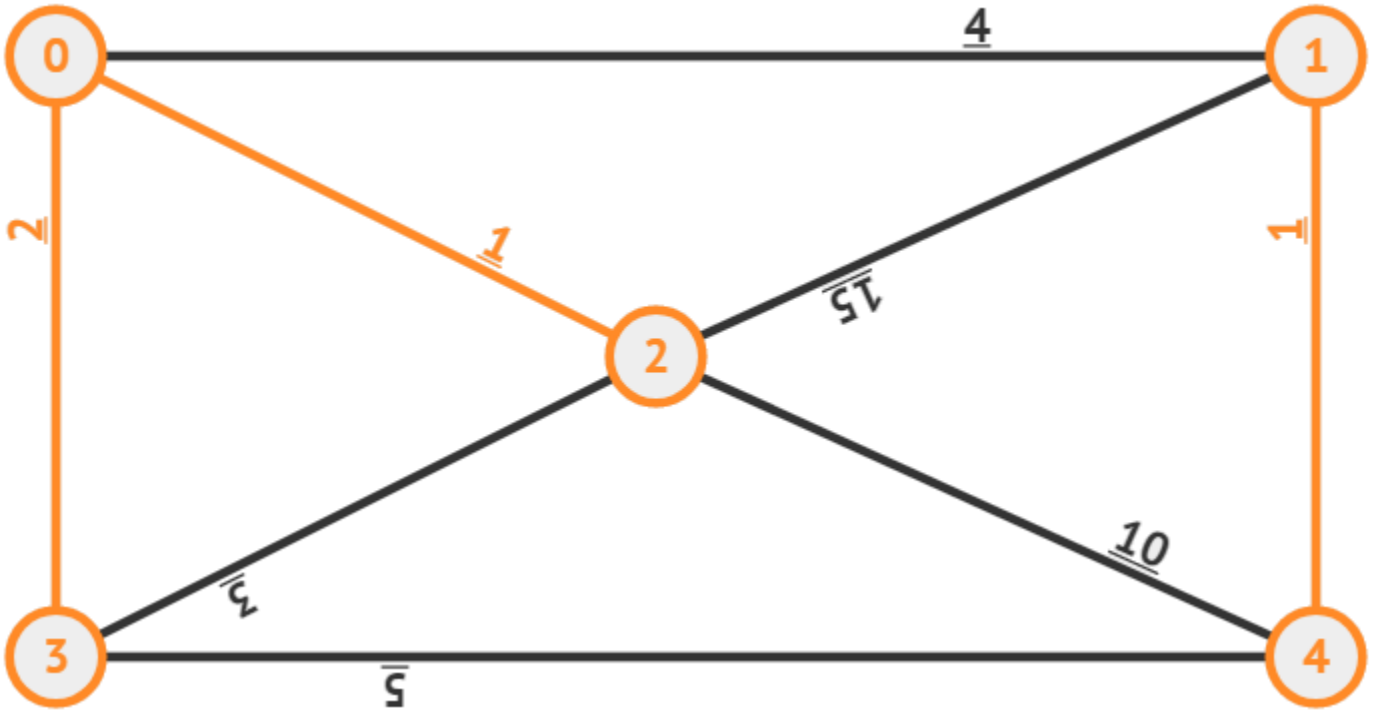
Ejercicio

Buscamos las aristas de Menor Valor, si alguna llega a formar un **CICLO** se descarta.
Seleccionamos aristas de Valor 2.



