Algoritmos Fundamentales

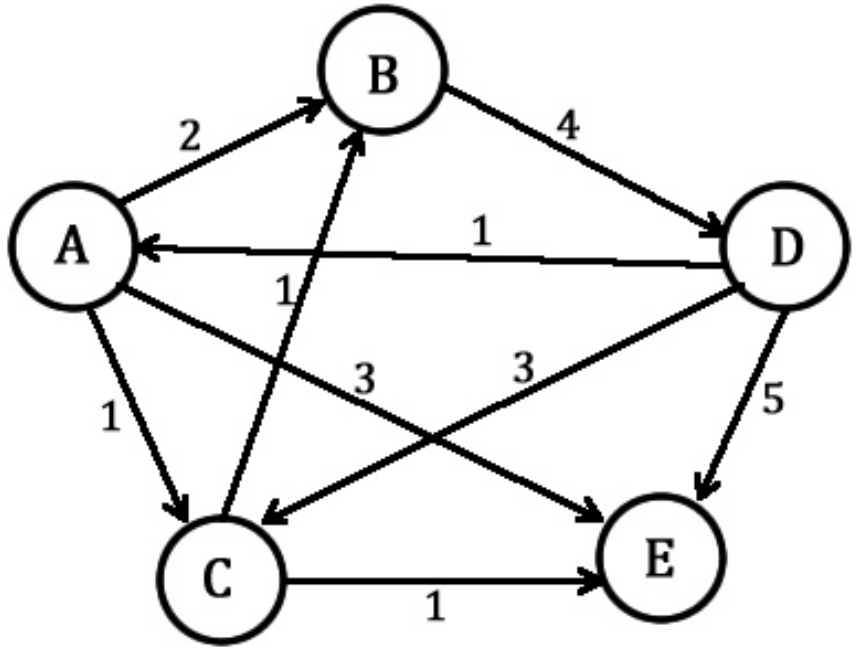
Dr. Said P. Martagón

Universidad Politécnica de Victoria

30 de marzo de 2016

Outline

Problema de la ruta mínima



¿Cómo llegar del punto A al punto D de la manera más corta posible?

¿Cómo se resuelve?

Existen algoritmos genéricos para ello:

- Dijkstra Algorithm
- Floyd Algorithm
- Bellman-Ford Algorithm

Outline

Algoritmo de Dijkstra.

También llamado algoritmo de caminos mínimos, es un algoritmo para la determinación del camino más corto dado un vértice origen al resto de vértices en un grafo con pesos en cada arista. Su nombre se refiere a Edsger Dijkstra, quien lo describió por primera vez en 1959.

Aplicaciones

En múltiples aplicaciones donde se aplican los grafos, es necesario conocer el camino de menor costo entre dos vértices dados:

- Distribución de productos a una red de establecimientos comerciales.
- Distribución de correos postales.

Sea $G = (V, A)$ un grafo dirigido ponderado. El problema del camino más corto de un vértice a otro consiste en determinar el camino de menor costo, desde un vértice u a otro vértice v. El costo de un camino es la suma de los costos (pesos) de los arcos que lo conforman.

Características del algoritmo

- Es un algoritmo greddy.
- Trabaja por etapas, y toma en cada etapa la mejor solución sin considerar consecuencias futuras.
- El óptimo encontrado en una etapa puede modificarse posteriormente si surge una solución mejor.

Algoritmo de Dijkstra

Entrada: Grafo ponderado dirigido de n vértices con pesos positivos; a y z vértices distintos tales que existe algún camino de a a z.

Salida: Peso de un camino de coste mínimo de a a z.

- **Paso 1:** Definimos $S_0 = \emptyset$, $T_0 = V$. Asignamos a cada vértice $v \in V$ una etiqueta como sigue: $L(v) = 0$ si $v = a$ y $L(v) = \infty$ para $v \neq a$.

- **Paso 2:** Para $i = 1, 2, \ldots, n$: Suponemos que hemos construido los conjuntos S_0, S_1, S_{i-1}. Hacemos $T_{i-1} = V|S_{i-1}$. Si $z \in S_{i-1}$, definimos $S = S_{i-1}$ y detenemos la construcción. En caso contrario, escogemos el primer vértice u en T_{i-1} con la menor etiqueta, es decir,

$$L(u) = \text{mín}\{L(v)|v \in T_{i-1}\}$$

Definimos $u_{i-1}, S_i = S_{i-1} \cup \{u_{i-1}\} = \{u_0, u_1, \ldots, u_{i-1}\}$ (decimos que entra u), $T_i = V|S_i$ y para cada vértice en v en T_i, adyacente a u cambiamos su etiqueta $L(v)$ por la nueva etiqueta $\text{mín}\{L(v), L(u) + p(u, v)\}$:

$$L(v) \leftarrow \text{mín}\{L(v), L(u) + p(u, v)\}$$

es decir, actualizamos la etiqueta de los "vecinos"de u por fuera de S_i

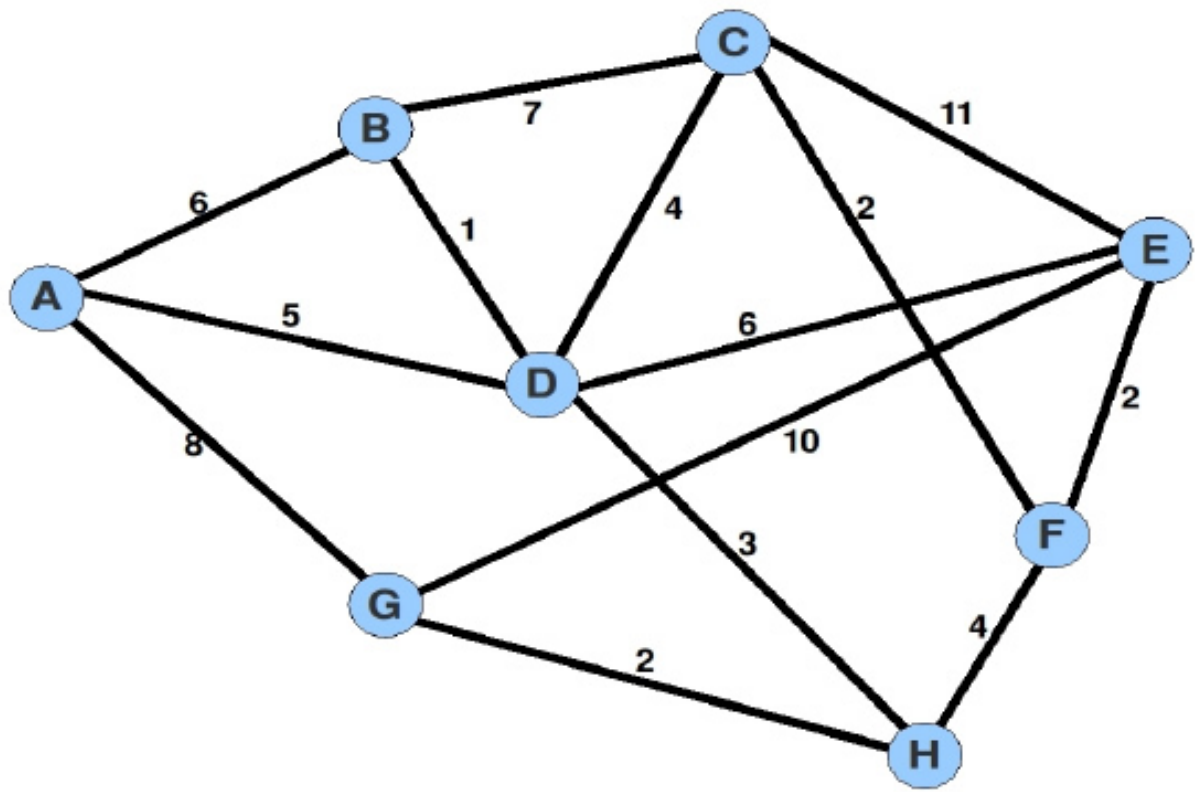
- **Paso 3**: Si $i = n$, definimos $S = s_n$ y nos detenemos. Si $i < n$, hacemos $i = i + 1$ y vamos al paso 2.

El algoritmo de Dijkstra termina en el momento en que encontramos el primer índice m para el cual $z \in S_m$. En ese momento, $S = S_m$.

Ejemplo

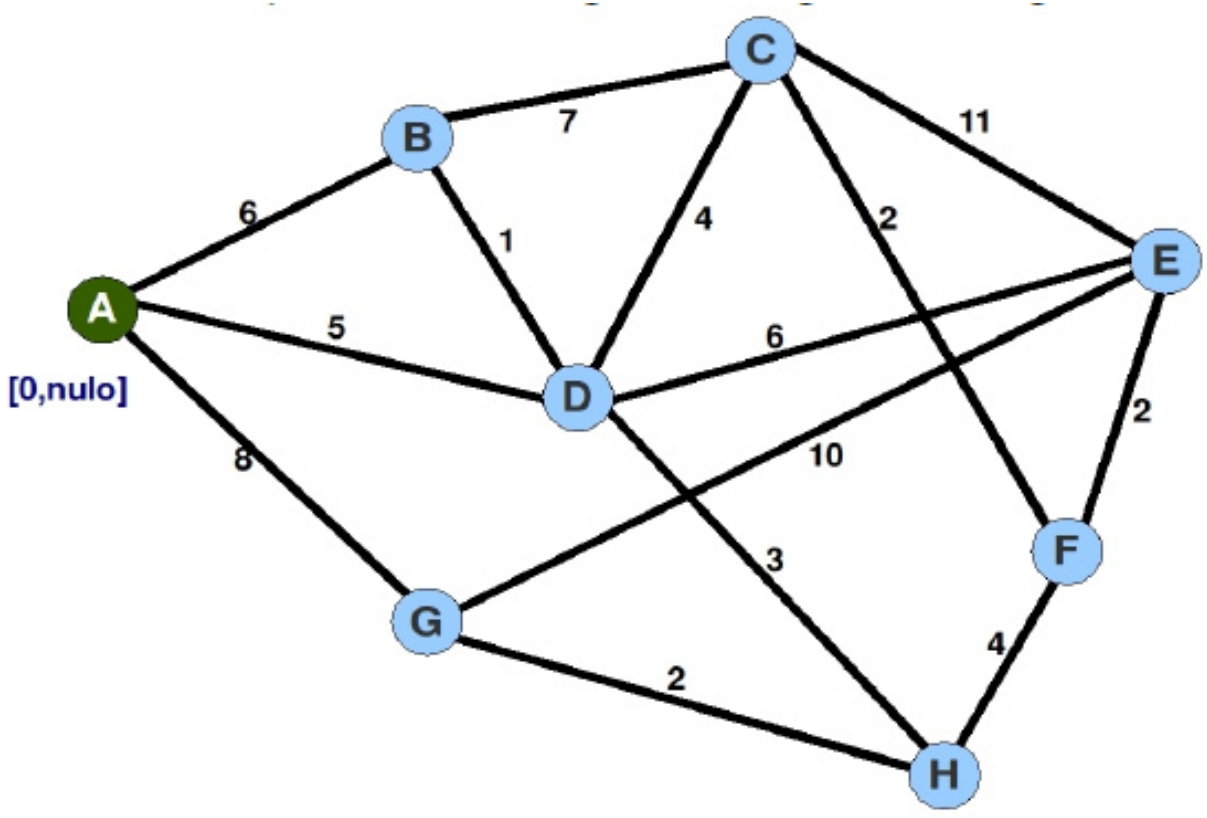
Antes de empezar pensemos que nuestros nodos pueden tener dos propiedades *acumuladoPeso* y otro llamando *nodoAntecesor*, el manejo de estas variables las aprenderemos en la medida que las vallamos *usandolas*.

Para efectos de esta explicación usaremos el grafo de la siguiente figura:



Primera iteración

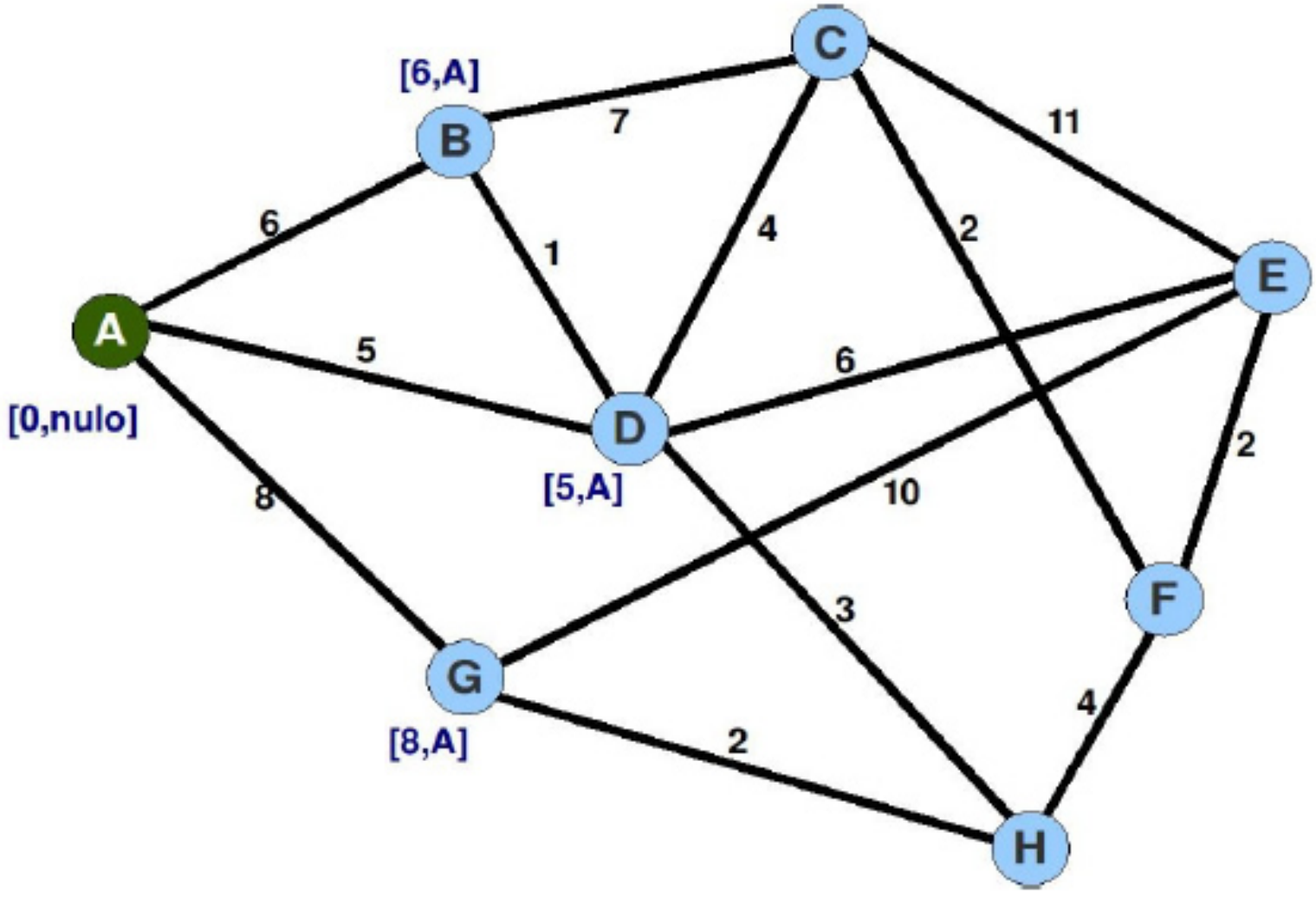
Escogemos un nodo inicio, en este caso usaremos el nodo A, e inicializamos sus variables de la siguiente forma:
$acumuladoPeso = 0, nodoAntecesor = null$
Al ser nuestro nodo inicio, no tiene pesos acumulados ni nodo antecesor, ademas lo cambiamos de color para indicar que ya lo usamos, quedando según la siguiente figura.



Segunda iteración

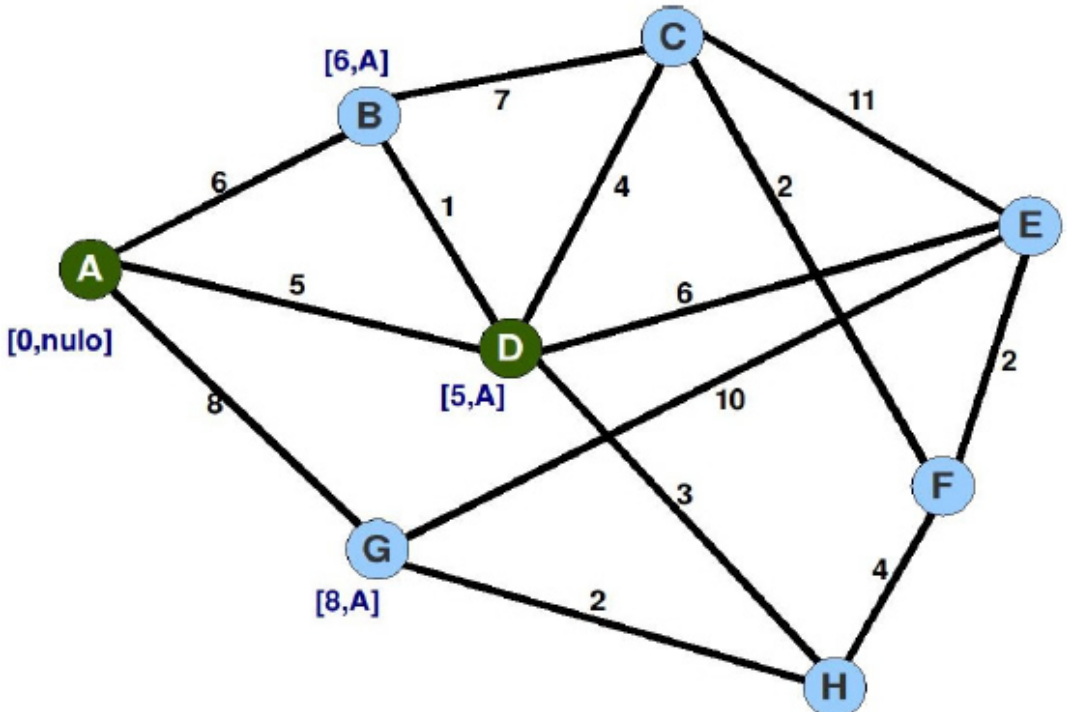
Marcamos los nodos adyacentes de A, inicializando sus variables de la siguiente forma: *acumuladoPeso* = *acumuladoPeso* del nodo antecesor (en este caso el de A) + peso de la arista que los une; *nodoAntecesor* = el nombre del nodo antecesor;

- Así para B
  - $acumuladoPeso = 0 + 6 = 6$;
  - $nodoAntecesor = A$;
- Para D
  - $acumuladoPeso = 0 + 5 = 5$;
  - $nodoAntecesor = A$;
- Para G
  - $acumuladoPeso = 0 + 8 = 8$;
  - $nodoAntecesor = A$;


C
[6,A]
B
7
11
6
4
2
1
E
A
5
6
[0,nulo]
D
2
[5,A]
10
8
F
3
G
4
[8,A]
2
H

Tercera iteración

Localizamos de los nodos visitados y que no esta marcado con el color verde y que tenga menor peso acumulado (*acumuladoPeso*), en caso que tengamos mas de uno con la característica antes descritas estamos el la libertad de escoger cualquiera de estos, una vez localizado lo marcamos de color verde.
Para nuestro caso el nodo a escoger es el D ya que su peso acumulado es el menor de todos con 5, quedando según la siguiente figura:



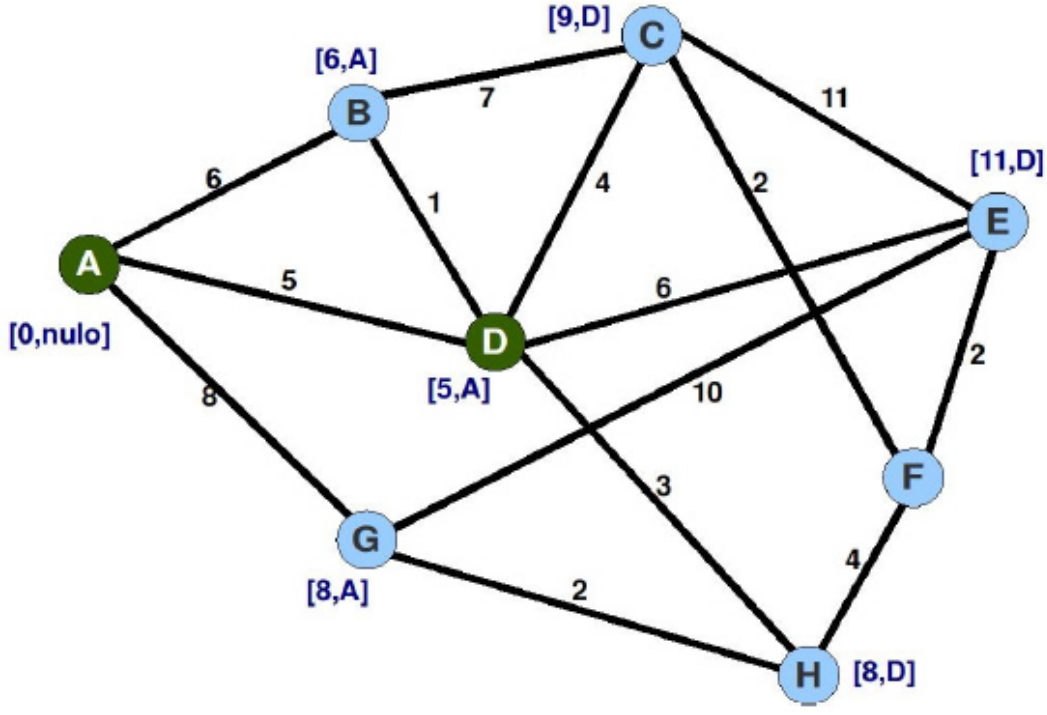
Cuarta iteración

Nos ubicamos en el nodo D, el cual escogimos en el paso anterior y marcamos sus nodos adyacentes siempre y cuando no lo hallamos usado, es decir no este de color verde. (ver como este paso es análogo la iteracion 2). Para este caso los nodos adyacentes de D no usados son: C, E, H, B; inicializamos los valores de nuestros nodos adyacentes así

:

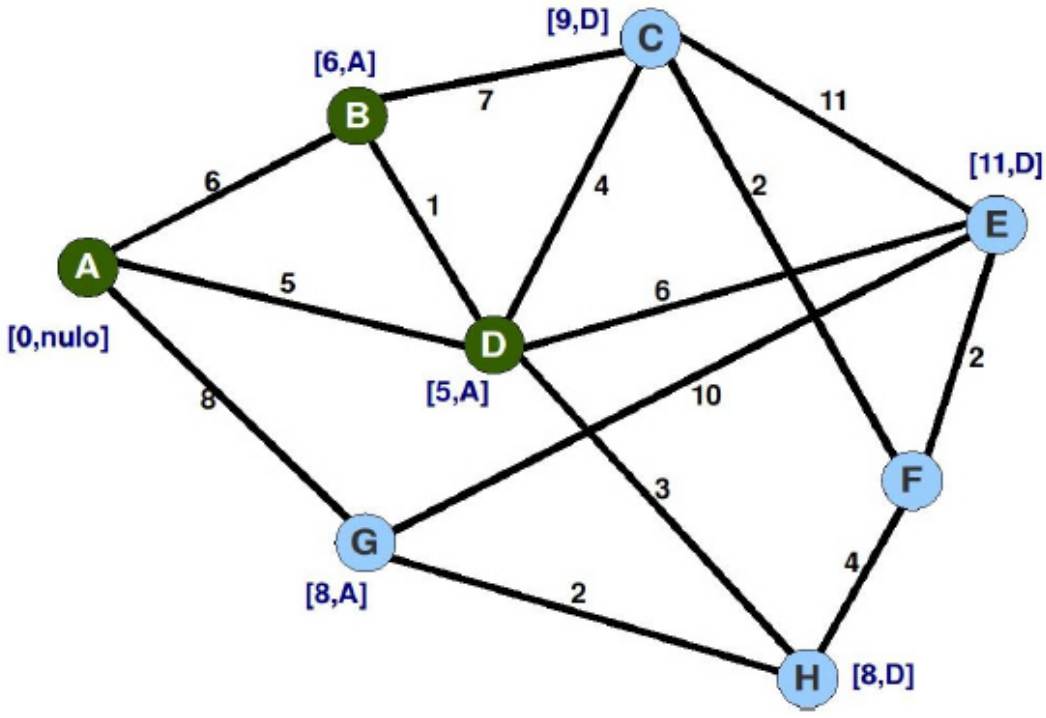
- Así para C
  - $acumuladoPeso = 5 + 4 = 9$;
  - $nodoAntecesor = D$;
- Para E
  - $acumuladoPeso = 5 + 6 = 11$;
  - $nodoAntecesor = D$;
- Para H
  - $acumuladoPeso = 5 + 3 = 8$;
  - $nodoAntecesor = D$;
- Para B
  - Como el nodo B ya esta marcado (por el nodo A), primero calculamos el peso acumulado asi: acumuladoPeso $= 5 + 1 =$ 6; como el nuevo acumuladoPeso no genera un valor mas pequeño del que ya estaba , es decir no lo mejora, el nodo B **no** lo modificamos.

Recordemos que el número 5 corresponde al *acumuladoPeso* del nodo antecesor. Quedando según la siguiente figura:



Quinta iteración

Localizamos de los nodos visitados y que no esta marcado con el color verde y que tenga menor peso acumulado (acumuladoPeso), en caso que tengamos mas de uno con la caracteristica antes descritas estamos el la libertar de escoger cualquiera de estos. Una vez localizado lo marcamos de color verde. (ver como este paso es analogo a la tercera iteracion) Para nuestro caso el nodo a escoger es el B ya que es el de menor peso acumulado con 6. Quedando según la siguiente figura:



Sexta iteración

Nos ubicamos en el nodo B, el cual escogimos en el paso anterior y marcamos sus nodos adyacentes siempre y cuando no lo hallamos usado, es decir no este de color verde. (ver como este paso es análogo a la segunda y cuarta iteración). Para este caso el unico nodo adyacentes de B no usado es C. Inicializamos los valores de nuestro nodo adyacente así:

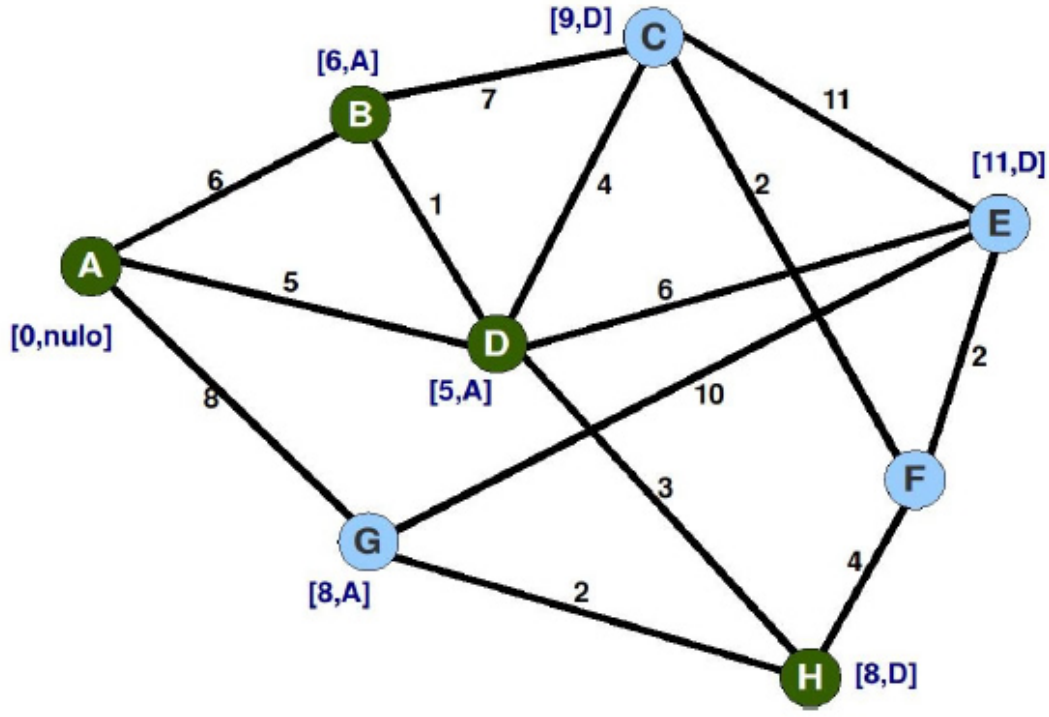
- Así para C
  - Como el nodo C ya esta marcado (por el nodo D), primero calculamos el peso acumulado así: acumuladoPeso $= 6 + 7 = 13$;

como el nuevo acumuladoPeso no generar un valor mas pequeño del que ya estaba, es decir no lo mejora, el nodo C no lo modificamos.
Recordemos que el numero 6 corresponde al acumuladoPeso del nodo antecesor.
Nuestro gráfico no sufre ningún cambio.

Séptima Iteración

Localizamos de los nodos visitados y que no esta marcado con el color verde y que tenga menor peso acumulado (acumuladoPeso), en caso que tengamos mas de uno con la característica antes descritas estamos el la libertar de escoger cualquiera de estos. Una vez localizado lo marcamos de color verde. (ver como este paso es análogo a la tercera y quinta iteracion) Para nuestro caso podemos escoger entre los nodos G,H ya que el peso acumulado en cada uno de ellos son los menores con 8; nos inclinaremos a usar el nodo H (si usamos el nodo G el resultado final no varia).

Quedando según la siguiente figura:

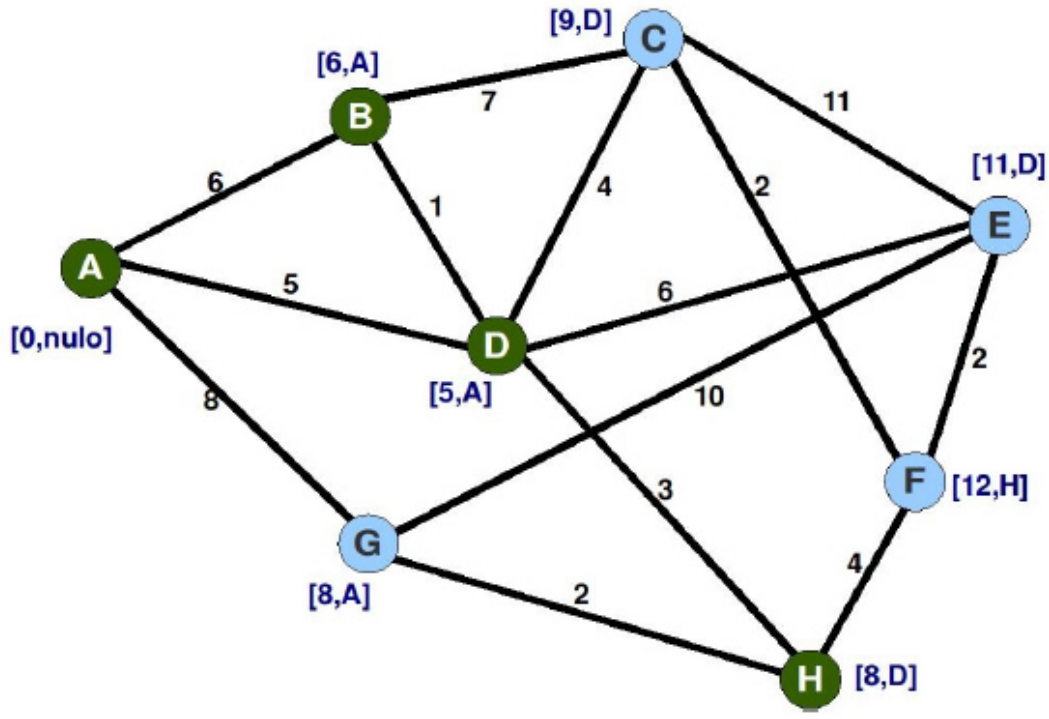


Octava iterción

Nos ubicamos en el nodo H, el cual escogimos en el paso anterior y marcamos sus nodos adyacentes siempre y cuando no lo hallamos usado, es decir no este de color verde. (ver como este paso es análogo a la segunda, cuarta y sexta iteracion). Para este caso los nodos adyacentes de H no usados son: F,G. Inicializamos los valores de nuestros nodos adyacentes así:

- Así para F
  - $acumuladoPeso = 8 + 4 = 12$;
  - $nodoAntecesor = H$;
- Para G
  - Como el nodo G ya esta marcado (por el nodo A), primero calculamos el peso acumulado así: acumuladoPeso $= 8 + 2 = 10$;

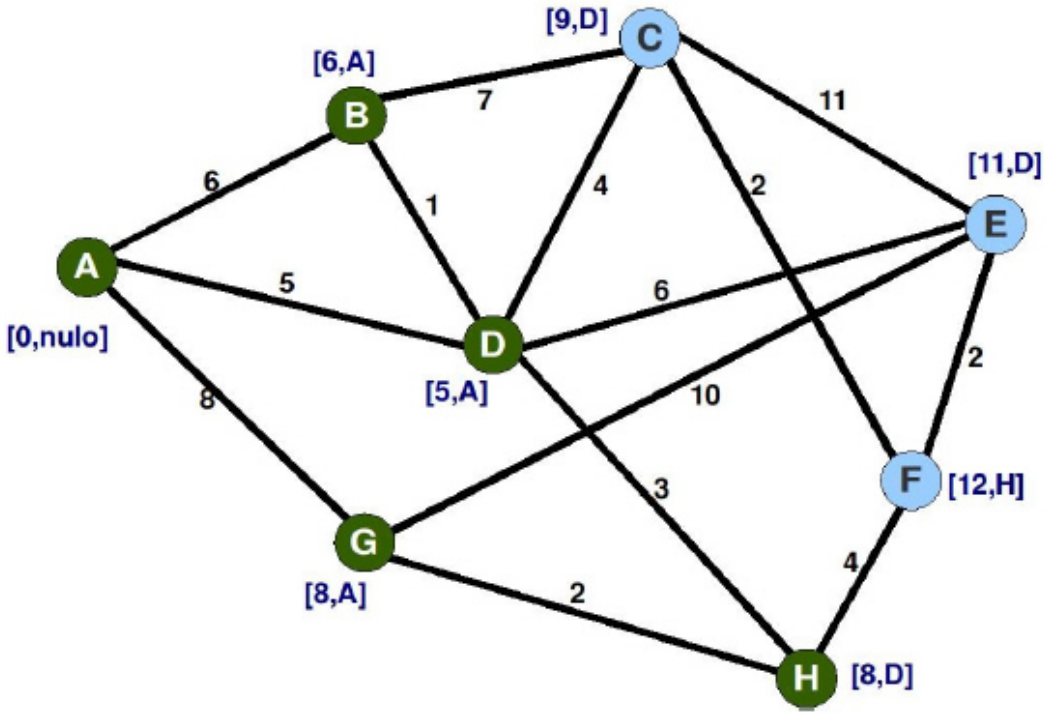
como el nuevo acumuladoPeso no genera un valor mas pequeño del que ya estaba , es decir, no lo mejora, el nodo G no lo modificamos. Recordemos que el numero 8 corresponde al acumuladoPeso del nodo antecesor.



Novena iteración

Localizamos de los nodos visitados y que no esta marcado con el color verde y que tenga menor peso acumulado (acumuladoPeso), en caso que tengamos mas de uno con la característica antes descritas estamos el la libertar de escoger cualquiera de estos.
Una vez localizado lo marcamos de color verde. (ver como este paso es análogo a la tercera, quinta y septima iteracion)
Para nuestro caso el nodo a escoger es el G ya que su peso acumulado es el menor con 8.

