CAMINOS MÍNIMOS CON UN SOLO ORIGEN

Problema de Camino Mínimo. Dado un grafo G **con pesos en las aristas**, el problema de camino mínimo entre dos nodos **u** y **v** consiste en encontrar un camino entre esos nodos cuyo peso sea menor o igual que el peso de cualquier otro camino entre **u** y **v**.

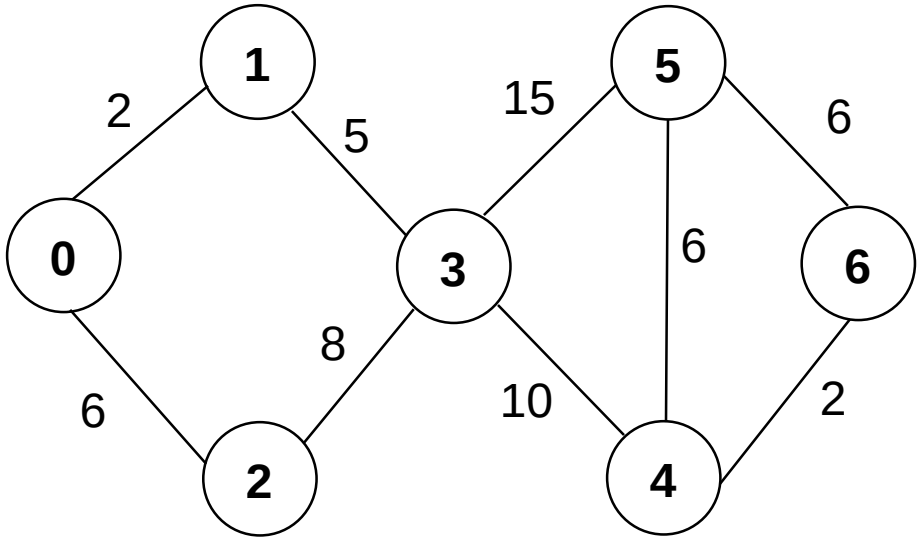
Según qué tipo de grafo analicemos, la solución al problema será diferente. Por ejemplo, si el grafo no tuviera pesos (o si todos los pesos fueran iguales, que a efectos prácticos es lo mismo), nos conviene usar otro algoritmo (Búsqueda en Anchura BFS).

UNA ALTERNATIVA: EL ALGORITMO DE DIJKSTRA.

Este algoritmo fue creado por uno de los padres de la computación, Edsger W. Dijkstra, en 1956. Sirve para cualquier grafo con pesos (dirigido o no) siempre y cuando sus pesos no sean negativos.

- El algoritmo calcula las distancias mínimas desde un nodo inicial a **todos los demás**. Para hacerlo, en cada paso se toma el nodo más cercano al inicial que aún no fue visitado (le diremos v).
- Luego, recalculamos todas los caminos mínimos, teniendo en cuenta a **v** como camino intermedio.
- Así, en cada paso tendremos un subconjunto de nodos que ya tienen calculada su mínima distancia y los demás tienen calculada su mínima distancia si solo puedo usar los nodos del conjunto como nodos intermedios.
- Con cada iteración agregaremos un nodo más a nuestro conjunto, hasta resolver el problema en su totalidad.

Ejemplo 1. Obtenga los caminos mínimos para el siguiente grafo, usando el algoritmo de Dijkstra, comenzando en el Nodo 0

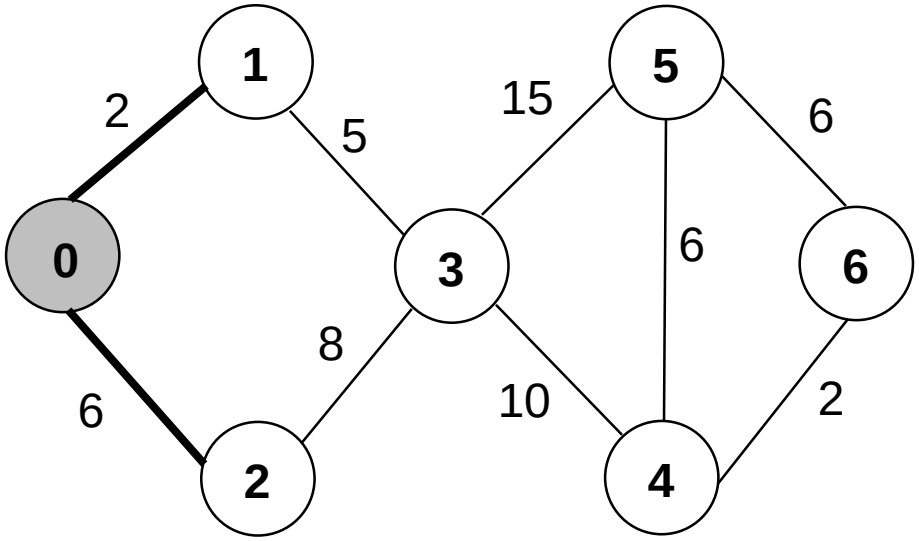


Situación inicial:

NO-Visitados = { 0 , 1 , 2 , 3 , 4 , 5 , 6 }

Del Nodo 0 al Nodo	0	1	2	3	4	5	6
Distancia	0	-	-	-	-	-	-
Marca							

Comenzamos por el Nodo 0:
Lo sacamos del conjunto de NO-Visitados. **NO-Visitados = { 1 , 2 , 3 , 4 , 5 , 6 }**
Se chequean adyacentes al Nodo 0 : Son los Nodos 1 y 2 (note las aristas en negrilla)



Consejo:

Esto NO significa que estemos añadiendo inmediatamente los dos nodos adyacentes al camino más corto.

Antes de añadir un nodo a este camino, tenemos que comprobar si hemos encontrado el camino más corto para llegar a él.

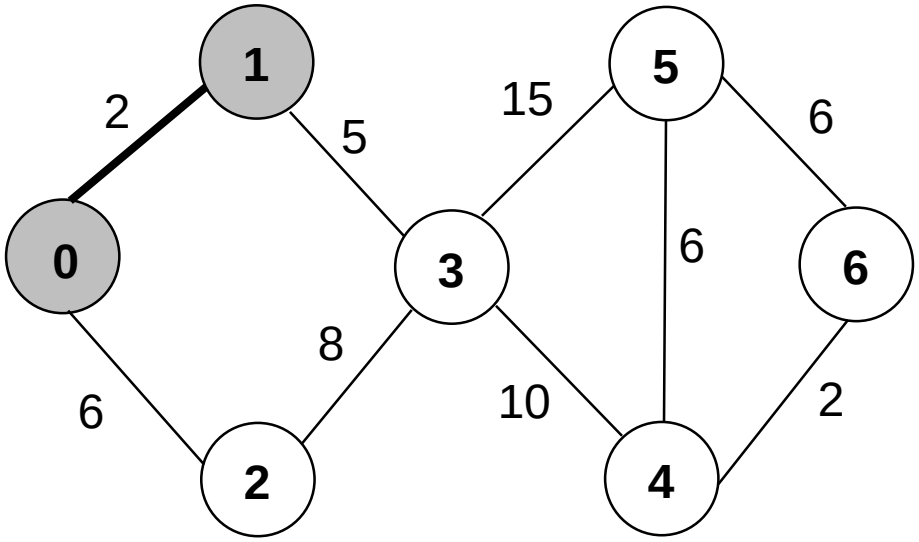
Simplemente estamos haciendo un proceso de examen inicial para ver las opciones disponibles.

Situación:

Del Nodo 0 al Nodo	0	1	2	3	4	5	6
Distancia	0	2	6	-	-	-	-

Después de actualizar las distancias de los nodos adyacentes, tenemos que:

- Seleccionar el nodo más cercano al nodo origen en función de las distancias actuales conocidas.
- Marcarlo como visitado. **NO-Visitados = { 2 , 3 , 4 , 5 , 6 }**
- Añadirlo a la ruta.
- Si comprobamos la lista de distancias, podemos ver que el nodo 1 tiene la distancia más corta al nodo origen (una distancia de 2), así que lo añadimos a la ruta.



Situación (Lo marcamos en la lista para representar que ha sido «visitado» y que hemos encontrado el camino más corto hacia ese nodo):

Del Nodo 0 al Nodo	**0**	**1**	**2**	**3**	**4**	**5**	**6**
Distancia	0	2	6	-	-	-	-
Marca		**x**					

Debemos analizar los nuevos nodos adyacentes para encontrar el camino más corto para llegar a ellos. Sólo analizaremos los nodos adyacentes a los nodos que ya forman parte del camino más corto (el camino marcado con aristas en negrilla).

El nodo 3 y el nodo 2 son adyacentes a nodos que ya están en el camino porque están conectados directamente al nodo 1 y al nodo 0, respectivamente, como se puede ver. Estos son los nodos que analizaremos en el siguiente paso.