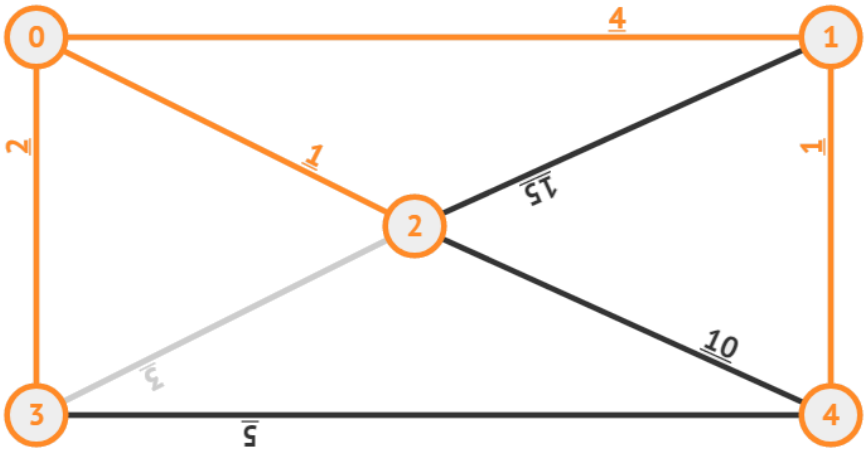
Ejercicio

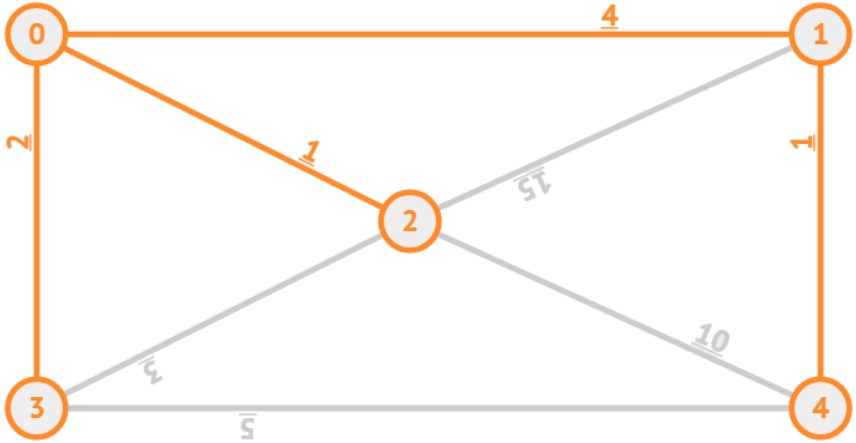
Buscamos las aristas de Menor Valor, si alguna llega a formar un **CICLO** se descarta.
Seleccionamos aristas de **Valor 3. PERO FORMA UN CICLO.**
Como forma un Ciclo se descarta y se continua con la siguiente: Valor 4



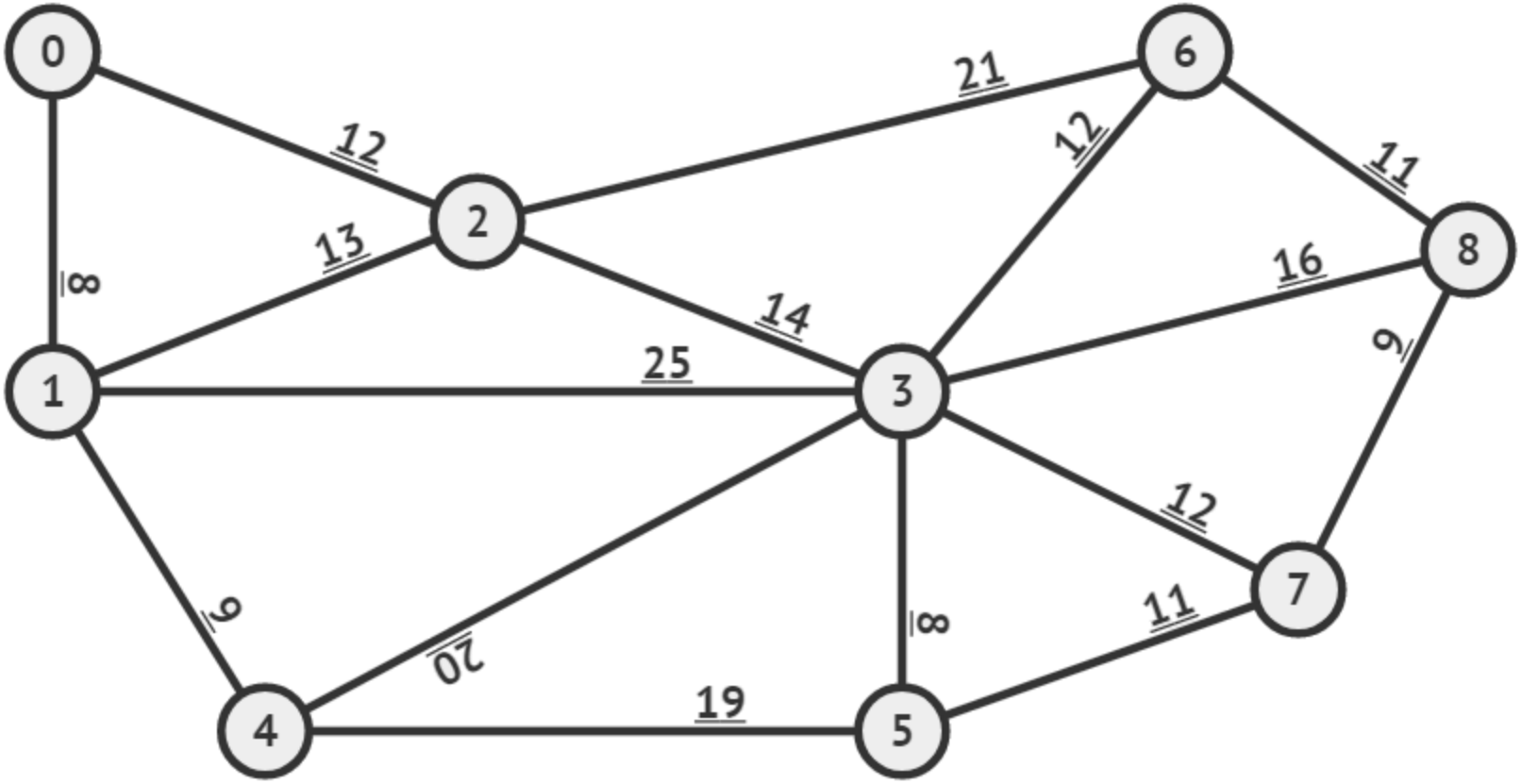
Ejercicio

Ahora todos los Vértices están conectados por al menos una arista, por lo tanto el proceso ha terminado.