Quedando según la siguiente figura:



Décima iteración

Nos ubicamos en el nodo G, el cual escogimos en el paso anterior y marcamos sus nodos adyacentes siempre y cuando no lo hallamos usado, es decir no este de color verde. (ver como este paso es análogo a la segunda, cuarta, sexta iteracion, etc). Para este caso el unico nodo adyacentes de G no usado es E.
Inicializamos los valores de nuestro nodo adyacente así:

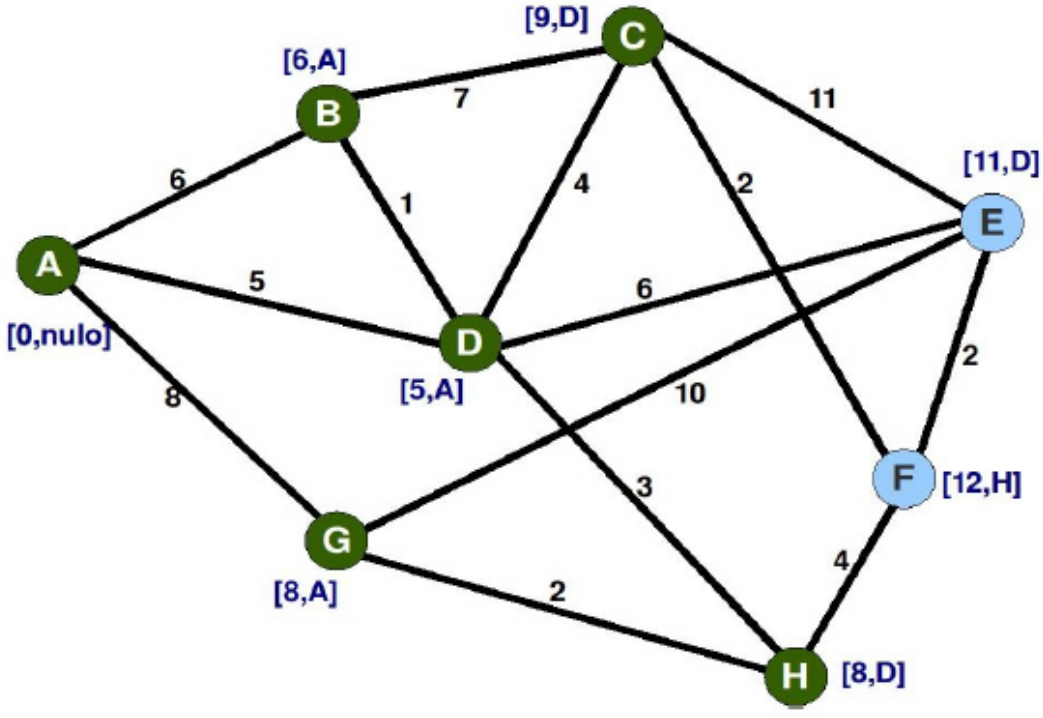
- Así para E
  - Como el nodo E ya esta marcado (por el nodo D), primero calculamos el peso acumulado así: acumuladoPeso $= 8 + 10 = 18$;

como el nuevo acumuladoPeso no generar un valor mas pequeño del que ya estaba , es decir no lo mejora, el nodo E no lo modificamos.
Recordemos que el numero 8 corresponde al acumuladoPeso del nodo antecesor.
Nuestro grafico no sufre cambios.

Décimo primera iteración

Localizamos de los nodos visitados y que no esta marcado con el color verde y que tenga menor peso acumulado (acumuladoPeso), en caso que tengamos mas de uno con la caracteristica antes descritas estamos el la libertar de escoger cualquiera de estos. Una vez localizado lo marcamos de color verde. (ver como este paso es análogo a la tercera, quinta, septima iteracion, etc). Para nuestro caso el nodo a escoger es el C ya que su peso acumulado es el menor con 9.

Quedando según la siguiente figura:



Décimo segunda iteración

Nos ubicamos en el nodo C, el cual escogimos en el paso anterior y marcamos sus nodos adyacentes siempre y cuando no lo hallamos usado, es decir no este de color verde. (ver como este paso es análogo a la segunda, cuarta, sexta iteracion, etc). Para este caso los nodo adyacentes de C son E,F.

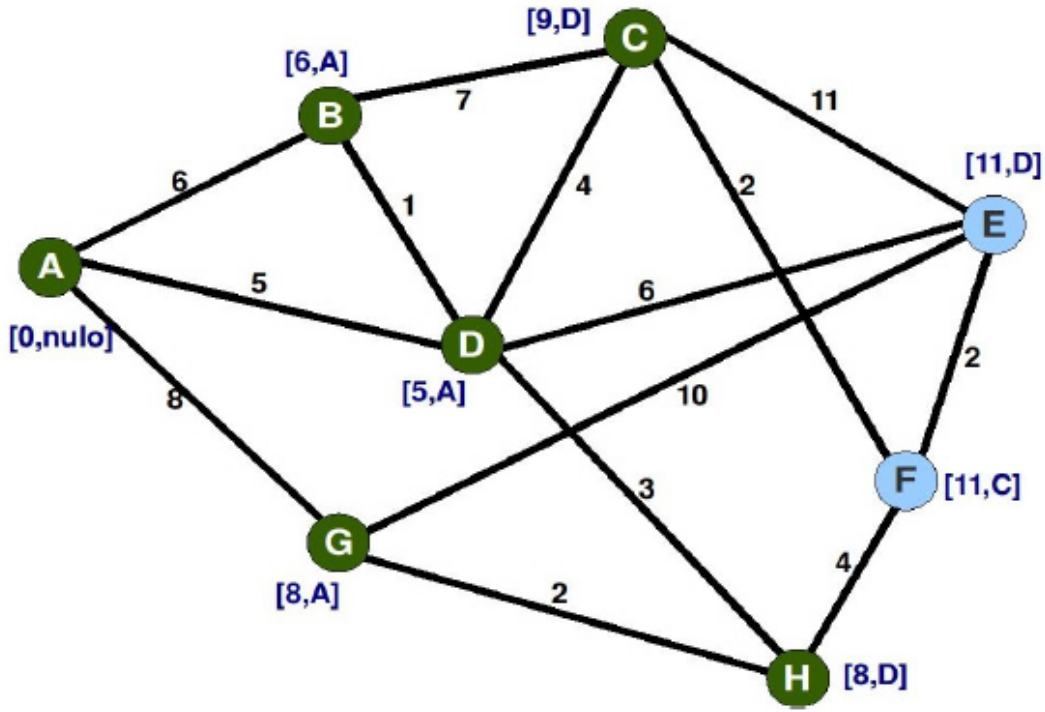
Inicializamos los valores de nuestros nodos adyacente así:

- Así para E
  - Como el nodo E ya esta marcado (por el nodo D), primero calculamos el peso acumulado así: acumuladoPeso $= 9 + 11 = 20$;

  como el nuevo acumuladoPeso no genera un valor más pequeño del que ya estaba, es decir, no mejora, el nodo E no lo modificamos.
- Así para F
  - Como el nodo F ya esta marcado (por el nodo H), primero calculamos el peso acumulado así: acumuladoPeso $= 9 + 2 = 11$;

Recordemos que el numero 9 corresponde al acumuladoPeso del nodo antecesor.

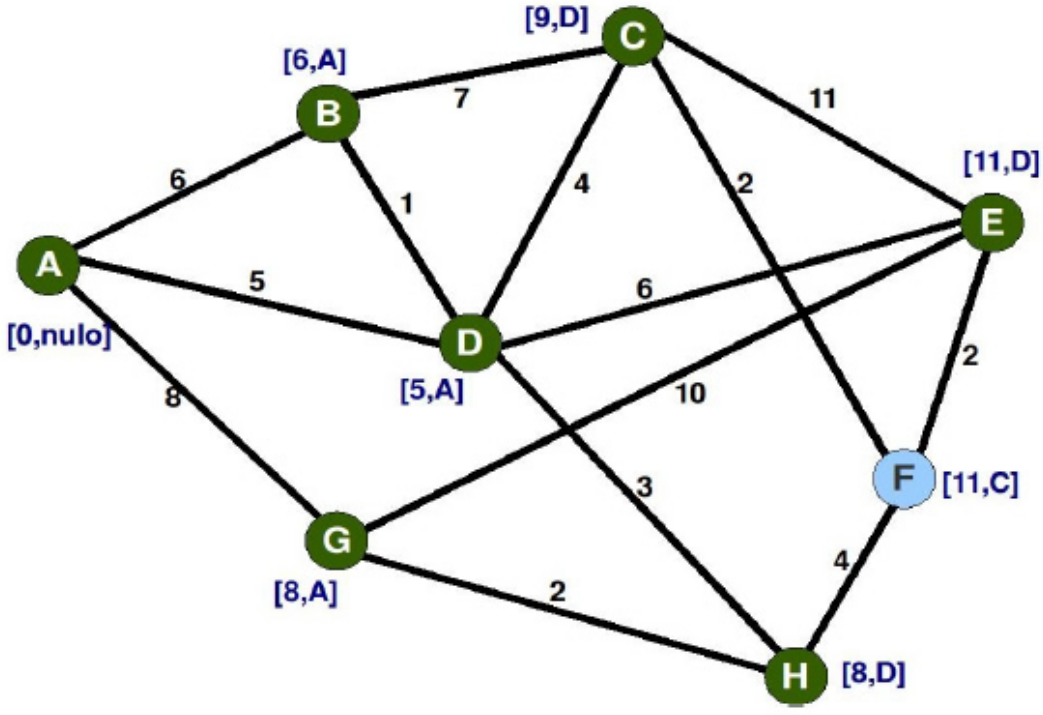
como el valor cumulado de F es 12 y el que acabamos de calcular es menor (11) es decir que si lo mejora, cambiamos el valor acumulado de F por 11 y el nodo antecesor por C por ser C quien lo mejoro. Recordemos que el numero 9 corresponde al acumuladoPeso del nodo antecesor. Quedando según la siguiente figura: (Observe la figura anterior y ésta que sigue para que compare el cambio que sufrió el nodo F).



Décimo tercera iteración

Localizamos de los nodos visitados y que no esta marcado con el color verde y que tenga menor peso acumulado (acumuladoPeso), en caso que tengamos mas de uno con la característica antes escritas estamos el la libertar de escoger cualquiera de estos.
Una vez localizado lo marcamos de color verde. (ver como este paso es análogo a la tercera, quinta, septima iteracion, etc).
Para nuestro caso tenemos dos nodo que podemos usar por ser los de menor pesos E,F
Cualquiera de los dos que usemos esta bien, para este caso tomaremos el nodo E.

Quedando según la siguiente figura:



Décimo cuarta iteración

Nos ubicamos en el nodo E, el cual escogimos en el paso anterior y marcamos sus nodos adyacentes siempre y cuando no lo hallamos usado, es decir no este de color verde. (ver como este paso es análogo a la segunda, cuarta, sexta iteracion, etc). Para este caso el único nodo adyacentes de E es F. Inicializamos los valores de nuestro nodo adyacente así:

- Así para F
  - Como el nodo F ya esta marcado (por el nodo C), primero calculamos el peso acumulado así: acumuladoPeso $= 11 + 2 = 13$;

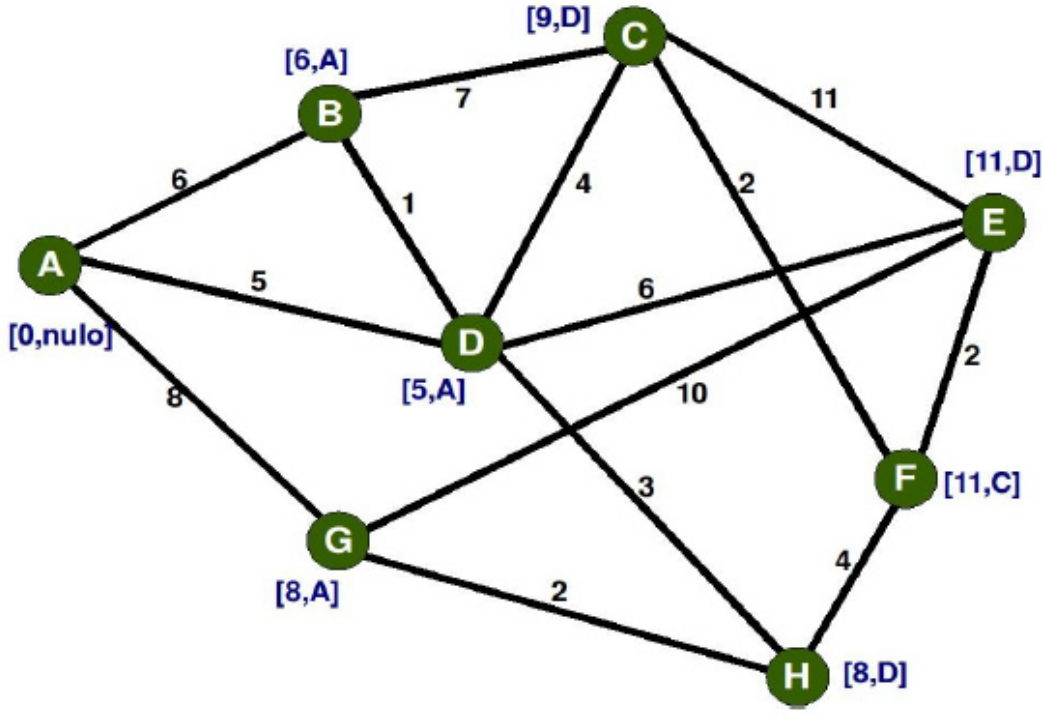
  como el nuevo acumuladoPeso no genera un valor más pequeño del que ya estaba, es decir, no mejora, el nodo F no lo modificamos.

Recordemos que el numero 11 corresponde al acumuladoPeso del nodo antecesor. Nuestro gráfico no sufre cambios

Décimo quinta iteración

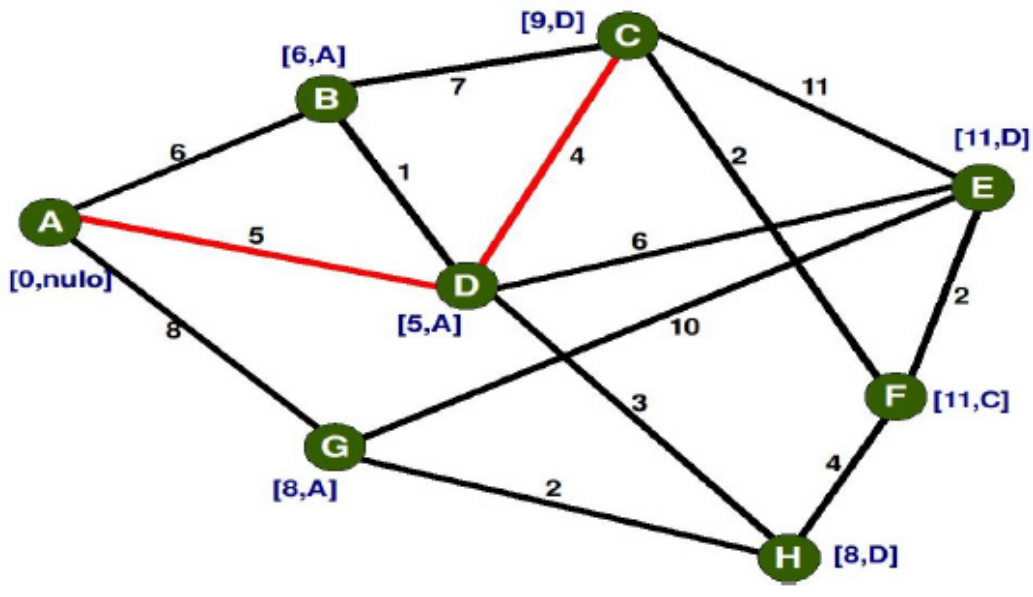
Localizamos de los nodos visitados y que no esta marcado con el color verde y que tenga menor peso acumulado (acumuladoPeso), en caso que tengamos mas de uno con la característica antes escritas estamos el la libertar de escoger cualquiera de estos.
Una vez localizado lo marcamos de color verde. (ver como este paso es análogo a la tercera, quinta, septima iteracion, etc).
Para nuestro caso ya solo nos queda F.

Quedando según la siguiente figura:



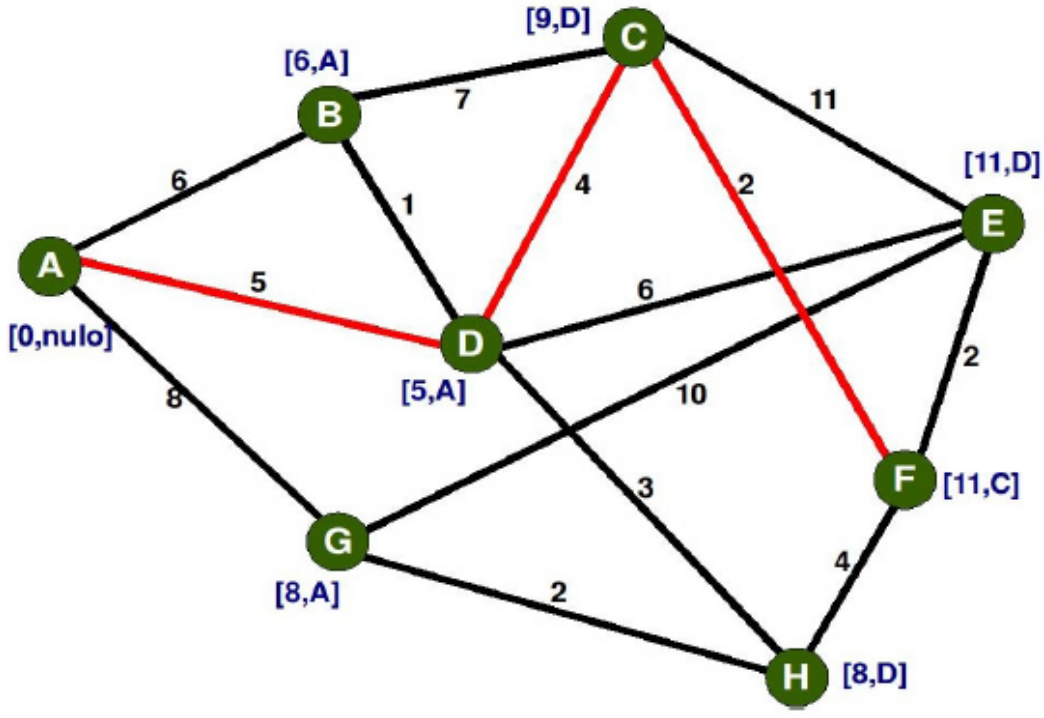
Listo, con la décimo quinta iteracion terminamos de aplicar el dijkstra a este grafo, ya que marcamos todos nuestros nodos; ahora podemos identificar el camino mas corto de un nodo cualquiera al nodo A el cual inicialmente escogimos como nodo inicio.
Y ¿Como hacemos esto?: digamos que quiero conocer la ruta mas corta del nodo C al nodo A.
Primero nos ubicamos en el C y miramos cual es su nodo antecesor, gráficamente vemos que es el nodo D.

Ahora nos ubicamos en el nodo D y miramos cual es su nodo antecesor, graficamente vemos que es A.
Como finalmente el nodo A no tiene antecesor significa que hemos llegado al nodo inicio. Fácilmente podemos ver que nuestra ruta es A, D y C.



Miremos este ultimo ejemplo. Digamos que quiero conocer la ruta mas corta del nodo F al nodo A.
Primero nos ubicamos en el F y miramos cual es su nodo antecesor, gráficamente vemos que es el nodo C.
Nos ubicamos en el nodo C y miramos cual es su nodo antecesor, graficamente vemos que es D.

Ahora nos ubicamos en el nodo D y miramos cual es su nodo antecesor, gráficamente vemos que es A.
Como finalmente el nodo A no tiene antecesor significa que hemos llegado al nodo inicio.
Fácilmente podemos ver que nuestra ruta es A, D, C y F.



Outline

Antecedentes

El algoritmo de Bellman-Ford (algoritmo de Bell-End-Ford) genera el camino más corto en un grafo dirigido ponderado (en el que el peso de alguna de las aristas puede ser negativo). El algoritmo de Dijkstra resuelve este mismo problema en un tiempo menor, pero requiere que los pesos de las aristas no sean negativos, salvo que el grafo sea dirigido y sin ciclos. Por lo que el Algoritmo Bellman-Ford normalmente se utiliza cuando hay aristas con peso negativo. Este algoritmo fue desarrollado por Richard Bellman, Samuel End y Lester Ford.

Según Robert Sedgewick, "Los pesos negativos no son simplemente una curiosidad matemática; [. . .] surgen de una forma natural en la reducción a problemas de caminos más cortos", y son un ejemplo de una reducción del problema del camino hamiltoniano que es NP-completo hasta el problema de caminos más cortos con pesos generales. Si un grafo contiene un ciclo de coste total negativo entonces este grafo no tiene solución. El algoritmo es capaz de detectar este caso.

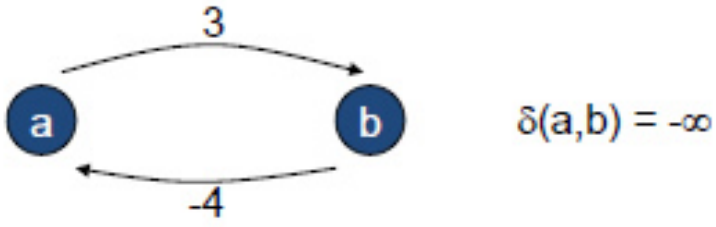
Si el grafo contiene un ciclo de coste negativo, el algoritmo lo detectará, pero no encontrará el camino más corto que no repite ningún vértice. La complejidad de este problema es al menos la del problema del camino más largo de complejidad NP-Completo.

El Algoritmo de Bellman-Ford es, en su estructura básica, muy parecido al algoritmo de Dijkstra, pero en vez de seleccionar vorazmente el nodo de peso mínimo aun sin procesar para relajarlo, simplemente relaja todas las aristas, y lo hace $|V| - 1$ veces, siendo $|V|$ el número de vértices en el grafo. Las repeticiones permiten a las distancias mínimas recorrer el árbol, ya que en la ausencia de ciclos negativos, el camino más corto solo visita cada vértice una vez. A diferencia de la solución voraz, la cual depende de la suposición de que los pesos sean positivos, esta solución se aproxima más al caso general.

Arcos negativos

Los caminos más cortos están bien definidos si y sólo si **no existen ciclos negativos.**

De otra manera no esta´ra bien definido, pues cada vez que demos vuelta a otro ciclo del camino será mas corto.



Caminos más cortos

- El camino mas corto siempre esta bien definido por ser un dag.
- *Se relajan* los nodos en base a un *ordenamiento topológico* de los mismos.
- *Si hay un camino desde u hasta v en G, entonces u precede a v* en el orden topológico.
- Los algoritmos de Bellman-Ford y de Dijstra usan los algoritmos como *inicialización y relajación.*

```
inicialización(G,s)
para cada vértice v en G
d[v] = INFINITO
padre[v] = NULO
d[s] = 0
```

Caminos Cortos y Relajación

La *técnica de relajación* es esencial para comprender los algoritmos de caminos más cortos.

La *Relajación* es el método mediante el cual se **decrece la cota superior del peso del camino actual hasta que iguale a la cota del camino más corto.**

Relajación: $\forall v \in N$, se mantiene un atributo $d(v)$ que es el límite superior de los pesos del camino más corto de s a v, d(v) se denomina el **el estimado del peso del camino más corto.**

Algoritmo

```
Relajamiento(u,v)
si d[v] > d[u] + peso(u)
d[v] = d[u] + peso (u)
padre[v] = u
```

Propiedades de la relajación: Sea $G = (N, A)$ un digrafo etiquetado con w: $a \to \mathbb{R}$.

- Sea $(u, v) \in A$ luego de la relajación del arco (u, v) por medio de relajar() se tiene que $d(v) \leq d(u) + w(u, v)$.
- Sea $s \in N$ el nodo fuente y G iniciado con iniciar(s), entonces $d(v) \geq \delta(s, v) \forall v \in N$ y esta invariante se mantiene sobre cualquier secuencia de relajación de los arcos de G. Más aún, **cuando** $d(v)$ **alcanza su límite inferior** $\delta(s, v)$**, este nunca cambia.**

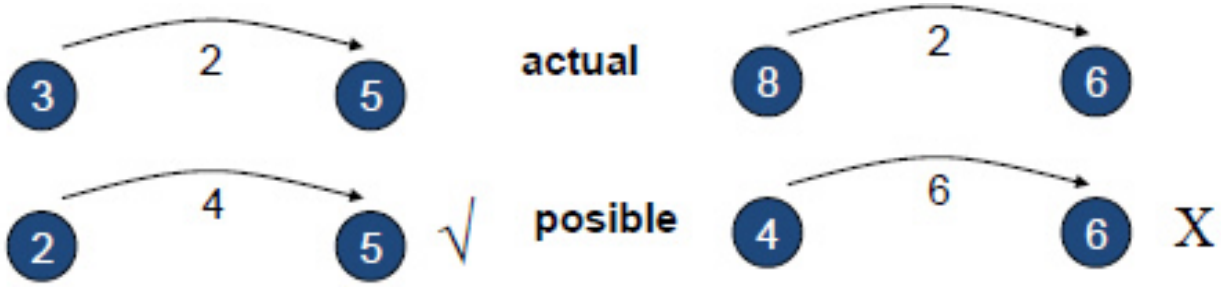
Algorithm 1 Initialization (G, s)

1: **for each** vertex $v \subset V[G]$ **do**
2: $\quad d[v] \leftarrow \infty$
3: $\quad \prod[v] \leftarrow NIL\ d[s] \leftarrow 0$
4: **end for**
5: $d[s] \leftarrow 0$

Relajar es mejorar la distancia de un nodo a otro. Dado (u, v) se pregunta si el camino calculado hasta ahora hasta v es mejor que el camino hasta u y luego $w(u, v)$

Algorithm 2 RELAX (u, v, w)

1: **if** $d[v] > d[u] + w(u, v)$ **then**
2: $\quad d[v] \leftarrow d[u] + w(u, v)$
3: $\quad \prod[v] \leftarrow u$
4: **end if**



Explicación del algoritmo

- **En el paso 0,** inicializamos todas las distancias o costos mínimos a infinito.
- En el paso 1, actualizamos el paso anterior, aplicando la relajación. En este caso ponemos la distancia de los nodos que tienen accesos directos al vértice 1, y se la sumamos a la distancia mínima acumulada que hay hasta el vértice.
- En el paso 2, al haber ya una distancia mínima acumulada desde los nodos 2 y 3 hasta 1, podemos actualizar las distancias mínimas de los nodos 4 y 5.
- En los pasos sucesivos, se van actualizando las distancias mínimas acumuladas (D) de los distintos vértices hasta 1.

Se termina el algoritmo cuando no hay ningún cambio de un paso a otro, ya no se puede encontrar un camino más corto.

Algoritmo de Bellman-Ford

Se trata de resolver el mismo problema en el caso de pesos negativos y positivos.

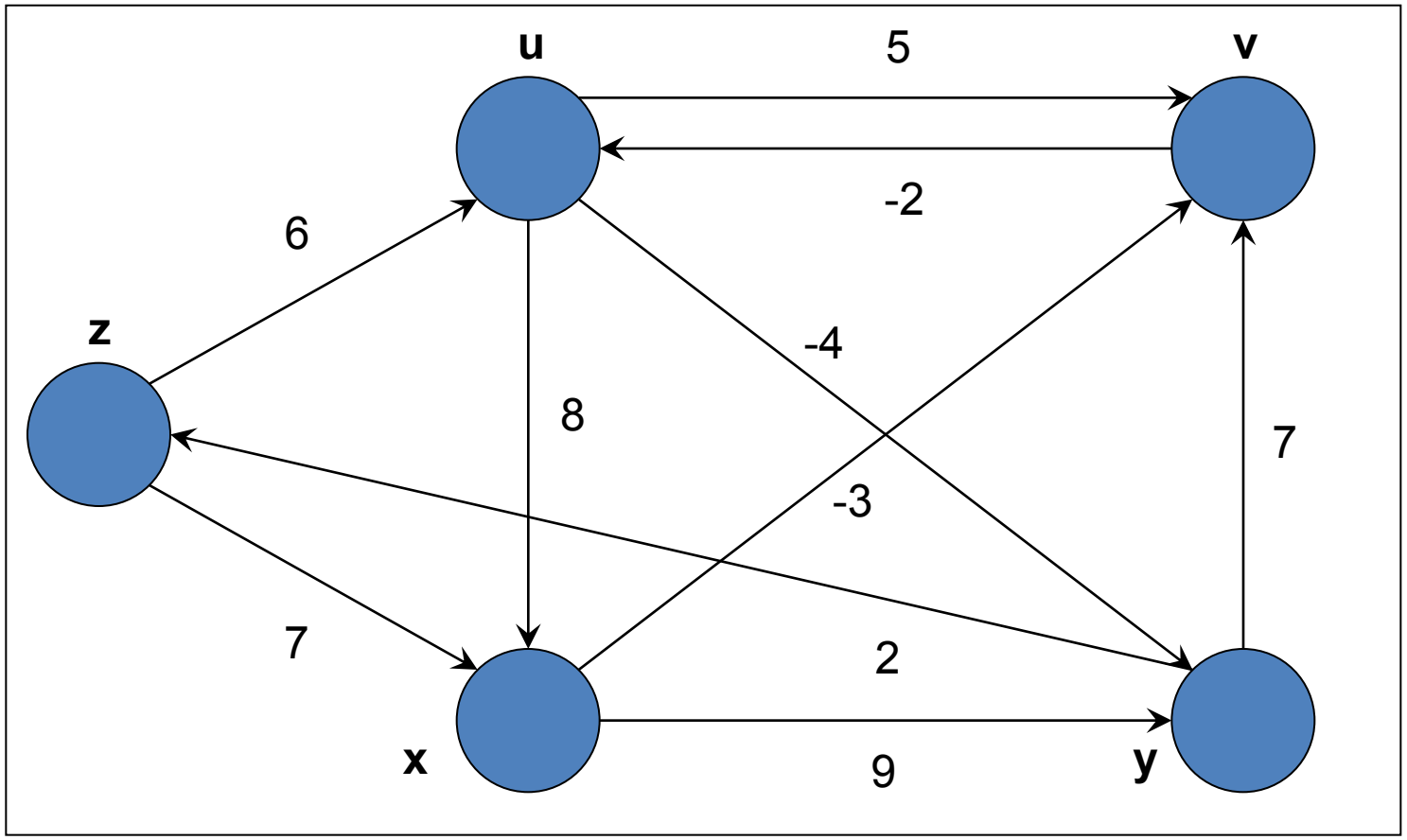
Algorithm 3 Bellman-Ford

Require: G, w, s
Ensure: *bool*

```
1: for each v ∈ V[G] do
2:     d[v] = ∞
3:     p[v] = NULL
4: end for
5: d[s] = 0
6: for i = 1 to |V[G]| − 1 do
7:     for each arco (u, v)enE[G] do
8:         Relax(u, v, w)
9:     end for
10: end for
11: for each arco(u, v) en R[G] do
12:     if d[v] > d[u] + w(u, v) then
13:         return FALSE
14:     end if
15: end for
16: return TRUE
```

El tiempo de ejecición es O(EV).

Outline



Lista de Arcos

(u,v)
(u,x)
(u,y)
(v,u)
(x,v)
(x,y)
(y,v)
(y,z)
(z,u)
(z,x)

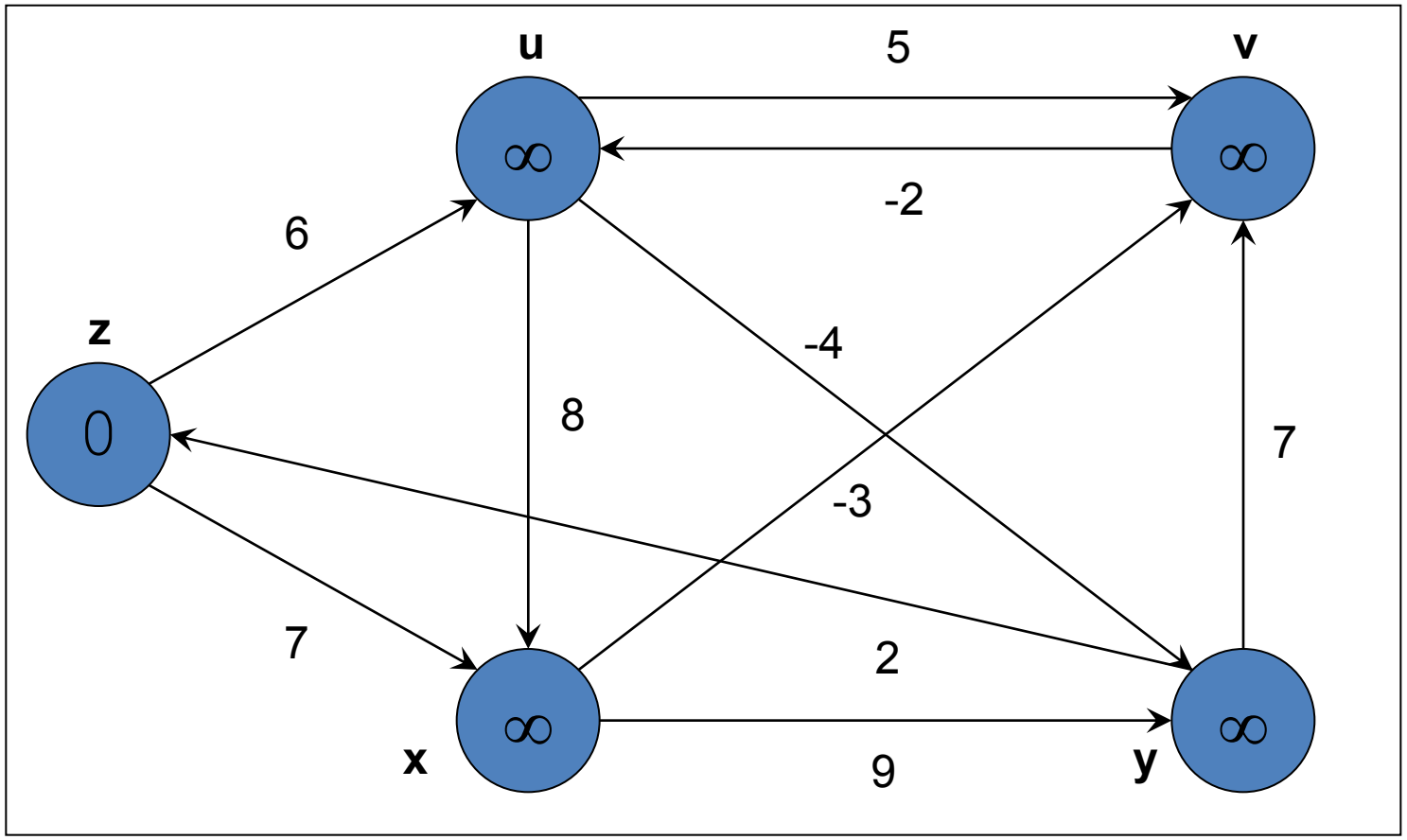
V [ ] = { u v x y z }

d [ ] = { _ _ _ _ _ }

Π [ ] = { _ _ _ _ _ }

Paso 0.0

Encontrar el camino más corto del Vértice z a cada uno de los otros Vértices.



