Camino más corto: algoritmo de Dijkstra

Rodrigo Maranzana

Algoritmo de Dijkstra: concepto

Algoritmo de Dijkstra

Algoritmo de resolución específico para camino más corto.

- Encuentra siempre la solución óptima.
- La complejidad depende del problema. Pero es muy eficiente.
- No sirve si los pesos de los arcos son negativos

Algoritmo de Dijkstra: pseudocódigo

Pseudo código

Inicializar etiquetas de todos los nodos a infinito

Para cada nodo del grafo no explorado:
    nodo_i = nodo actual

    Para cada vecino del nodo_i:
        nodo_j = vecino actual
        nueva_etiqueta(**vecino_j**) = etiqueta(**nodo_i**) + distancia(**nodo_i**, **nodo_j**)

        Si nueva_etiqueta(**vecino_j**) < etiqueta(**vecino_j**)
            Actualizar etiqueta(**vecino_j**)

Algoritmo en ejemplo básico

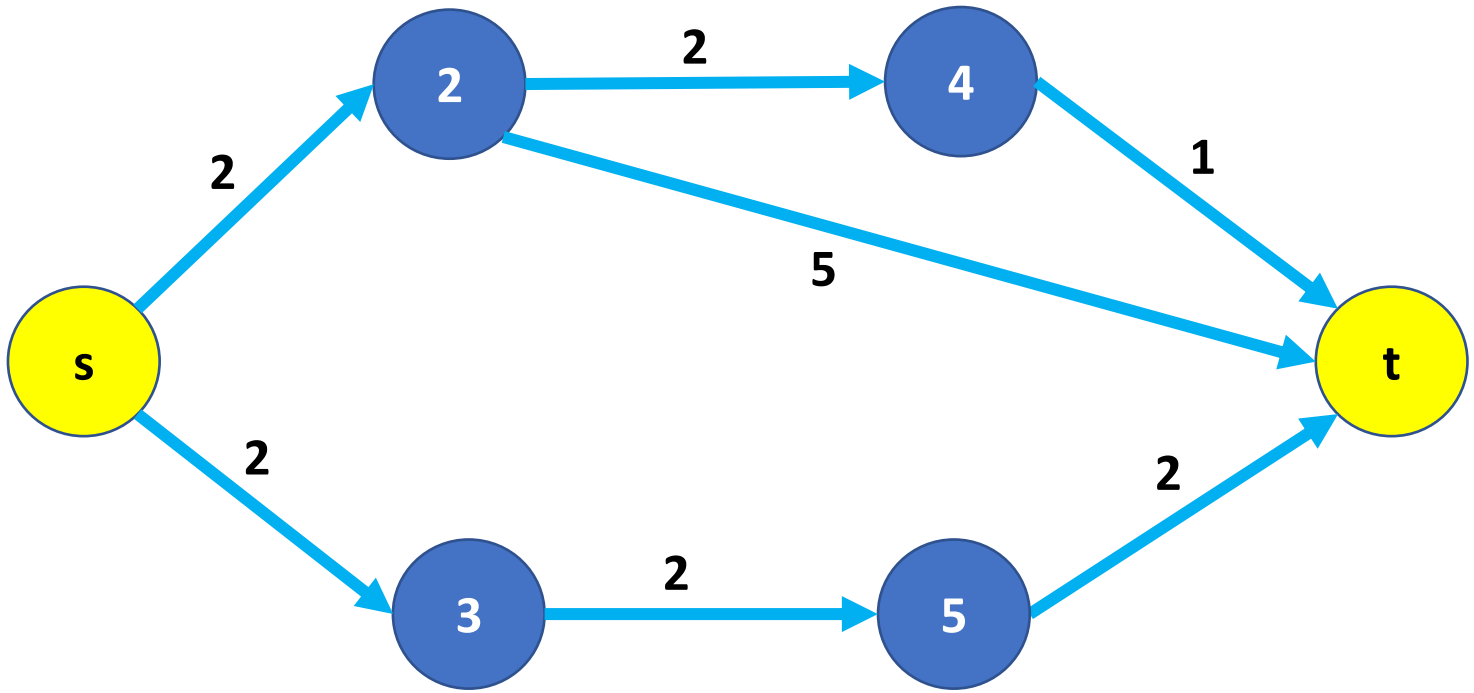

Tabla de precedencias y grafo de actualización

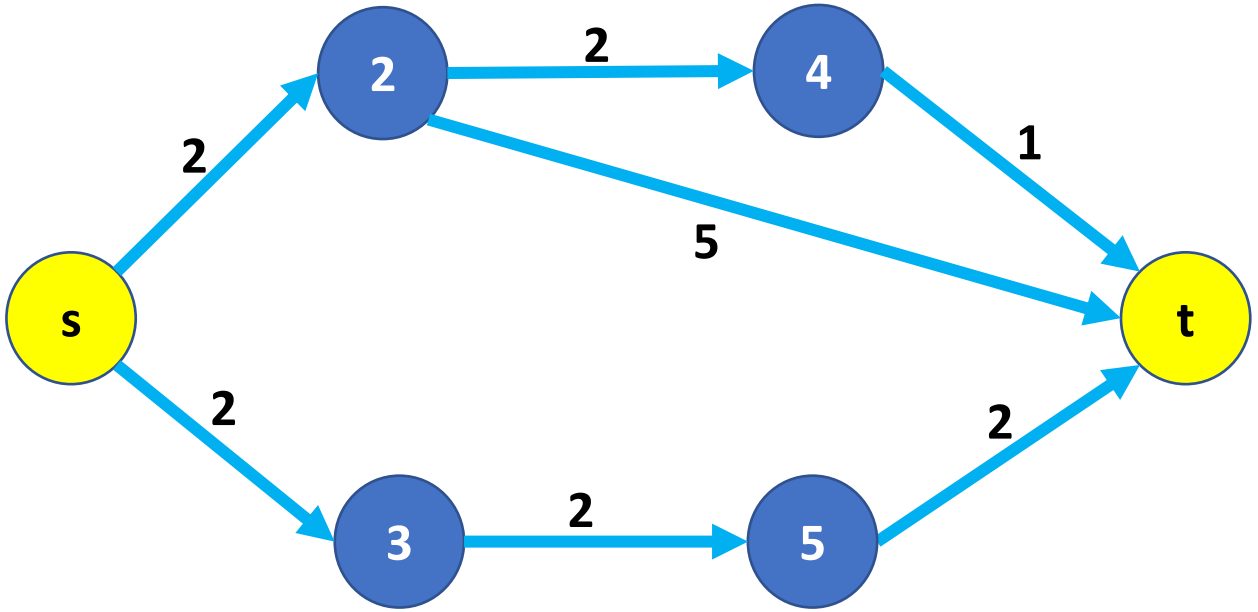

Tabla de precedencias

S	2	3	4	5	t

Inicialización

INICIALIZACIÓN

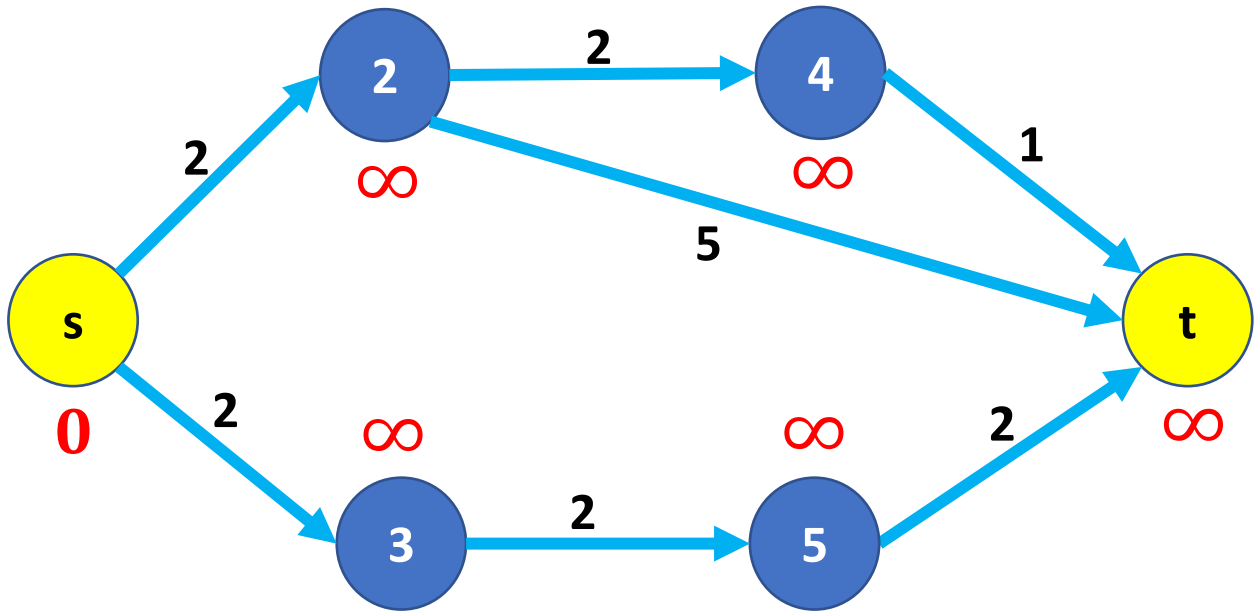

Tabla de precedencias

S	2	3	4	5	t

Iteración 0: vecinos de “s”

VECINOS DE “s”

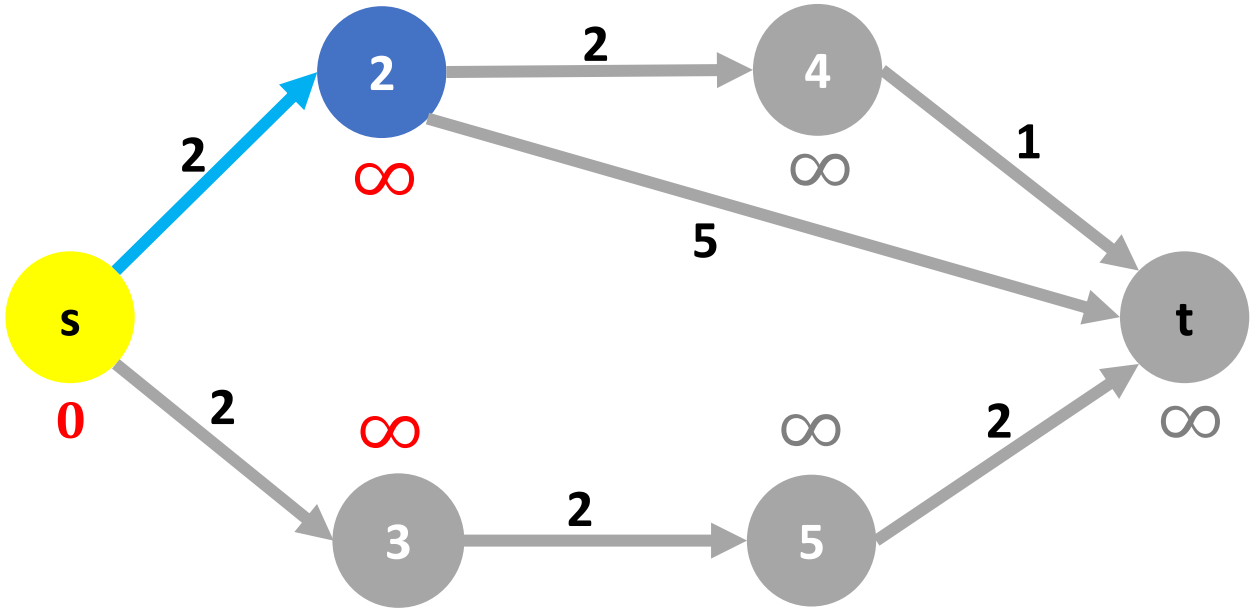

Tabla de precedencias

S	2	3	4	5	t
\					
\					
\					
\					
\					

Iteración 0: vecinos de “s”

VECINOS DE “s”