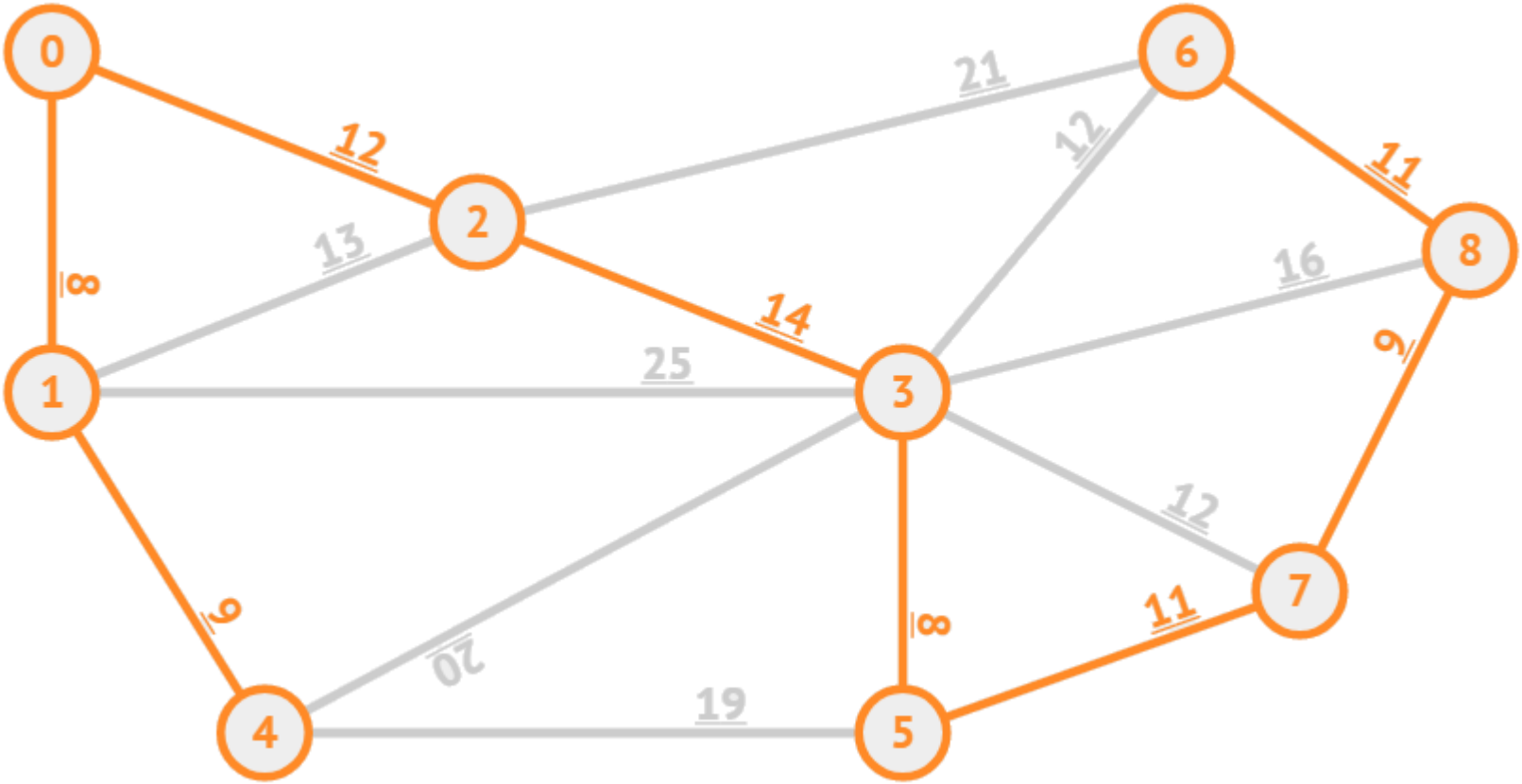




Ejercicio



Solución



Para practicar grafos dirigidos y no dirigidos de una manera visual, visitar:
https://visualgo.net/es/graphds

Para practicar Kruskal, visitar:
https://visualgo.net/en/mst