Como ya tenemos anotada en nuestra lista la distancia del nodo origen al nodo 2, no necesitamos actualizar la distancia esta vez. Sólo tenemos que actualizar la distancia del nodo origen al nuevo nodo adyacente (nodo 3):

Del Nodo 0 al Nodo	**0**	**1**	**2**	**3**	**4**	**5**	**6**
Distancia	0	2	6	7	-	-	-
Marca		**x**					

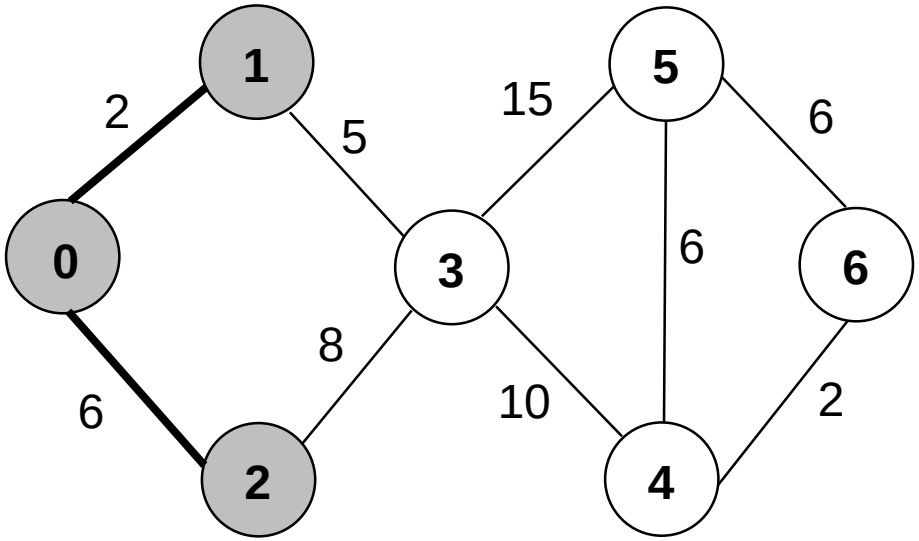
Esta distancia es 7. Veamos por qué.

Para hallar la distancia del nodo origen a otro nodo (en este caso, el nodo 3), sumamos los pesos de todas las aristas que forman el camino más corto para llegar a ese nodo.

Para el nodo 3: la distancia total es 7 porque sumamos los pesos de las aristas que forman el camino 0 -> 1 -> 3 (peso 2 para la arista 0 -> 1 y peso 5 para la arista 1 -> 3).

Ahora que tenemos la distancia a los nodos adyacentes, tenemos que elegir qué nodo se añadirá a la ruta. Debemos seleccionar el nodo no visitado con la distancia más corta (actualmente conocida) al nodo de origen.

De la lista de distancias, podemos detectar inmediatamente que se trata del nodo 2 con la distancia 6.



También lo marcamos como visitado en la lista de distancias y lo excluimos de la lista de NO-Visitados:

Del Nodo 0 al Nodo	**0**	**1**	**2**	**3**	**4**	**5**	**6**
Distancia	0	2	6	7	-	-	-
Marca		**x**	**x**				

NO-Visitados = { 3 , 4 , 5 , 6 }

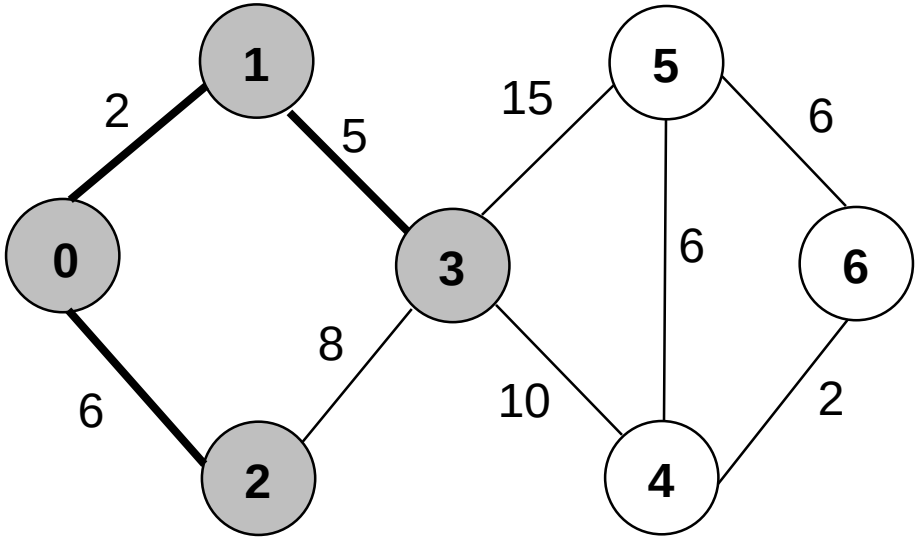
Ahora tenemos que repetir el proceso para encontrar el camino más corto desde el nodo origen hasta el nuevo nodo adyacente, que es el nodo 3. Puede verse que tenemos dos posibles caminos 0 -> 1 -> 3 o 0 -> 2 -> 3.
Veamos cómo podemos decidir cuál es el camino más corto.

El nodo 3 ya tiene una distancia en la lista que se registró anteriormente (7, ver la lista de abajo).
Esta distancia fue el resultado de un paso anterior, en el que sumamos los pesos 5 y 2 de las dos aristas que necesitábamos cruzar para seguir el camino 0 -> 1 -> 3.

Pero ahora tenemos otra alternativa. Si elegimos el camino 0 -> 2 -> 3, necesitaríamos seguir dos aristas 0 -> 2 y 2 -> 3 con pesos 6 y 8, respectivamente, lo que representa una distancia total de 14.

Claramente, la primera distancia (existente) es más corta (7 frente a 14), por lo que optaremos por mantener el camino original 0 -> 1 -> 3. Sólo actualizamos la distancia si el nuevo camino es más corto.

Por lo tanto, añadimos este nodo a la ruta utilizando la primera alternativa: 0 -> 1 -> 3.



Marcamos este nodo como visitado y lo excluimos de la lista de nodos NO-Visitados:

Del Nodo 0 al Nodo	**0**	**1**	**2**	**3**	**4**	**5**	**6**
Distancia	0	2	6	7	-	-	-
Marca		**x**	**x**	**x**			

NO-Visitados = { 4 , 5 , 6 }

Tenemos que comprobar los nuevos nodos adyacentes que no hemos visitado hasta ahora. Esta vez, estos nodos son el nodo 4 y el nodo 5, ya que son adyacentes al nodo 3.

Actualizamos las distancias de estos nodos al nodo origen, intentando siempre encontrar un camino más corto, si es posible:

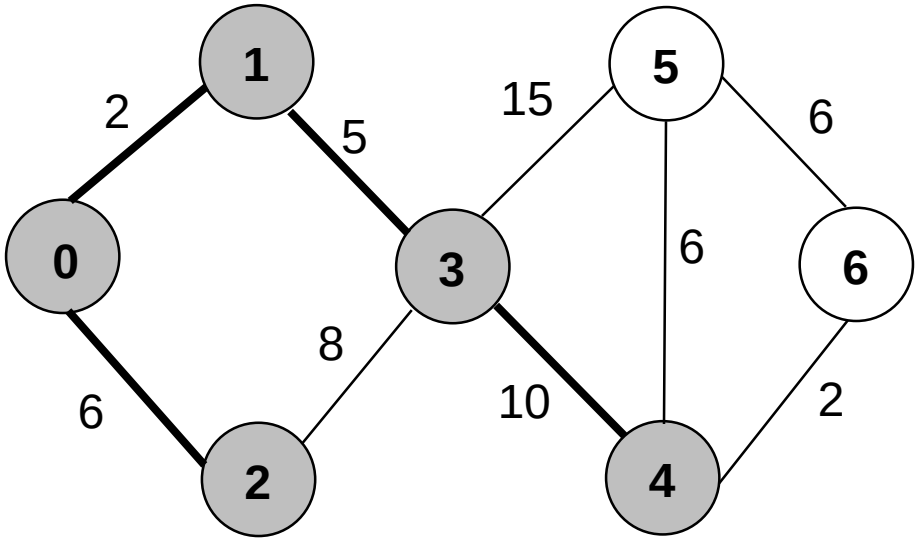
Para el nodo 4: la distancia es 17 desde la ruta 0 -> 1 -> 3 -> 4.

Para el nodo 5: la distancia es 22 desde el camino 0 -> 1 -> 3 -> 5.

Consejo: Observa que sólo podemos considerar la ampliación del camino más corto (marcado en negrilla). No podemos considerar caminos que nos lleven a través de aristas que no han sido añadidas al camino más corto (por ejemplo, no podemos formar un camino que pase por la arista 2 -> 3).

Del Nodo 0 al Nodo	**0**	**1**	**2**	**3**	**4**	**5**	**6**
Distancia	0	2	6	7	17	22	-
Marca		**x**	**x**	**x**			

Tenemos que elegir cuál nodo no visitado se marcará como visitado ahora. En este caso, es el nodo 4 porque tiene la distancia más corta en la lista de distancias. Lo añadimos en el diagrama:



Marcamos este nodo como visitado y lo excluimos de la lista de nodos NO-Visitados:

Del Nodo 0 al Nodo	**0**	**1**	**2**	**3**	**4**	**5**	**6**
Distancia	0	2	6	7	17	22	-
Marca		**x**	**x**	**x**	**x**		

NO-Visitados = { 5 , 6 }

Comprobamos los nodos adyacentes: nodo 5 y nodo 6. Tenemos que analizar cada posible camino para llegar a ellos desde los nodos que ya han sido marcados como visitados y añadidos al camino.