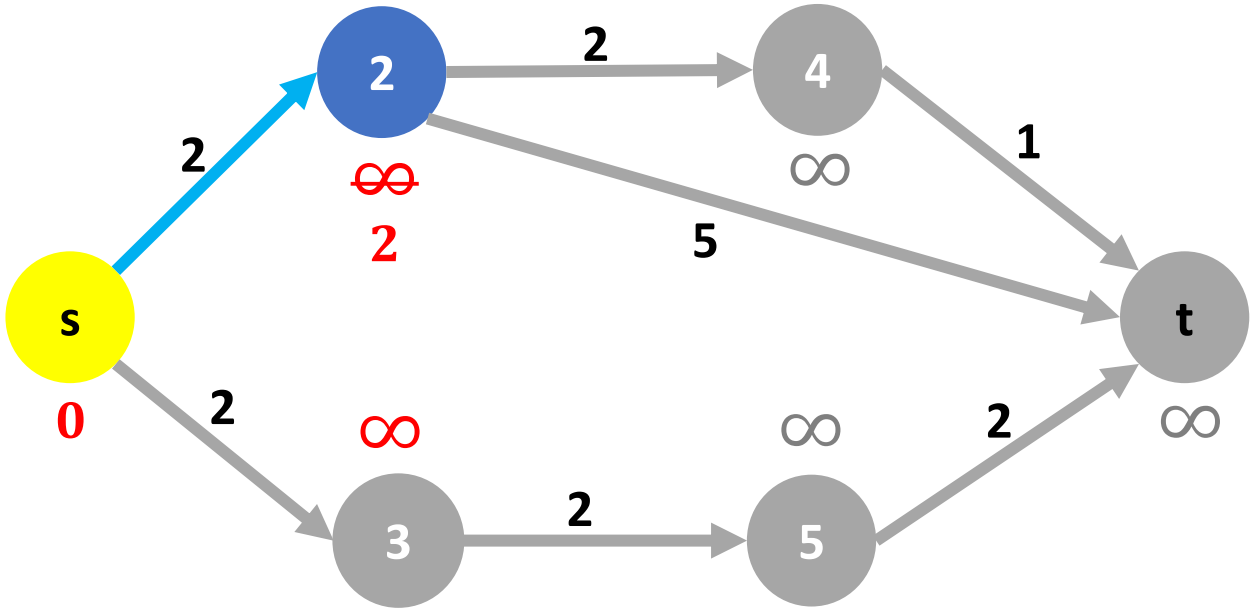

Tabla de precedencias

S	2	3	4	5	t
	S				

Iteración 0: vecinos de “s”

VECINOS DE “s”

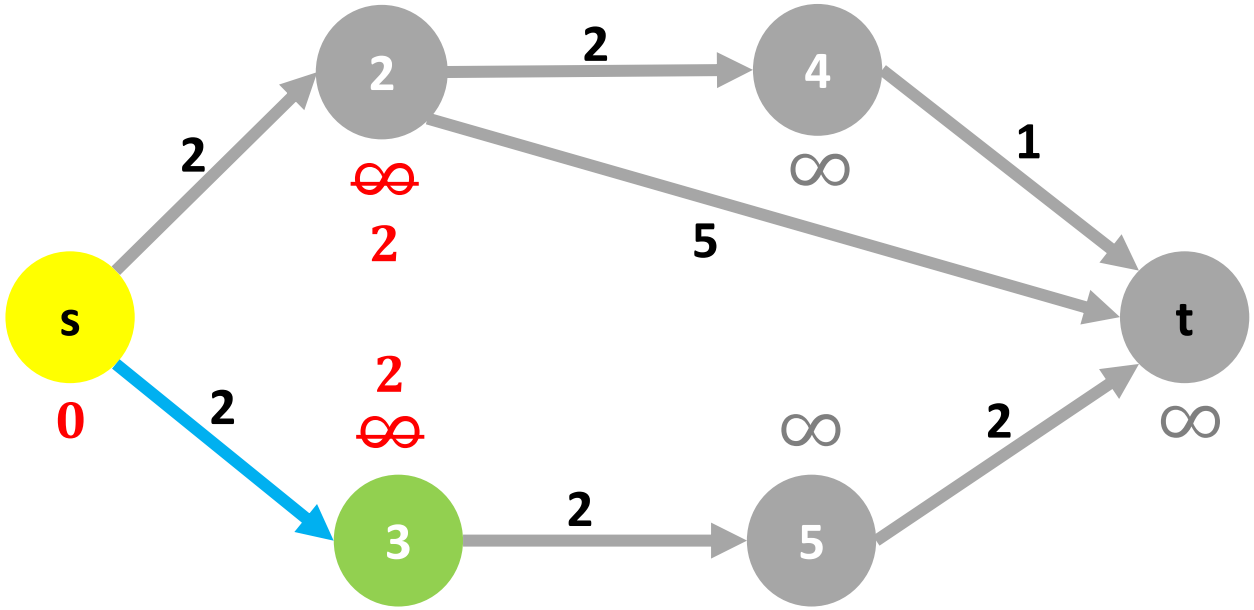

Tabla de precedencias

S	2	3	4	5	t
	S	S			

Iteración 0: vecinos de “s”

S ya explorado

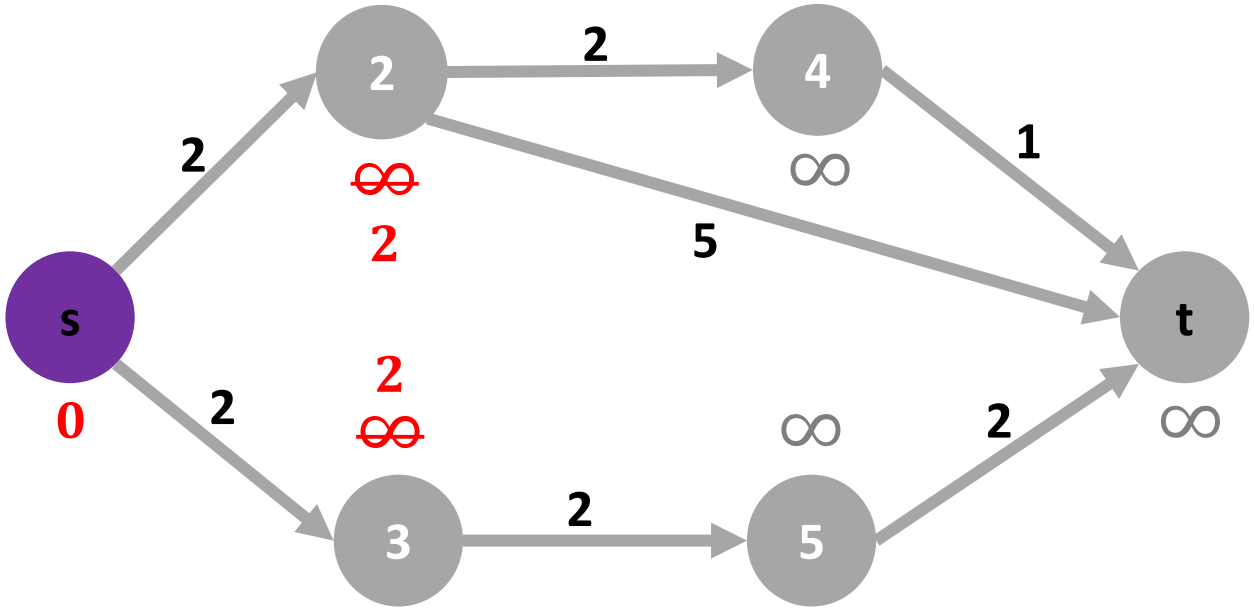

Tabla de precedencias

S	2	3	4	5	t
	S	S			

Iteración 1: vecinos de “2”

