UTEC
UNIVERSIDAD DE INGENIERÍA
Y TECNOLOGÍA

Welcome to Algorithms and Data Structures! - CS2100

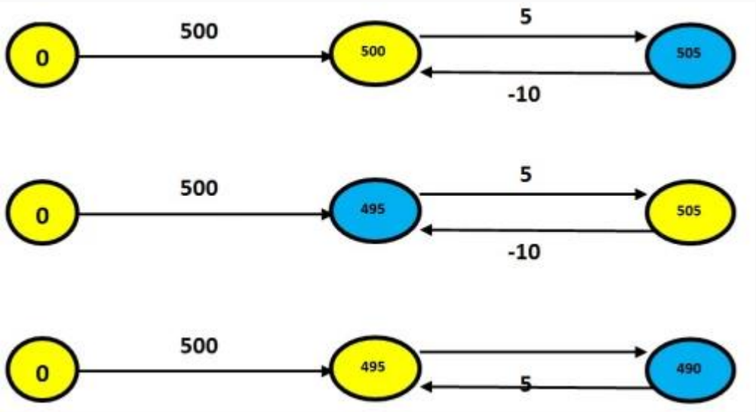
Floyd Warshall

Es un algoritmo de búsqueda del camino más corto en grafos para todo par de vértices que funciona tanto con aristas positivas como negativas (pero no con ciclos negativos)

Un ciclo negativo, es un ciclo en el cual la suma de sus aristas es un valor negativo. En dichos casos Floyd Warshall puede ser usado para detectar dichos ciclos

Es un ejemplo de programación dinámica que se ejecutará siempre en $O(|V|^3)$

Este algoritmo suele utilizarse con grafos densos dirigidos



Floyd Warshall

https://www.youtube.com/watch?v=4OQeCuLYj-4

Se irá iterando sobre filas y columnas.

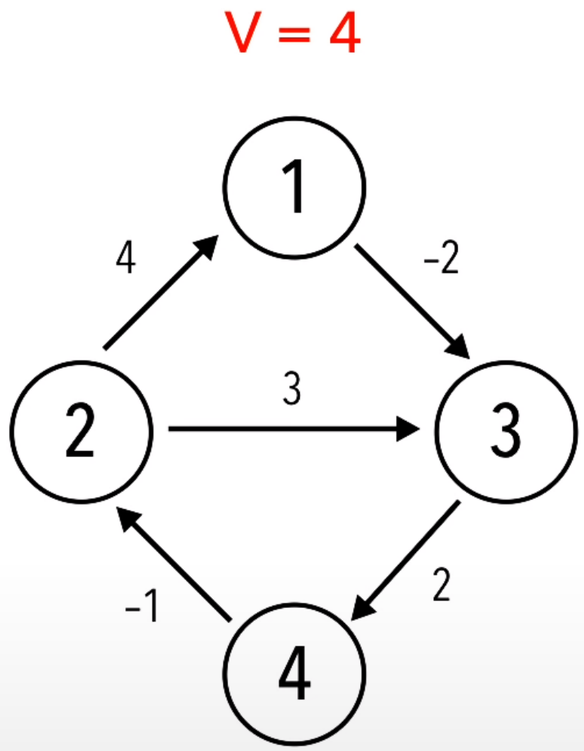
Durante las iteraciones, se irán operando (sumando) cada elemento de cada fila y columna y se comparará con su elemento intersección.

Se reemplazará el elemento intersección por el resultado de la operación, solo cuando esta es menor.

```
let V = number of vertices in graph
let dist = V × V array of minimum distances initialized to ∞
for each vertex v
   dist [v][v] ← 0
for each edge (u,v)
   dist [u][v] ← weight(u,v)
for k from 1 to V
   for i from 1 to V
      for j from 1 to V
         if dist [i][j] > dist [i][k] + dist [k][j]
            dist [i][j] ← dist [i][k] + dist [k][j]
         end if
```

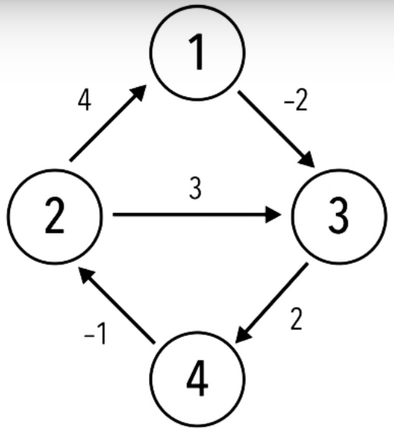
Floyd - Warshall

```
let V = number of vertices in graph
let dist = V × V array of minimum distances
for each vertex v
   dist [v][v] ← 0
for each edge (u,v)
   dist [u][v] ← weight(u,v)
for k from 1 to V
   for i from 1 to V
      for j from 1 to V
         if dist [i][j] > dist [i][k] + dist [k][j]
            dist [i][j] ← dist [i][k] + dist [k][j]
         end if
```