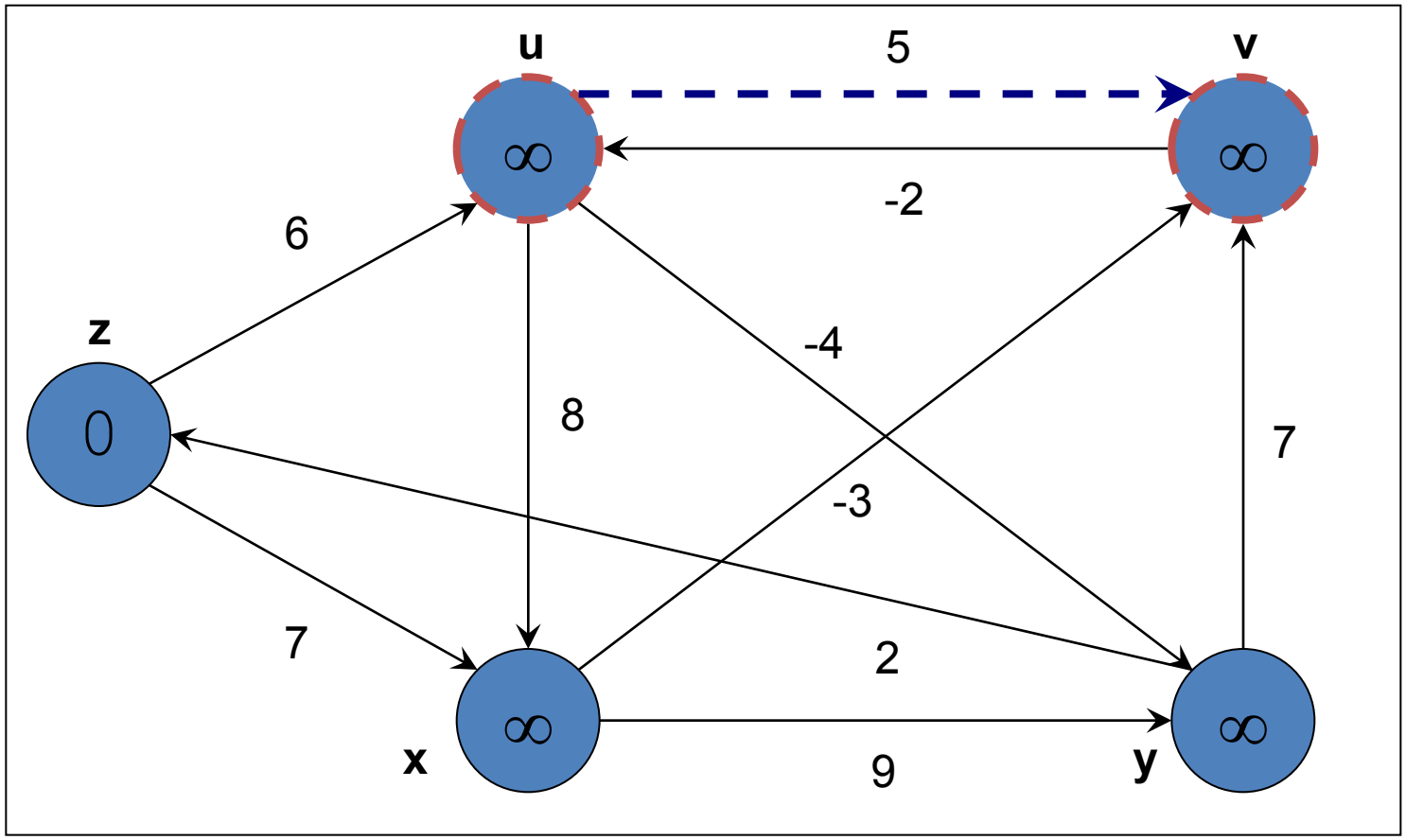
Lista de Arcos

(u,v)
(u,x)
(u,y)
(v,u)
(x,v)
(x,y)
(y,v)
(y,z)
(z,u)
(z,x)

V	**[**	**]**	**=**	**{**	**u**	**v**	**x**	**y**	**z**	**}**
d	**[**	**]**	**=**	**{**	∞	∞	∞	∞	**0**	**}**
Π	**[**	**]**	**=**	**{**						**}**

Paso 0.1

Inicializar los vectores d y Π.



Lista de Arcos

(u,v)←
(u,x)
(u,y)
(v,u)
(x,v)
(x,y)
(y,v)
(y,z)
(z,u)
(z,x)

V	**[**	**]**	**=**	**{**	**u**	**v**	**x**	**y**	**z**	**}**
d	**[**	**]**	**=**	**{**	∞	∞	∞	∞	**0**	**}**
Π	**[**	**]**	**=**	**{**						**}**

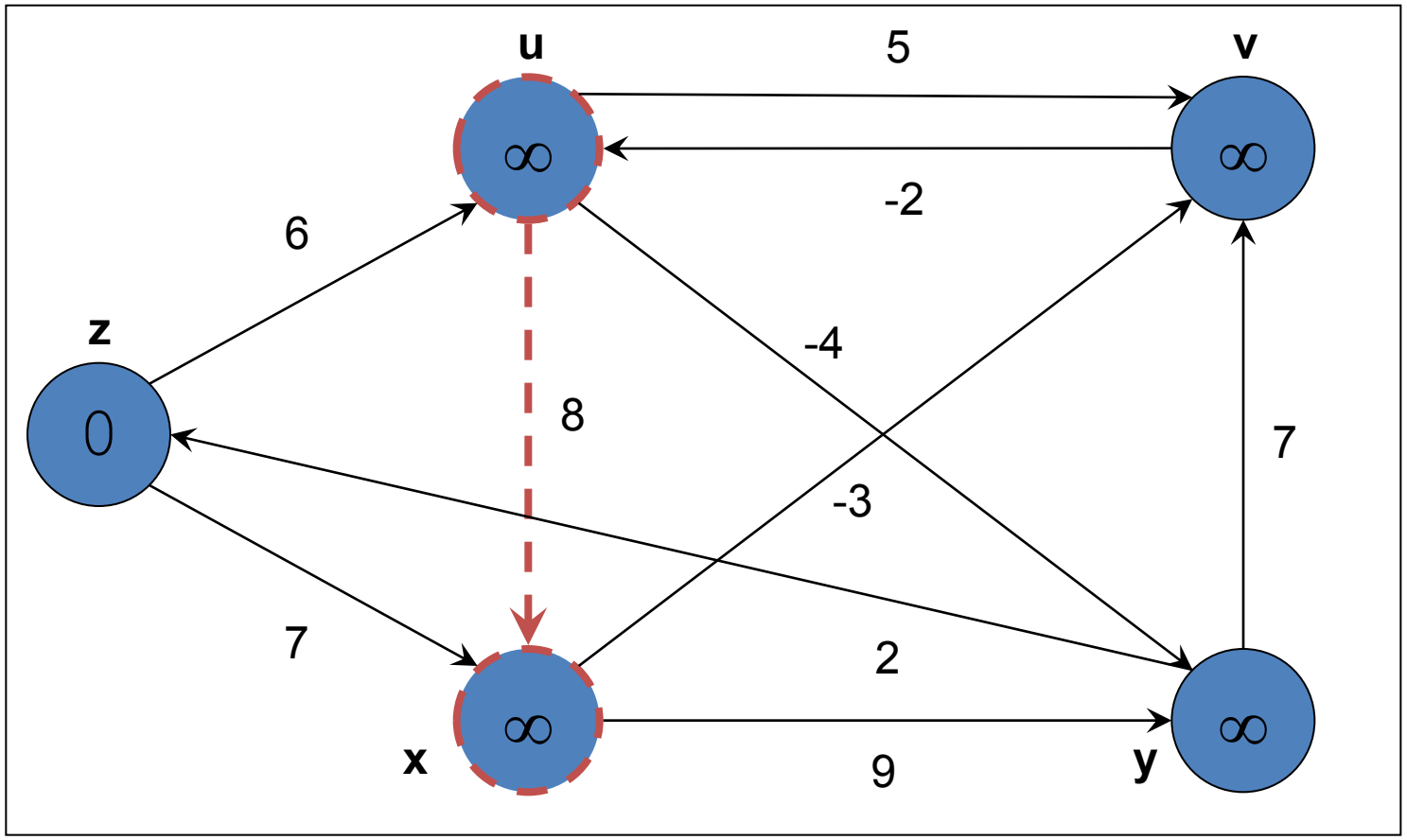
Paso 1.1

Aplicar Relax al Arco (u,v)

Pregunta: ¿ d[v] > d[u] + w(u , v) ?

Respuesta: NO

Proceso: No se hace nada.



Lista de Arcos

(u,v)
(u,x) ←
(u,y)
(v,u)
(x,v)
(x,y)
(y,v)
(y,z)
(z,u)
(z,x)

V	**[**	**]**	**=**	**{**	**u**	**v**	**x**	**y**	**z**	**}**
d	**[**	**]**	**=**	**{**	∞	∞	∞	∞	**0**	**}**
Π	**[**	**]**	**=**	**{**						**}**

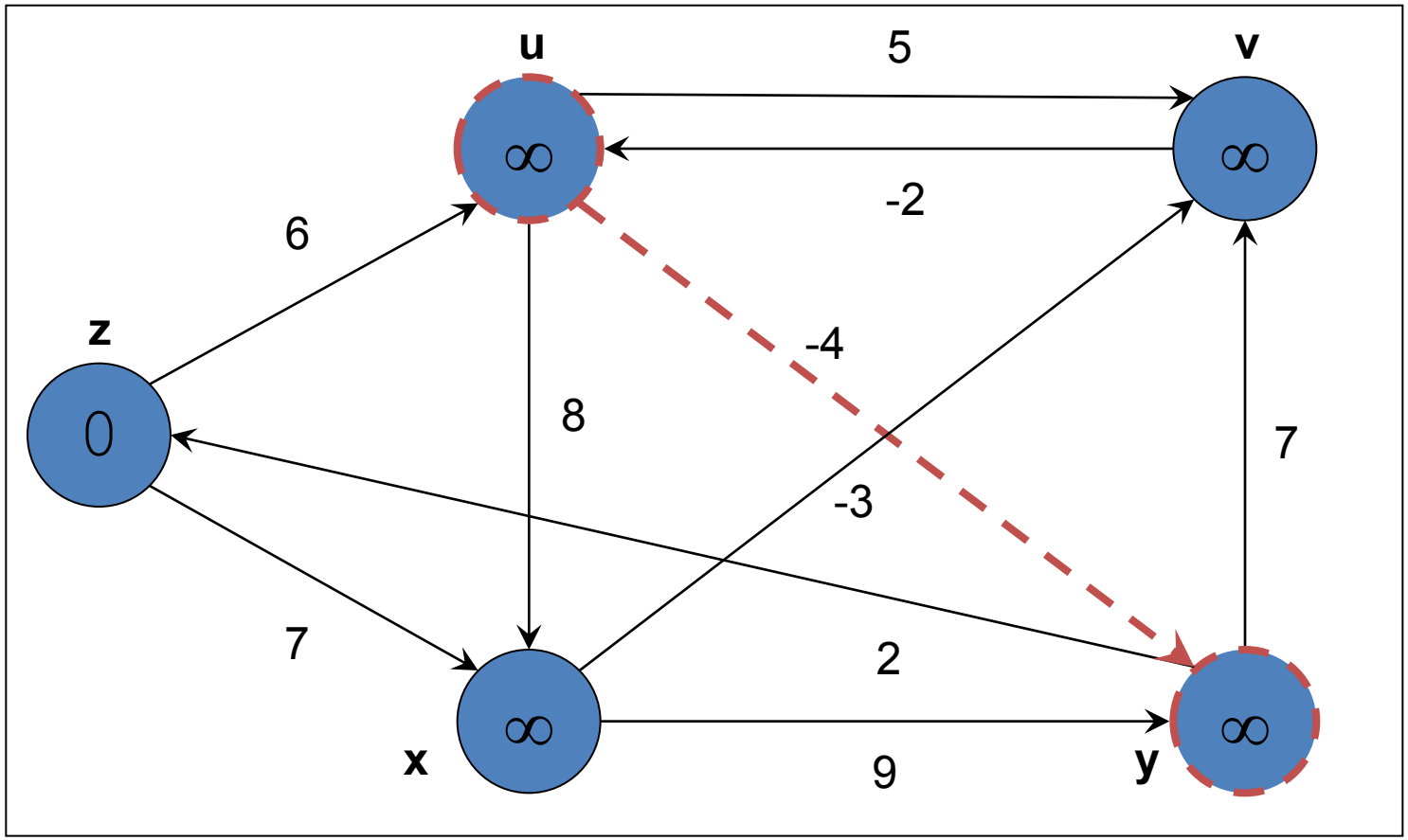
Paso 1.2

Aplicar Relax al Arco (u,x)

Pregunta: ¿ d[x] > d[u] + w(u , x) ?

Respuesta: NO

Proceso: No se hace nada.



Lista de Arcos

(u,v)
(u,x)
(u,y) ←
(v,u)
(x,v)
(x,y)
(y,v)
(y,z)
(z,u)
(z,x)

V	**[**	**]**	**=**	**{**	**u**	**v**	**x**	**y**	**z**	**}**
d	**[**	**]**	**=**	**{**	**∞**	**∞**	**∞**	**∞**	**0**	**}**
Π	**[**	**]**	**=**	**{**						**}**

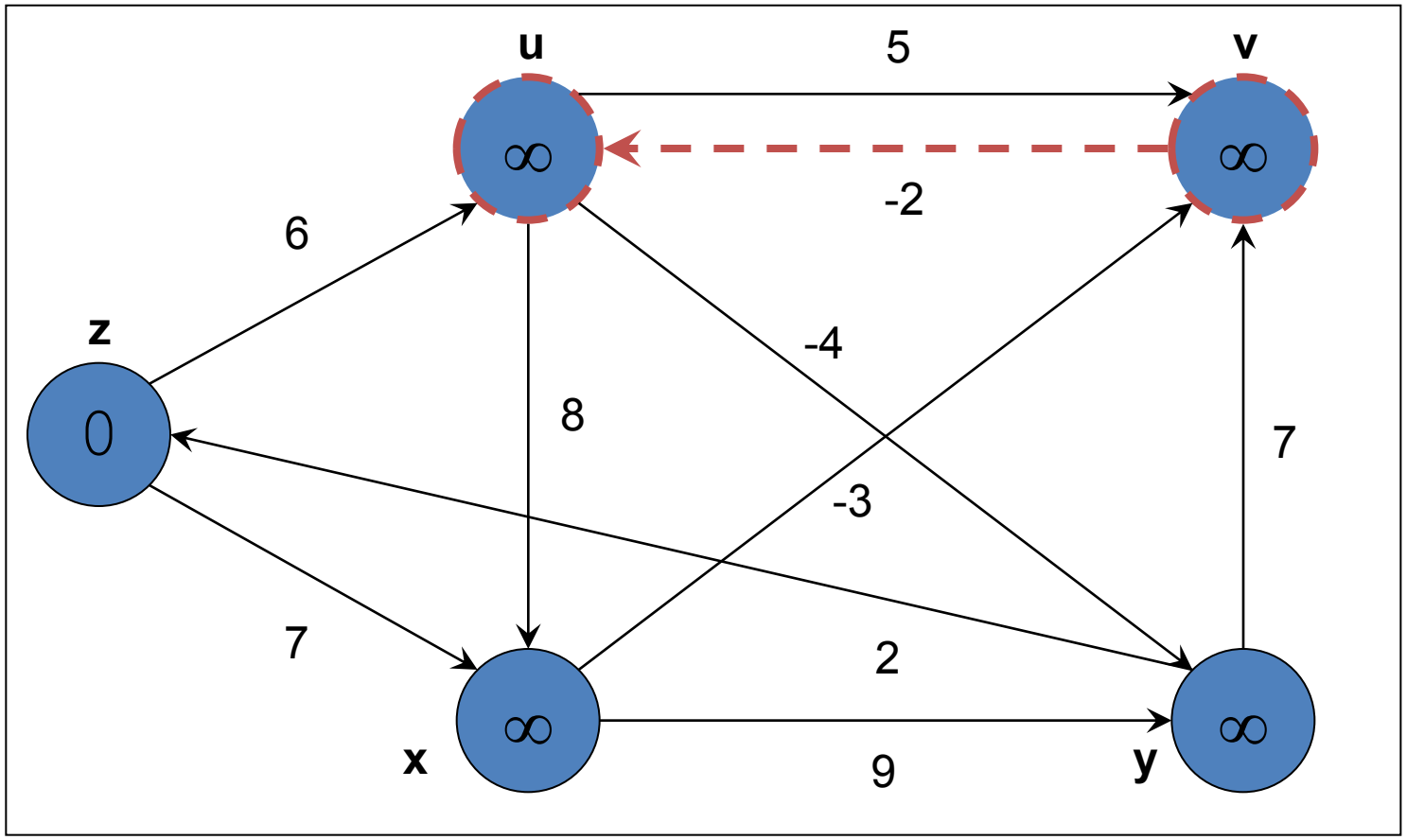
Paso 1.3

Aplicar Relax al Arco (u,y)

Pregunta: ¿ d[y] > d[u] + w(u , y) ?

Respuesta: NO

Proceso: No se hace nada.



Lista de Arcos

(u,v)
(u,x)
(u,y)
(v,u) ←
(x,v)
(x,y)
(y,v)
(y,z)
(z,u)
(z,x)

V	**[**	**]**	**=**	**{**	**u**	**v**	**x**	**y**	**z**	**}**
d	**[**	**]**	**=**	**{**	∞	∞	∞	∞	**0**	**}**
Π	**[**	**]**	**=**	**{**						**}**

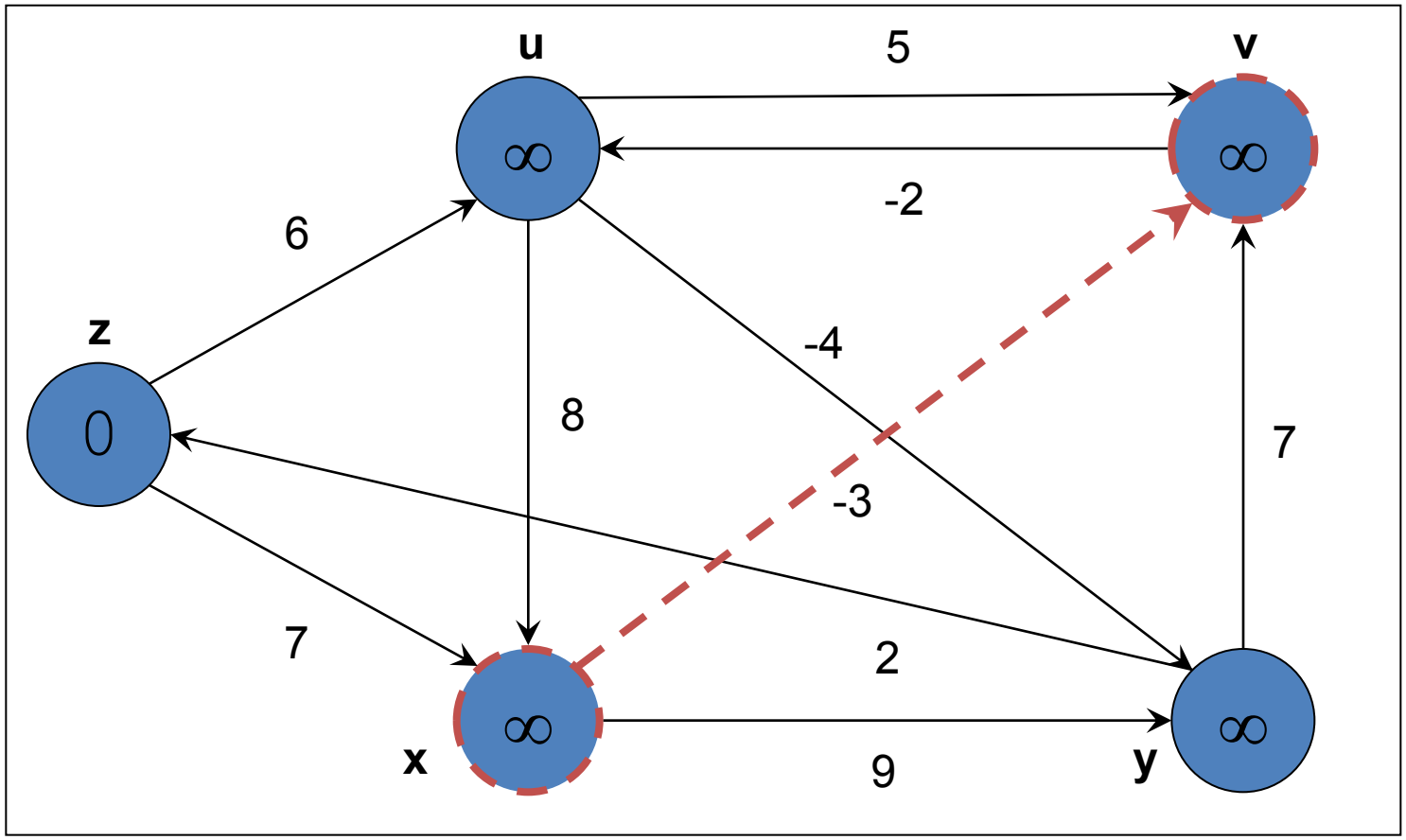
Paso 1.4

Aplicar Relax al Arco (v,u)

Pregunta: ¿ d[u] > d[v] + w(v , u) ?

Respuesta: NO

Proceso: No se hace nada.



Lista de Arcos

(u,v)
(u,x)
(u,y)
(v,u)
(x,v) ←
(x,y)
(y,v)
(y,z)
(z,u)
(z,x)

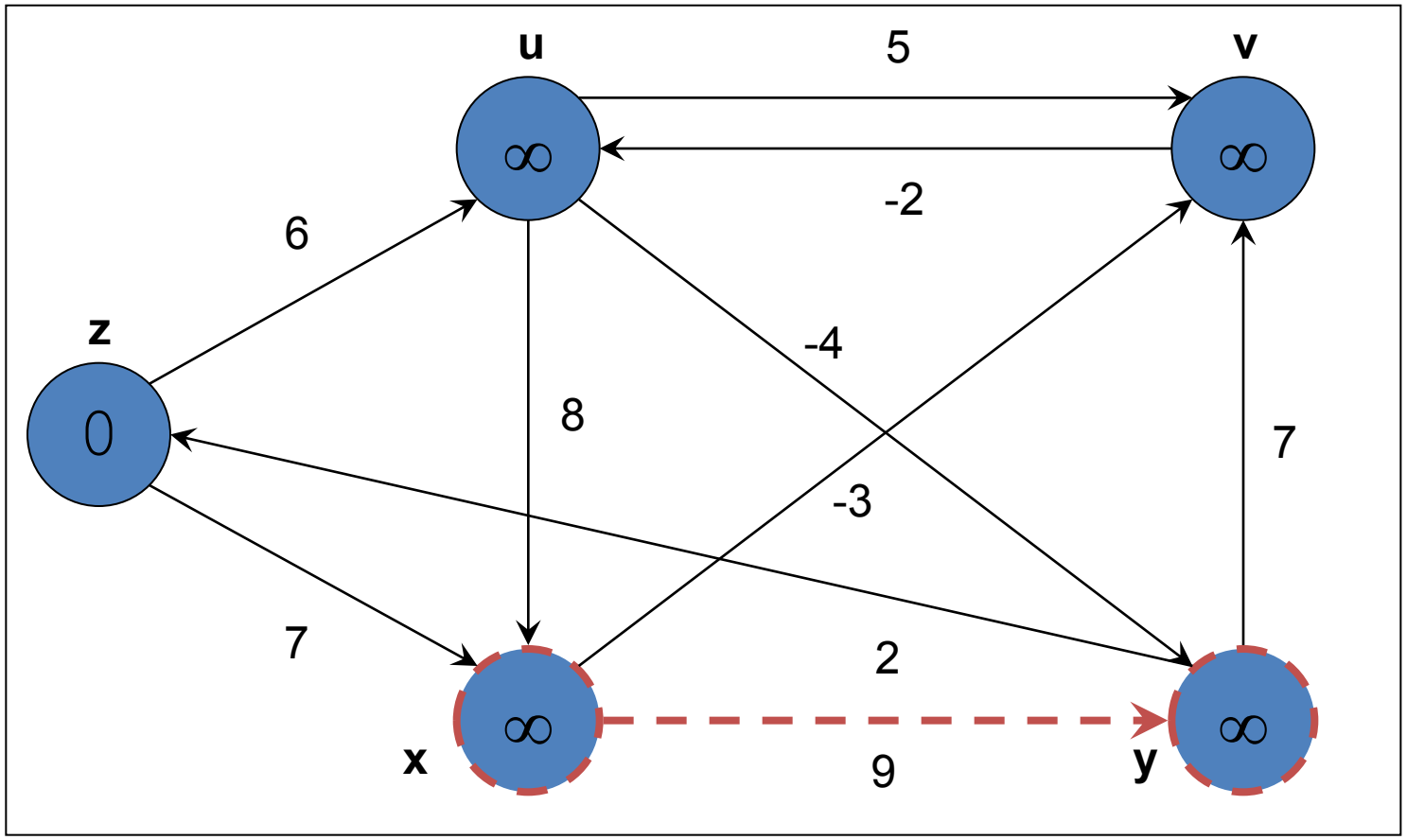
V	[ ]	=	{	u	v	x	y	z	}
d	[ ]	=	{	∞	∞	∞	∞	0	}
Π	[ ]	=	{						}

Paso 1.5 Aplicar Relax al Arco (x,v)

Pregunta: ¿ d[v] > d[x] + w(x , v) ?

Respuesta: NO

Proceso: No se hace nada.



Lista de Arcos

(u,v)
(u,x)
(u,y)
(v,u)
(x,v)
(x,y) ←
(y,v)
(y,z)
(z,u)
(z,x)

V	**[**	**]**	**=**	**{**	**u**	**v**	**x**	**y**	**z**	**}**
d	**[**	**]**	**=**	**{**	∞	∞	∞	∞	**0**	**}**
Π	**[**	**]**	**=**	**{**						**}**

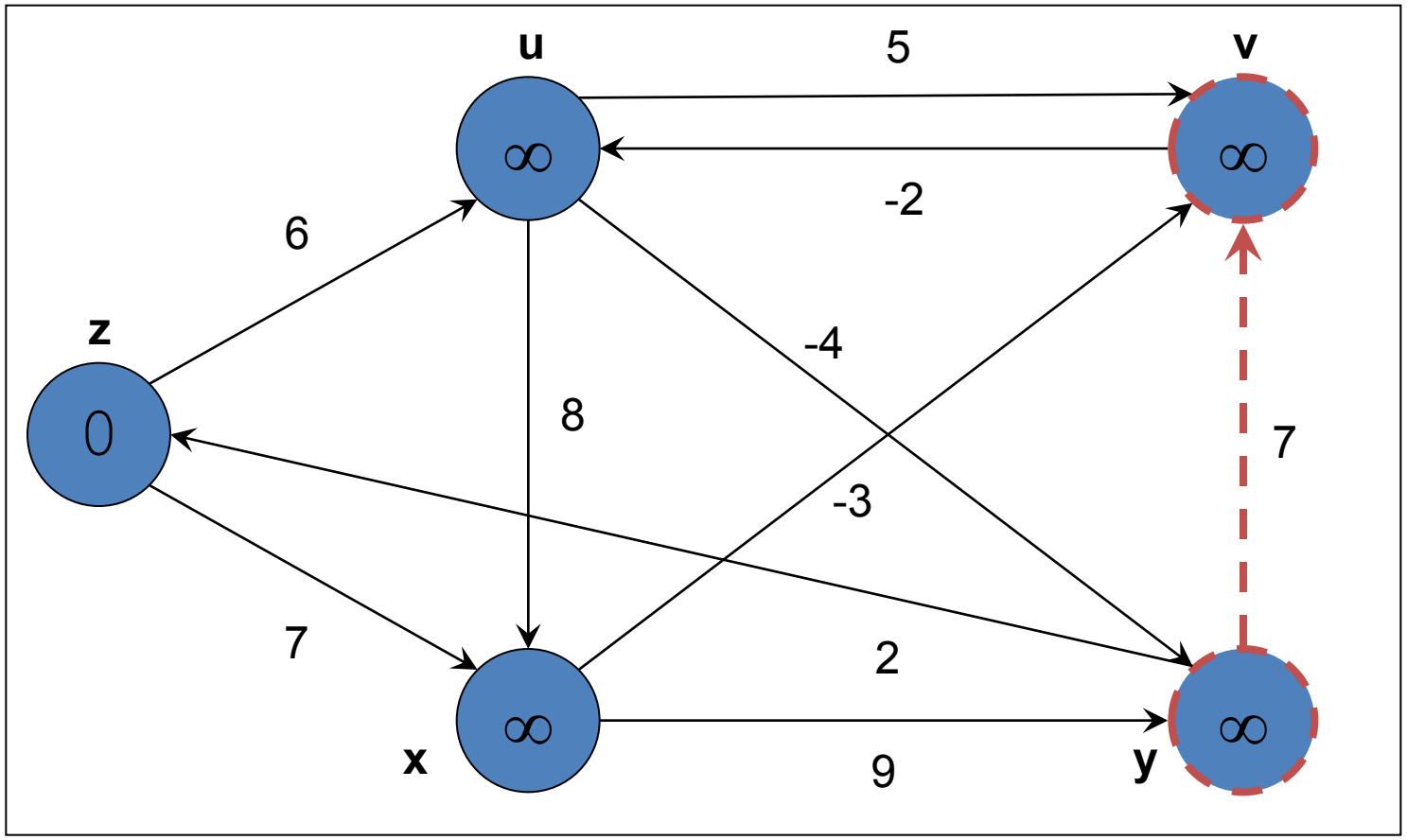
Paso 1.6

Aplicar Relax al Arco (x,y)

Pregunta: ¿ d[y] > d[x] + w(x , y) ?

Respuesta: NO

Proceso: No se hace nada.



Lista de Arcos

(u,v)
(u,x)
(u,y)
(v,u)
(x,v)
(x,y)
(y,v) ←
(y,z)
(z,u)
(z,x)

V	**[**	**]**	**=**	**{**	**u**	**v**	**x**	**y**	**z**	**}**
d	**[**	**]**	**=**	**{**	∞	∞	∞	∞	**0**	**}**
Π	**[**	**]**	**=**	**{**						**}**

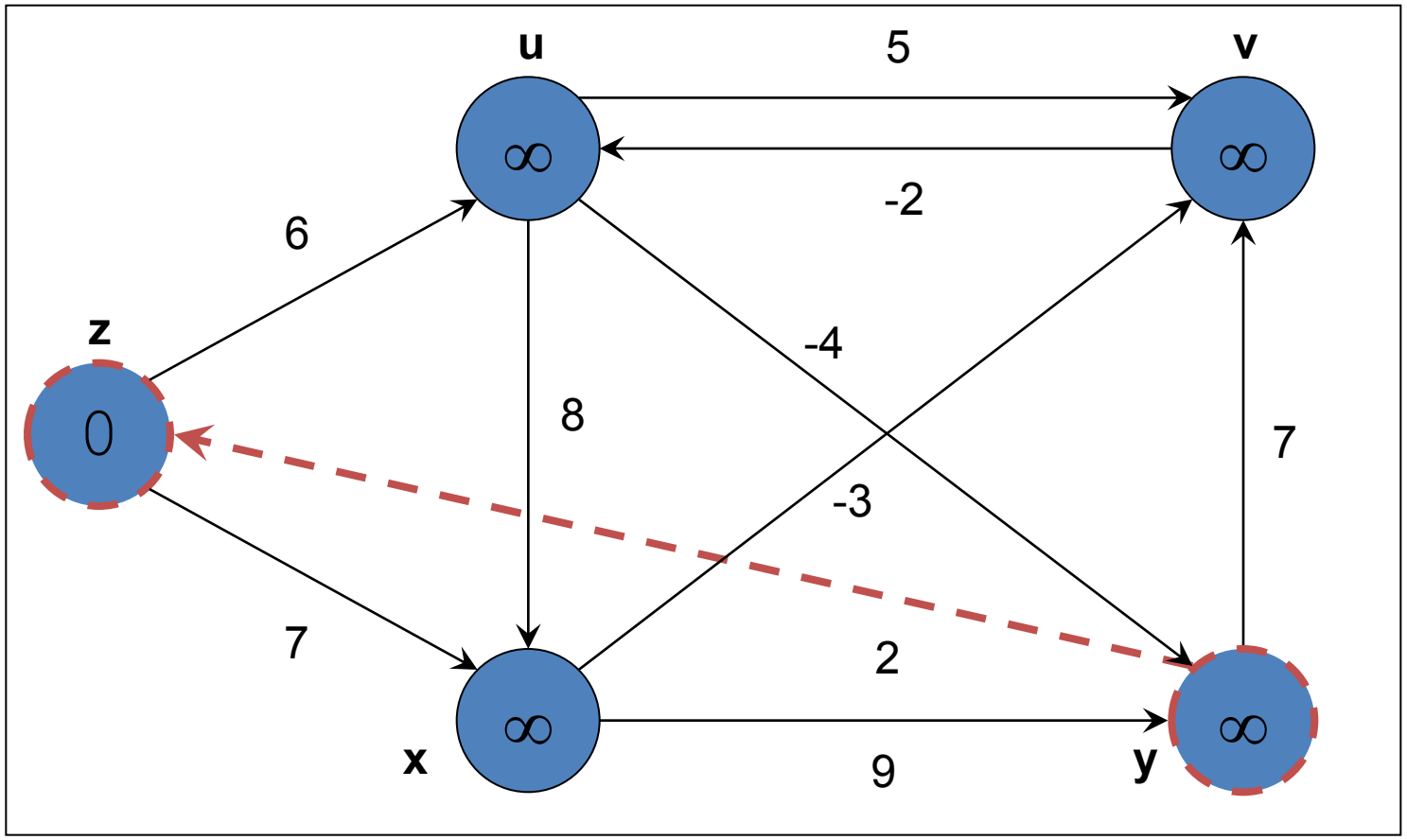
Paso 1.7

Aplicar Relax al Arco (y,v)

Pregunta: ¿ d[v] > d[y] + w(y , v) ?

Respuesta: NO

Proceso: No se hace nada.