Para el nodo 5:

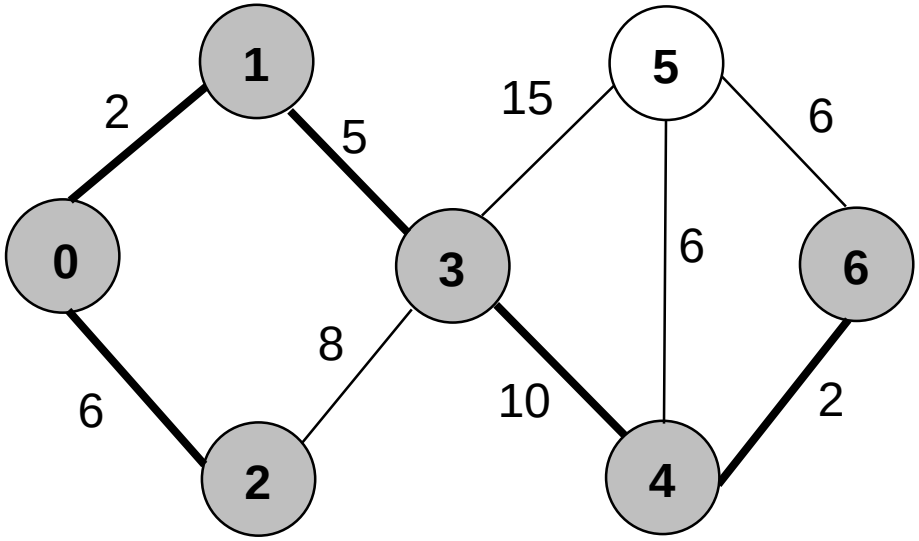
- La primera opción es el camino 0 -> 1 -> 3 -> 5, que tiene una distancia de 22 desde el nodo origen (2 + 5 + 15).
- La segunda opción sería el camino 0 -> 1 -> 3 -> 4 -> 5, con distancia de 23 desde el nodo origen (2 + 5 + 10 + 6).
- Claramente, el primer camino es más corto, así que lo elegimos para el nodo 5.

Para el nodo 6: El camino disponible es 0 -> 1 -> 3 -> 4 -> 6, con distancia de 19 desde el nodo origen (2 + 5 + 10 + 2).

El camino más corto es el que lleva al Nodo 6, y así se marca, además de excluirlo de los NO-visitados.

Del Nodo 0 al Nodo	**0**	**1**	**2**	**3**	**4**	**5**	**6**
Distancia	0	2	6	7	17	22	19
Marca		**x**	**x**	**x**	**x**		**x**

NO-Visitados = { 5 }



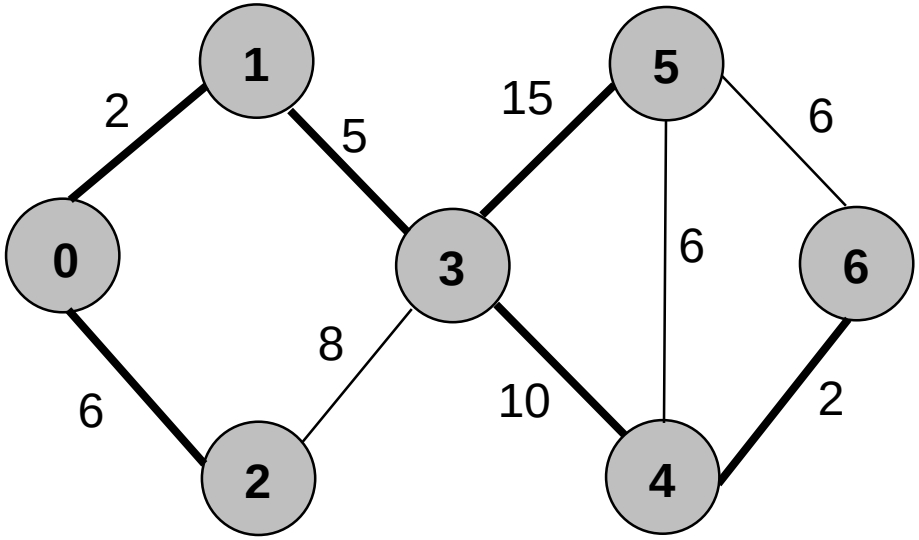
Sólo un nodo no ha sido visitado todavía, el nodo 5. Veamos cómo podemos incluirlo en el camino.

Hay tres caminos diferentes que podemos tomar para llegar al nodo 5 desde los nodos que se han añadido al camino:

Opción 1: 0 -> 1 -> 3 -> 5 con una distancia de 22 (2 + 5 + 15).
Opción 2: 0 -> 1 -> 3 -> 4 -> 5 con una distancia de 23 (2 + 5 + 10 + 6).
Opción 3: 0 -> 1 -> 3 -> 4 -> 6 -> 5 con una distancia de 25 (2 + 5 + 10 + 2 + 6).

Seleccionamos el camino más corto 0 -> 1 -> 3 -> 5 con una distancia de 22.

Del Nodo 0 al Nodo	**0**	**1**	**2**	**3**	**4**	**5**	**6**
Distancia	0	2	6	7	17	22	19
Marca		**x**	**x**	**x**	**x**		**x**

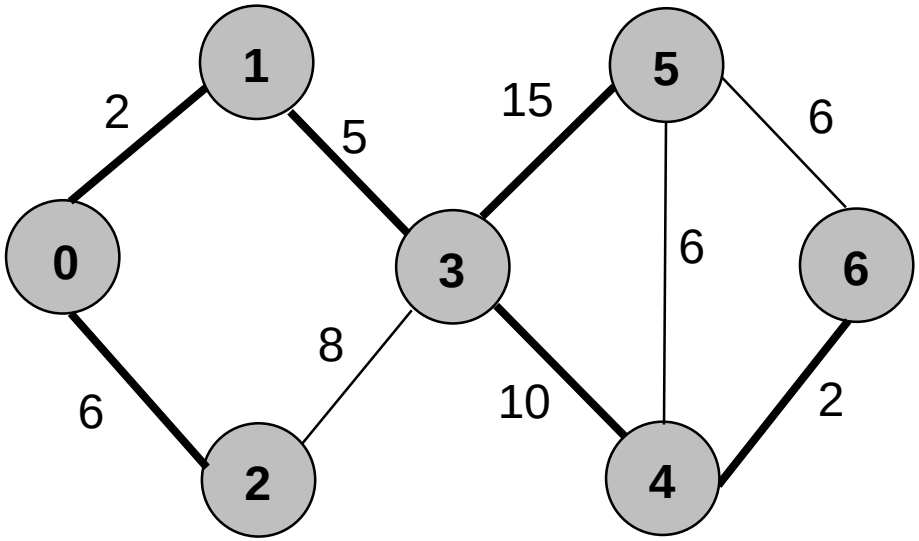


Marcamos este nodo como visitado y lo excluimos de la lista de nodos NO-Visitados:

Del Nodo 0 al Nodo	**0**	**1**	**2**	**3**	**4**	**5**	**6**
Distancia	0	2	6	7	17	22	19
Marca		**x**	**x**	**x**	**x**	**x**	**x**

NO-Visitados = { }

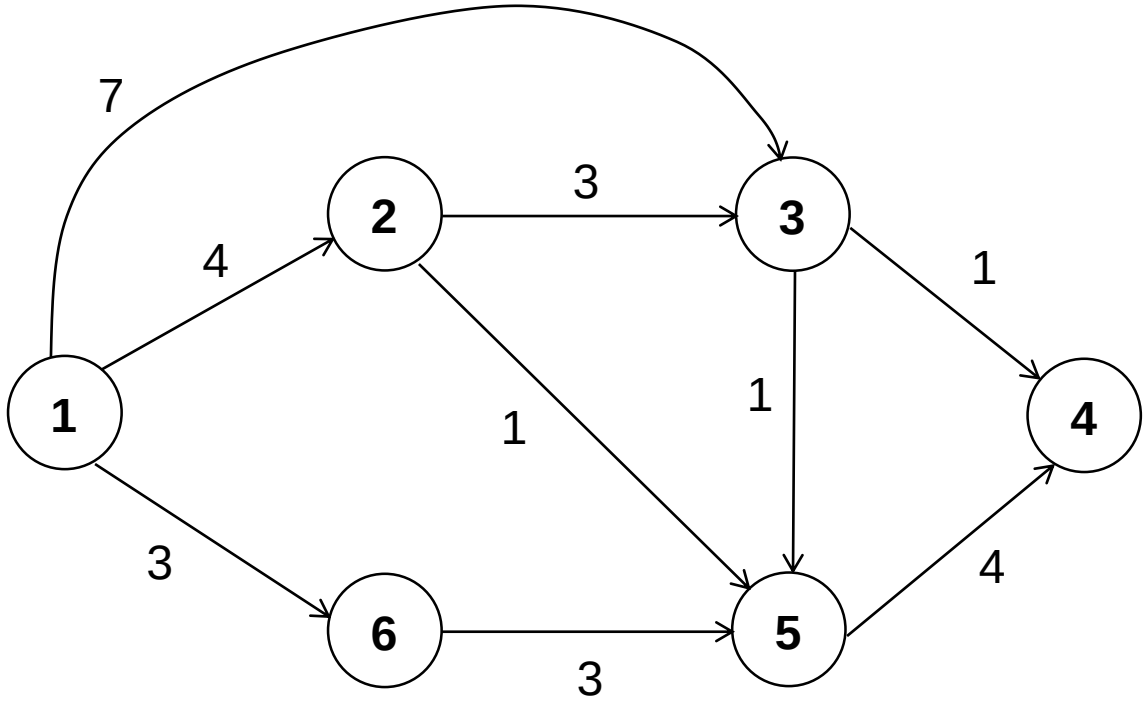
Y terminamos porque la lista de nodos NO-visitados está vacía.
Así tenemos el resultado final con el camino más corto desde el nodo 0 a cada nodo del grafo.