VECINOS DE “2”

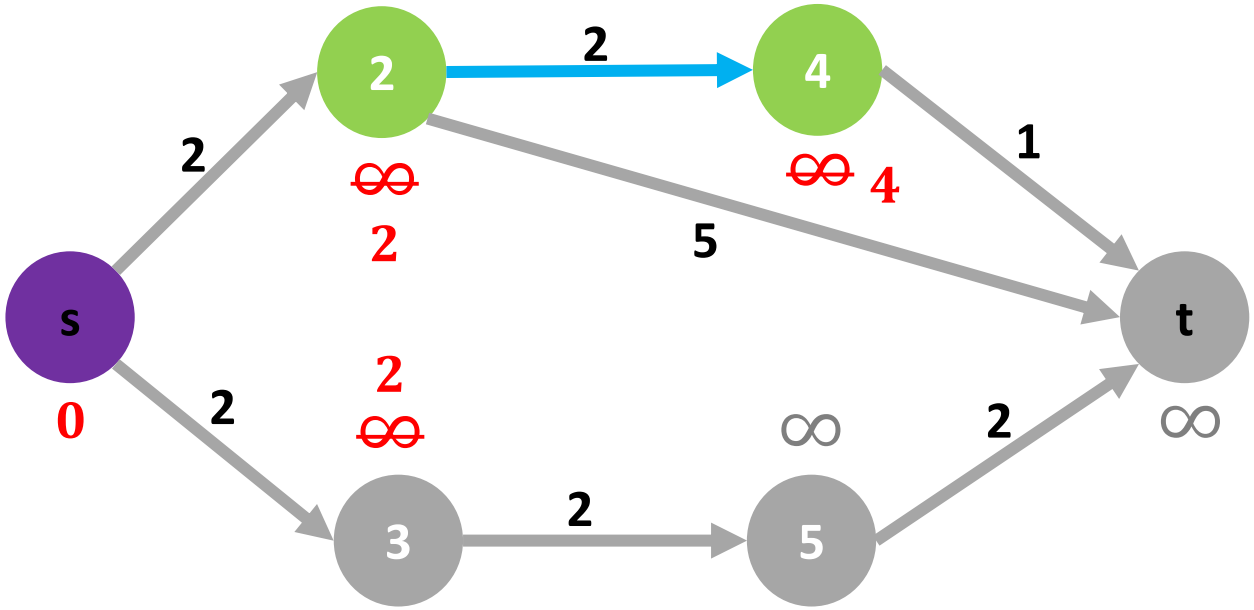

Tabla de precedencias

S	2	3	4	5	t
	s	s	2		

Iteración 1: vecinos de “2”

VECINOS DE “2”

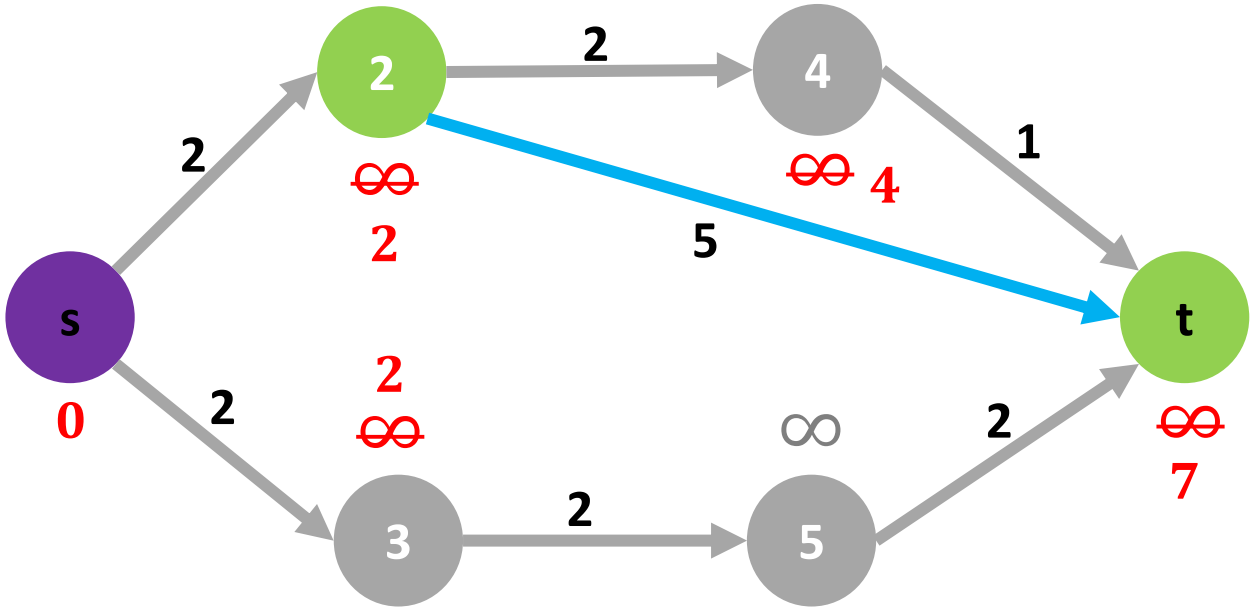

Tabla de precedencias

S	2	3	4	5	t
	s	s	2		2

Iteración 1: vecinos de “2”