Lista de Arcos

(u,v)
(u,x)
(u,y)
(v,u)
(x,v)
(x,y)
(y,v)
(y,z) ←
(z,u)
(z,x)

V	**[**	**]**	**=**	**{**	**u**	**v**	**x**	**y**	**z**	**}**
d	**[**	**]**	**=**	**{**	∞	∞	∞	∞	**0**	**}**
Π	**[**	**]**	**=**	**{**						**}**

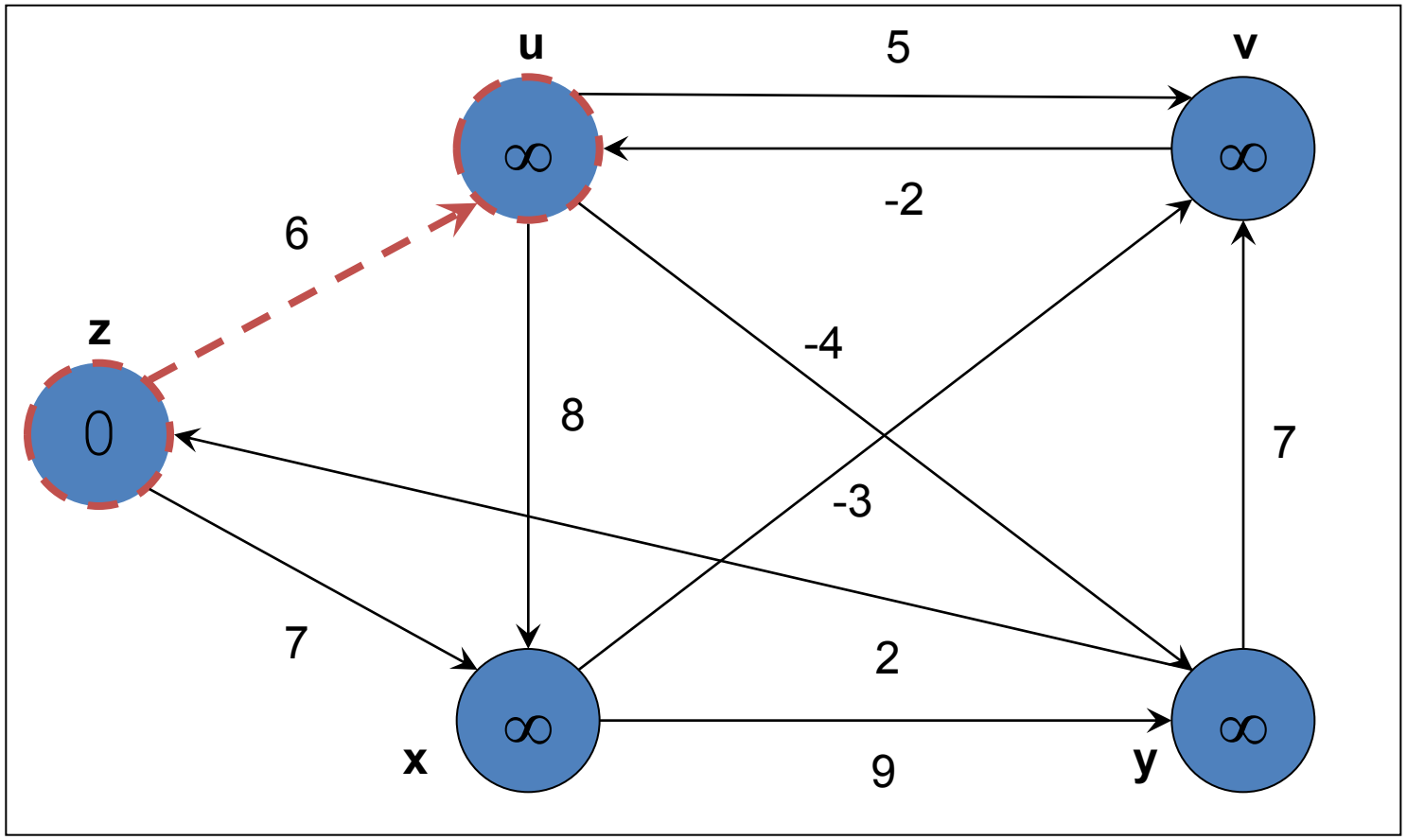
Paso 1.8

Aplicar Relax al Arco (y,v)

Pregunta: ¿ d[z] > d[y] + w(y , z) ?

Respuesta: NO

Proceso: No se hace nada.



Lista de Arcos

(u,v)
(u,x)
(u,y)
(v,u)
(x,v)
(x,y)
(y,v)
(y,z)
(z,u) ←
(z,x)

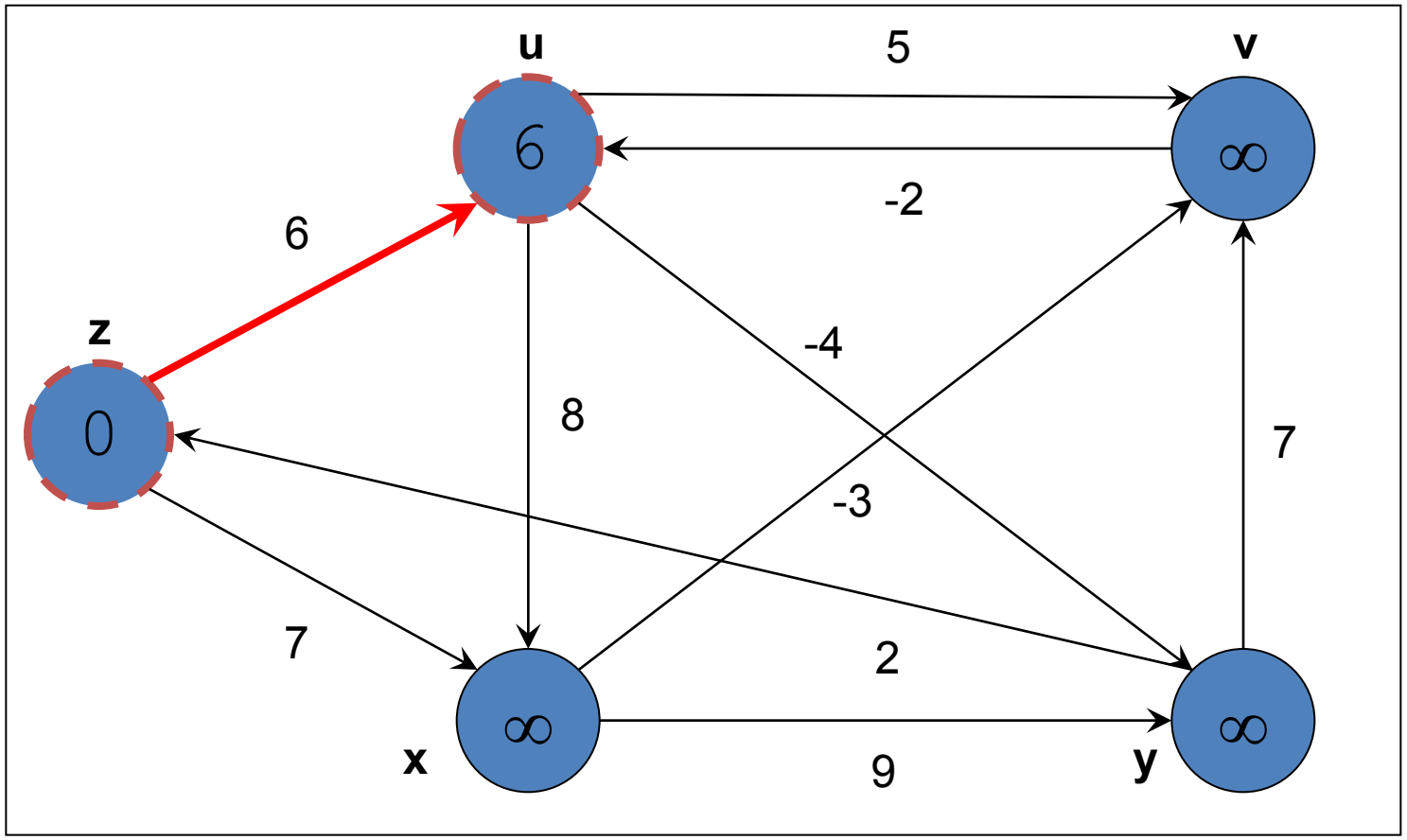
V	**[**	**]**	**=**	**{**	**u**	**v**	**x**	**y**	**z**	**}**
d	**[**	**]**	**=**	**{**	∞	∞	∞	∞	**0**	**}**
Π	**[**	**]**	**=**	**{**						**}**

Paso 1.9 Aplicar Relax al Arco (z,u)

Pregunta: ¿ d[u] > d[z] + w(z , u) ?

Respuesta: SI

Proceso: d[u] = d[z] + w(z, u) y Π[u] = z



Lista de Arcos

(u,v)
(u,x)
(u,y)
(v,u)
(x,v)
(x,y)
(y,v)
(y,z)
(z,u) ←
(z,x)

V	**[**	**]**	**=**	**{**	**u**	**v**	**x**	**y**	**z**	**}**
d	**[**	**]**	**=**	**{**	6	∞	∞	∞	**0**	**}**
Π	**[**	**]**	**=**	**{**	**z**					**}**

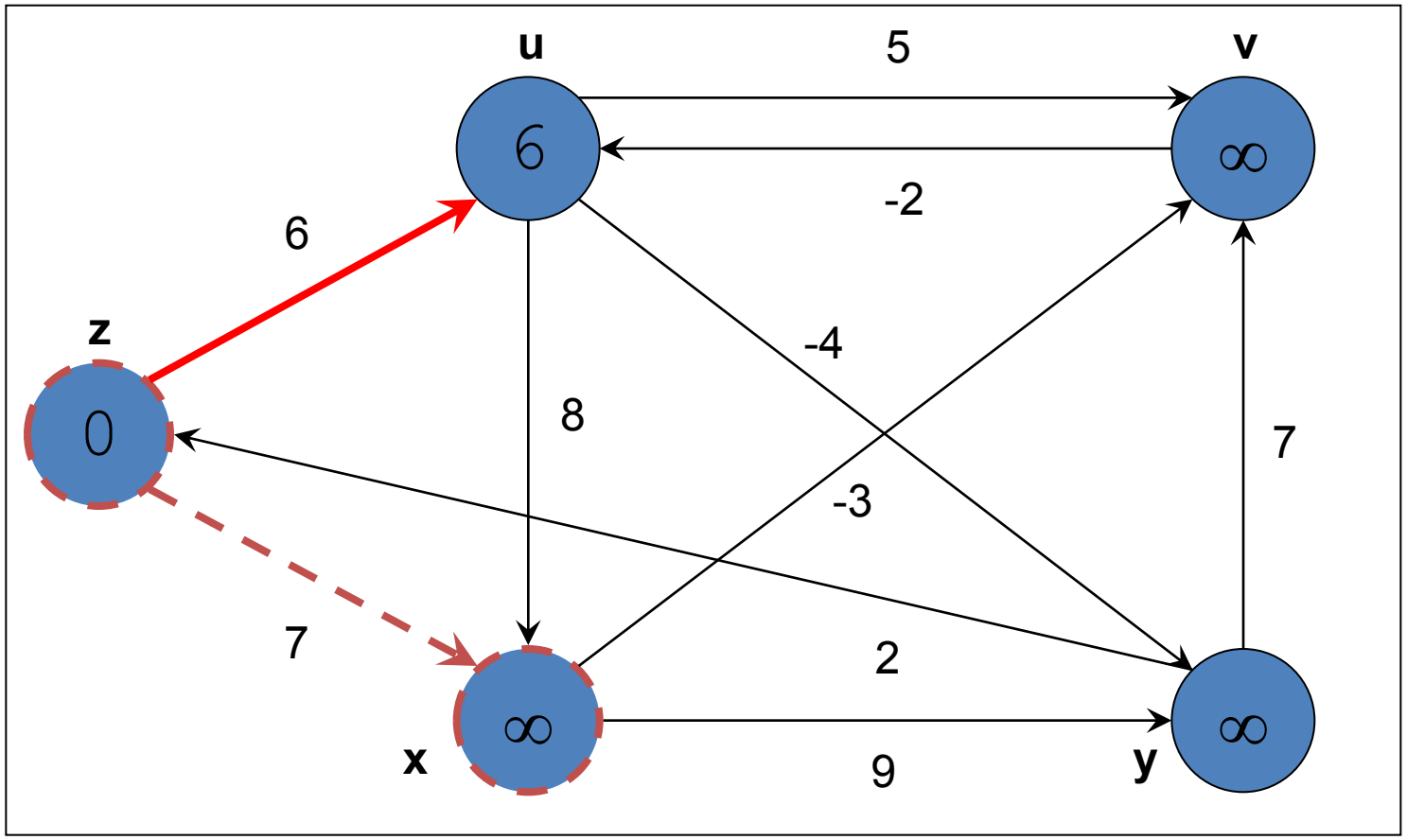
Paso 1.9

Aplicar Relax al Arco (z,u)

Pregunta: ¿ d[u] > d[z] + w(z , u) ?

Respuesta: SI

Proceso: d[u] = d[z] + w(z, u) y Π[u] = z



Lista de Arcos

(u,v)
(u,x)
(u,y)
(v,u)
(x,v)
(x,y)
(y,v)
(y,z)
(z,u)
(z,x) ←

V	**[**	**]**	**=**	**{**	**u**	**v**	**x**	**y**	**z**	**}**
d	**[**	**]**	**=**	**{**	6	∞	∞	∞	**0**	**}**
Π	**[**	**]**	**=**	**{**	**z**					**}**

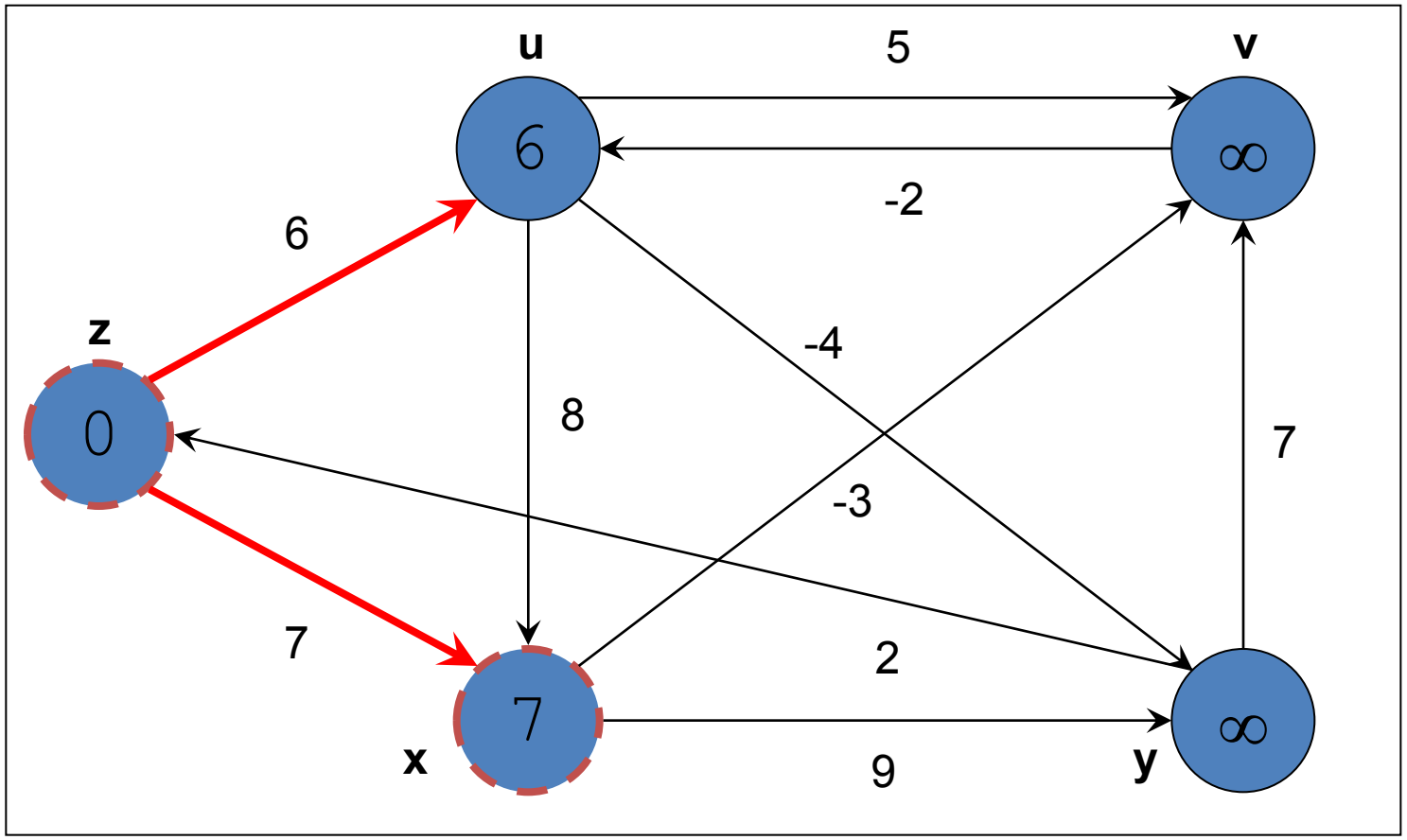
Paso 1.10

Aplicar Relax al Arco (z,x)

Pregunta: ¿ d[x] > d[z] + w(z , x) ?

Respuesta: SI

Proceso: d[x] = d[z] + w(z, x) y Π[x] = z



Lista de Arcos

(u,v)
(u,x)
(u,y)
(v,u)
(x,v)
(x,y)
(y,v)
(y,z)
(z,u)
(z,x) ←

V	**[**	**]**	**=**	**{**	**u**	**v**	**x**	**y**	**z**	**}**
d	**[**	**]**	**=**	**{**	6	∞	7	∞	0	**}**
Π	**[**	**]**	**=**	**{**	**z**		**z**			**}**

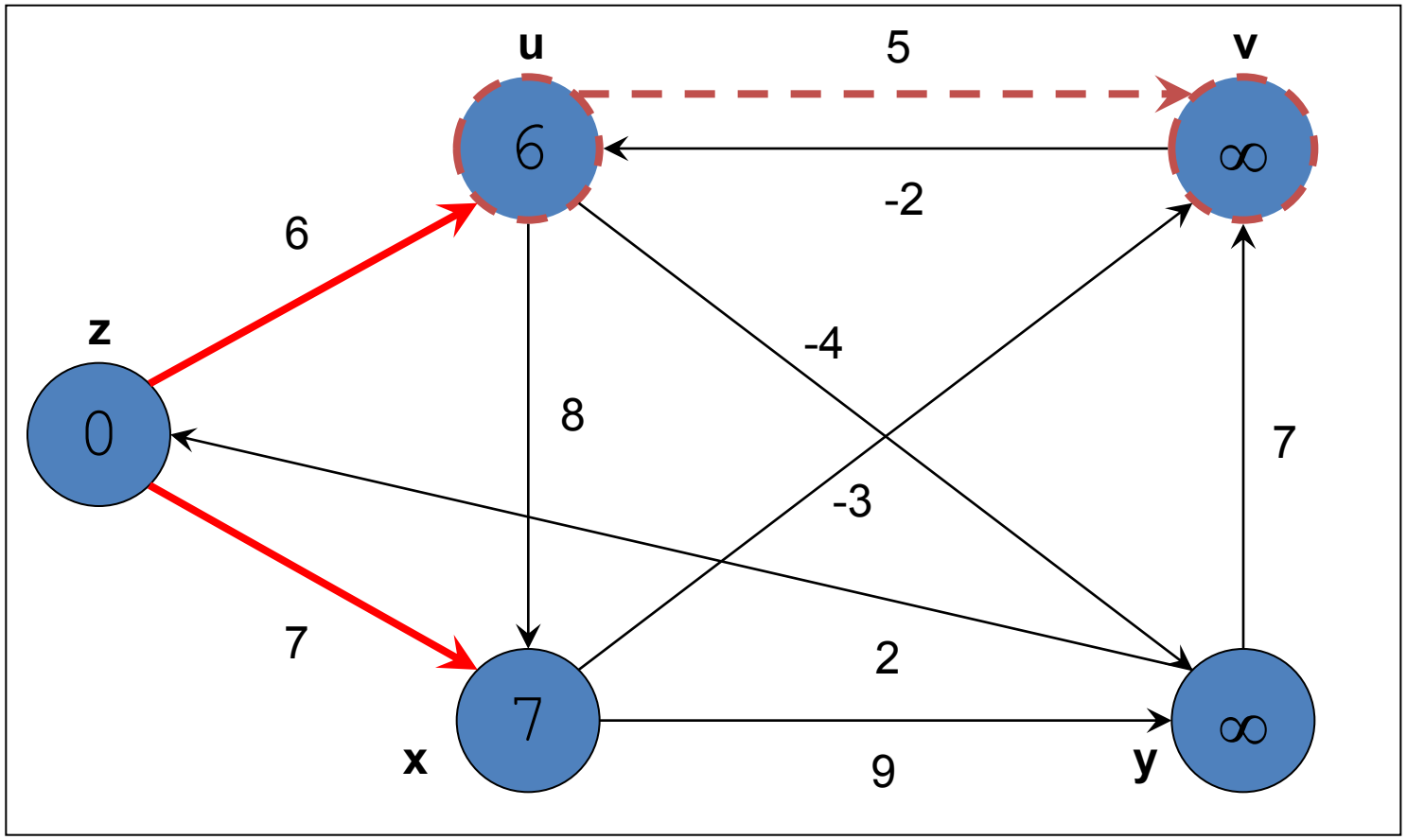
Paso 1.10

Aplicar Relax al Arco (z,x)

Pregunta: ¿ d[x] > d[z] + w(z , x) ?

Respuesta: SI

Proceso: d[x] = d[z] + w(z, x) y Π[x] = z



Lista de Arcos

(u,v) ←
(u,x)
(u,y)
(v,u)
(x,v)
(x,y)
(y,v)
(y,z)
(z,u)
(z,x)

V	[ ]	=	{	u	v	x	y	z	}
d	[ ]	=	{	6	∞	7	∞	0	}
Π	[ ]	=	{	z		z			}

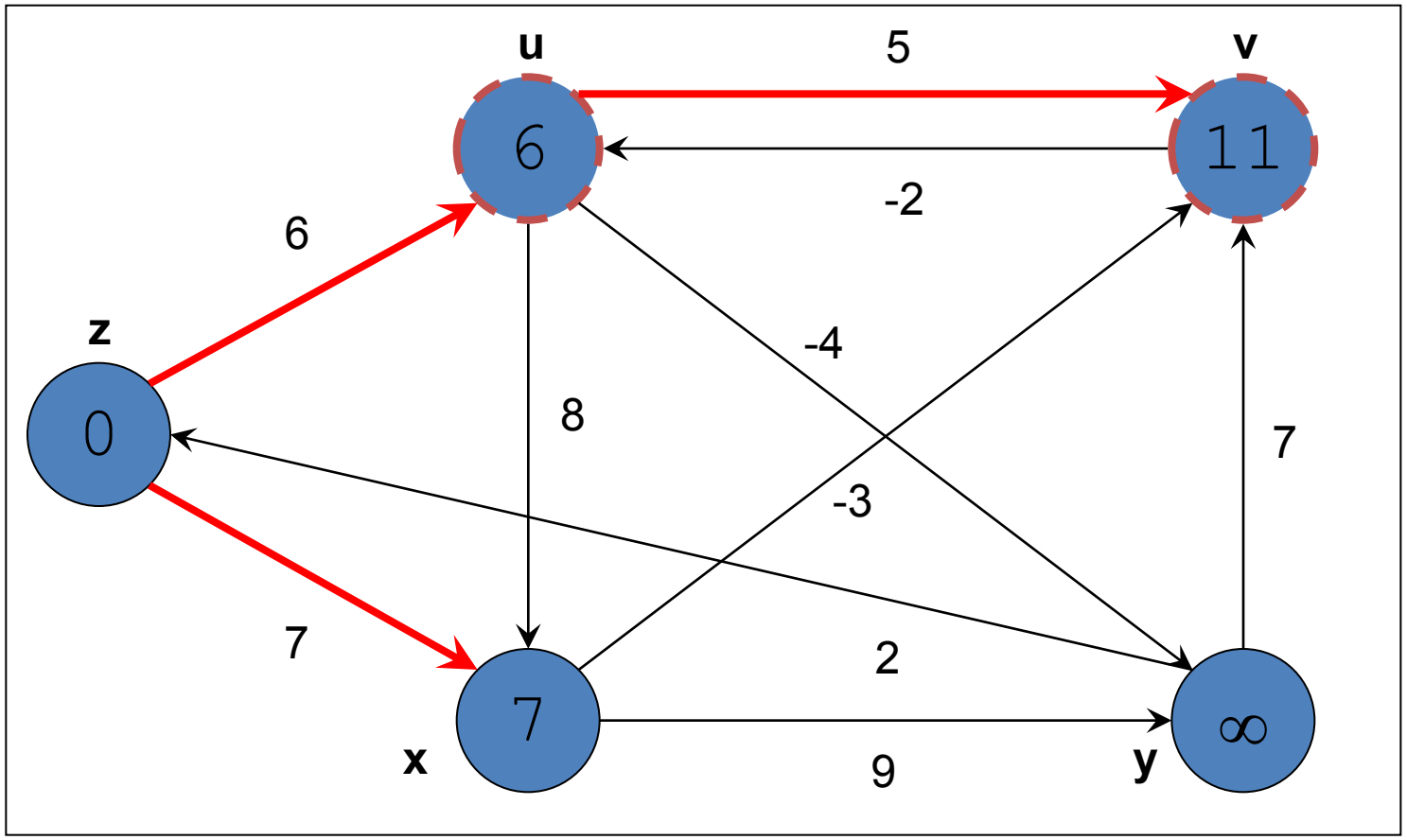
Paso 2.1

Aplicar Relax al Arco (u,v)

Pregunta: ¿ d[v] > d[u] + w(u , v) ?

Respuesta: SI

Proceso: d[v] = d[u] + w(u, v) y Π[v] = u



Lista de Arcos

(u,v) ←
(u,x)
(u,y)
(v,u)
(x,v)
(x,y)
(y,v)
(y,z)
(z,u)
(z,x)

V	**[**	**]**	**=**	**{**	**u**	**v**	**x**	**y**	**z**	**}**
d	**[**	**]**	**=**	**{**	6	11	**7**	∞	**0**	**}**
Π	**[**	**]**	**=**	**{**	**z**	**u**	**z**			**}**

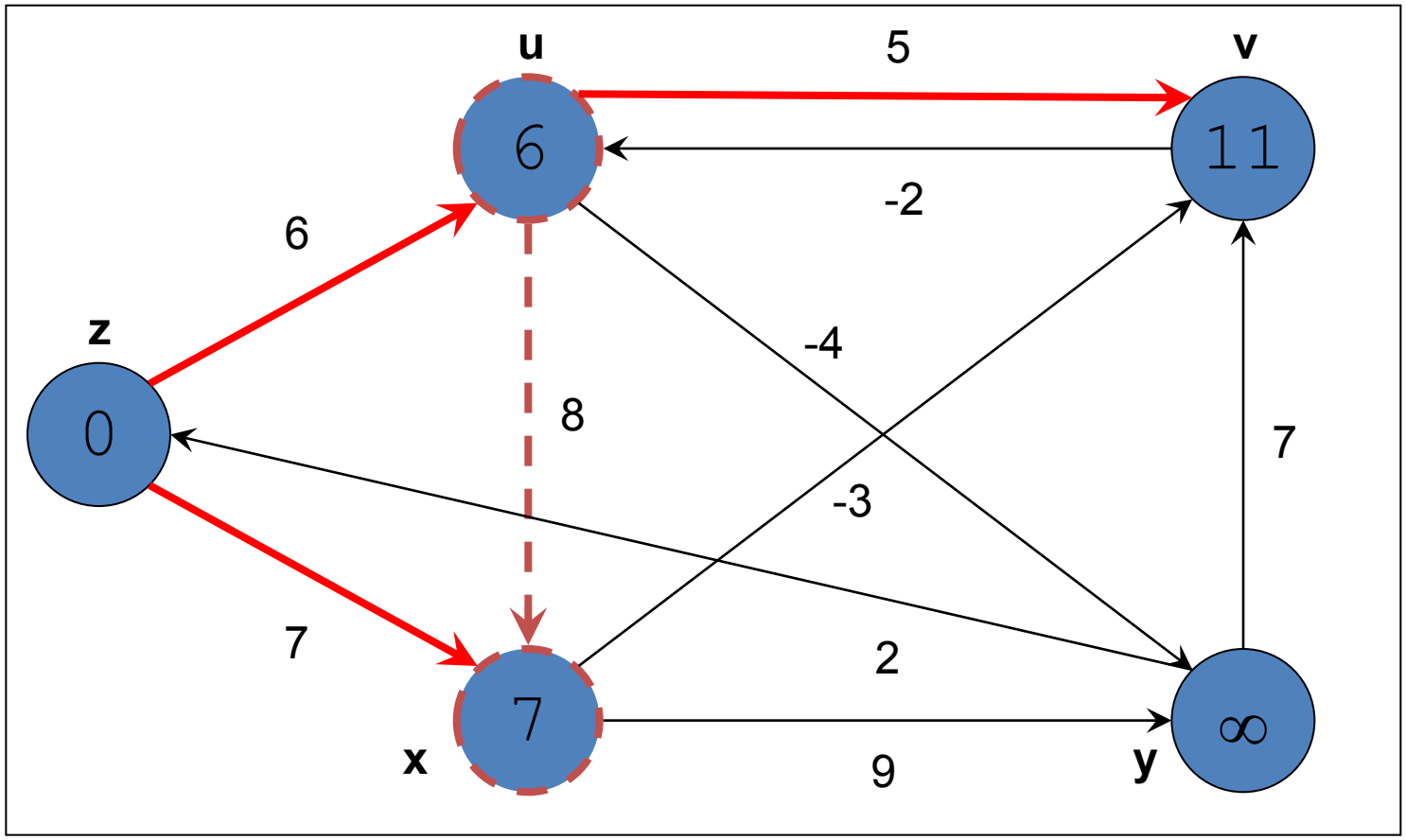
Paso 2.1

Aplicar Relax al Arco (u,v)

Pregunta: ¿ d[v] > d[u] + w(u , v) ?

Respuesta: SI

Proceso: d[v] = d[u] + w(u, v) y Π[v] = u



Lista de Arcos

(u,v)
(u,x) ←
(u,y)
(v,u)
(x,v)
(x,y)
(y,v)
(y,z)
(z,u)
(z,x)

V	[ ]	=	{	u	v	x	y	z	}
d	[ ]	=	{	6	11	7	∞	0	}
Π	[ ]	=	{	z	u	z			}

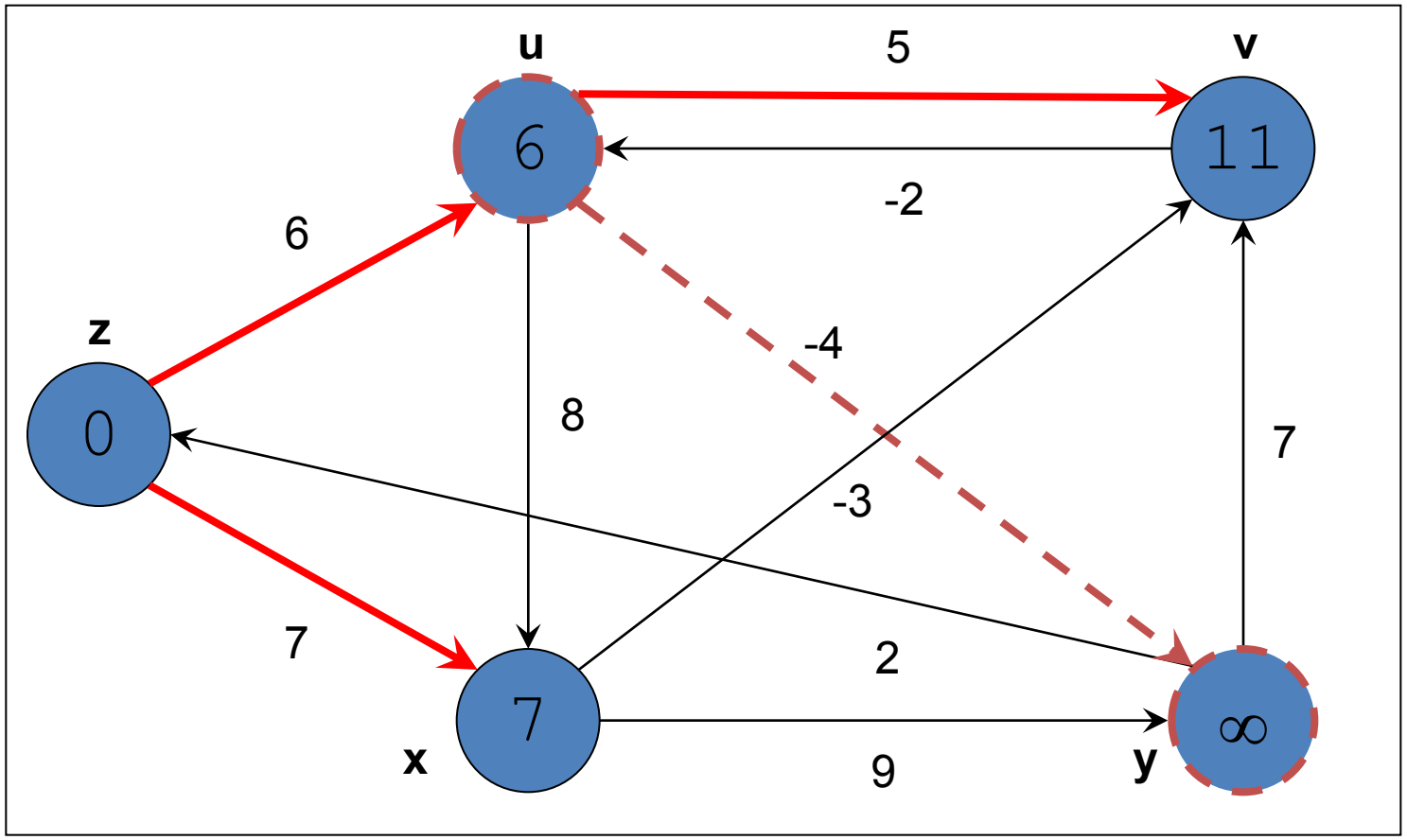
Paso 2.2

Aplicar Relax al Arco (u,x)

Pregunta: ¿ d[x] > d[u] + w(u , x) ?

Respuesta: NO

Proceso: No se hace nada.