En el diagrama, las líneas en negrilla marcan las aristas que pertenecen al camino más corto. Tienes que seguir estas aristas para seguir el camino más corto para llegar a un nodo determinado del grafo partiendo del nodo 0.

Por ejemplo, si quieres llegar al nodo 6 partiendo del nodo 0, sólo tienes que seguir las aristas en negrilla y estarás siguiendo el camino más corto 0 -> 1 -> 3 -> 4 - > 6 automáticamente.

Ejemplo 2. Obtenga los caminos mínimos para el siguiente grafo, usando el algoritmo de Dijkstra, comenzando en el Nodo 1.

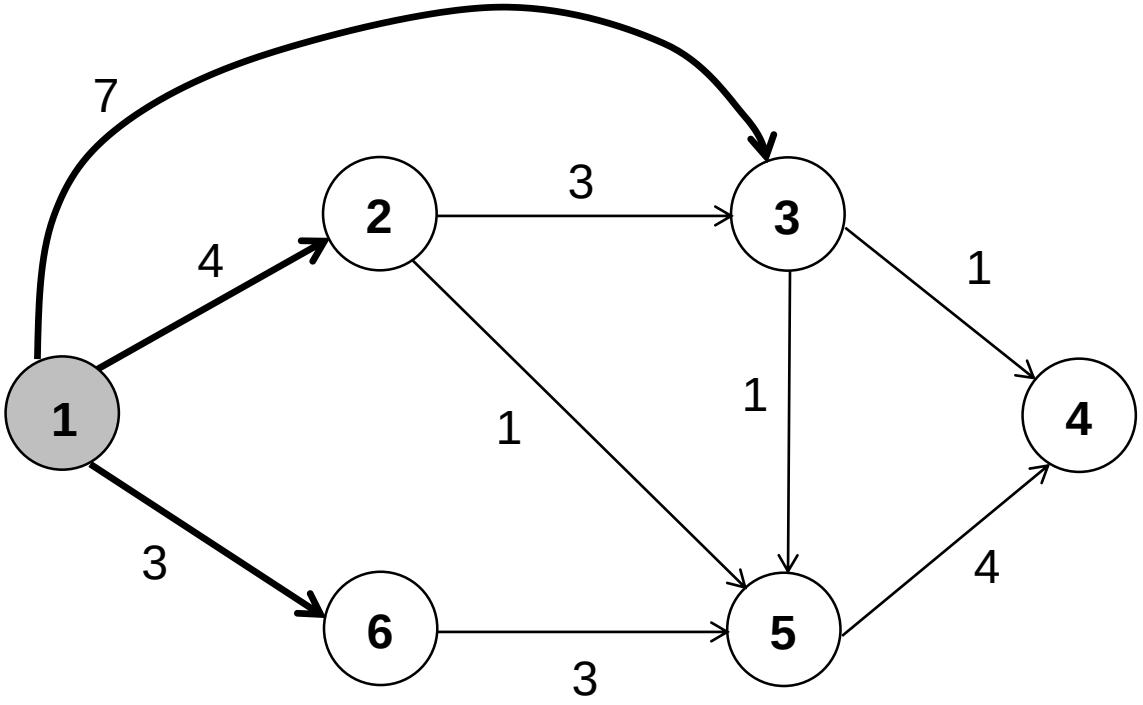


Situación inicial:

NO-Visitados = { 1 , 2 , 3 , 4 , 5 , 6 }

Del Nodo 1 al Nodo	1	2	3	4	5	6
Distancia	0	-	-	-	-	-
Marca						

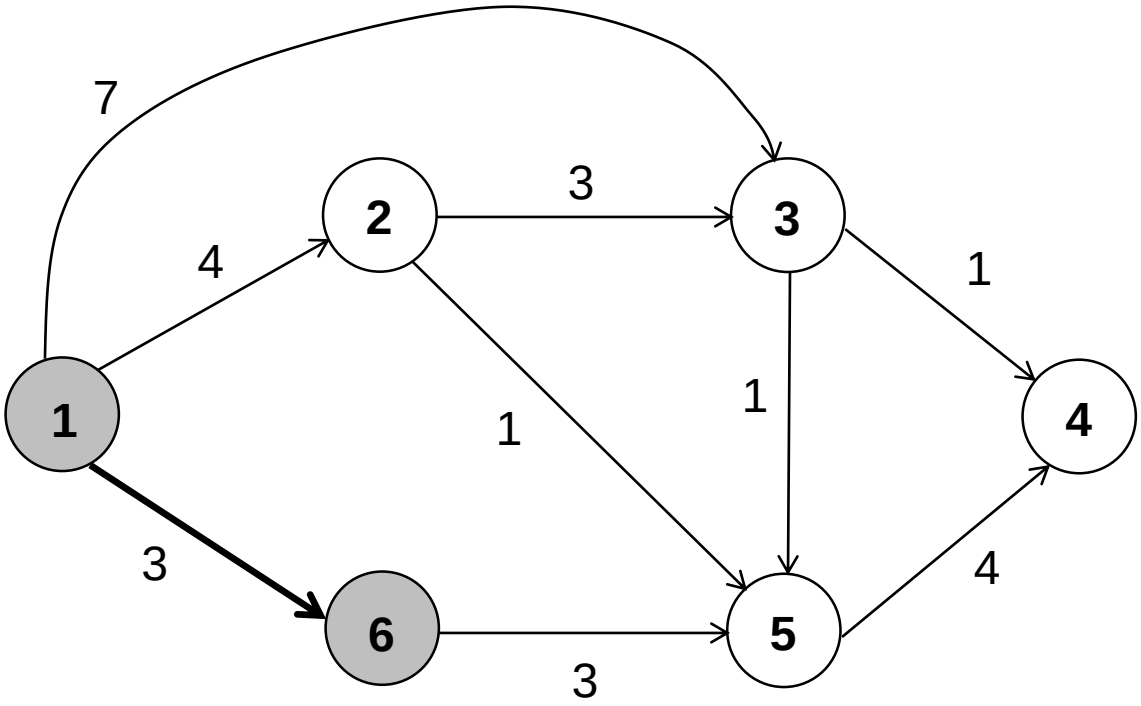
Comenzando por el nodo 1.
Lo sacamos del conjunto de NO-Visitados. **NO-Visitados = { 2 , 3 , 4 , 5 , 6 }**
Se chequean adyacentes al Nodo 1: Son los Nodos 2 , 3 y 6 (note las aristas en negrilla)



Situación:

Del Nodo 1 al Nodo	1	2	3	4	5	**6**
Distancia	0	4	7	-	-	**3**

De la lista de distancias, se detecta que se trata del nodo 6 con distancia 3.

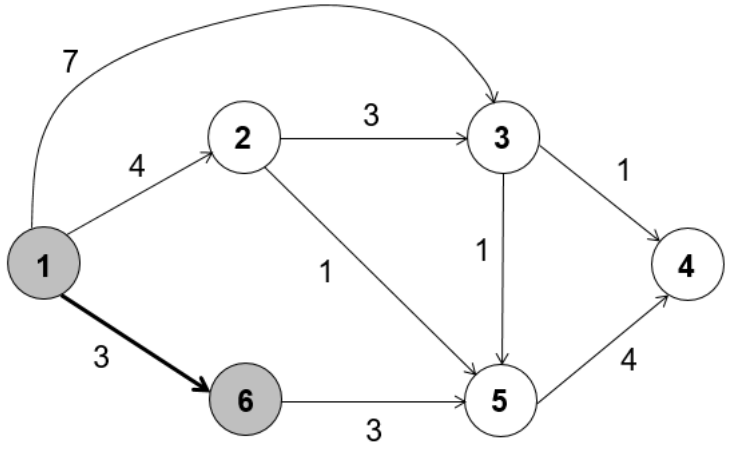


Lo marcamos como visitado en la lista de distancias y lo excluimos de la lista de NO-Visitados:

Del Nodo 1 al Nodo	1	2	3	4	5	6
Distancia	0	4	7	-	-	3
Marca						**x**

NO-Visitados = { 2 , 3 , 4 , 5 }

Debemos analizar los nuevos nodos adyacentes para encontrar el camino más corto para llegar a ellos. Sólo analizaremos los nodos adyacentes a los nodos que ya forman parte del camino más corto (el camino marcado con aristas en negrilla).