“2” ya explorado

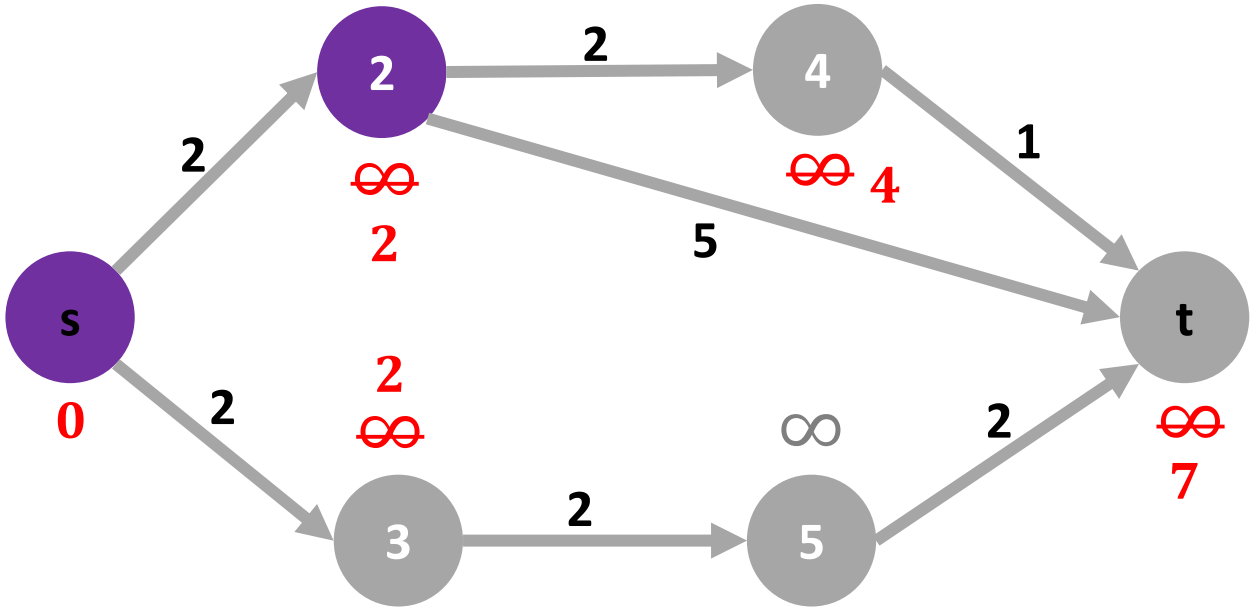

Tabla de precedencias

S	2	3	4	5	t
	s	s	2		2

Iteración 2: vecinos de “3”

VECINOS DE “3”

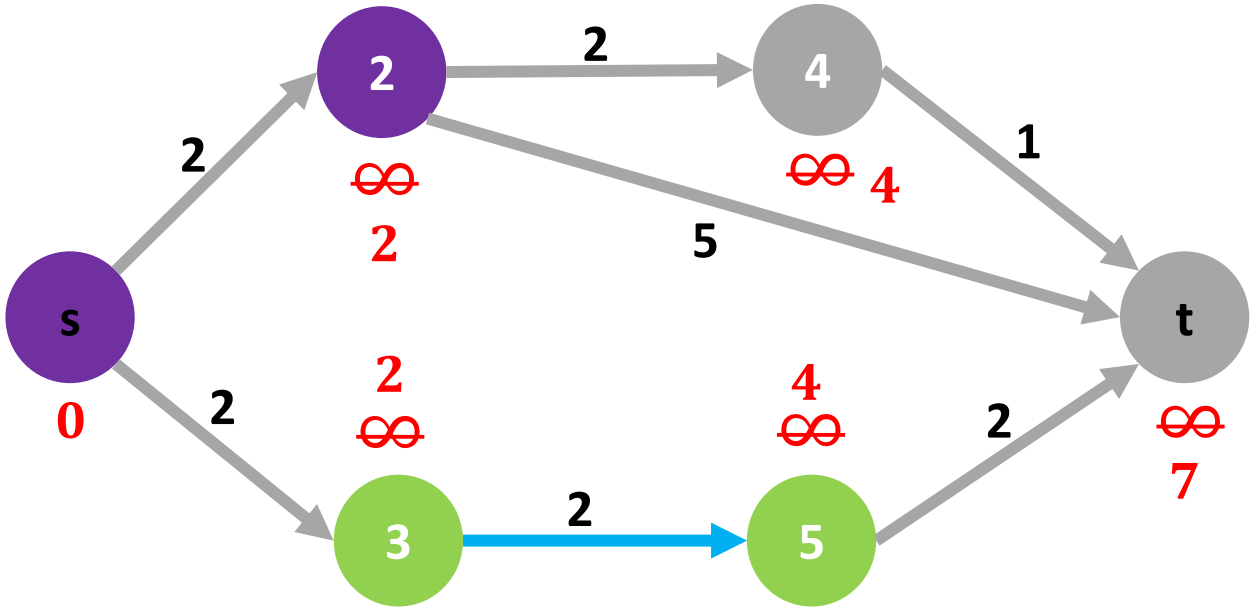

Tabla de precedencias

S	2	3	4	5	t
	s	s	2	3	2

Iteración 2: vecinos de “3”

“3” ya explorado

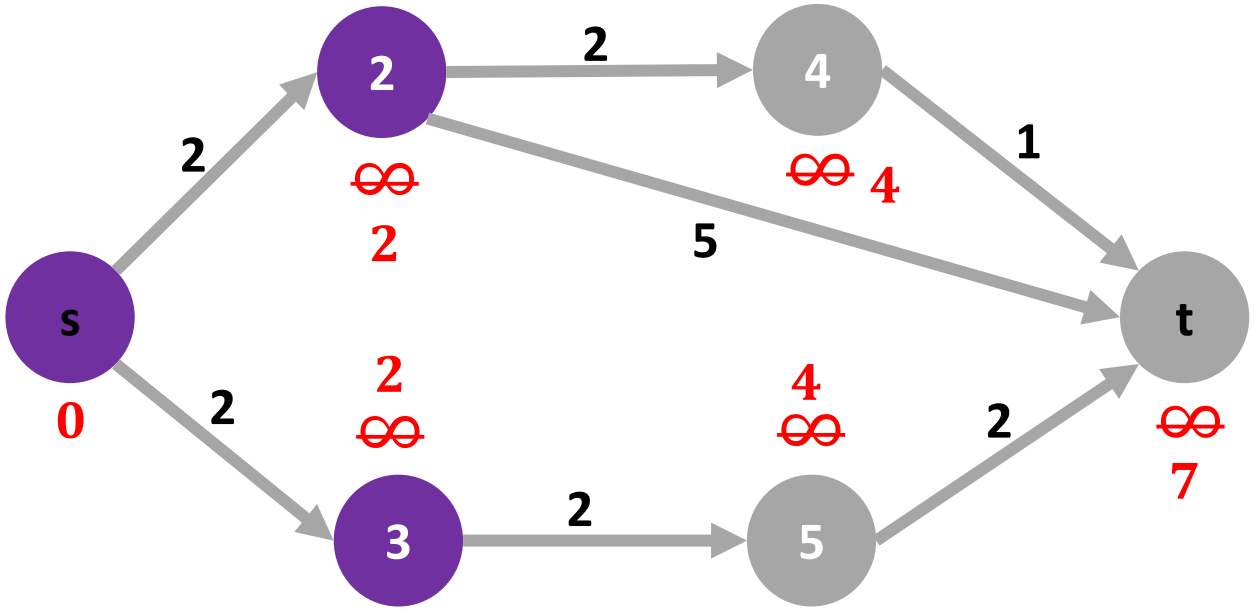

Tabla de precedencias

S	2	3	4	5	t
	s	s	2	3	2

Iteración 3: vecinos de “4”