Lista de Arcos

(u,v)
(u,x)
(u,y) ←
(v,u)
(x,v)
(x,y)
(y,v)
(y,z)
(z,u)
(z,x)

V	**[**	**]**	**=**	**{**	**u**	**v**	**x**	**y**	**z**	**}**
d	**[**	**]**	**=**	**{**	6	11	7	∞	0	**}**
Π	**[**	**]**	**=**	**{**	**z**	**u**	**z**			**}**

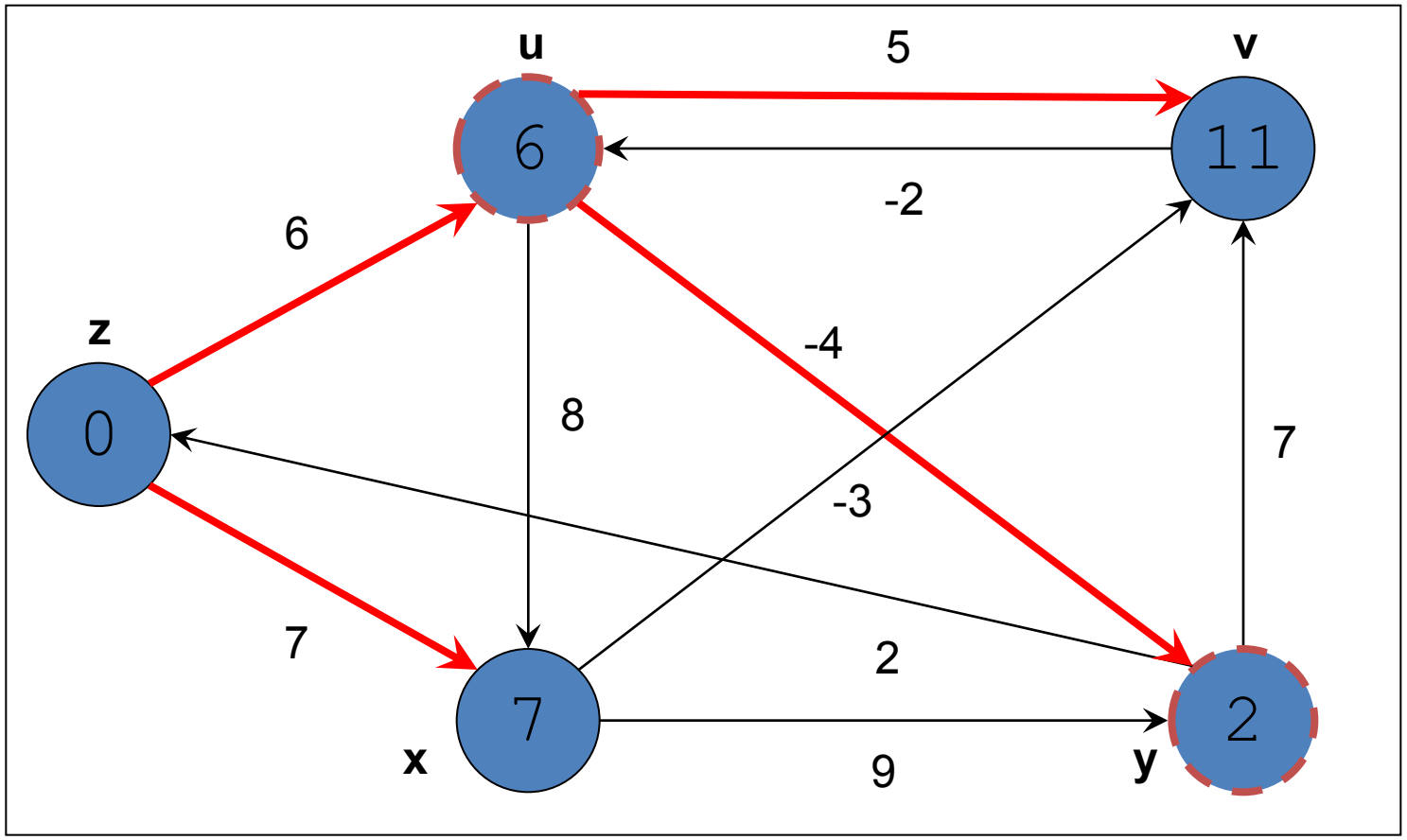
Paso 2.3

Aplicar Relax al Arco (u,y)

Pregunta: ¿ d[y] > d[u] + w(u , y) ?

Respuesta: SI

Proceso: d[y] = d[u] + w(u, y) y Π[y] = u



Lista de Arcos

(u,v)
(u,x)
(u,y) ←
(v,u)
(x,v)
(x,y)
(y,v)
(y,z)
(z,u)
(z,x)

V	**[**	**]**	**=**	**{**	**u**	**v**	**x**	**y**	**z**	**}**
d	**[**	**]**	**=**	**{**	6	11	7	2	0	**}**
Π	**[**	**]**	**=**	**{**	**z**	**u**	**z**	**u**		**}**

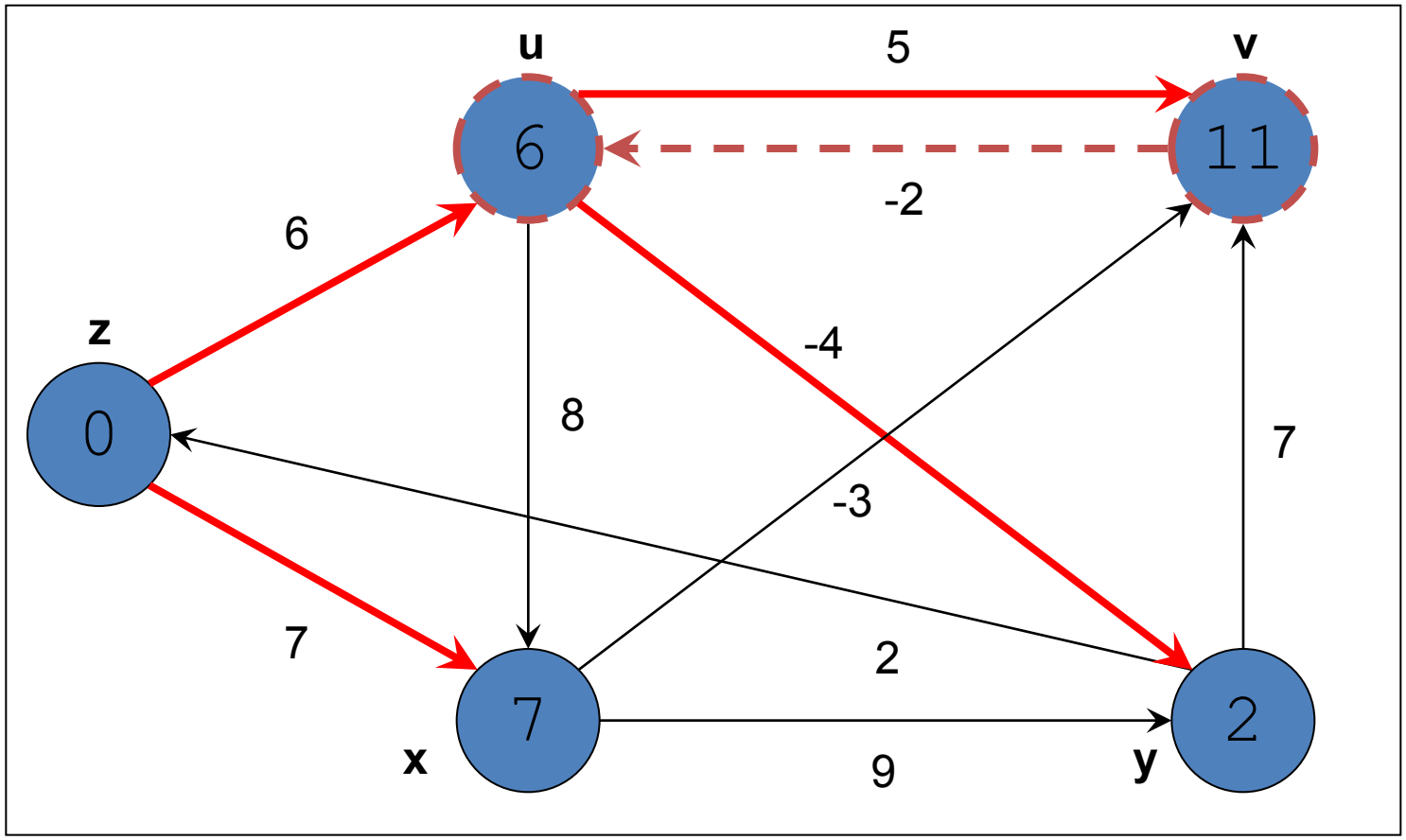
Paso 2.3

Aplicar Relax al Arco (u,y)

Pregunta: ¿ d[y] > d[u] + w(u , y) ?

Respuesta: SI

Proceso: d[y] = d[u] + w(u, y) y Π[y] = u



Lista de Arcos

(u,v)
(u,x)
(u,y)
(v,u) ←
(x,v)
(x,y)
(y,v)
(y,z)
(z,u)
(z,x)

V	**[**	**]**	**=**	**{**	**u**	**v**	**x**	**y**	**z**	**}**
d	**[**	**]**	**=**	**{**	6	11	7	2	0	**}**
Π	**[**	**]**	**=**	**{**	**z**	**u**	**z**	**u**		**}**

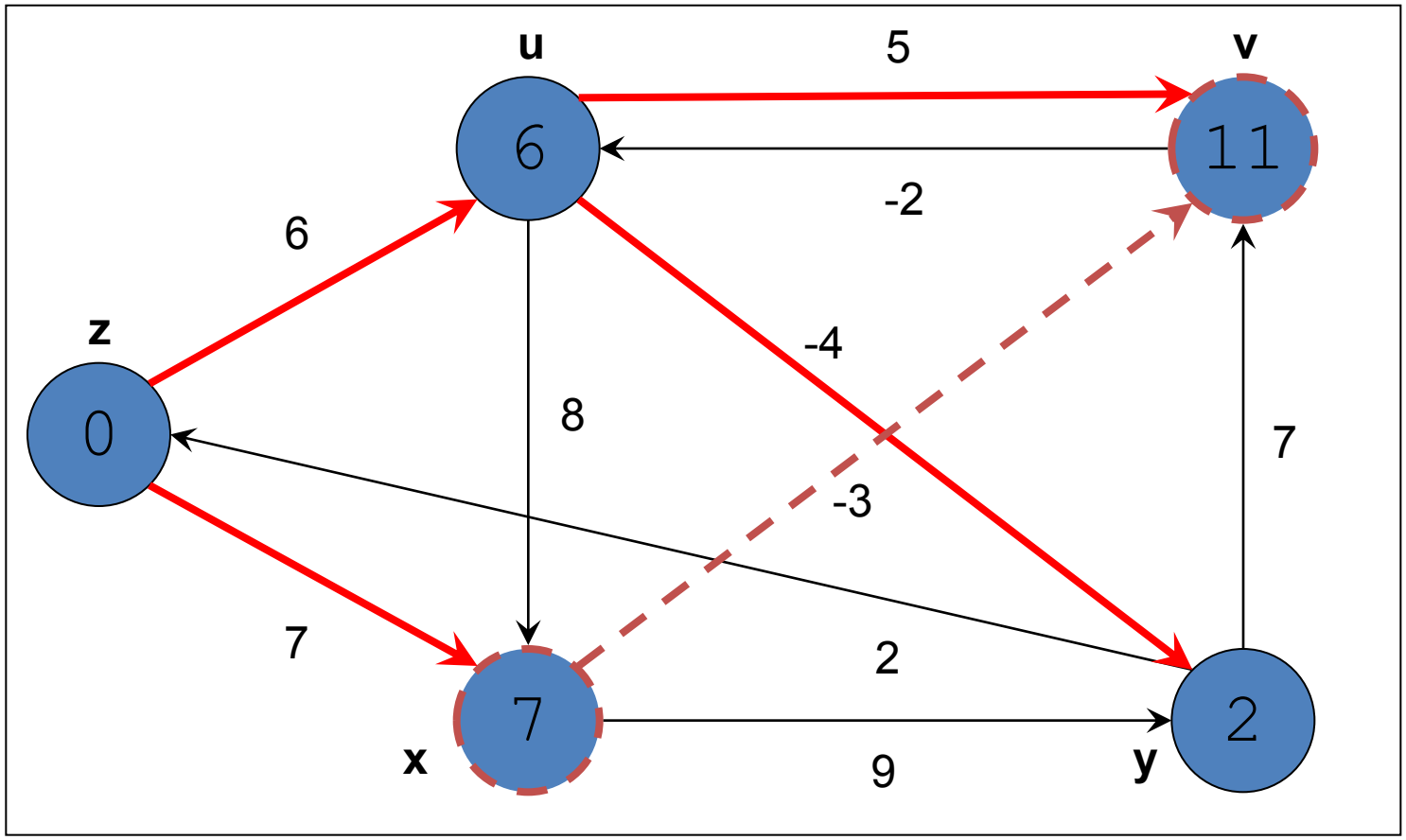
Paso 2.4

Aplicar Relax al Arco (v,u)

Pregunta: **¿ d[u] > d[v] + w(v , u) ?**

Respuesta: NO

Proceso: **No se hace nada.**



Lista de Arcos

(u,v)
(u,x)
(u,y)
(v,u)
(x,v) ←
(x,y)
(y,v)
(y,z)
(z,u)
(z,x)

V	**[**	**]**	**=**	**{**	**u**	**v**	**x**	**y**	**z**	**}**
d	**[**	**]**	**=**	**{**	6	11	7	2	0	**}**
Π	**[**	**]**	**=**	**{**	**z**	**u**	**z**	**u**		**}**

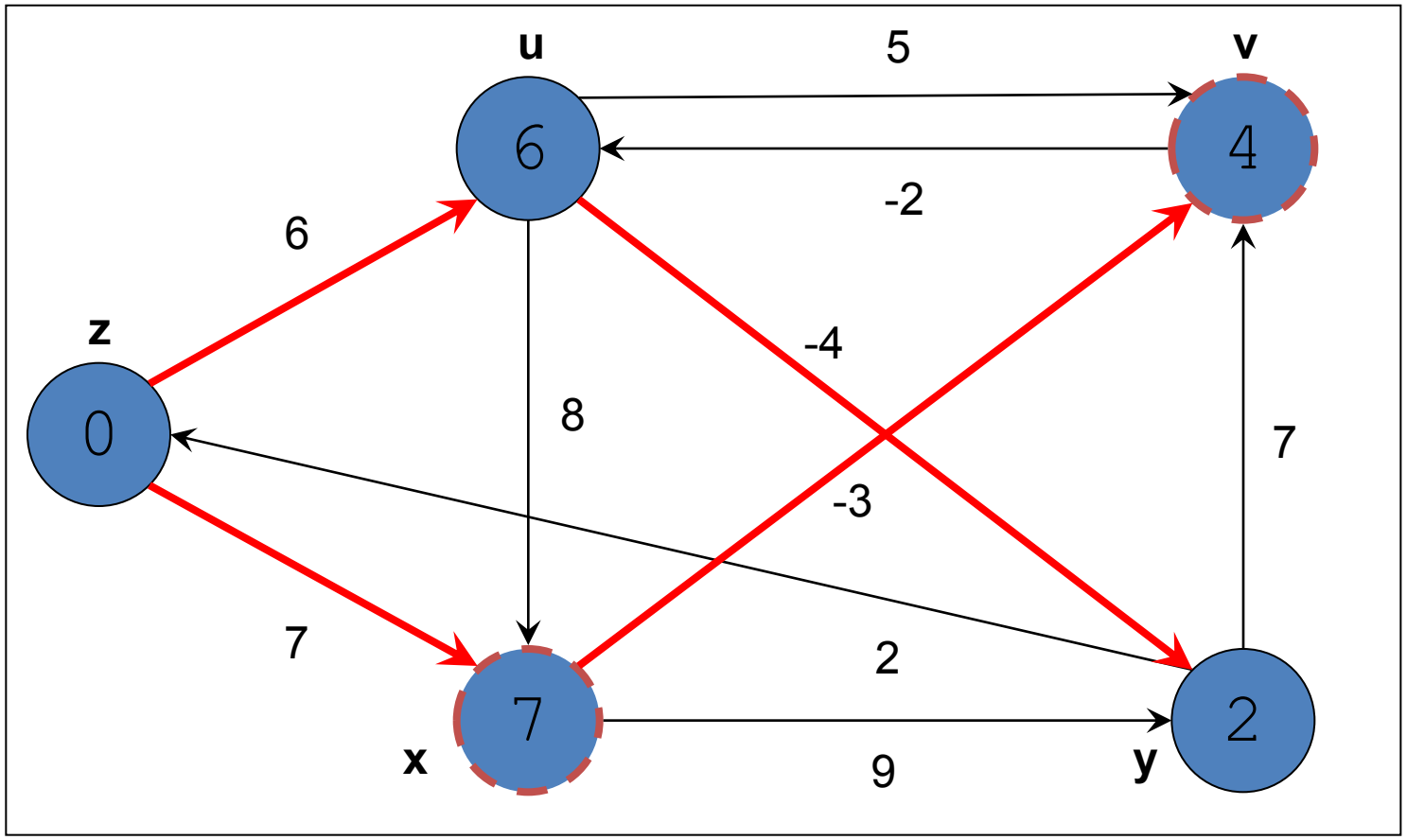
Paso 2.5

Aplicar Relax al Arco (x,v)

Pregunta: ¿ d[v] > d[x] + w(x , v) ?

Respuesta: SI

Proceso: d[y] = d[x] + w(x, v) y Π[y] = x



Lista de Arcos

(u,v)
(u,x)
(u,y)
(v,u)
(x,v) ←
(x,y)
(y,v)
(y,z)
(z,u)
(z,x)

V	**[**	**]**	**=**	**{**	**u**	**v**	**x**	**y**	**z**	**}**
d	**[**	**]**	**=**	**{**	6	4	7	2	0	**}**
Π	**[**	**]**	**=**	**{**	**z**	**x**	**z**	**u**		**}**

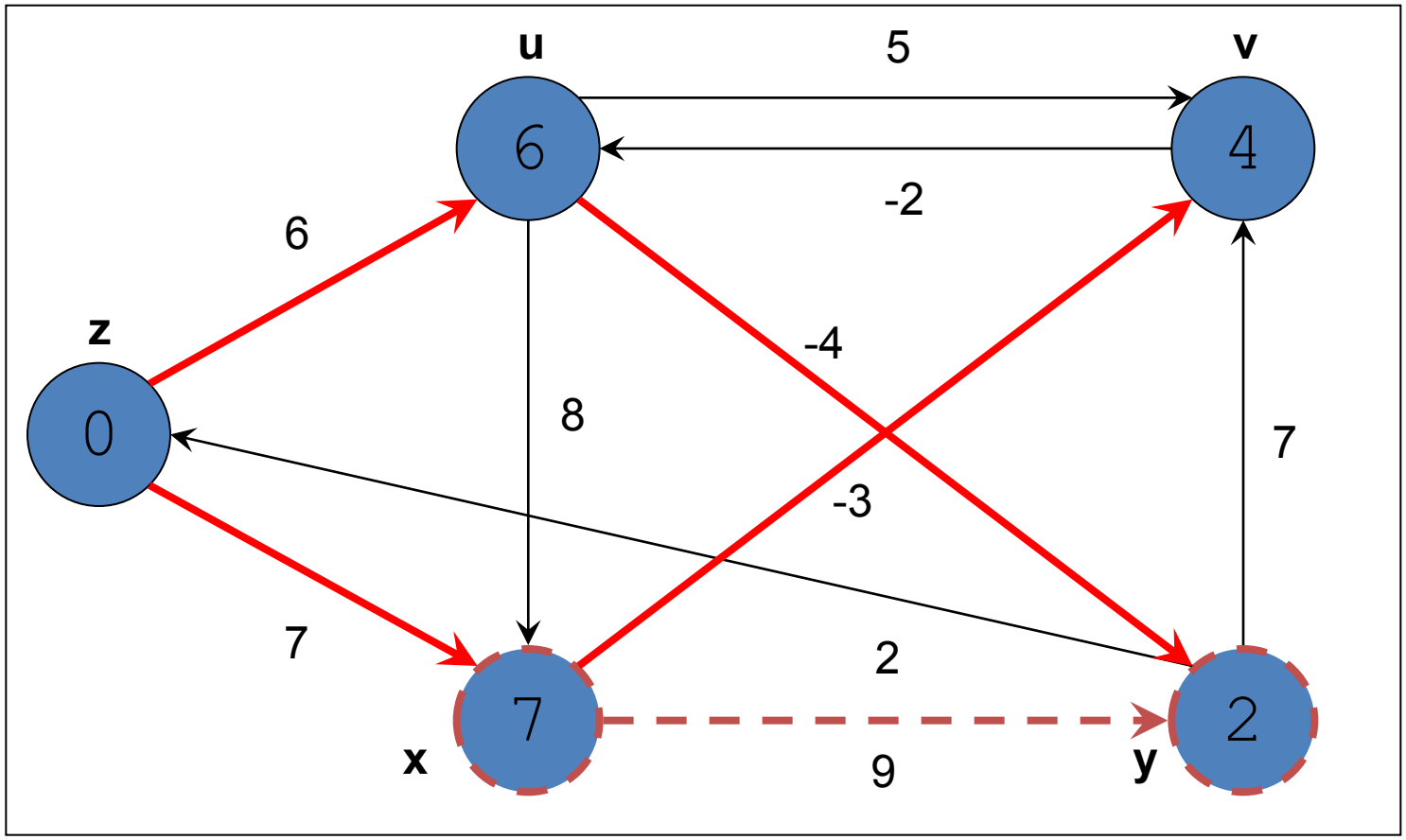
Paso 2.5

Aplicar Relax al Arco (x,v)

Pregunta: ¿ d[v] > d[x] + w(x , v) ?

Respuesta: SI

Proceso: d[y] = d[x] + w(x, v) y Π[y] = x



Lista de Arcos

(u,v)
(u,x)
(u,y)
(v,u)
(x,v)
(x,y) ←
(y,v)
(y,z)
(z,u)
(z,x)

V	**[**	**]**	**=**	**{**	**u**	**v**	**x**	**y**	**z**	**}**
d	**[**	**]**	**=**	**{**	6	4	7	2	0	**}**
Π	**[**	**]**	**=**	**{**	**z**	**x**	**z**	**u**		**}**

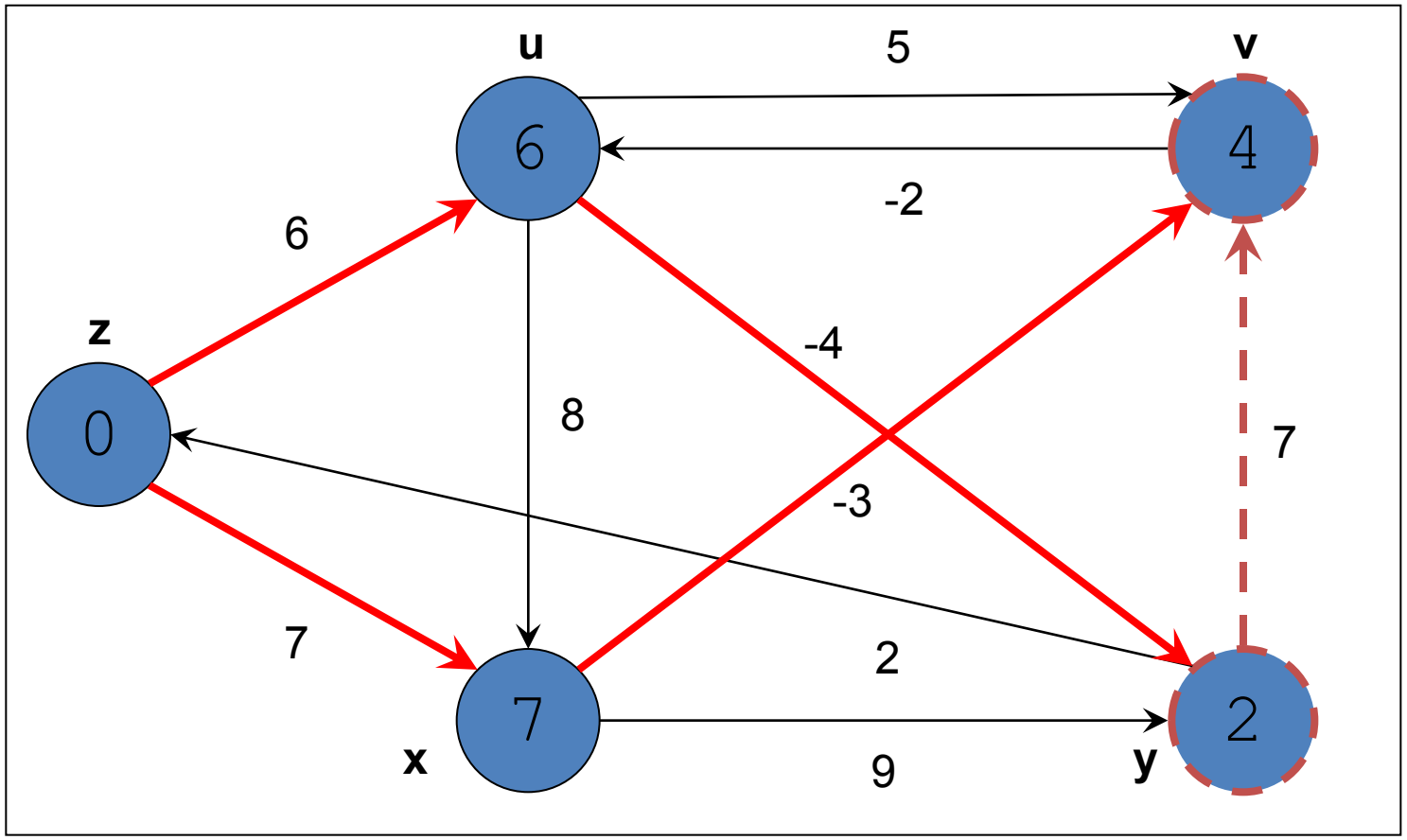
Paso 2.6

Aplicar Relax al Arco (x,y)

Pregunta: **¿ d[y] > d[x] + w(x , y) ?**

Respuesta: NO

Proceso: **No se hace nada.**



Lista de Arcos

(u,v)
(u,x)
(u,y)
(v,u)
(x,v)
(x,y)
(y,v) ←
(y,z)
(z,u)
(z,x)

V	**[**	**]**	**=**	**{**	**u**	**v**	**x**	**y**	**z**	**}**
d	**[**	**]**	**=**	**{**	6	4	7	2	0	**}**
Π	**[**	**]**	**=**	**{**	**z**	**x**	**z**	**u**		**}**

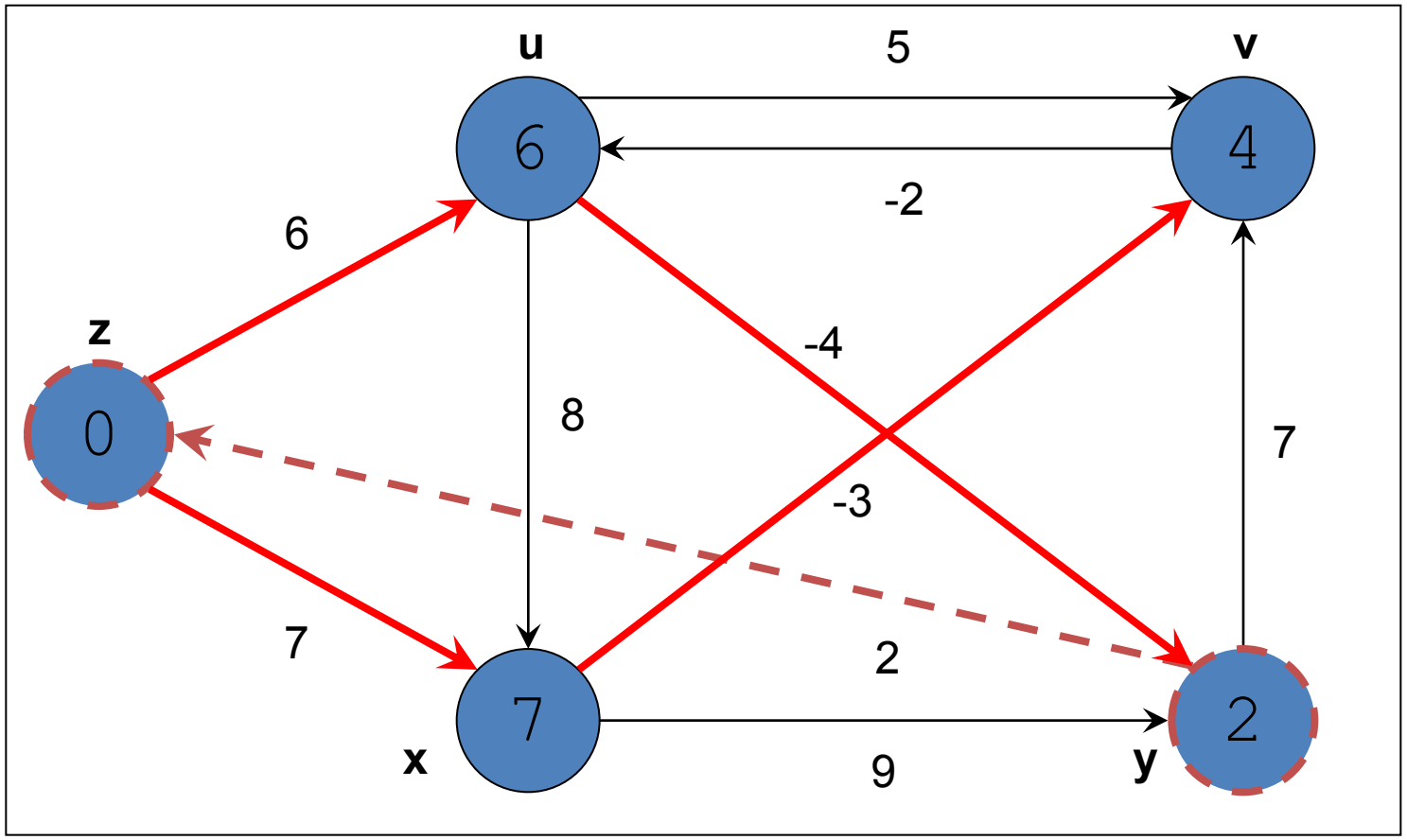
Paso 2.7

Aplicar Relax al Arco (y,v)

Pregunta: **¿ d[v] > d[y] + w(y , v) ?**

Respuesta: NO

Proceso: **No se hace nada.**



Lista de Arcos

(u,v)
(u,x)
(u,y)
(v,u)
(x,v)
(x,y)
(y,v)
(y,z) ←
(z,u)
(z,x)

V	**[**	**]**	**=**	**{**	**u**	**v**	**x**	**y**	**z**	**}**
d	**[**	**]**	**=**	**{**	6	4	7	2	0	**}**
Π	**[**	**]**	**=**	**{**	**z**	**x**	**z**	**u**		**}**

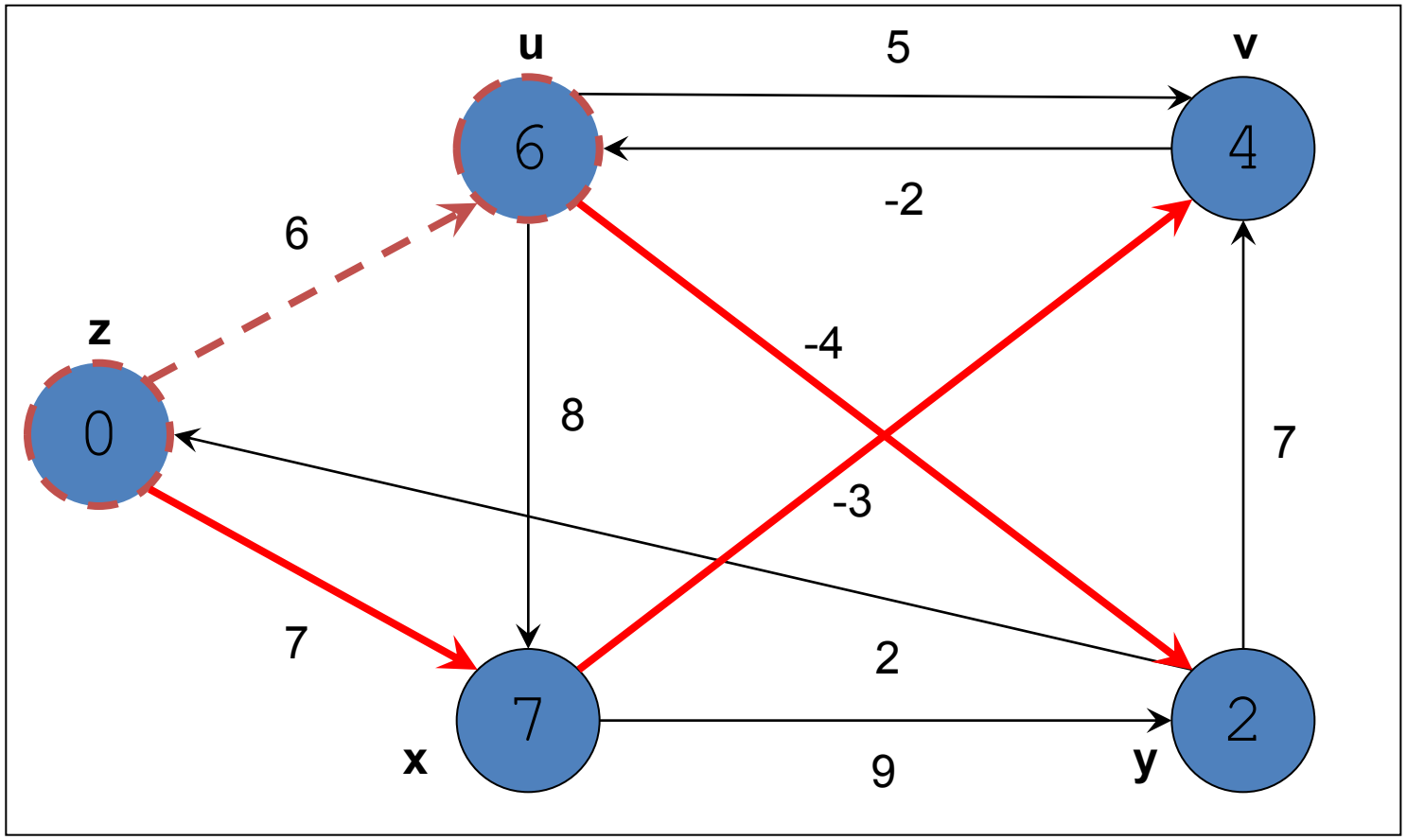
Paso 2.8

Aplicar Relax al Arco (y,z)

Pregunta: **¿ d[z] > d[y] + w(y , z) ?**

Respuesta: NO

Proceso: **No se hace nada.**



Lista de Arcos

(u,v)
(u,x)
(u,y)
(v,u)
(x,v)
(x,y)
(y,v)
(y,z)
(z,u) ←
(z,x)

V	**[**	**]**	**=**	**{**	**u**	**v**	**x**	**y**	**z**	**}**
d	**[**	**]**	**=**	**{**	6	4	7	2	0	**}**
Π	**[**	**]**	**=**	**{**	**z**	**x**	**z**	**u**		**}**

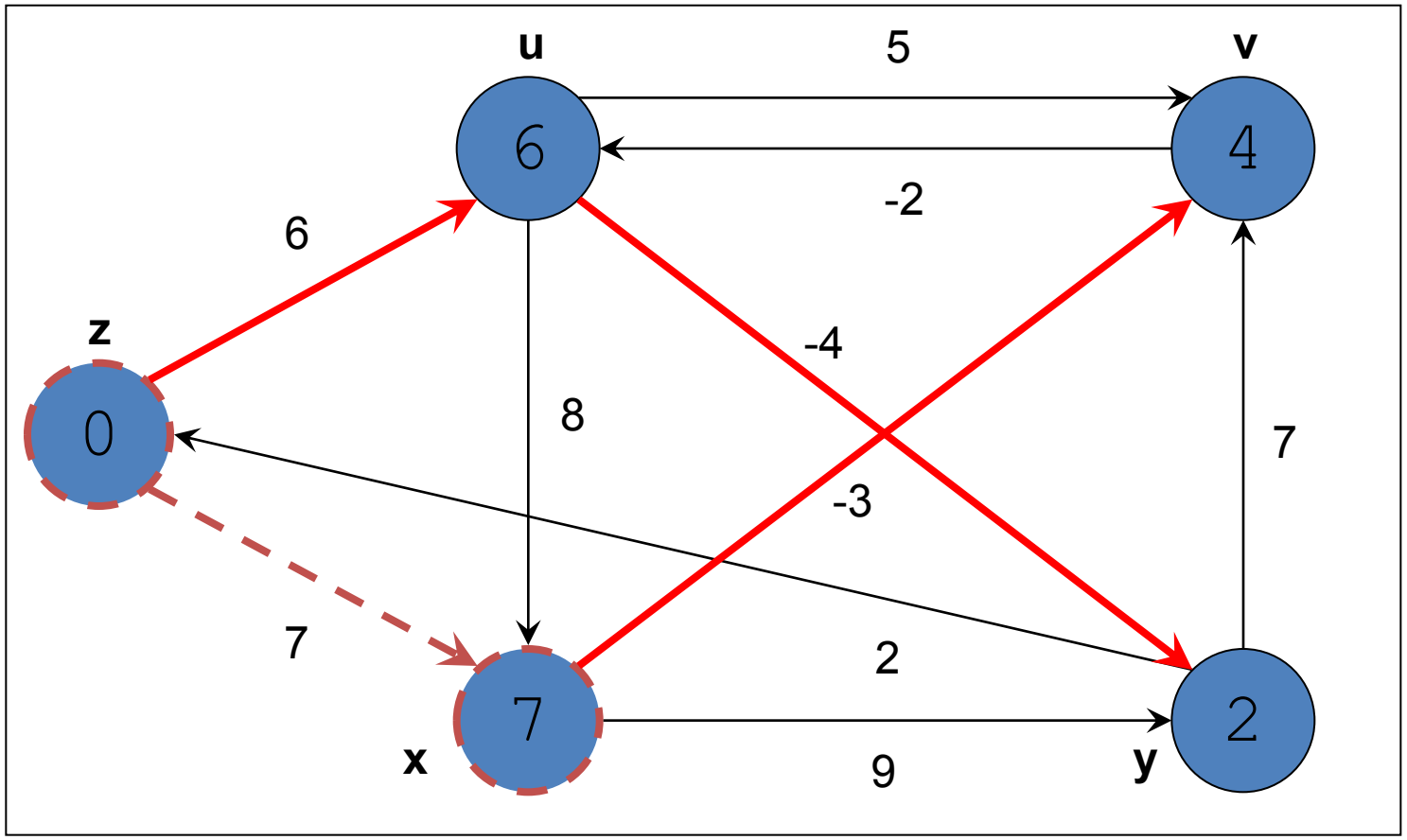
Paso 2.9

Aplicar Relax al Arco (z,u)