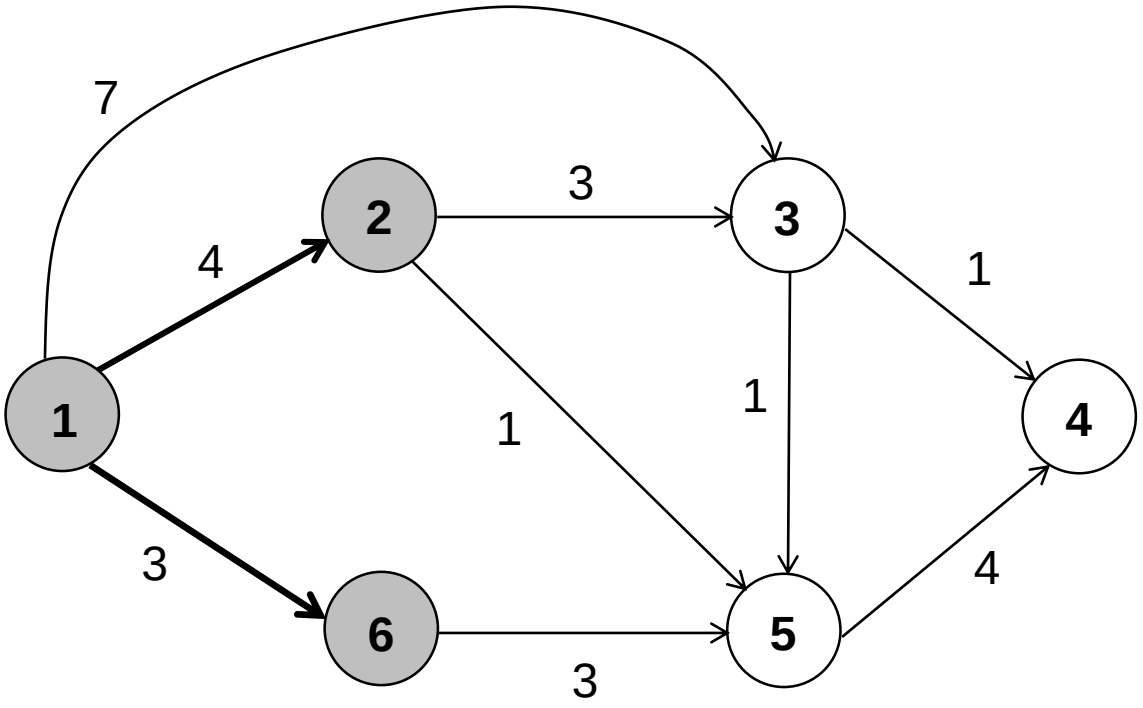


Con respecto al Nodo 5.
Opción: 1 -> 6 -> 5 con una distancia de 6 (3 + 3).

Del Nodo 1 al Nodo	1	**2**	3	4	5	6
Distancia	0	**4**	7	-	6	3
Marca						**x**

De la lista de distancias, se detecta que se trata del nodo 2 con distancia 4.

Recuerden: Sólo actualizamos la distancia si el nuevo camino es más corto.

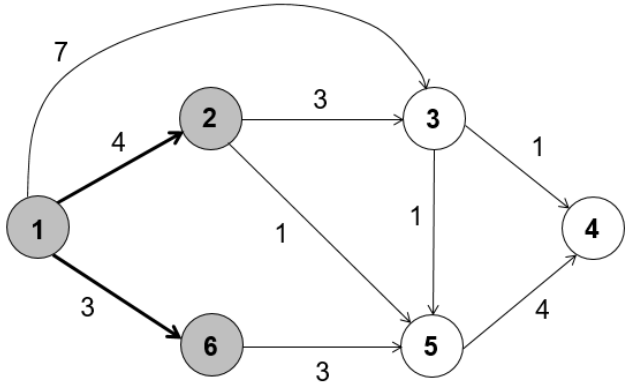


Lo marcamos como visitado en la lista de distancias y lo excluimos de la lista de NO-Visitados:
*

Del Nodo 1 al Nodo	1	2	3	4	5	6
Distancia	0	4	7	-	6	3
Marca		**x**				**x**

NO-Visitados = { 3 , 4 , 5 }

Debemos analizar los nuevos nodos adyacentes para encontrar el camino más corto para llegar a ellos. Sólo analizaremos los nodos adyacentes a los nodos que ya forman parte del camino más corto (el camino marcado con aristas en negrilla).



Con respecto al Nodo 3.

Hay DOS caminos diferentes que podemos tomar para llegar al nodo 3 desde los nodos que se han añadido al camino:
Opción 1: 1 -> 3 con una distancia de 7.
Opción 2: 1 -> 2 -> 3 con una distancia de 7 (4 + 3).

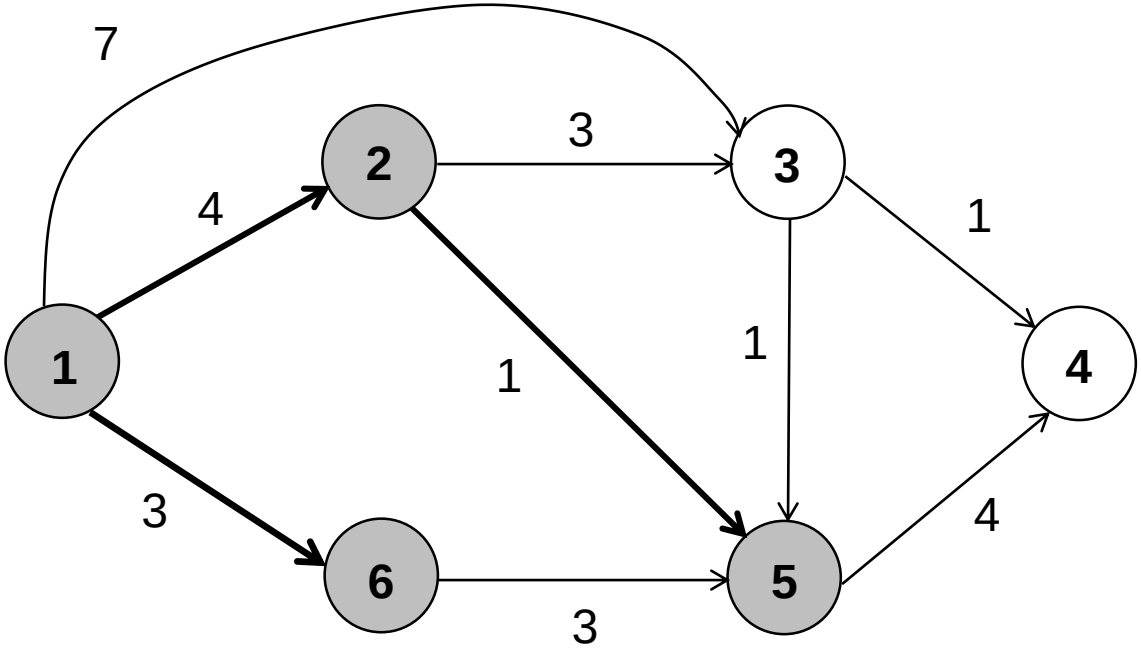
Con respecto al Nodo 5.

Hay DOS caminos diferentes que podemos tomar para llegar al nodo 5 desde los nodos que se han añadido al camino:

Opción 1: 1 -> 2 -> 5 con una distancia de 5 (4 + 1). ¡Encontramos una distancia menor a la ya registrada! ¡Actualizar!
Opción 2: 1 -> 6 -> 5 con una distancia de 6 (3 + 3).

Del Nodo 1 al Nodo	1	2	3	4	**5**	6
Distancia	0	4	7	-	**5**	3
Marca		**x**				**x**

De la lista de distancias, se detecta que se trata del nodo 5 con distancia 5.

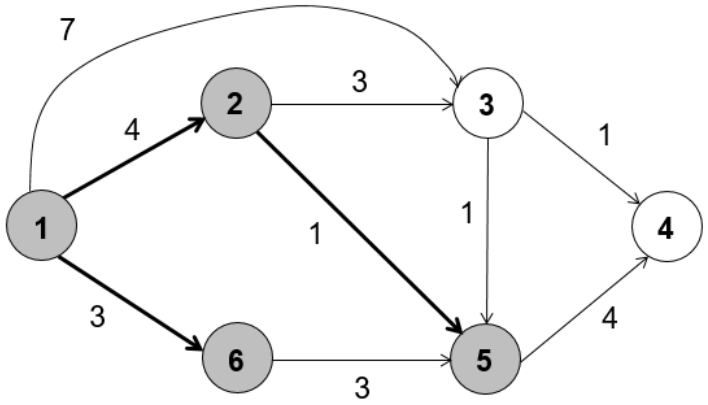


Lo marcamos como visitado en la lista de distancias y lo excluimos de la lista de NO-Visitados:

Del Nodo 1 al Nodo	1	2	3	4	5	6
Distancia	0	4	7	-	5	3
Marca		**x**			**x**	**x**

NO-Visitados = { 3 , 4 }

Debemos analizar los nuevos nodos adyacentes para encontrar el camino más corto para llegar a ellos. Sólo analizaremos los nodos adyacentes a los nodos que ya forman parte del camino más corto (el camino marcado con aristas en negrilla).