VECINOS DE “4”

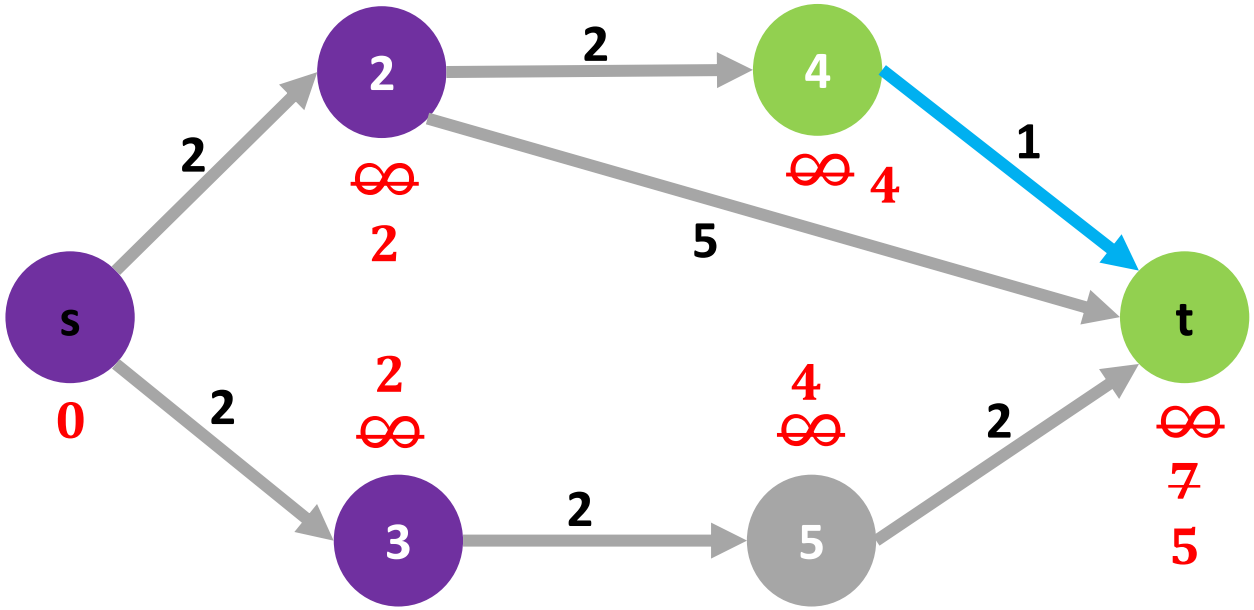

Tabla de precedencias

S	2	3	4	5	t
	s	s	2	3	~~2~~
					4

Iteración 3: vecinos de “4”

“4” ya explorado

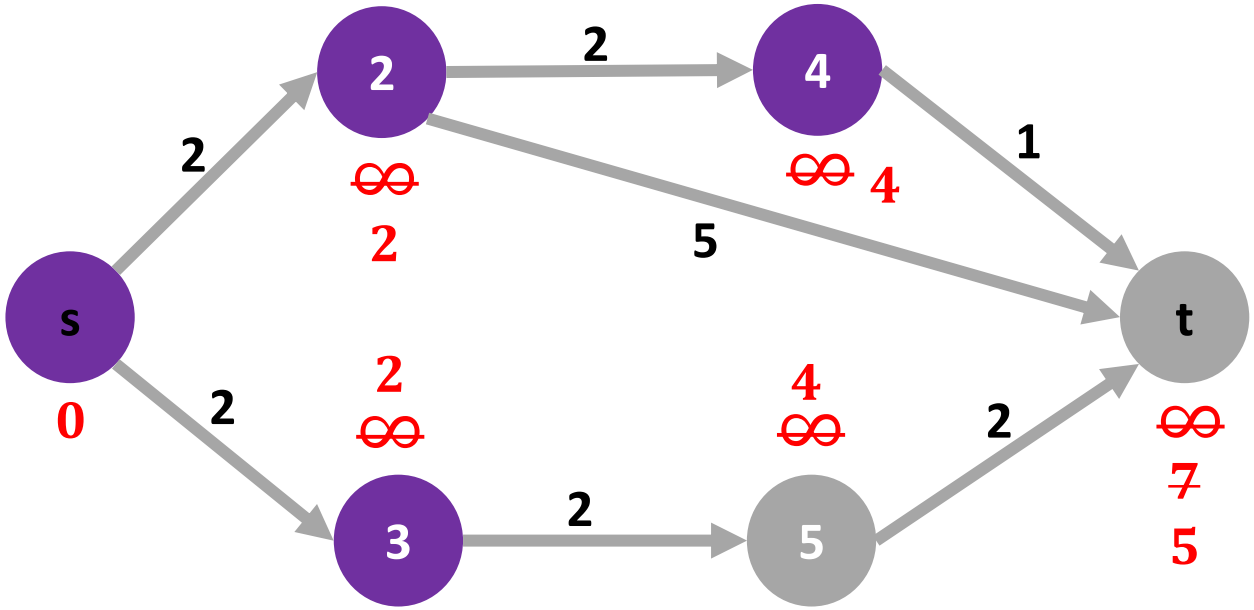

Tabla de precedencias

S	2	3	4	5	t
	s	s	2	3	2
					4

Iteración 4: vecinos de “5”

VECINOS DE “5”