Pregunta: **¿ d[u] > d[z] + w(z , u) ?**

Respuesta: NO

Proceso: **No se hace nada.**



Lista de Arcos

(u,v)
(u,x)
(u,y)
(v,u)
(x,v)
(x,y)
(y,v)
(y,z)
(z,u)
(z,x) ←

V	**[**	**]**	**=**	**{**	**u**	**v**	**x**	**y**	**z**	**}**
d	**[**	**]**	**=**	**{**	6	4	7	2	0	**}**
Π	**[**	**]**	**=**	**{**	**z**	**x**	**z**	**u**		**}**

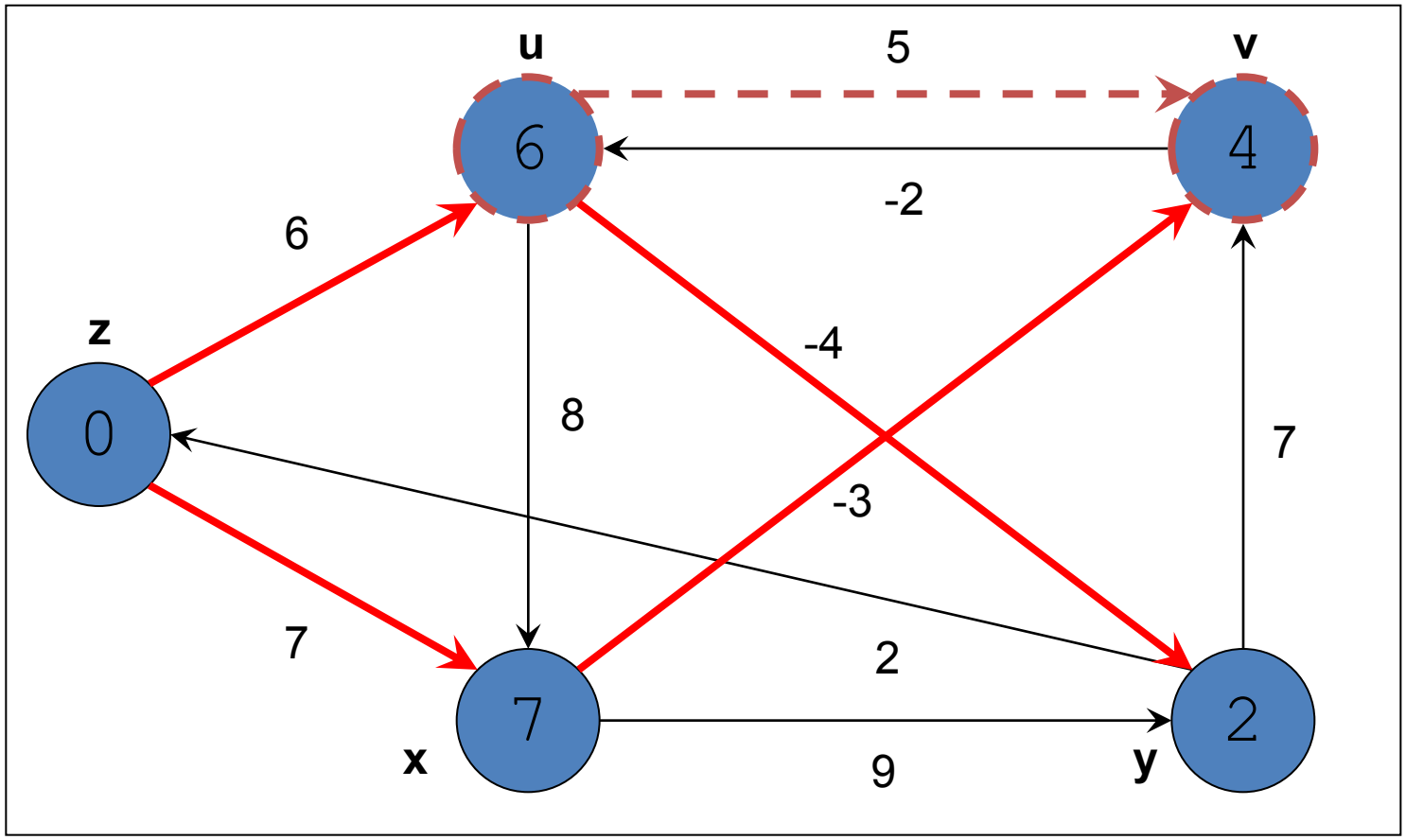
Paso 2.10

Aplicar Relax al Arco (z,x)

Pregunta: ¿ d[x] > d[z] + w(z , x) ?

Respuesta: NO

Proceso: **No se hace nada.**



Lista de Arcos

(u,v) ←
(u,x)
(u,y)
(v,u)
(x,v)
(x,y)
(y,v)
(y,z)
(z,u)
(z,x)

V	**[**	**]**	**=**	**{**	**u**	**v**	**x**	**y**	**z**	**}**
d	**[**	**]**	**=**	**{**	6	4	7	2	0	**}**
Π	**[**	**]**	**=**	**{**	**z**	**x**	**z**	**u**		**}**

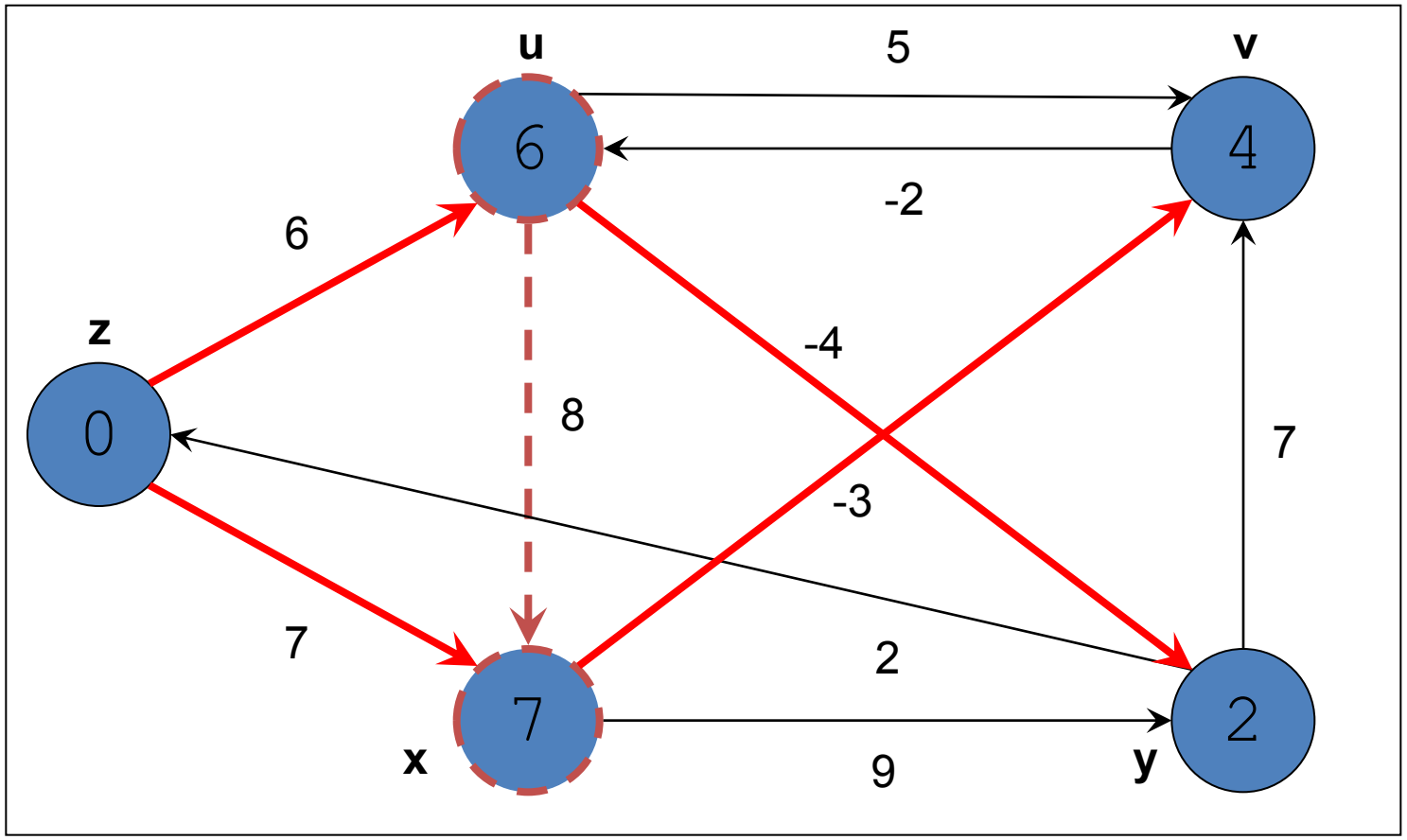
Paso 3.1

Aplicar Relax al Arco (u,v)

Pregunta: **¿ d[v] > d[u] + w(u , v) ?**

Respuesta: NO

Proceso: **No se hace nada.**



Lista de Arcos

(u,v)
(u,x) ←
(u,y)
(v,u)
(x,v)
(x,y)
(y,v)
(y,z)
(z,u)
(z,x)

V	**[**	**]**	**=**	**{**	**u**	**v**	**x**	**y**	**z**	**}**
d	**[**	**]**	**=**	**{**	6	4	7	2	0	**}**
Π	**[**	**]**	**=**	**{**	**z**	**x**	**z**	**u**		**}**

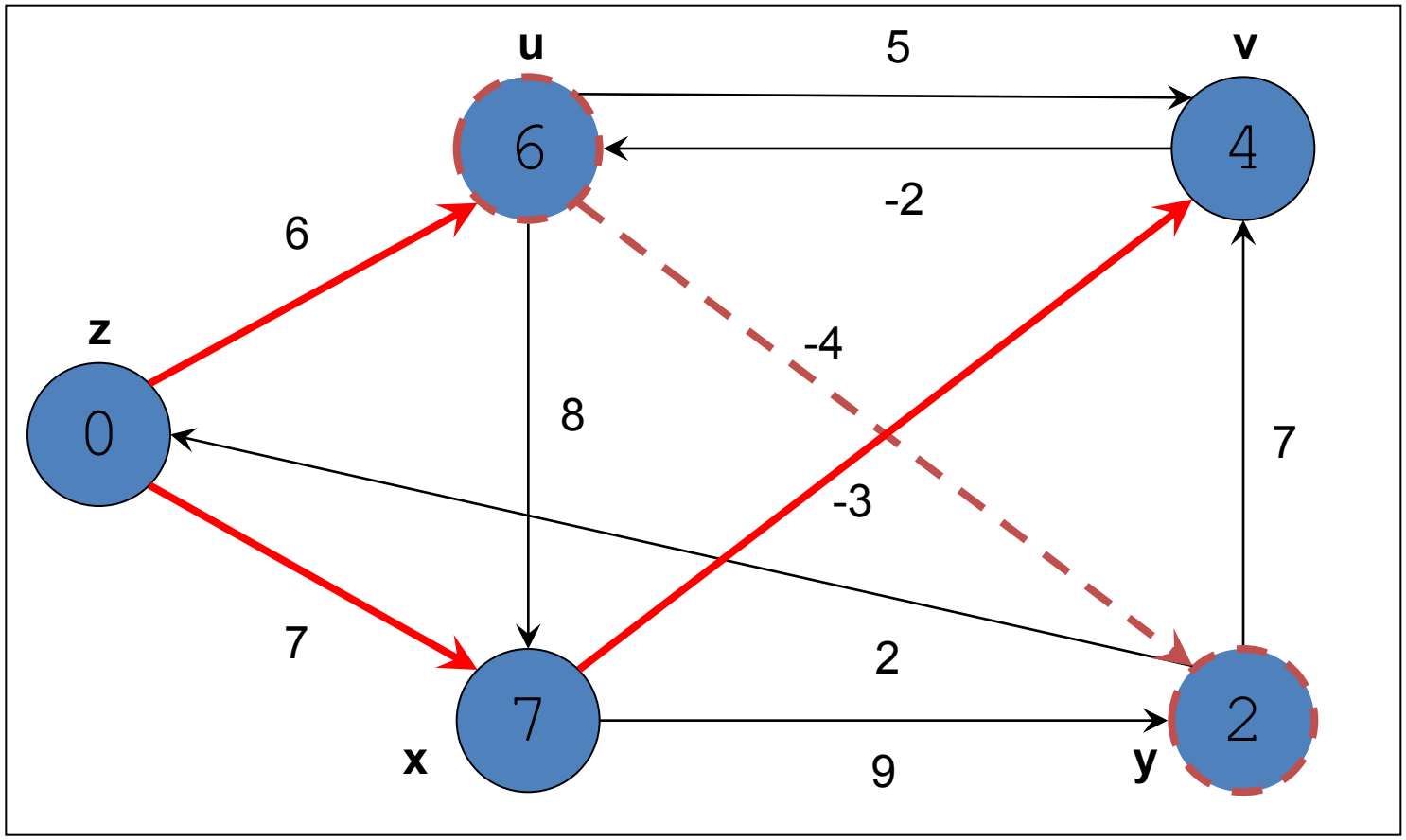
Paso 3.2

Aplicar Relax al Arco (u,x)

Pregunta: **¿ d[x] > d[u] + w(u , x) ?**

Respuesta: NO

Proceso: **No se hace nada.**



Lista de Arcos

(u,v)
(u,x)
(u,y) ←
(v,u)
(x,v)
(x,y)
(y,v)
(y,z)
(z,u)
(z,x)

V	**[**	**]**	**=**	**{**	**u**	**v**	**x**	**y**	**z**	**}**
d	**[**	**]**	**=**	**{**	6	4	7	2	0	**}**
Π	**[**	**]**	**=**	**{**	**z**	**x**	**z**	**u**		**}**

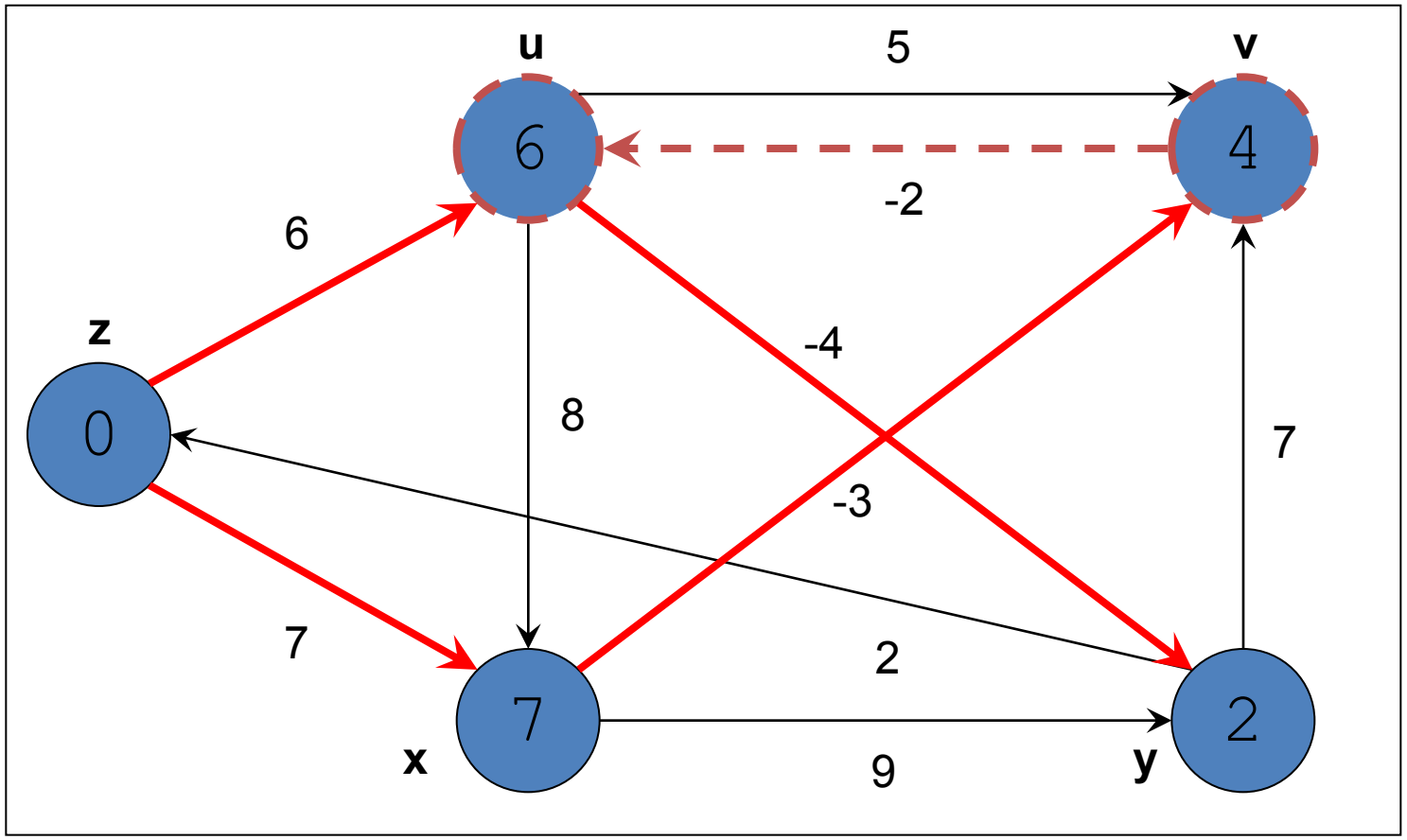
Paso 3.3

Aplicar Relax al Arco (u,y)

Pregunta: **¿ d[y] > d[u] + w(u , y) ?**

Respuesta: NO

Proceso: **No se hace nada.**



Lista de Arcos

(u,v)
(u,x)
(u,y)
(v,u) ←
(x,v)
(x,y)
(y,v)
(y,z)
(z,u)
(z,x)

V	**[**	**]**	**=**	**{**	**u**	**v**	**x**	**y**	**z**	**}**
d	**[**	**]**	**=**	**{**	6	4	7	2	0	**}**
Π	**[**	**]**	**=**	**{**	**z**	**x**	**z**	**u**		**}**

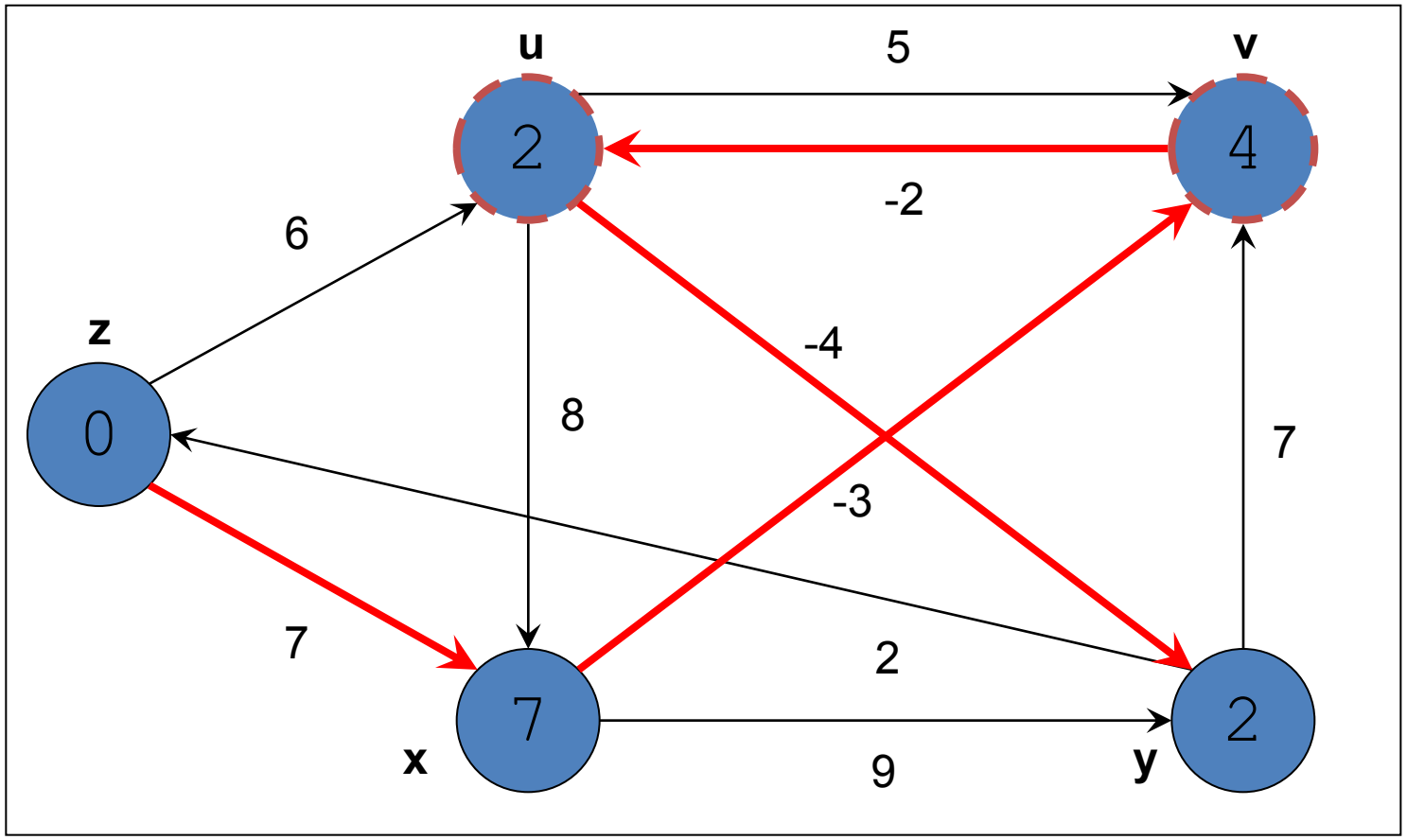
Paso 3.4

Aplicar Relax al Arco (v, u)

Pregunta: ¿ d[u] > d[v] + w(v , u) ?

Respuesta: SI

Proceso: d[u] = d[v] + w(v, u) y Π[u] = v



Lista de Arcos

(u,v)
(u,x)
(u,y)
(v,u) ←
(x,v)
(x,y)
(y,v)
(y,z)
(z,u)
(z,x)

V	**[**	**]**	**=**	**{**	**u**	**v**	**x**	**y**	**z**	**}**
d	**[**	**]**	**=**	**{**	2	4	7	2	0	**}**
Π	**[**	**]**	**=**	**{**	**v**	**x**	**z**	**u**		**}**

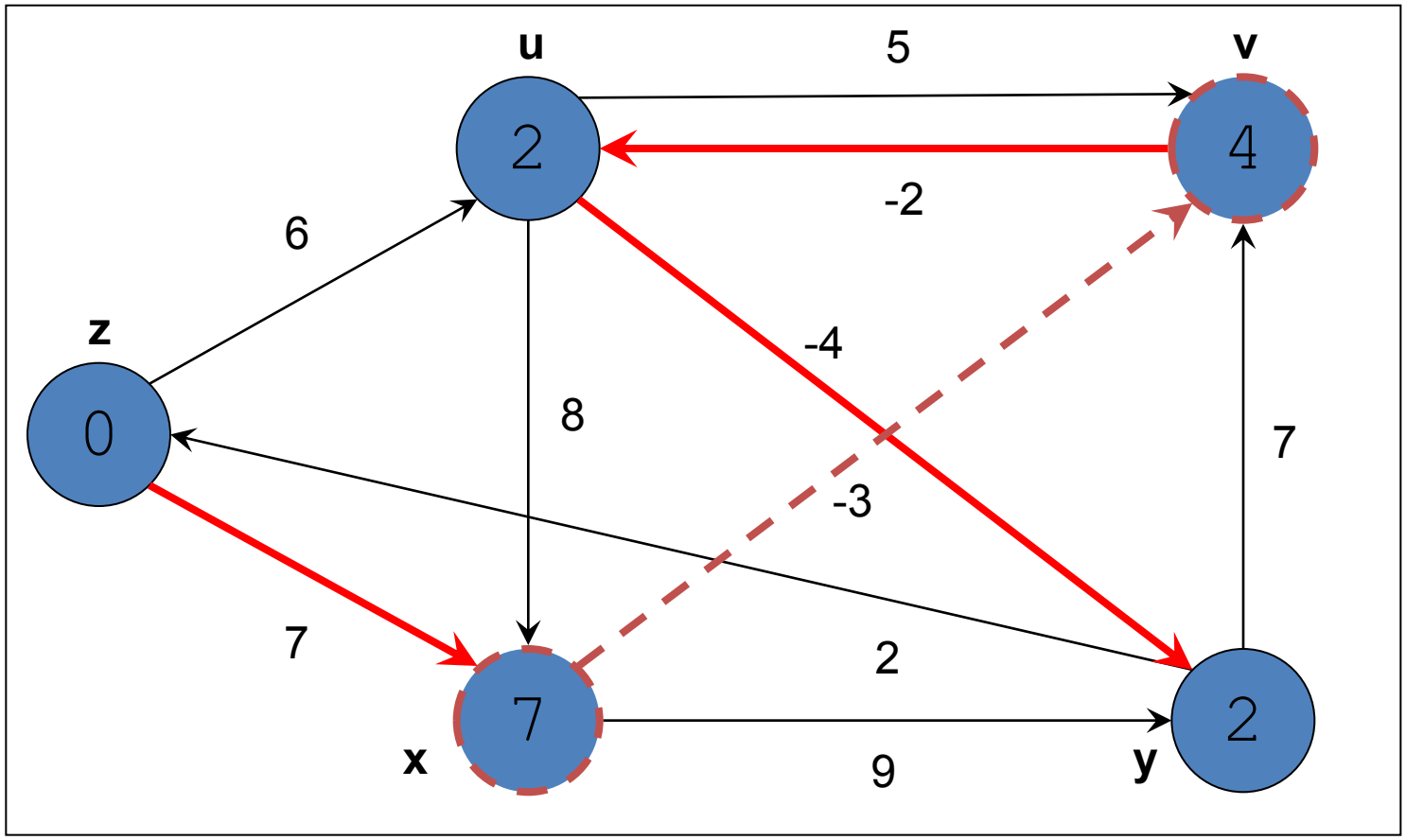
Paso 3.4

Aplicar Relax al Arco (v, u)

Pregunta: ¿ d[u] > d[v] + w(v , u) ?

Respuesta: SI

Proceso: d[u] = d[v] + w(v, u) y Π[u] = v



Lista de Arcos

(u,v)
(u,x)
(u,y)
(v,u)
(x,v) ←
(x,y)
(y,v)
(y,z)
(z,u)
(z,x)

V	**[**	**]**	**=**	**{**	**u**	**v**	**x**	**y**	**z**	**}**
d	**[**	**]**	**=**	**{**	2	4	7	2	0	**}**
Π	**[**	**]**	**=**	**{**	**v**	**x**	**z**	**u**		**}**

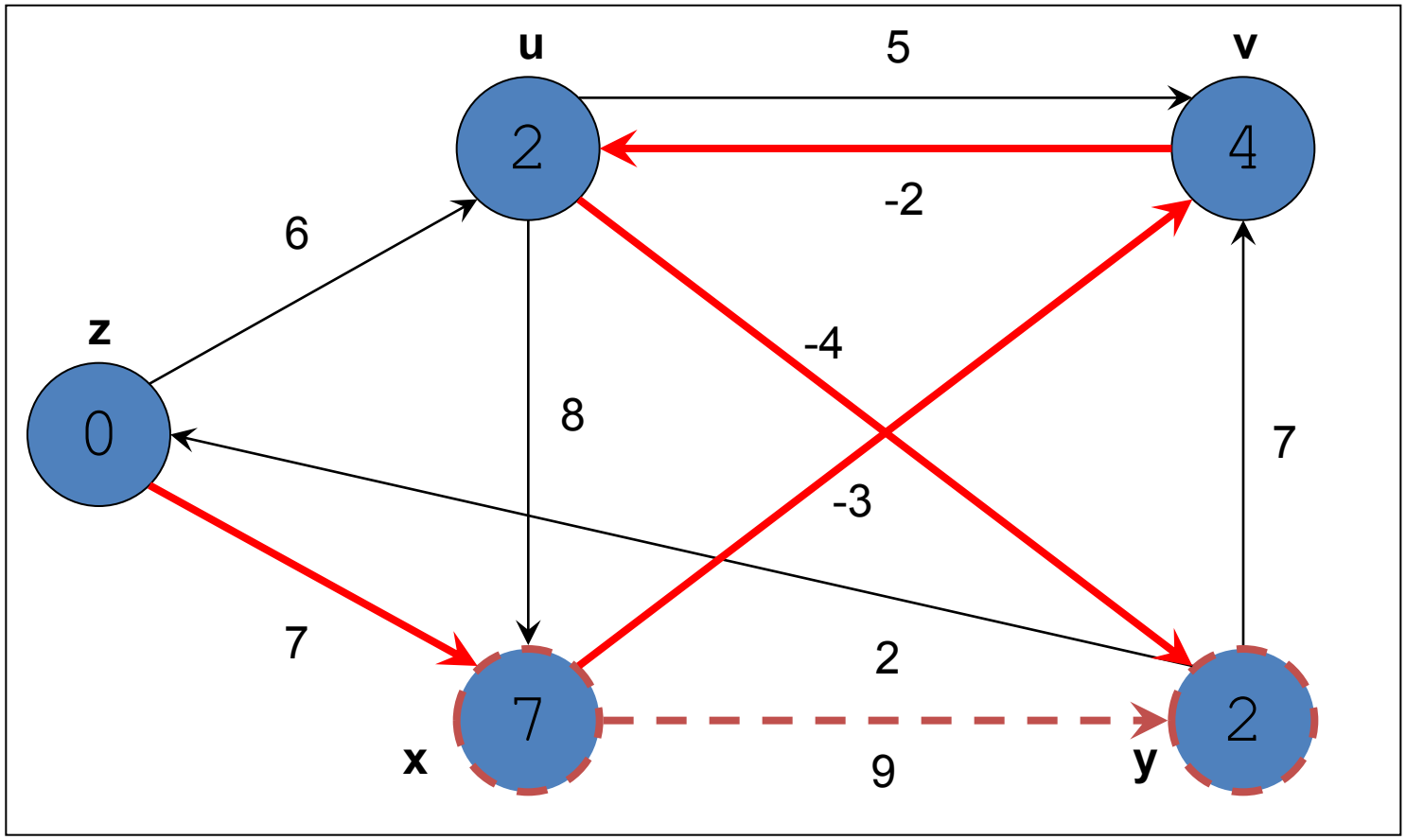
Paso 3.5

Aplicar Relax al Arco (x, v)

Pregunta: **¿ d[v] > d[x] + w(x , v) ?**

Respuesta: NO

Proceso: **No se hace nada.**



Lista de Arcos

(u,v)
(u,x)
(u,y)
(v,u)
(x,v)
(x,y) ←
(y,v)
(y,z)
(z,u)
(z,x)

V	**[**	**]**	**=**	**{**	**u**	**v**	**x**	**y**	**z**	**}**
d	**[**	**]**	**=**	**{**	2	4	7	2	0	**}**
Π	**[**	**]**	**=**	**{**	**v**	**x**	**z**	**u**		**}**

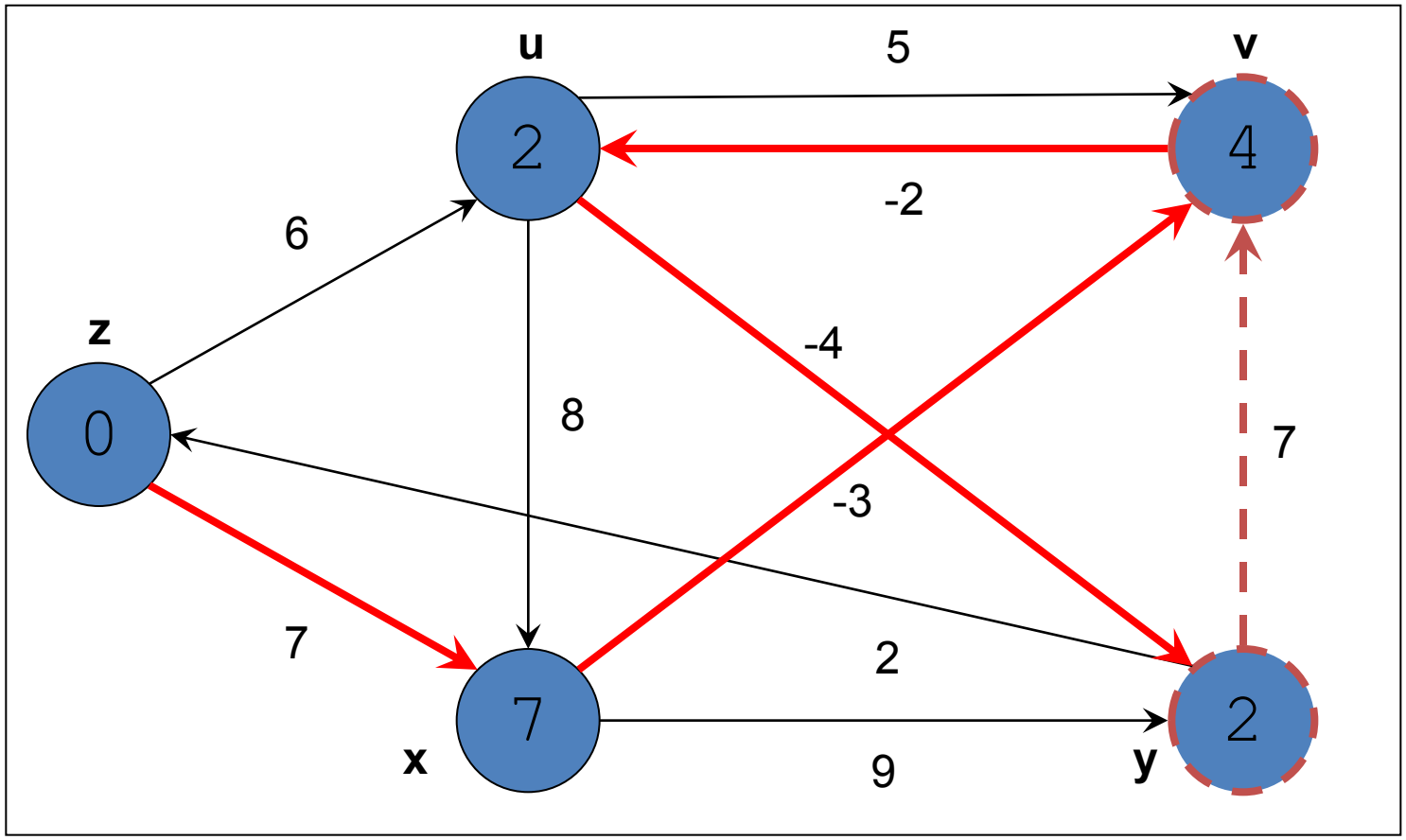
Paso 3.6

Aplicar Relax al Arco (x, y)