Con respecto al Nodo 3.

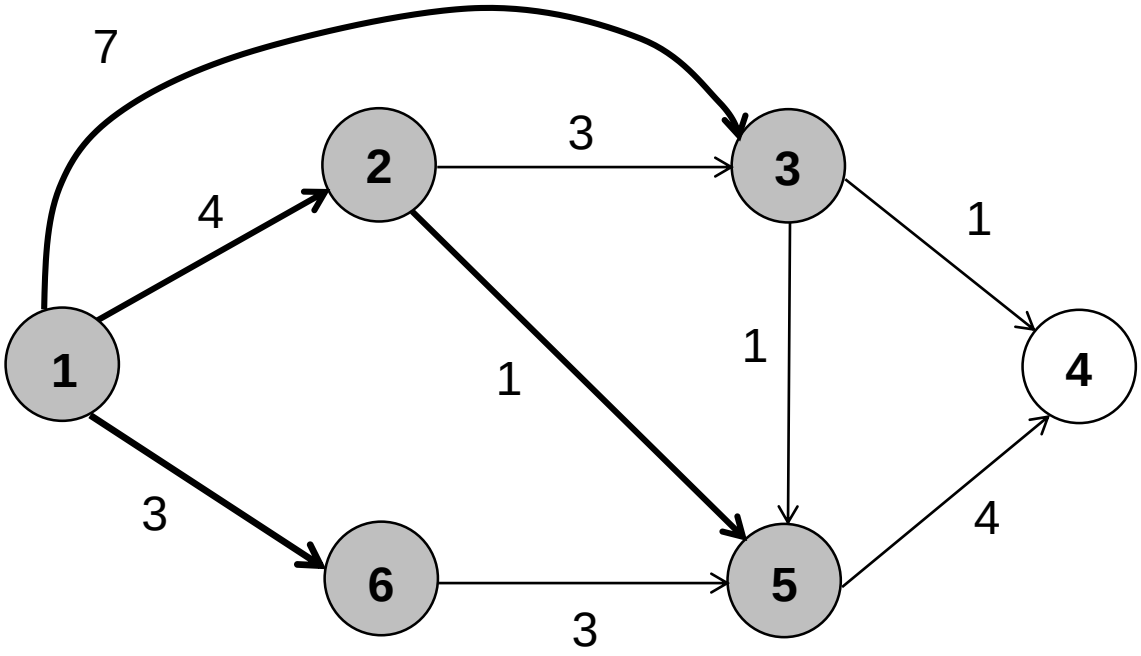
Hay DOS caminos diferentes que podemos tomar para llegar al nodo 3 desde los nodos que se han añadido al camino:
Opción 1: 1 -> 3 con una distancia de 7.
Opción 2: 1 -> 2 -> 3 con una distancia de 7 (4 + 3).

Con respecto al Nodo 4.

Opción: 1 -> 2 -> 5 -> 4 con una distancia de 9 (4 + 1 + 4).

Del Nodo 1 al Nodo	1	2	**3**	4	5	6
Distancia	0	4	**7**	9	5	3
Marca		**x**			**x**	**x**

De la lista de distancias, se detecta que se trata del nodo 3 con distancia 7. Opción 1 -> 3.

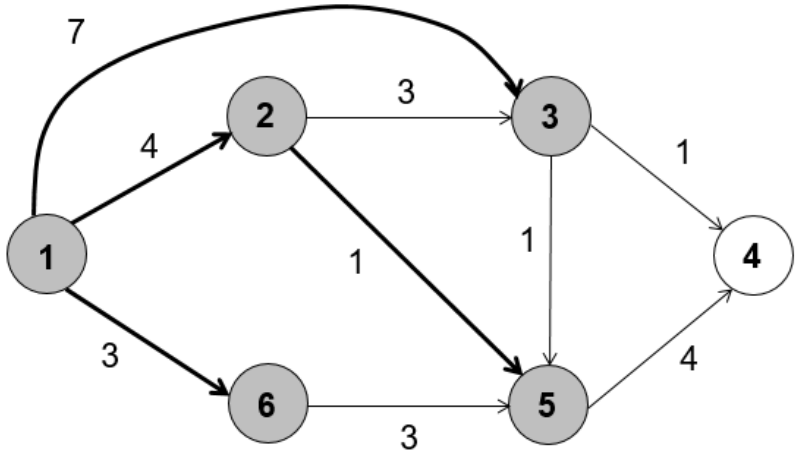


Lo marcamos como visitado en la lista de distancias y lo excluimos de la lista de NO-Visitados:

Del Nodo 1 al Nodo	1	2	3	4	5	6
Distancia	0	4	7	9	5	3
Marca		**x**	**x**		**x**	**x**

NO-Visitados = { 4 }

Debemos analizar los nuevos nodos adyacentes para encontrar el camino más corto para llegar a ellos. Sólo analizaremos los nodos adyacentes a los nodos que ya forman parte del camino más corto (el camino marcado con aristas en negrilla).

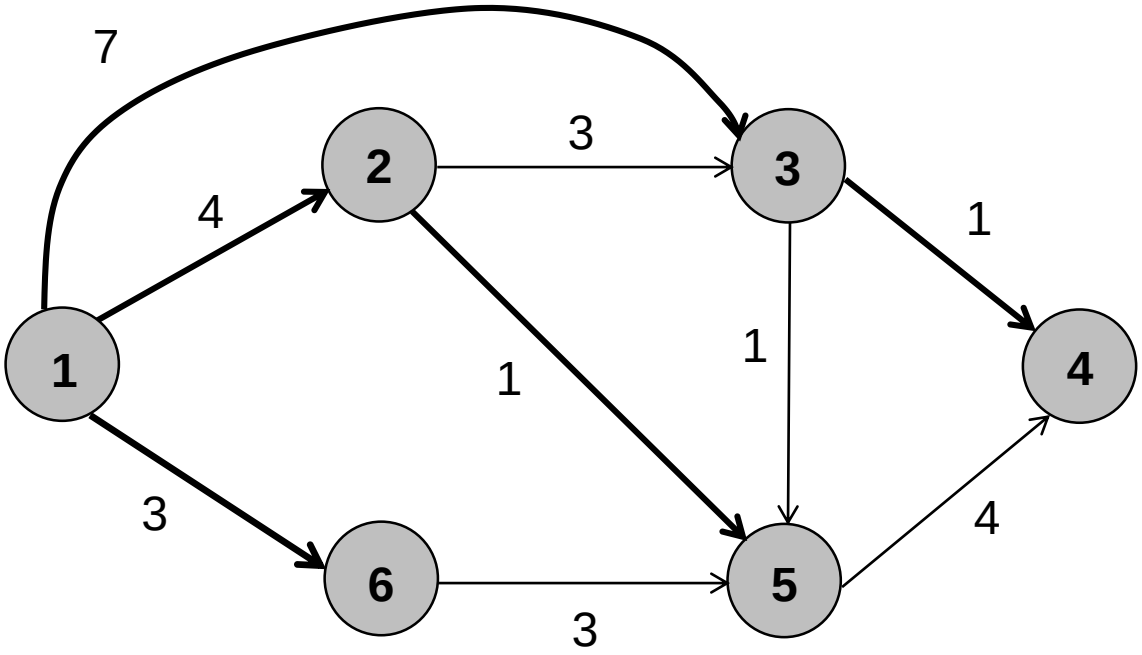


Con respecto al Nodo 4.

Opción 1: 1 -> 3 -> 4 con una distancia de 8 (7 + 1). ¡Encontramos una distancia menor a la ya registrada! ¡Actualizar!

Opción 2: 1 -> 2 -> 5 -> 4 con una distancia de 9 (7 + 1).

Del Nodo 1 al Nodo	1	2	3	**4**	5	6
Distancia	0	4	7	**8**	5	3
Marca		**x**	**x**		**x**	**x**



Lo marcamos como visitado en la lista de distancias y lo excluimos de la lista de NO-Visitados:

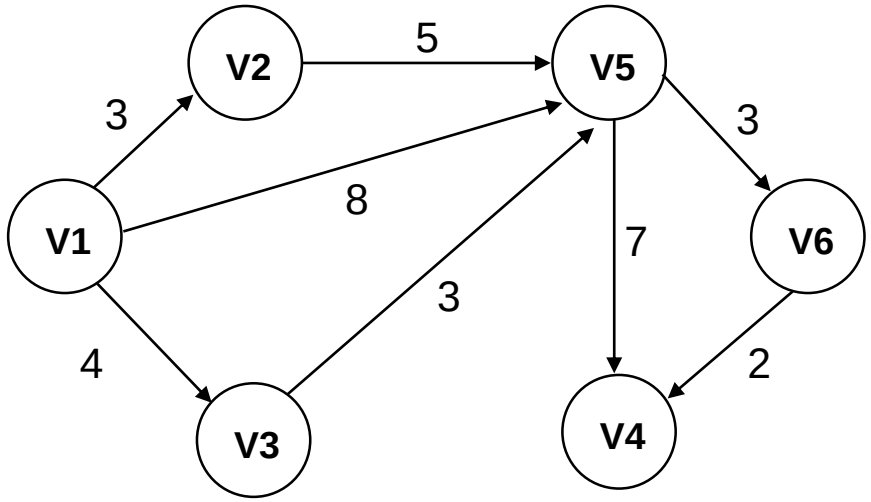
Del Nodo 1 al Nodo	1	2	3	4	5	6
Distancia	0	4	7	8	5	3
Marca		**x**	**x**	**x**	**x**	**x**

NO-Visitados = { }

Y terminamos porque la lista de nodos NO-visitados está vacía.