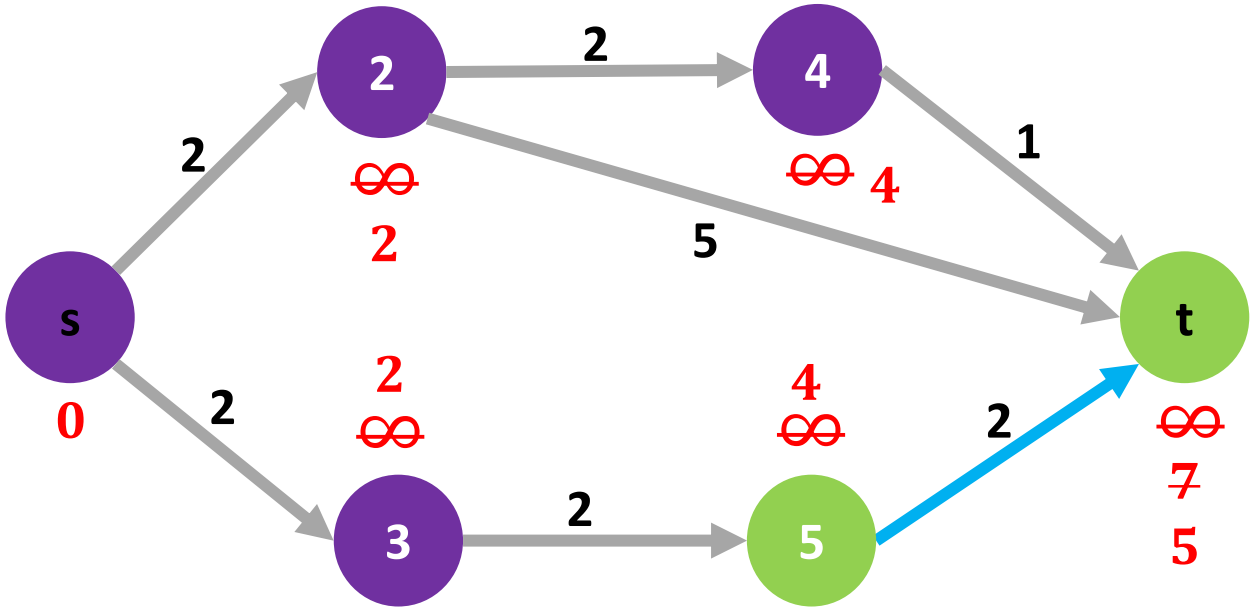

Tabla de precedencias

S	2	3	4	5	t
	s	s	2	3	2
					4

6 No es menor que 5, no actualiza!

Iteración 4: vecinos de “5”

“5” ya explorado

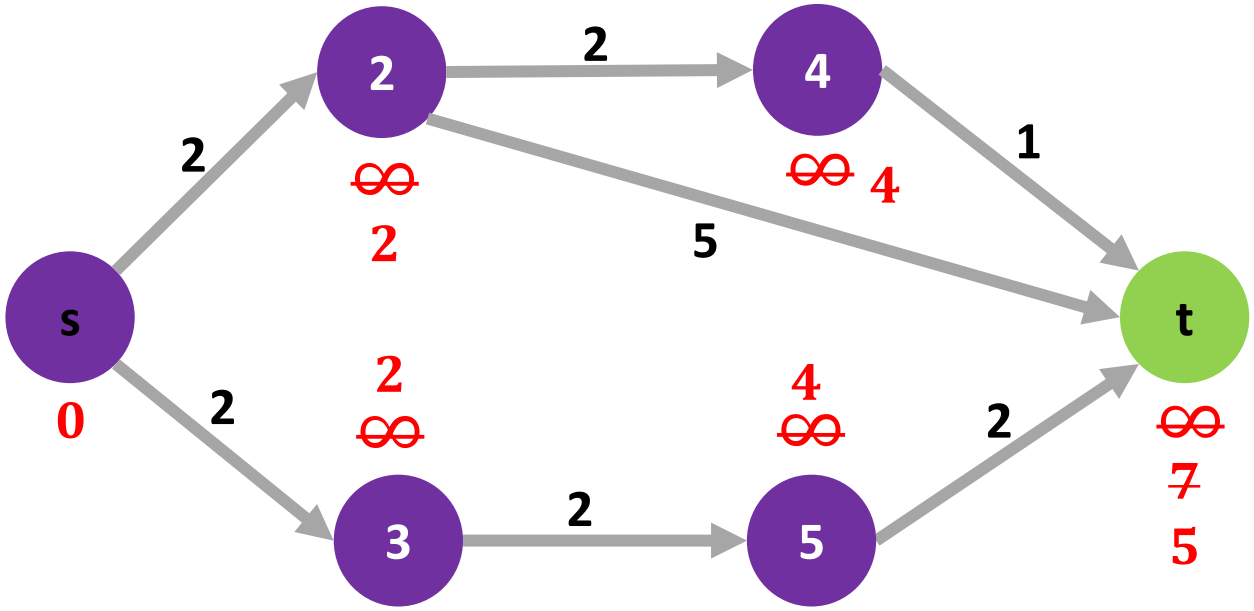

Tabla de precedencias

S	2	3	4	5	t
	s	s	2	3	2
					4

Iteración 5: vecinos de “t”