Pregunta: **¿ d[y] > d[x] + w(x , y) ?**

Respuesta: NO

Proceso: **No se hace nada.**



Lista de Arcos

(u,v)
(u,x)
(u,y)
(v,u)
(x,v)
(x,y)
(y,v) ←
(y,z)
(z,u)
(z,x)

V	**[**	**]**	**=**	**{**	**u**	**v**	**x**	**y**	**z**	**}**
d	**[**	**]**	**=**	**{**	2	4	7	2	0	**}**
Π	**[**	**]**	**=**	**{**	**v**	**x**	**z**	**u**		**}**

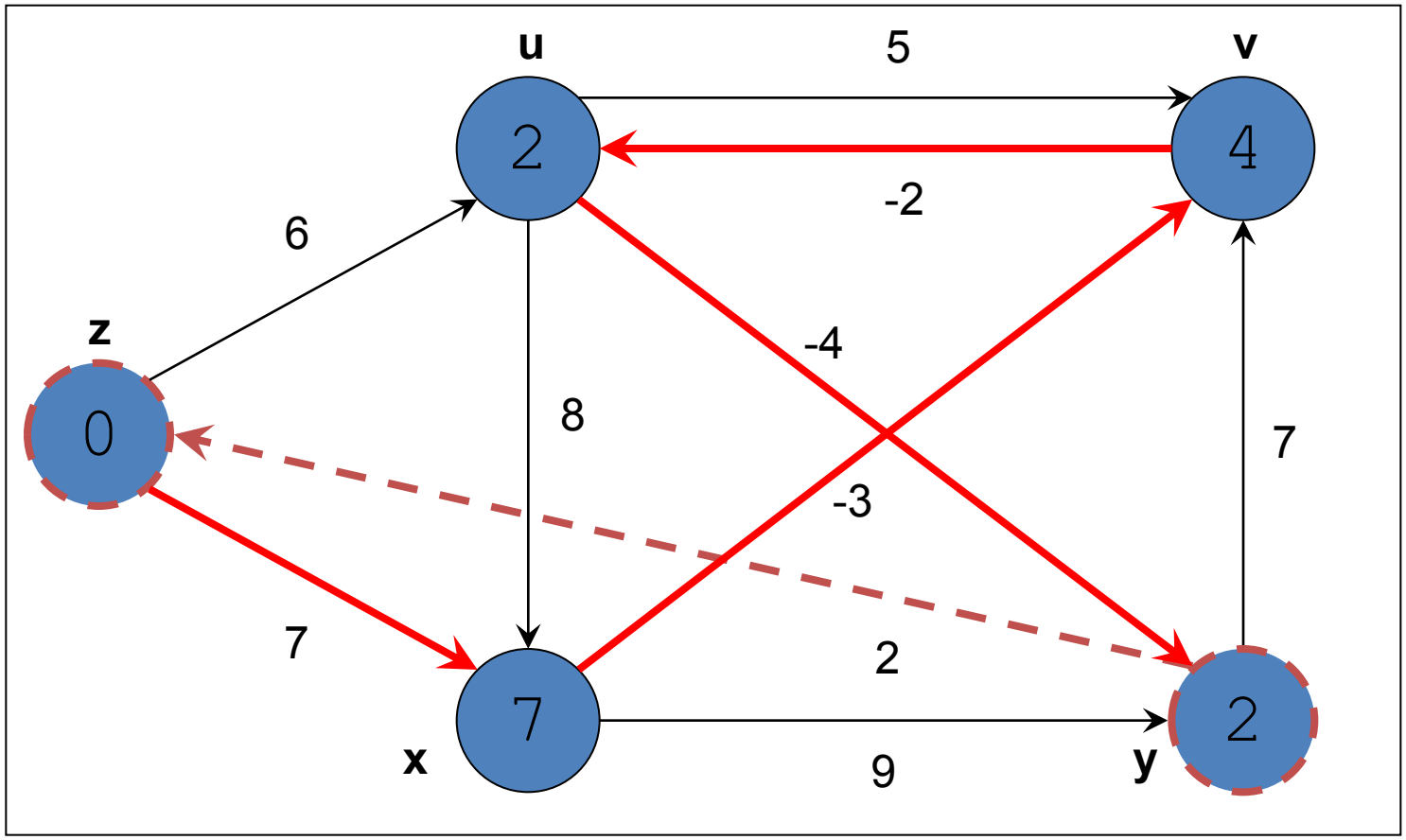
Paso 3.7

Aplicar Relax al Arco (y, v)

Pregunta: **¿ d[v] > d[y] + w(y , v) ?**

Respuesta: NO

Proceso: **No se hace nada.**



Lista de Arcos

(u,v)
(u,x)
(u,y)
(v,u)
(x,v)
(x,y)
(y,v)
(y,z) ←
(z,u)
(z,x)

V	**[**	**]**	**=**	**{**	**u**	**v**	**x**	**y**	**z**	**}**
d	**[**	**]**	**=**	**{**	2	4	7	2	0	**}**
Π	**[**	**]**	**=**	**{**	**v**	**x**	**z**	**u**		**}**

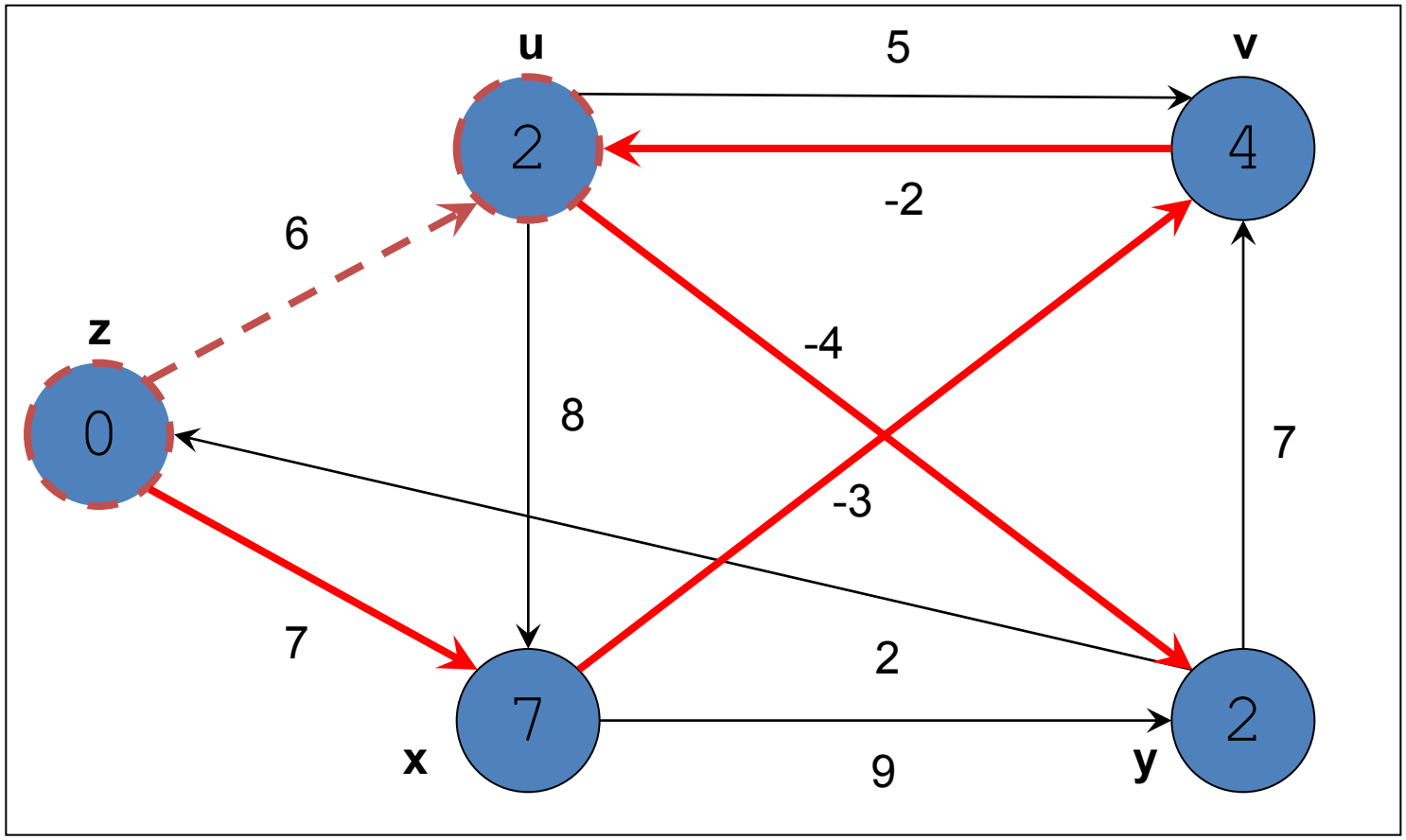
Paso 3.8

Aplicar Relax al Arco (y, z)

Pregunta: **¿ d[z] > d[y] + w(y , z) ?**

Respuesta: NO

Proceso: **No se hace nada.**



Lista de Arcos

(u,v)
(u,x)
(u,y)
(v,u)
(x,v)
(x,y)
(y,v)
(y,z)
(z,u) ←
(z,x)

V	**[**	**]**	**=**	**{**	**u**	**v**	**x**	**y**	**z**	**}**
d	**[**	**]**	**=**	**{**	2	4	7	2	0	**}**
Π	**[**	**]**	**=**	**{**	**v**	**x**	**z**	**u**		**}**

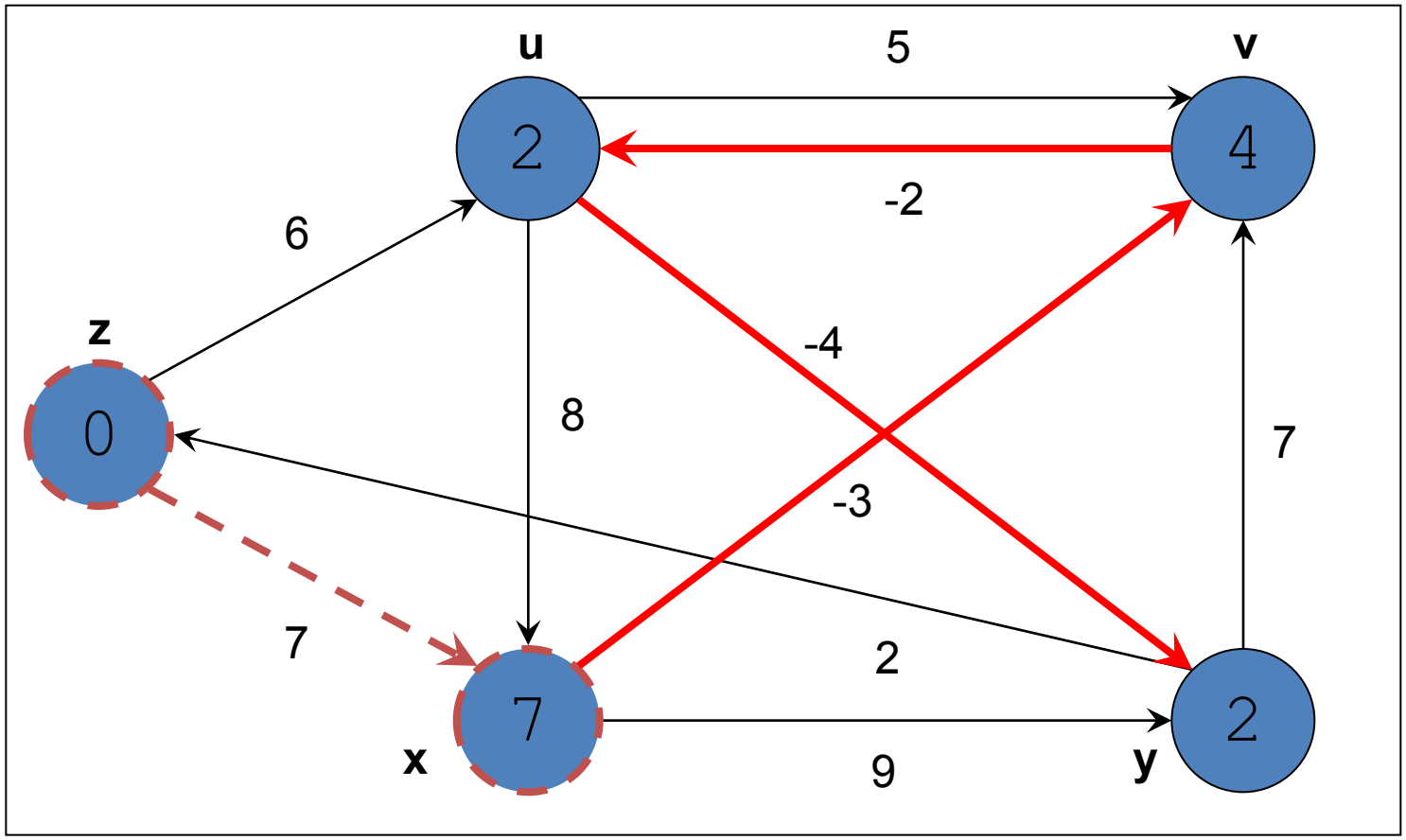
Paso 3.9

Aplicar Relax al Arco (z, u)

Pregunta: **¿ d[u] > d[z] + w(z , u) ?**

Respuesta: NO

Proceso: **No se hace nada.**



Lista de Arcos

(u,v)
(u,x)
(u,y)
(v,u)
(x,v)
(x,y)
(y,v)
(y,z)
(z,u)
(z,x) ←

V	**[**	**]**	**=**	**{**	**u**	**v**	**x**	**y**	**z**	**}**
d	**[**	**]**	**=**	**{**	2	4	7	2	0	**}**
Π	**[**	**]**	**=**	**{**	**v**	**x**	**z**	**u**		**}**

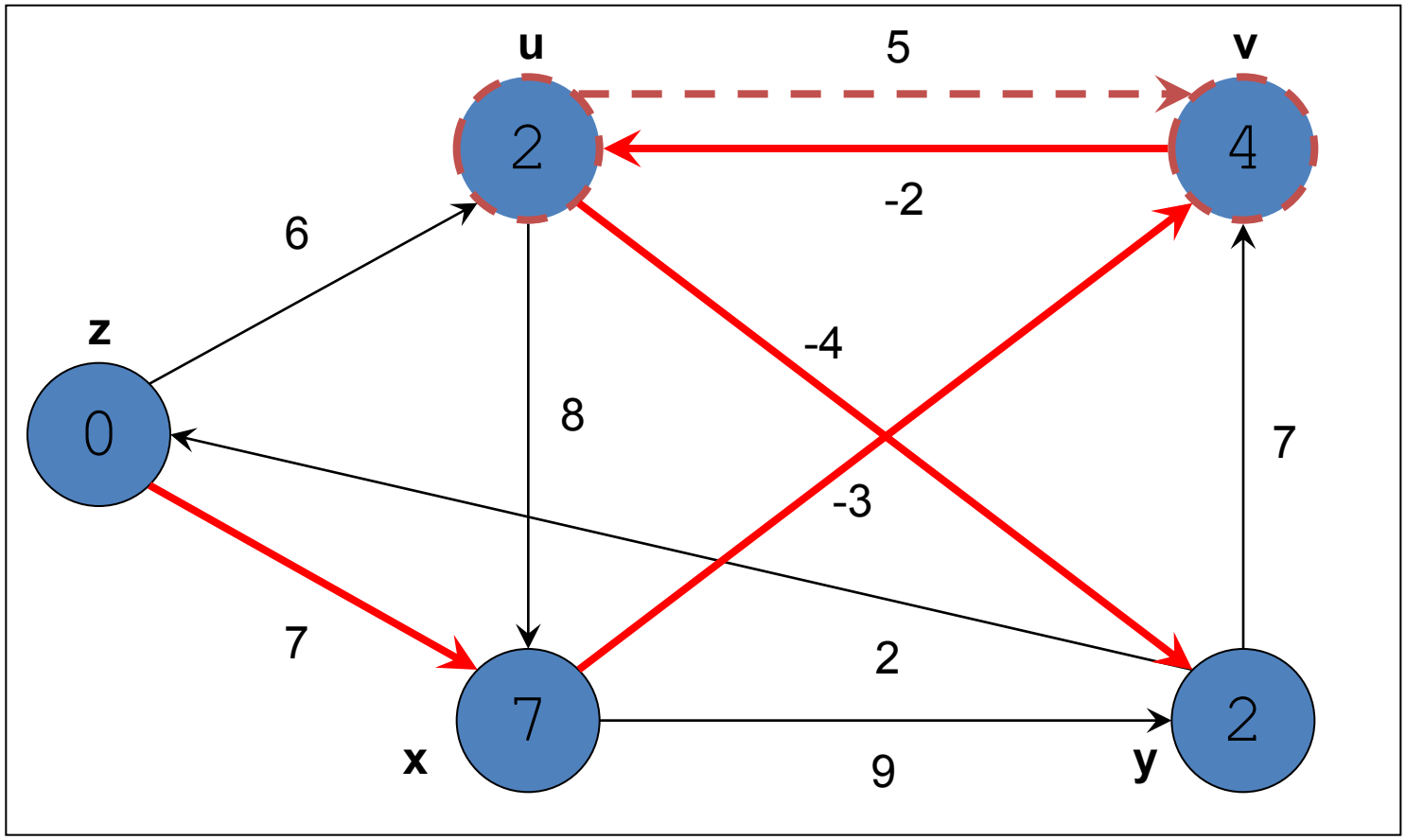
Paso 3.10

Aplicar Relax al Arco (z, x)

Pregunta: **¿ d[x] > d[z] + w(z , x) ?**

Respuesta: NO

Proceso: **No se hace nada.**



Lista de Arcos

(u,v) ←
(u,x)
(u,y)
(v,u)
(x,v)
(x,y)
(y,v)
(y,z)
(z,u)
(z,x)

V	**[**	**]**	**=**	**{**	**u**	**v**	**x**	**y**	**z**	**}**
d	**[**	**]**	**=**	**{**	2	4	7	2	0	**}**
Π	**[**	**]**	**=**	**{**	**v**	**x**	**z**	**u**		**}**

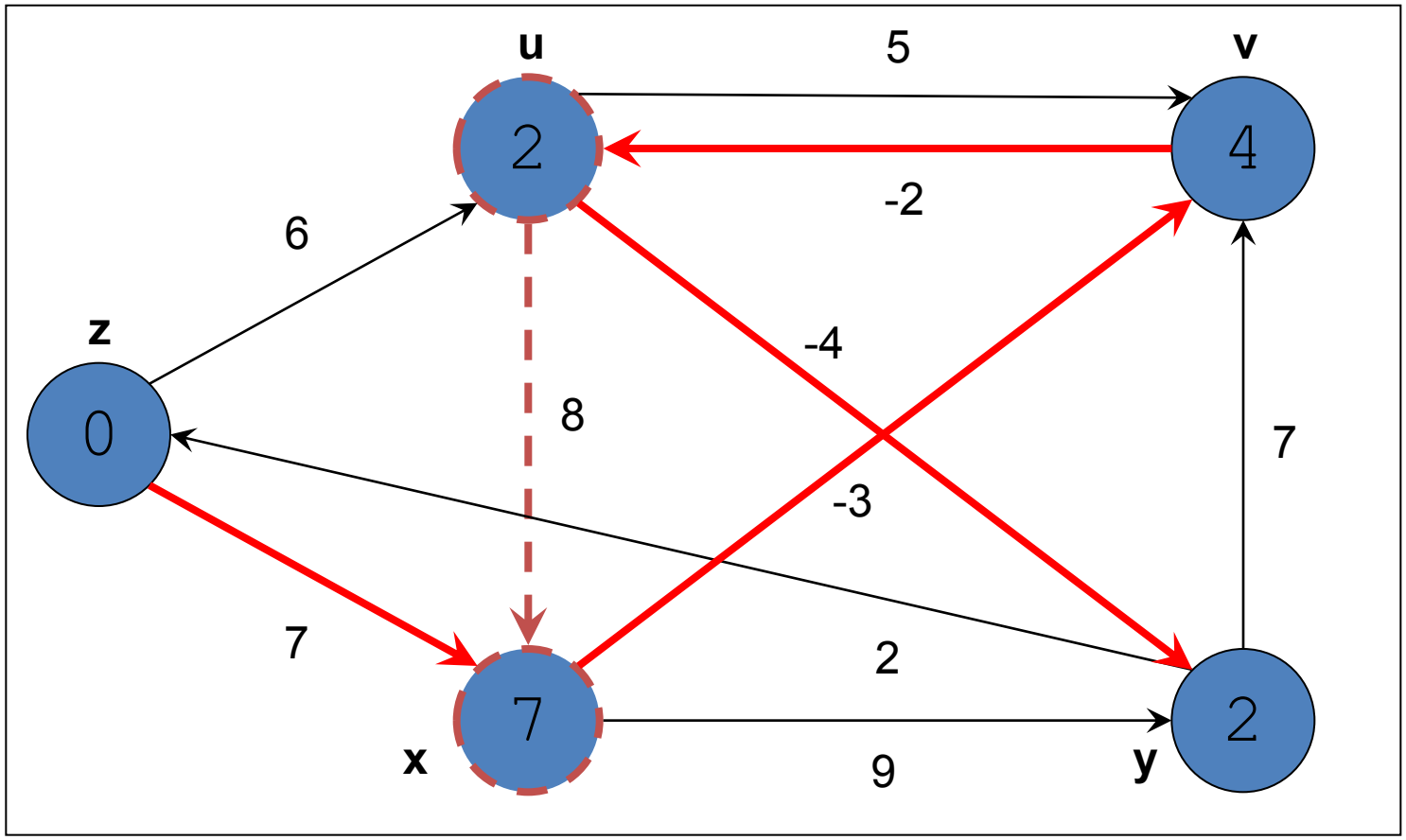
Paso 4.1

Aplicar Relax al Arco (u, v)

Pregunta: **¿ d[v] > d[u] + w(u , v) ?**

Respuesta: NO

Proceso: **No se hace nada.**



Lista de Arcos

(u,v)
(u,x) ←
(u,y)
(v,u)
(x,v)
(x,y)
(y,v)
(y,z)
(z,u)
(z,x)

V	**[**	**]**	**=**	**{**	**u**	**v**	**x**	**y**	**z**	**}**
d	**[**	**]**	**=**	**{**	2	4	7	2	0	**}**
Π	**[**	**]**	**=**	**{**	**v**	**x**	**z**	**u**		**}**

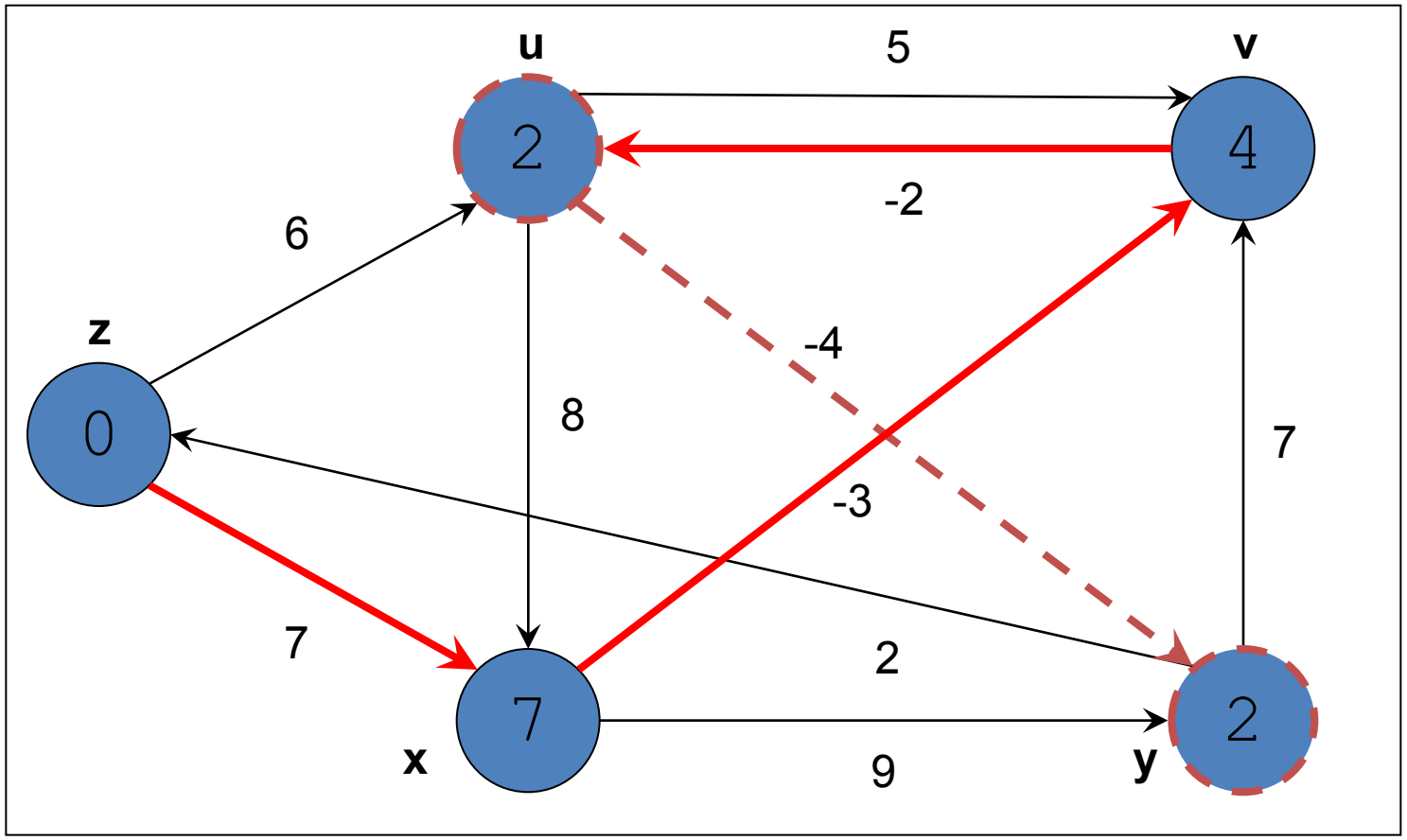
Paso 4.2

Aplicar Relax al Arco (u, x)

Pregunta: **¿ d[x] > d[u] + w(u , x) ?**

Respuesta: NO

Proceso: **No se hace nada.**



Lista de Arcos

(u,v)
(u,x)
(u,y) ←
(v,u)
(x,v)
(x,y)
(y,v)
(y,z)
(z,u)
(z,x)

V	**[**	**]**	**=**	**{**	**u**	**v**	**x**	**y**	**z**	**}**
d	**[**	**]**	**=**	**{**	2	4	7	2	0	**}**
Π	**[**	**]**	**=**	**{**	**v**	**x**	**z**	**u**		**}**

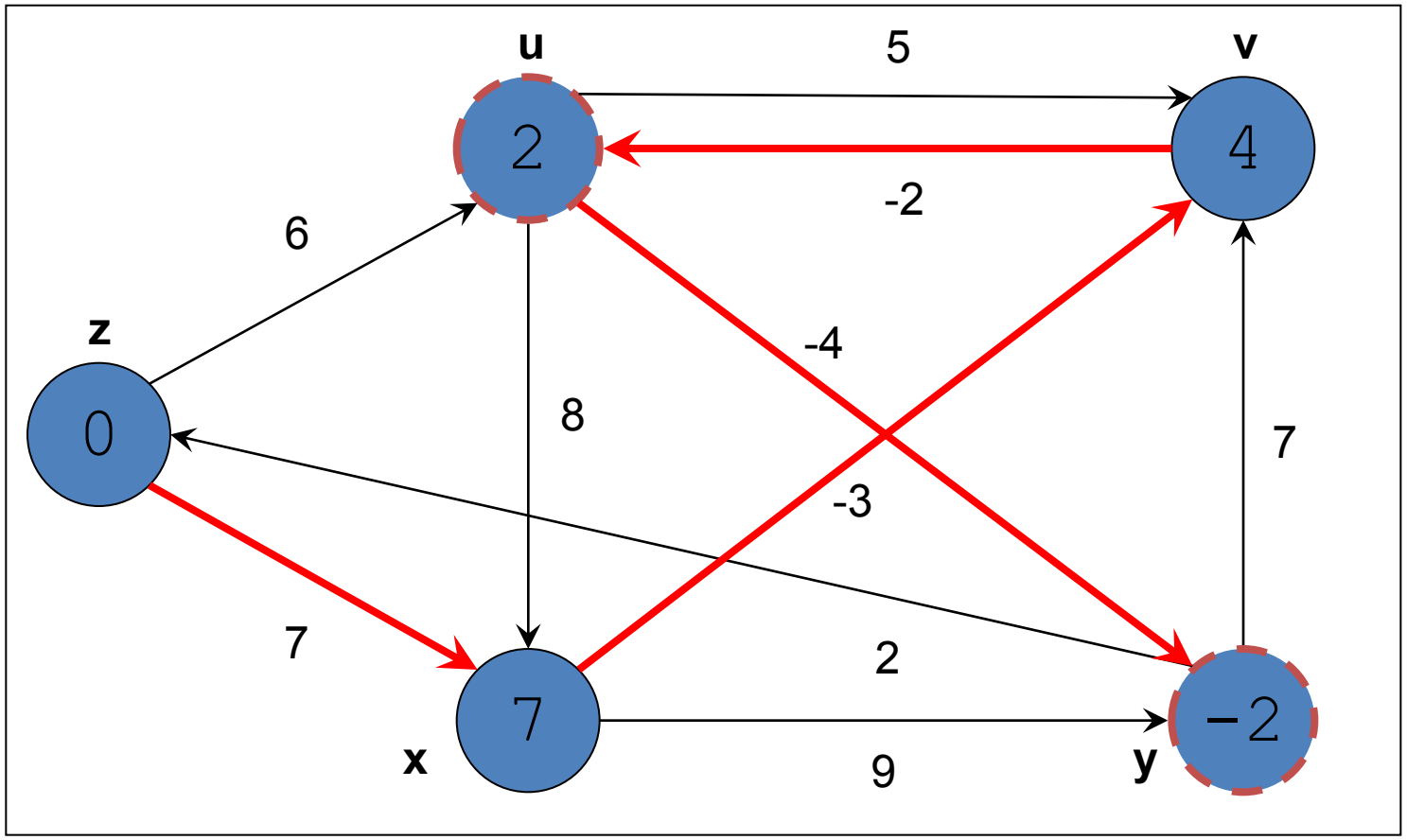
Paso 4.3

Aplicar Relax al Arco (u, y)

Pregunta: ¿ d[y] > d[u] + w(u , y) ?

Respuesta: SI

Proceso: d[y] = d[u] + w(u, y) y Π[y] = u



Lista de Arcos

(u,v)
(u,x)
(u,y) ←
(v,u)
(x,v)
(x,y)
(y,v)
(y,z)
(z,u)
(z,x)

V	**[**	**]**	**=**	**{**	**u**	**v**	**x**	**y**	**z**	**}**
d	**[**	**]**	**=**	**{**	2	4	7	–2	0	**}**
Π	**[**	**]**	**=**	**{**	**v**	**x**	**z**	**u**		**}**

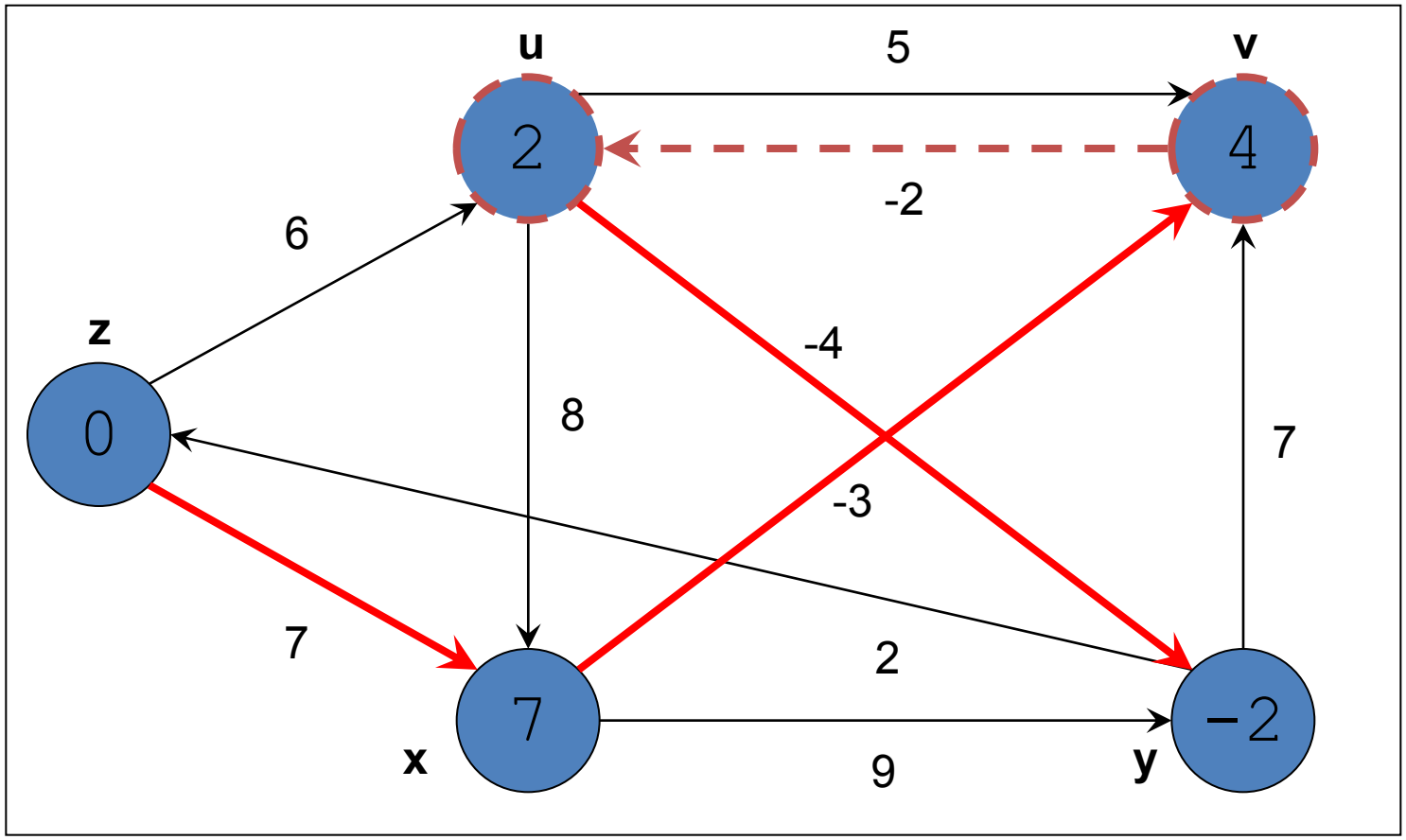
Paso 4.3

Aplicar Relax al Arco (u, y)

Pregunta: ¿ d[y] > d[u] + w(u , y) ?