Así tenemos el resultado final con el camino más corto desde el nodo 1, a cada nodo del grafo.

Ejemplo 3. Obtenga los caminos mínimos para el siguiente grafo, usando el algoritmo de Dijkstra, comenzando en el Vértice 1.

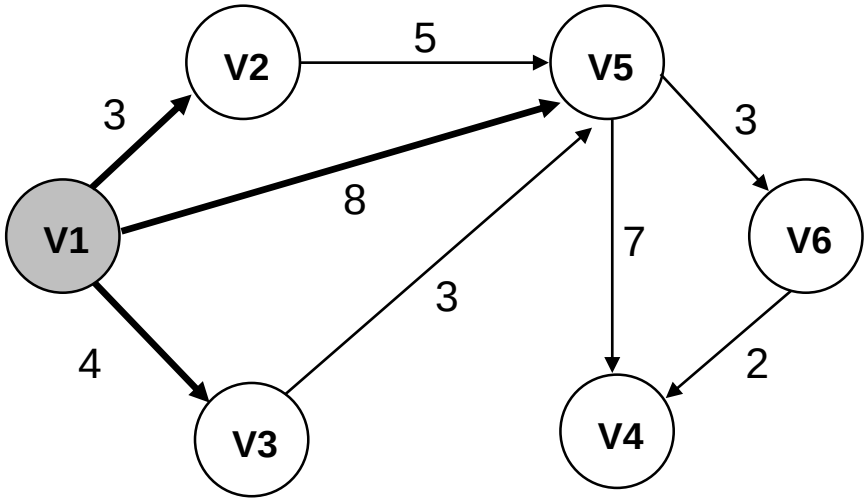


Situación inicial:

NO-Visitados = { V1 , V2 , V3 , V4 , V5 , V6 }

Del Nodo V1 al Nodo	V1	V2	V3	V4	V5	V6
Distancia	0	-	-	-	-	-
Marca						

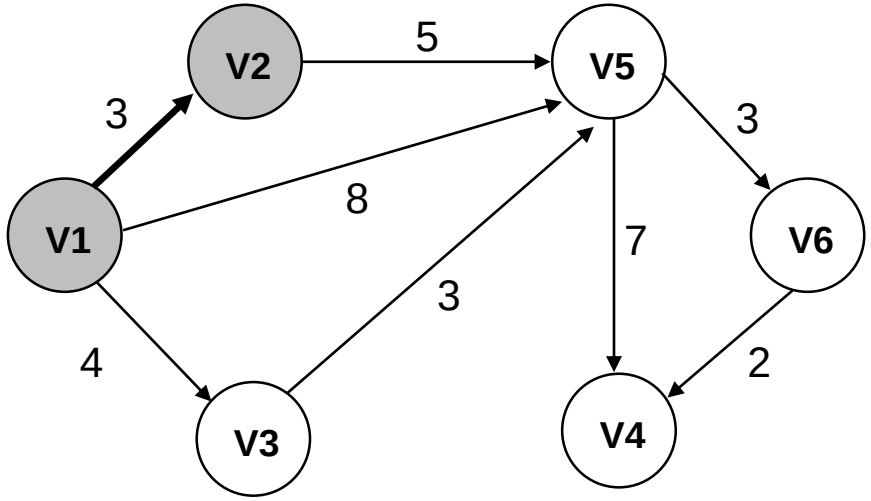
Comenzando por el nodo V1.
Lo sacamos del conjunto de NO-Visitados. **NO-Visitados = { V2 , V3 , V4 , V5 , V6 }**
Se chequean adyacentes al Nodo V1: Son los Nodos V2 , V3 y V5 (note las aristas en negrilla)



Situación:

Del Nodo V1 al Nodo	V1	**V2**	V3	V4	V5	V6
Distancia	0	**3**	4	-	8	-
Marca						

De la lista de distancias, se detecta que se trata del nodo V2 con distancia 3.

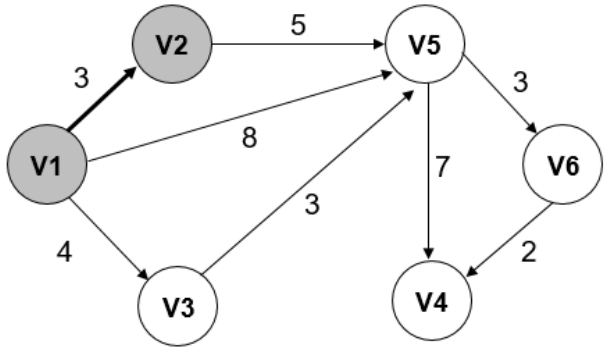


Lo marcamos como visitado en la lista de distancias y lo excluimos de la lista de NO-Visitados:

Del Nodo V1 al Nodo	V1	V2	V3	V4	V5	V6
Distancia	0	3	4	-	8	-
Marca		**x**				

NO-Visitados = { V3 , V4 , V5 , V6 }

Debemos analizar los nuevos nodos adyacentes para encontrar el camino más corto para llegar a ellos. Sólo analizaremos los nodos adyacentes a los nodos que ya forman parte del camino más corto (el camino marcado con aristas en negrilla).

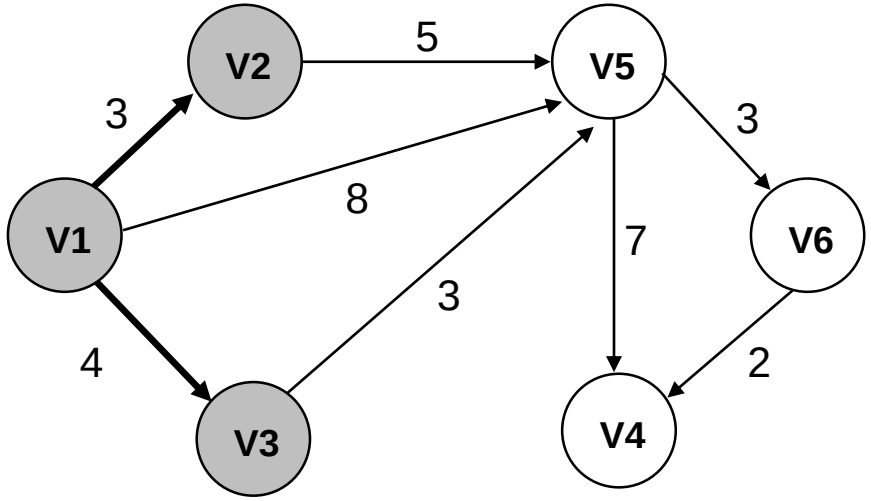


Con respecto al Nodo V5.
Opción 1: V1 -> V5 con una distancia de 8.
Opción 2: V1 -> V2 -> V5 con una distancia de 8 (3 + 5).

Del Nodo V1 al Nodo	V1	V2	**V3**	V4	V5	V6
Distancia	0	3	**4**	-	8	-
Marca		**X**				

De la lista de distancias, se detecta que se trata del nodo V3 con distancia 4.

Recuerden: Sólo actualizamos la distancia si el nuevo camino es más corto.

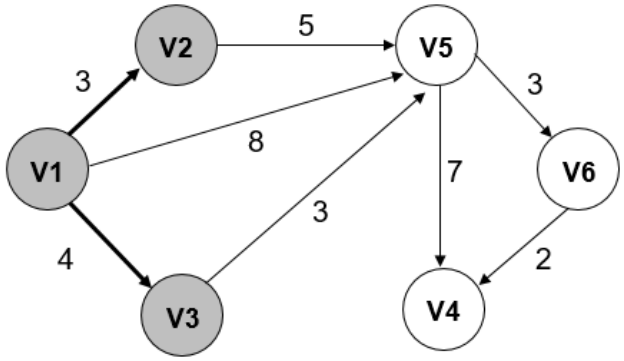


Lo marcamos como visitado en la lista de distancias y lo excluimos de la lista de NO-Visitados:

Del Nodo V1 al Nodo	V1	V2	V3	V4	V5	V6
Distancia	0	3	4	-	8	-
Marca		**x**	**x**			

NO-Visitados = { V4 , V5 , V6 }

Debemos analizar los nuevos nodos adyacentes para encontrar el camino más corto para llegar a ellos. Sólo analizaremos los nodos adyacentes a los nodos que ya forman parte del camino más corto (el camino marcado con aristas en negrilla).