Nodo “t” final, sin vecinos. Fin del algoritmo

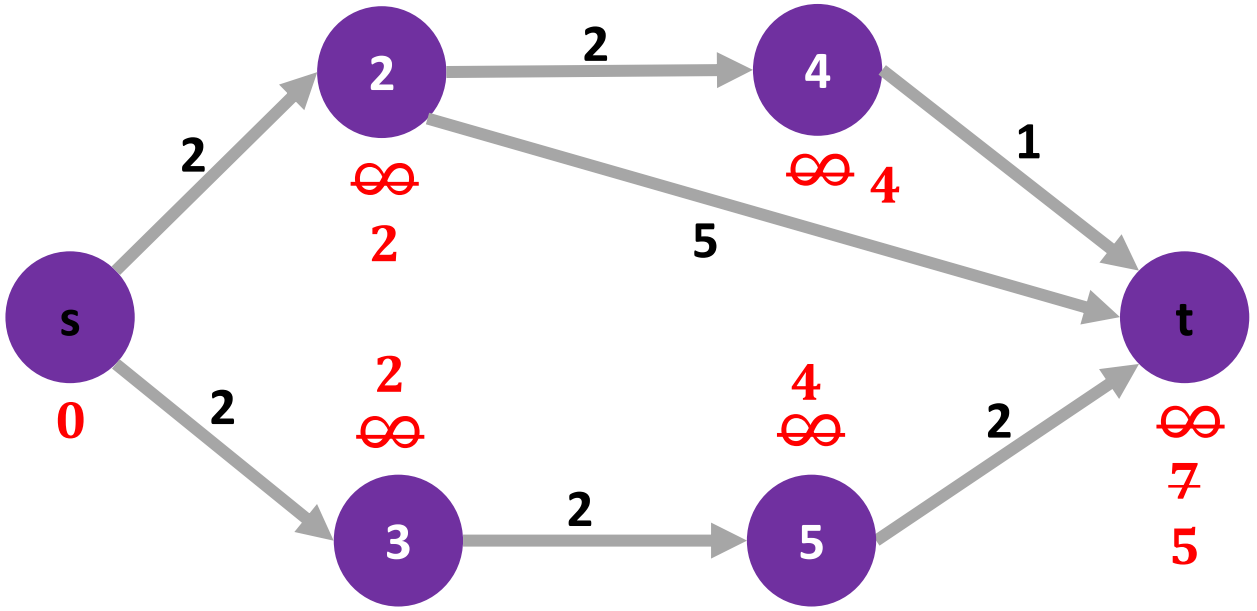

Tabla de precedencias

S	2	3	4	5	t
	s	s	2	3	~~2~~
					4

Distancia mínima s-t

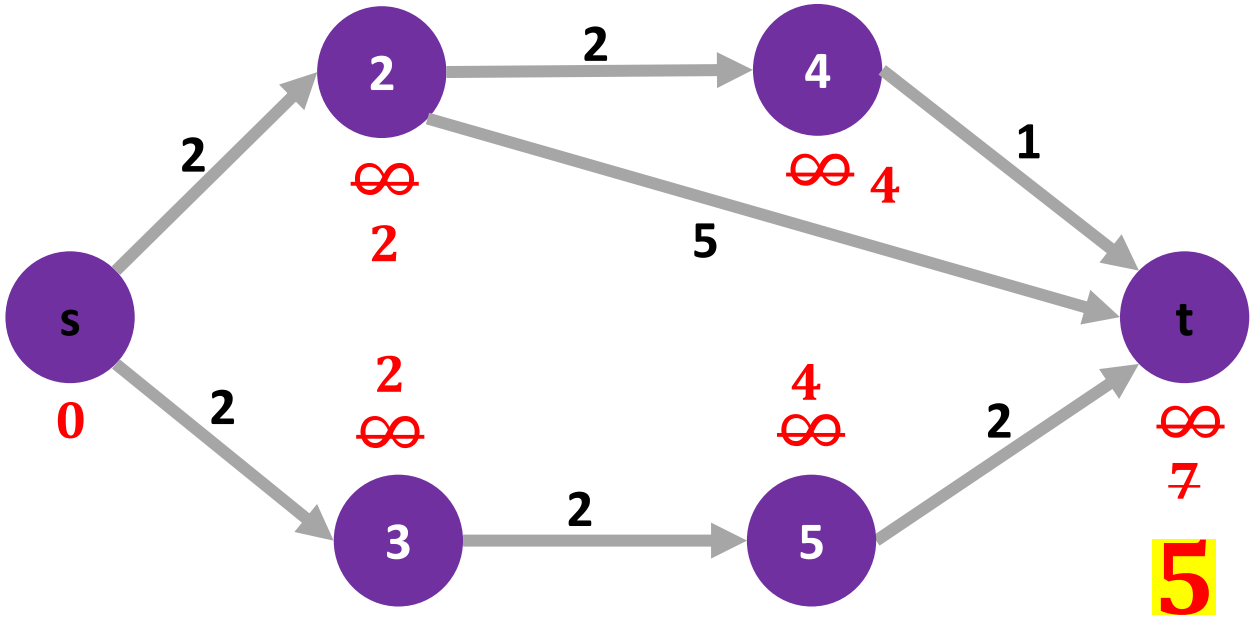

Tabla de precedencias

S	2	3	4	5	t
	s	s	2	3	~~2~~
					4

Distancia mínima s-t

Reconstrucción inversa con tabla de precedencias

Reconstrucción del camino más corto

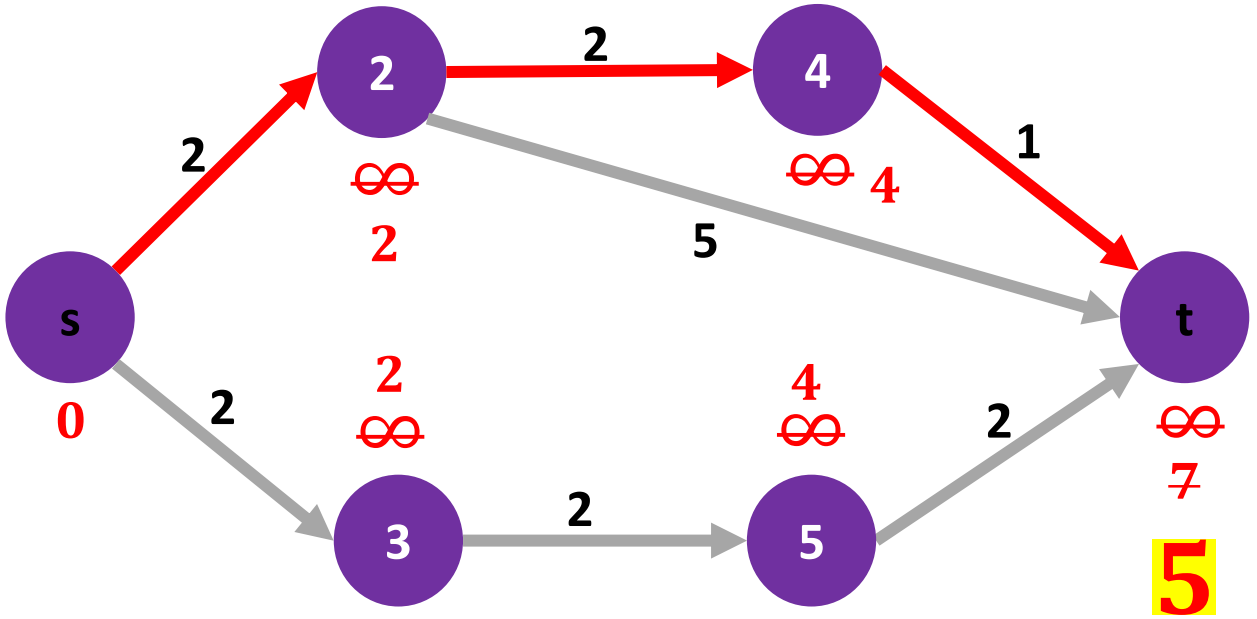

Tabla de precedencias

S	2	3	4	5	t
	s	s	2	3	2
					4

Distancia mínima s-t

Comentarios adicionales sobre Dijkstra

- Dijkstra puede resolver todos los caminos más cortos desde cualquier nodo.
- No tiene heurística.
- Para acelerarlo, agregar heurística. Ej: Algoritmo A*