Respuesta: SI

Proceso: d[y] = d[u] + w(u, y) y Π[y] = u



Lista de Arcos

(u,v)
(u,x)
(u,y)
(v,u) ←
(x,v)
(x,y)
(y,v)
(y,z)
(z,u)
(z,x)

V	**[**	**]**	**=**	**{**	**u**	**v**	**x**	**y**	**z**	**}**
d	**[**	**]**	**=**	**{**	2	4	7	–2	0	**}**
Π	**[**	**]**	**=**	**{**	**v**	**x**	**z**	**u**		**}**

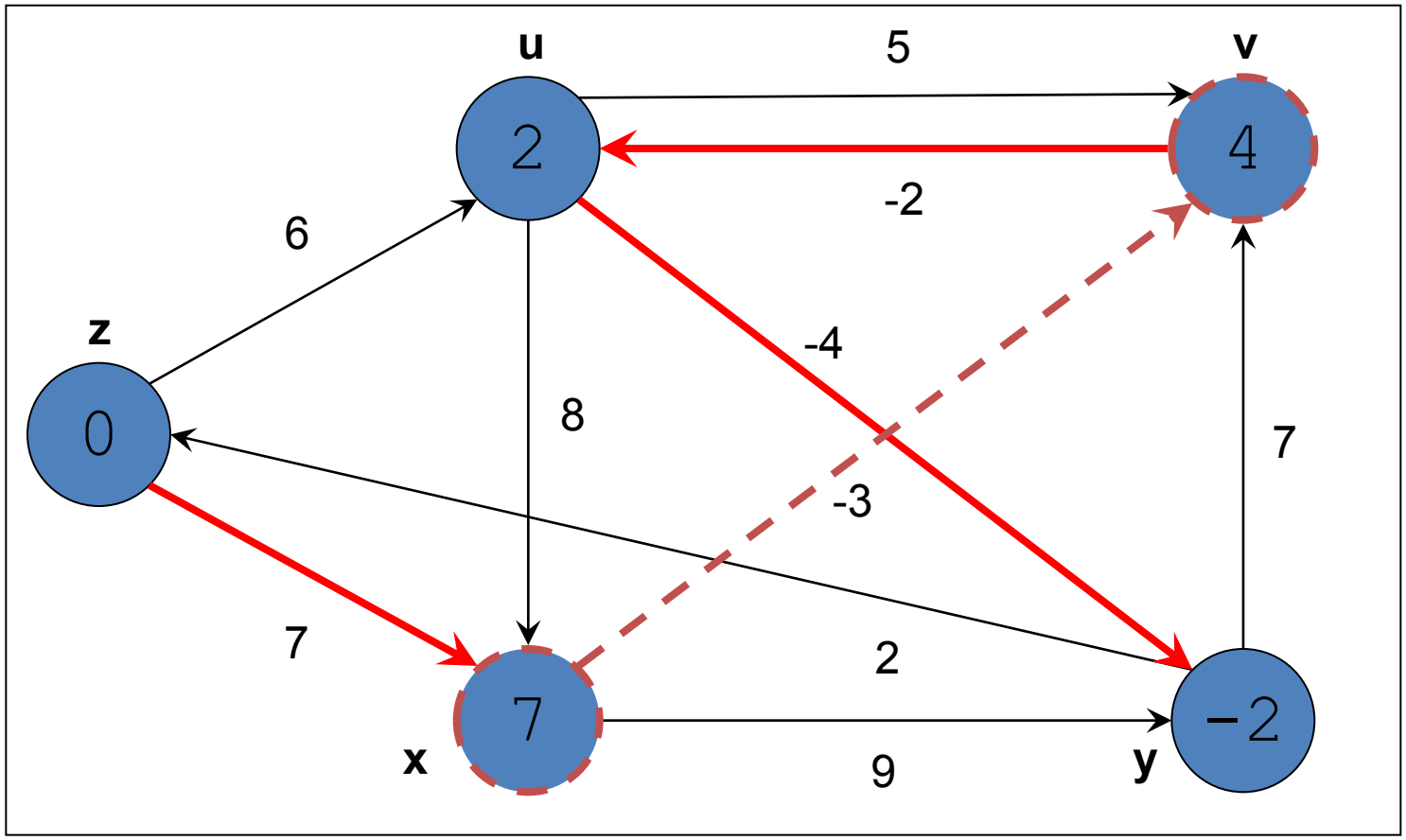
Paso 4.4

Aplicar Relax al Arco (v, u)

Pregunta: **¿ d[u] > d[v] + w(v , u) ?**

Respuesta: NO

Proceso: **No se hace nada.**



Lista de Arcos

(u,v)
(u,x)
(u,y)
(v,u)
(x,v) ←
(x,y)
(y,v)
(y,z)
(z,u)
(z,x)

V	**[**	**]**	**=**	**{**	**u**	**v**	**x**	**y**	**z**	**}**
d	**[**	**]**	**=**	**{**	2	4	7	–2	0	**}**
Π	**[**	**]**	**=**	**{**	**v**	**x**	**z**	**u**		**}**

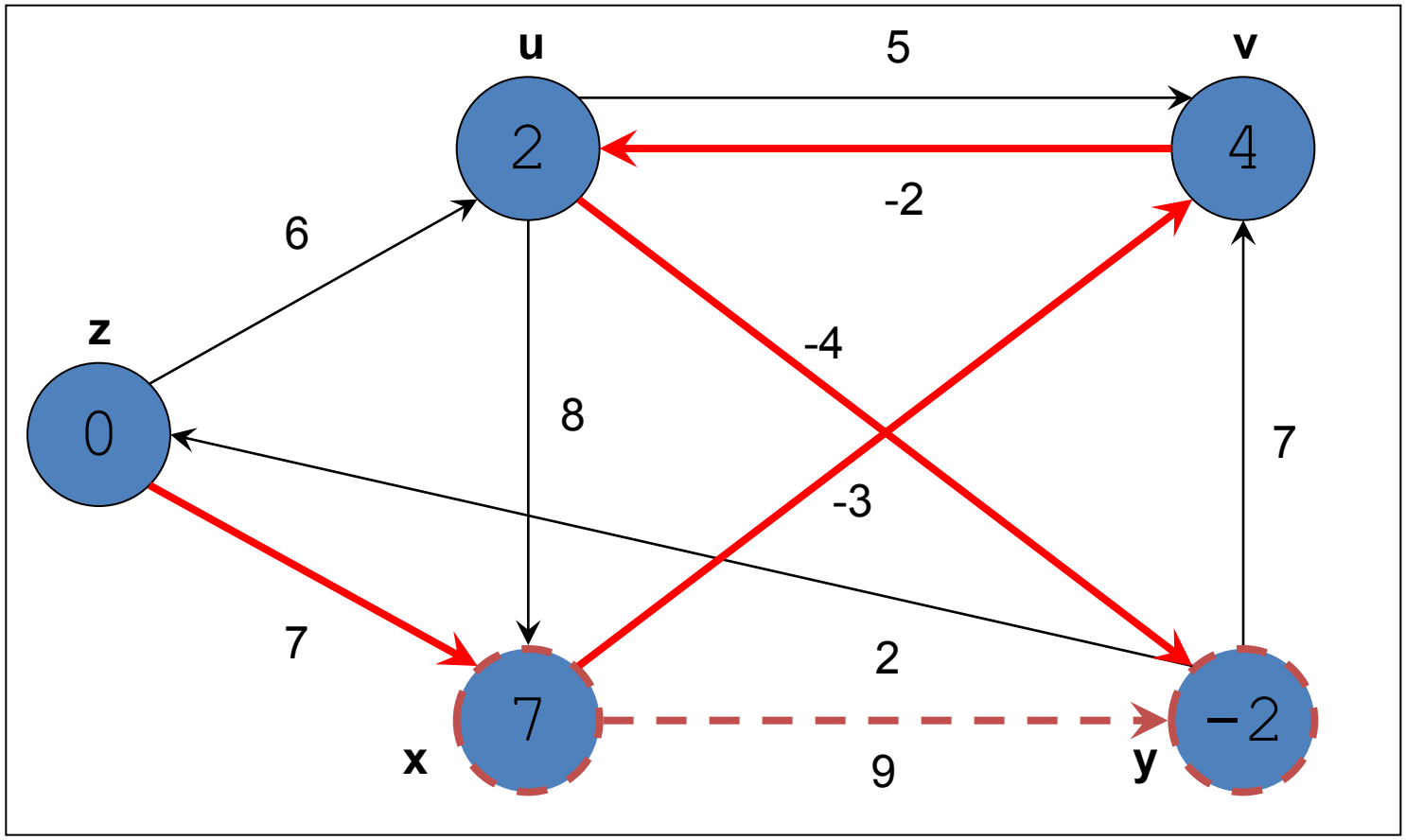
Paso 4.5

Aplicar Relax al Arco (x, v)

Pregunta: **¿ d[v] > d[x] + w(x , v) ?**

Respuesta: NO

Proceso: **No se hace nada.**



Lista de Arcos

(u,v)
(u,x)
(u,y)
(v,u)
(x,v)
(x,y) ←
(y,v)
(y,z)
(z,u)
(z,x)

V	**[**	**]**	**=**	**{**	**u**	**v**	**x**	**y**	**z**	**}**
d	**[**	**]**	**=**	**{**	2	4	7	–2	0	**}**
Π	**[**	**]**	**=**	**{**	**v**	**x**	**z**	**u**		**}**

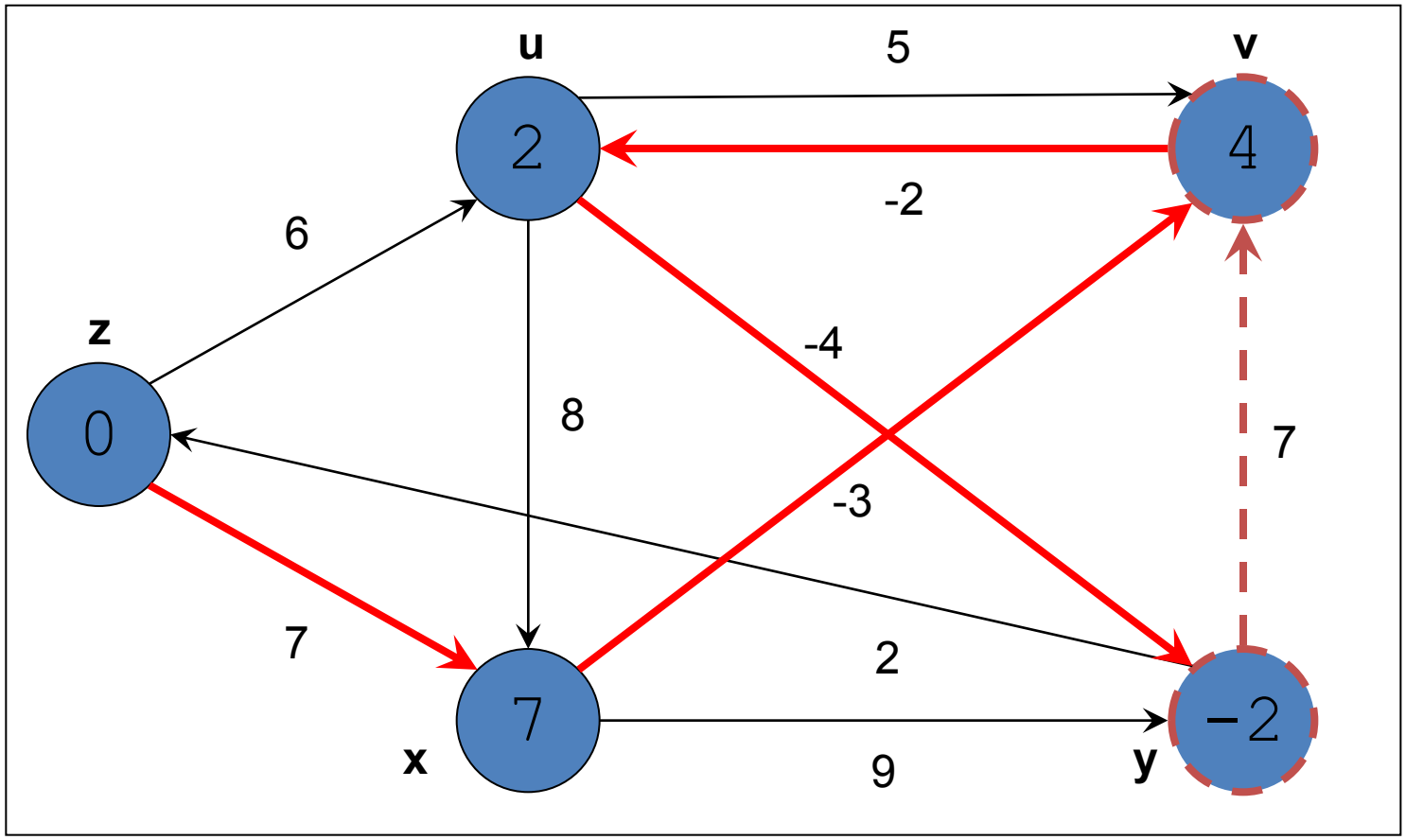
Paso 4.6

Aplicar Relax al Arco (x, y)

Pregunta: ¿ d[y] > d[x] + w(x , y) ?

Respuesta: NO

Proceso: **No se hace nada.**



Lista de Arcos

(u,v)
(u,x)
(u,y)
(v,u)
(x,v)
(x,y)
(y,v) ←
(y,z)
(z,u)
(z,x)

V	**[**	**]**	**=**	**{**	**u**	**v**	**x**	**y**	**z**	**}**
d	**[**	**]**	**=**	**{**	2	4	7	–2	0	**}**
Π	**[**	**]**	**=**	**{**	**v**	**x**	**z**	**u**		**}**

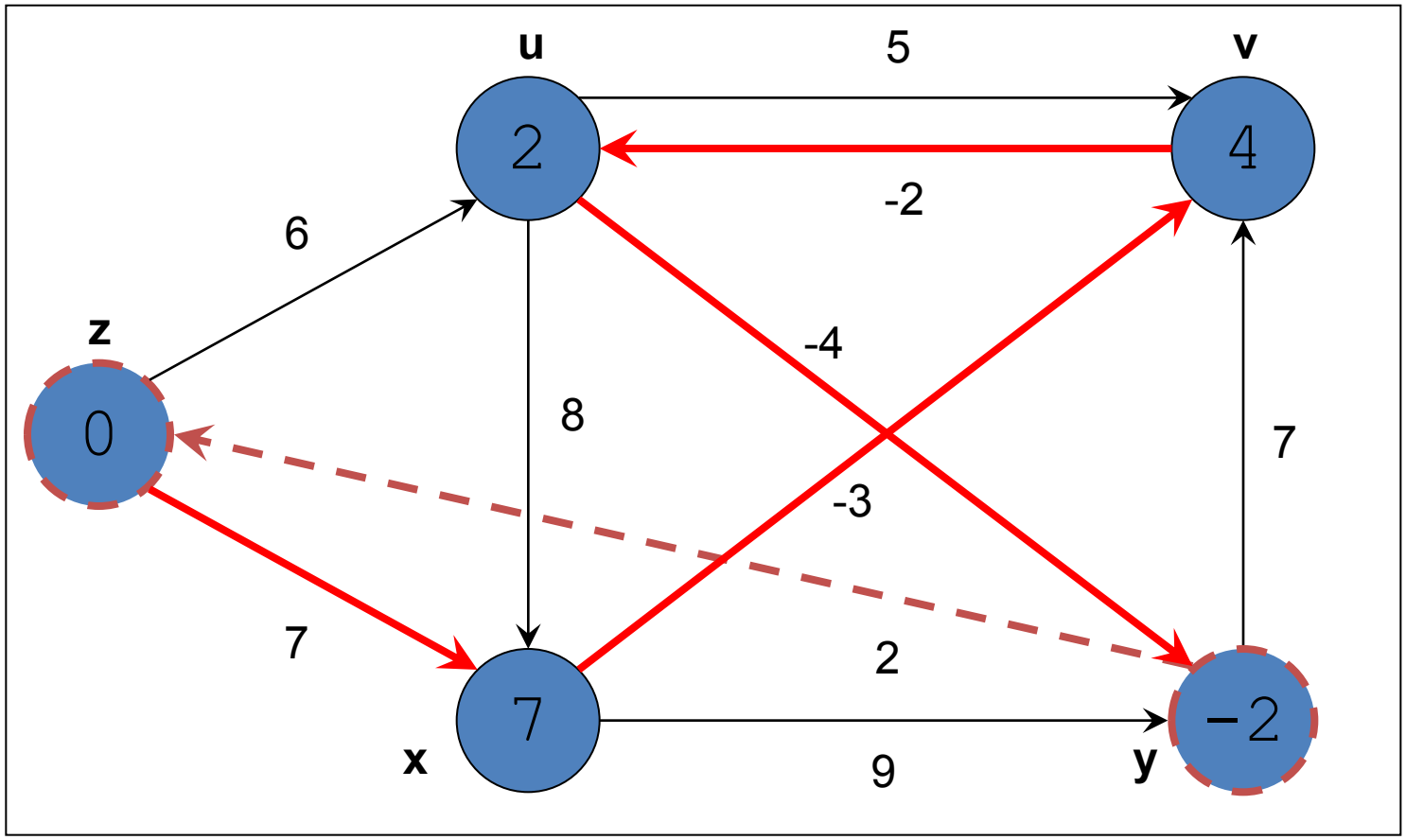
Paso 4.7

Aplicar Relax al Arco (y, v)

Pregunta: **¿ d[v] > d[y] + w(y , v) ?**

Respuesta: NO

Proceso: **No se hace nada.**



Lista de Arcos

(u,v)
(u,x)
(u,y)
(v,u)
(x,v)
(x,y)
(y,v)
(y,z) ←
(z,u)
(z,x)

V	**[**	**]**	**=**	**{**	**u**	**v**	**x**	**y**	**z**	**}**
d	**[**	**]**	**=**	**{**	2	4	7	–2	0	**}**
Π	**[**	**]**	**=**	**{**	**v**	**x**	**z**	**u**		**}**

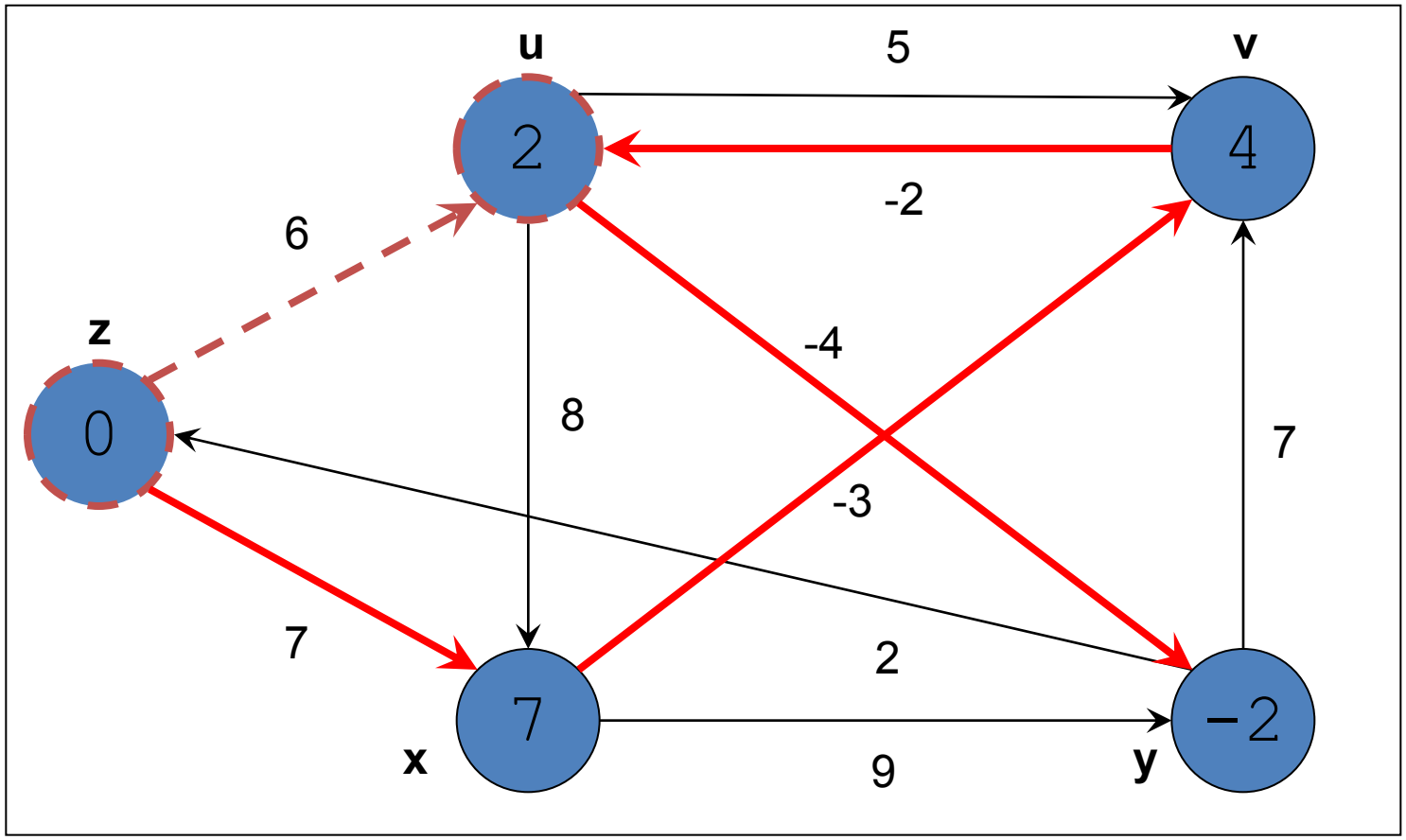
Paso 4.8

Aplicar Relax al Arco (y, z)

Pregunta: **¿ d[z] > d[y] + w(y , z) ?**

Respuesta: NO

Proceso: **No se hace nada.**



Lista de Arcos

(u,v)
(u,x)
(u,y)
(v,u)
(x,v)
(x,y)
(y,v)
(y,z)
(z,u) ←
(z,x)

V	**[**	**]**	**=**	**{**	**u**	**v**	**x**	**y**	**z**	**}**
d	**[**	**]**	**=**	**{**	2	4	7	–2	0	**}**
Π	**[**	**]**	**=**	**{**	**v**	**x**	**z**	**u**		**}**

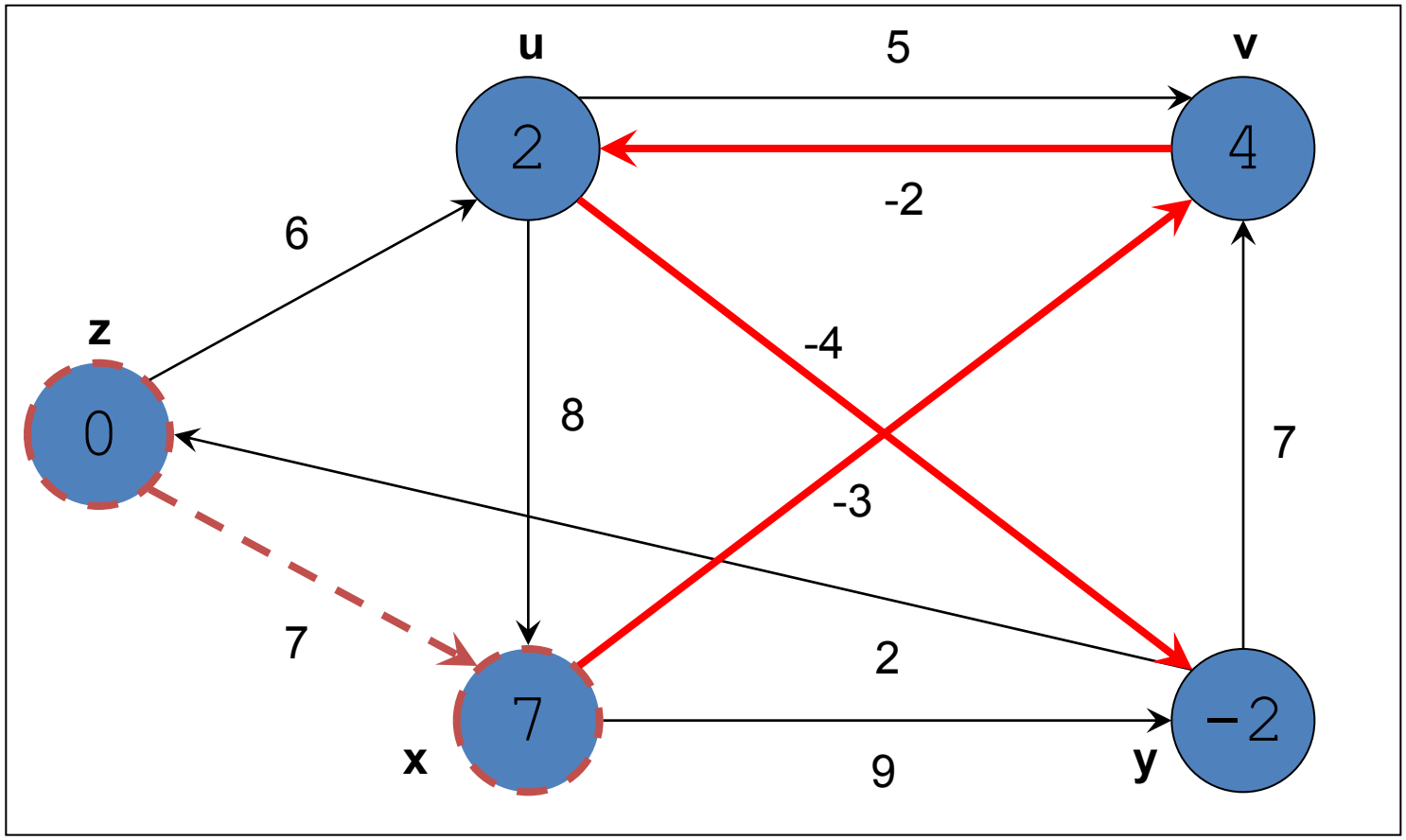
Paso 4.9

Aplicar Relax al Arco (z, u)

Pregunta: **¿ d[u] > d[z] + w(z , u) ?**

Respuesta: NO

Proceso: **No se hace nada.**



Lista de Arcos

(u,v)
(u,x)
(u,y)
(v,u)
(x,v)
(x,y)
(y,v)
(y,z)
(z,u)
(z,x) ←

V	**[**	**]**	**=**	**{**	**u**	**v**	**x**	**y**	**z**	**}**
d	**[**	**]**	**=**	**{**	2	4	7	–2	0	**}**
Π	**[**	**]**	**=**	**{**	**v**	**x**	**z**	**u**		**}**

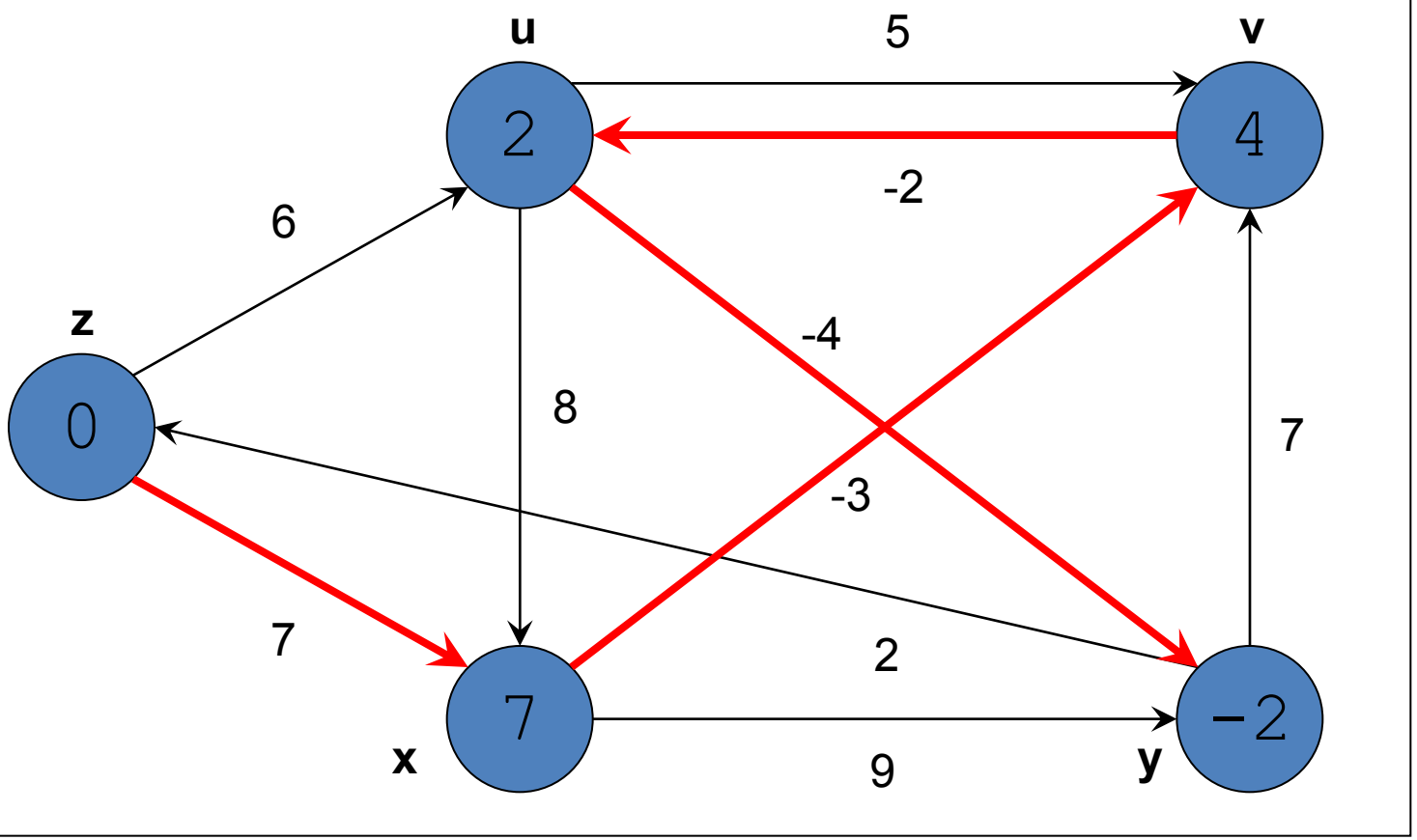
Paso 4.10

Aplicar Relax al Arco (z, x)

Pregunta: **¿ d[x] > d[z] + w(z , x) ?**

Respuesta: NO

Proceso: **No se hace nada.**



Lista de Arcos

(u,v)
(u,x)
(u,y)
(v,u)
(x,v)
(x,y)
(y,v)
(y,z)
(z,u)
(z,x)

V	**[**	**]**	**=**	**{**	**u**	**v**	**x**	**y**	**z**	**}**
d	**[**	**]**	**=**	**{**	2	4	7	–2	0	**}**
Π	**[**	**]**	**=**	**{**	**v**	**x**	**z**	**u**		**}**

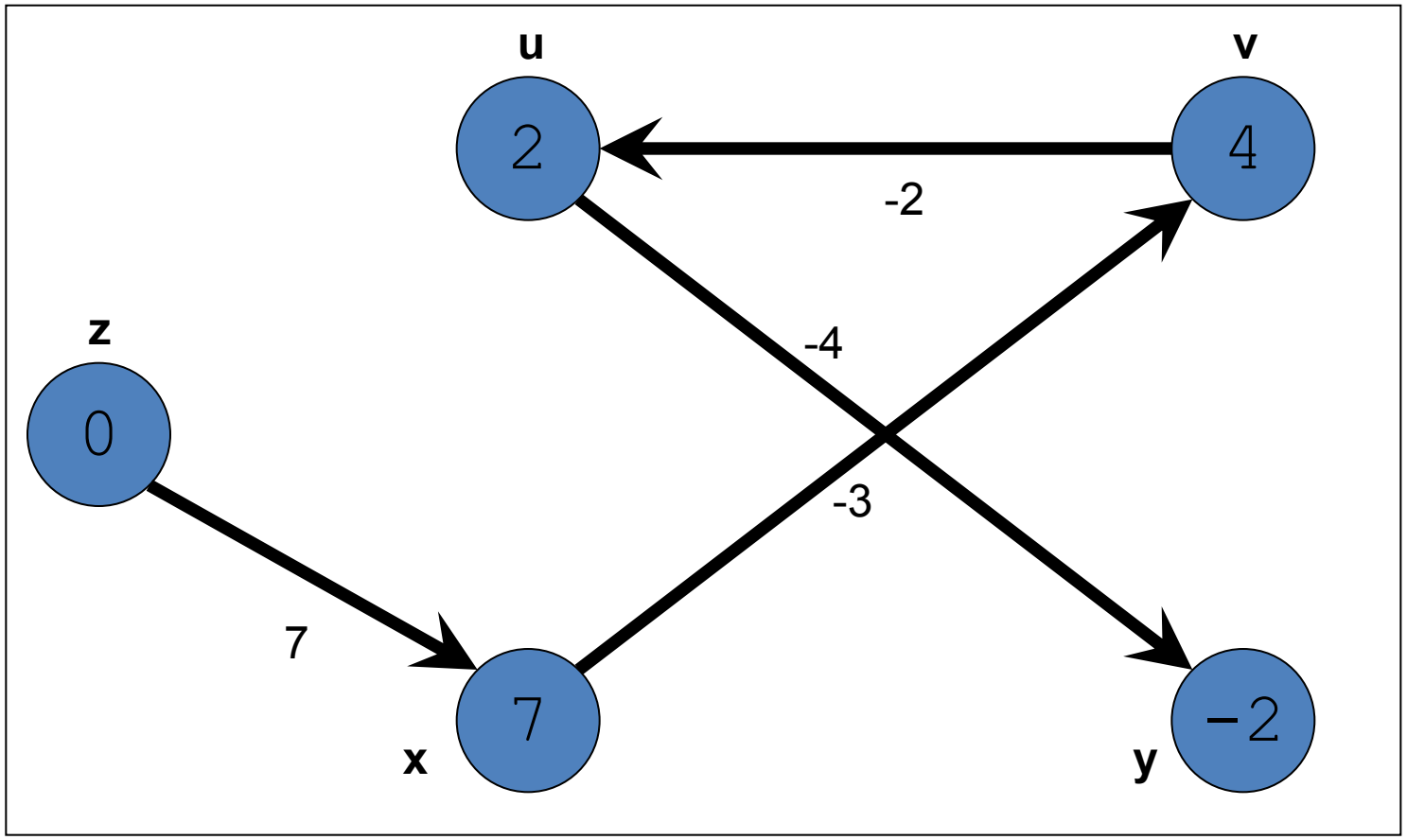
Paso 5.0

Verificar en cada arco que se cumple la condición:

d[Vf] <= d[Vi] + w(Vi , Vf)

Si no se cumple:

=> NO EXISTE SOLUCIÓN.



Lista de Arcos

(u,v)
(u,x)
(u,y)
(v,u)
(x,v)
(x,y)
(y,v)
(y,z)
(z,u)
(z,x)

V	**[**	**]**	**=**	**{**	**u**	**v**	**x**	**y**	**z**	**}**
d	**[**	**]**	**=**	**{**	2	4	7	–2	0	**}**
Π	**[**	**]**	**=**	**{**	**v**	**x**	**z**	**u**		**}**

SOLUCIÓN