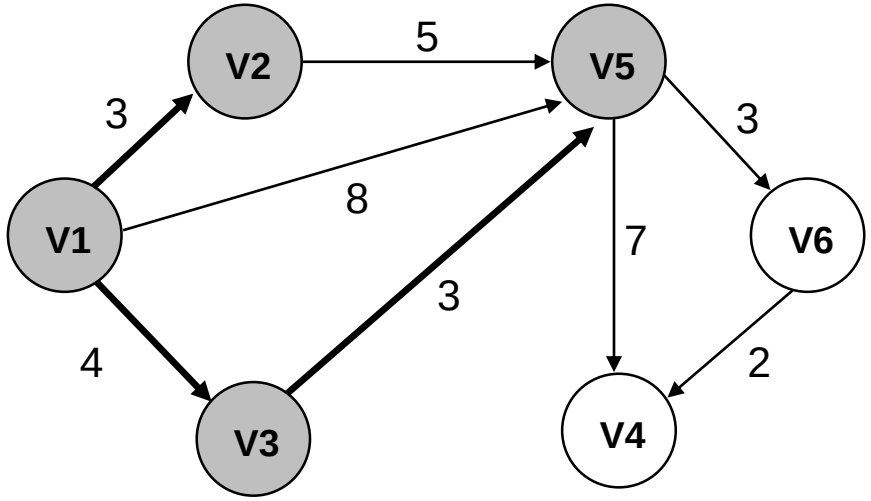


Con respecto al Nodo V5.
Opción 1: V1 -> V5 con una distancia de 8.
Opción 2: V1 -> V2 -> V5 con una distancia de 8 (3 + 5).
Opción 3: V1 -> V3 -> V5 con una distancia de 7 (4 + 3) ¡Encontramos una distancia menor a la ya registrada! ¡Actualizar!

Del Nodo V1 al Nodo	V1	V2	V3	V4	**V5**	V6
Distancia	0	3	4	-	**7**	-
Marca		**X**	**X**			

De la lista de distancias, se detecta que se trata del nodo V5 con distancia 7.

Recuerden: Sólo actualizamos la distancia si el nuevo camino es más corto.

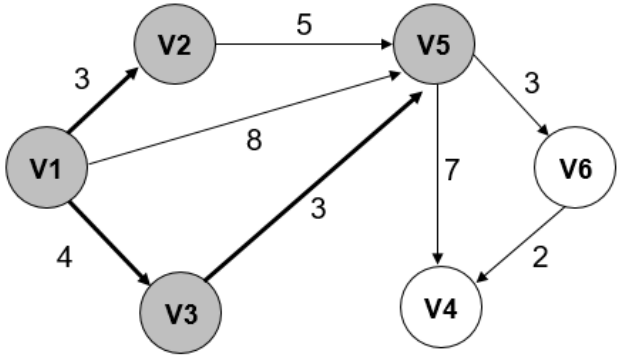


Lo marcamos como visitado en la lista de distancias y lo excluimos de la lista de NO-Visitados:

Del Nodo V1 al Nodo	V1	V2	V3	V4	V5	V6
Distancia	0	3	4	-	7	-
Marca		**x**	**x**		**x**	

NO-Visitados = { V4 , V6 }

Debemos analizar los nuevos nodos adyacentes para encontrar el camino más corto para llegar a ellos. Sólo analizaremos los nodos adyacentes a los nodos que ya forman parte del camino más corto (el camino marcado con aristas en negrilla).



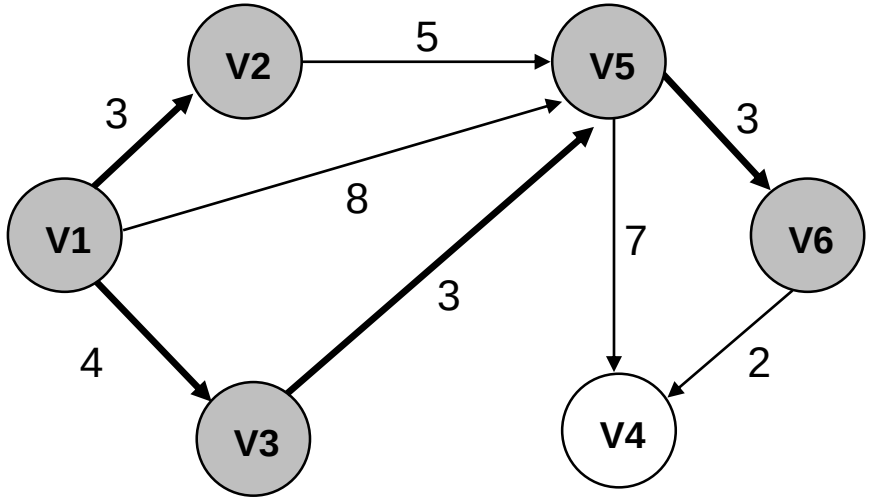
Con respecto al Nodo V4.
Opción: V1 -> V3 ->V5 -> V4 con una distancia de 14 (4 + 3 + 7)

Con respecto al Nodo V6.
Opción: V1 -> V3 ->V5 -> V6 con una distancia de 10 (4 + 3 + 3)

Del Nodo V1 al Nodo	V1	V2	V3	V4	V5	**V6**
Distancia	0	3	4	14	7	**10**
Marca		**X**	**X**		**X**	

De la lista de distancias, se detecta que se trata del nodo V6 con distancia 10.

Recuerden: Sólo actualizamos la distancia si el nuevo camino es más corto.

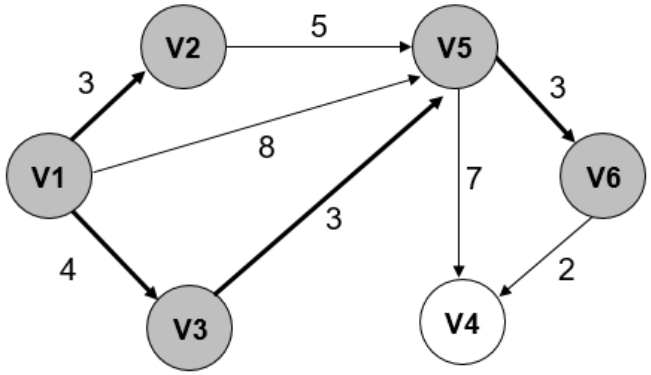


Lo marcamos como visitado en la lista de distancias y lo excluimos de la lista de NO-Visitados:

Del Nodo V1 al Nodo	V1	V2	V3	V4	V5	V6
Distancia	0	3	4	14	7	10
Marca		**x**	**x**		**x**	**x**

NO-Visitados = { V4 }

Debemos analizar los nuevos nodos adyacentes para encontrar el camino más corto para llegar a ellos. Sólo analizaremos los nodos adyacentes a los nodos que ya forman parte del camino más corto (el camino marcado con aristas en negrilla).

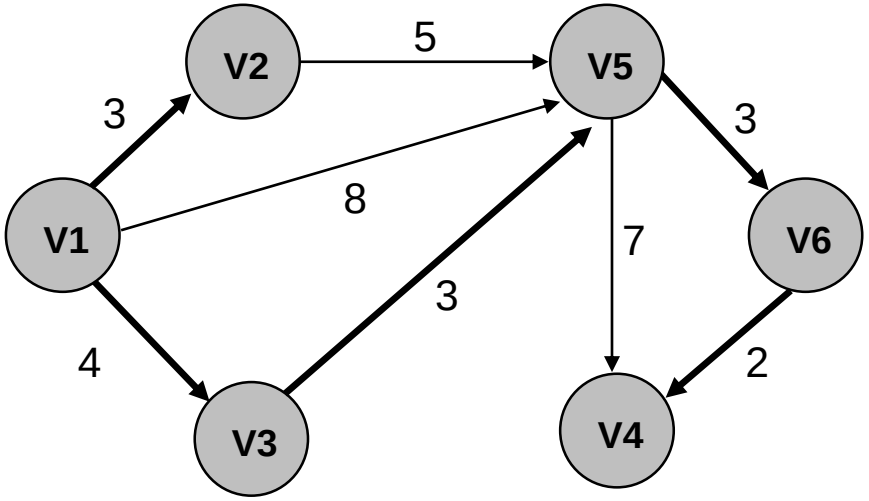


Con respecto al Nodo V4.
Opción 1:
V1 -> V3 ->V5 -> V4 con una distancia de 14 (4 + 3 + 7)

Opción 2: V1 -> V3 ->V5 -> V6 -> V4 con una distancia de 12 (4 + 3 + 3 + 2)
¡Encontramos una distancia menor a la ya registrada! ¡Actualizar!

Del Nodo V1 al Nodo	V1	V2	V3	V4	V5	**V6**
Distancia	0	3	4	**12**	7	10
Marca		**X**	**X**		**X**	**X**

Recuerden: Sólo actualizamos la distancia si el nuevo camino es más corto.



Lo marcamos como visitado en la lista de distancias y lo excluimos de la lista de NO-Visitados:

Del Nodo V1 al Nodo	V1	V2	V3	V4	V5	V6
Distancia	0	3	4	12	7	10
Marca		**x**	**x**	**x**	**x**	**x**

NO-Visitados = { }

Y terminamos porque la lista de nodos NO-visitados está vacía.

Así tenemos el resultado final con el camino más corto desde el nodo V1 a cada nodo del grafo.

---------- FIN DEL DOCUMENTO