Floyd - Warshall

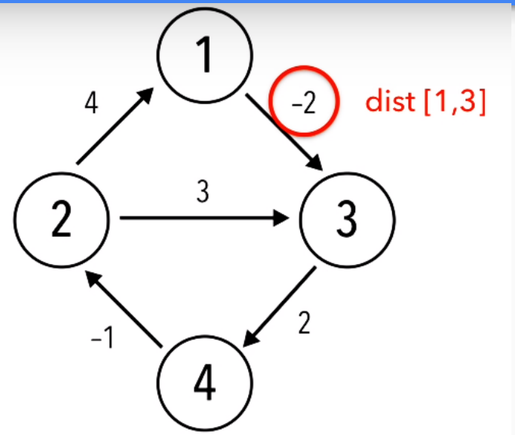
```
let V = number of vertices in graph
let dist = V × V array of minimum distances
for each vertex v
  dist [v][v] ← 0
for each edge (u,v)
  dist [u][v] ← weight(u,v)
for k from 1 to V
  for i from 1 to V
    for j from 1 to V
      if dist [i][j] > dist [i][k] + dist [k][j]
        dist [i][j] ← dist [i][k] + dist [k][j]
      end if
```



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

Floyd - Warshall

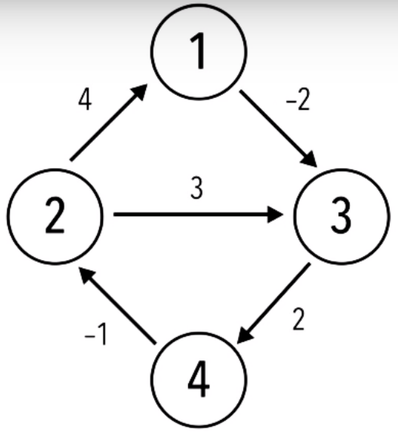
```
let V = number of vertices in graph
let dist = V × V array of minimum distances
for each vertex v
   dist [v][v] ← 0
for each edge (u,v)
   dist [u][v] ← weight(u,v)
for k from 1 to V
   for i from 1 to V
      for j from 1 to V
         if dist [i][j] > dist [i][k] + dist [k][j]
            dist [i][j] ← dist [i][k] + dist [k][j]
         end if
```



	1	2	3	4
1	0		-2	
2		0		
3			0	
4				0

Floyd - Warshall

```
let V = number of vertices in graph
let dist = V × V array of minimum distances
for each vertex v
   dist [v][v] ← 0
for each edge (u,v)
   dist [u][v] ← weight(u,v)
for k from 1 to V
   for i from 1 to V
      for j from 1 to V
         if dist [i][j] > dist [i][k] + dist [k][j]
            dist [i][j] ← dist [i][k] + dist [k][j]
         end if
```

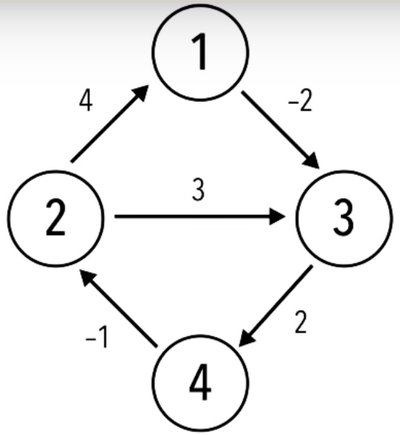


	1	2	3	4
1	0		-2	
2	4	0	3	
3			0	2
4		-1		0

Floyd - Warshall

k = 1 2 3 4
i = 1 2 3 4
j = 1 2 3 4

if dist [i][j] > dist [i][k] + dist [k][j]
dist [i][j] ← dist [i][k] + dist [k][j]

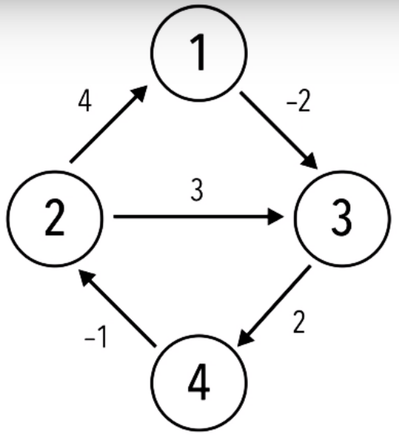


	1	2	3	4
1	0		-2	
2	4	0	3	
3			0	2
4		-1		0

Floyd - Warshall

k = 1 2 3 4
i = 1 2 3 4
j = 1 2 3 4

dist [i][j] > dist [i][k] + dist [k][j]

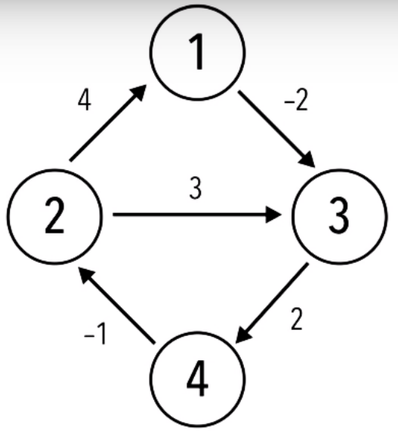


	1	2	3	4
1	0		-2	
2	4	0	3	
3			0	2
4		-1		0

Floyd - Warshall

k = 1 2 3 4
i = 1 2 3 4
j = 1 2 3 4

dist [i][j] > dist [i][k] + dist [k][j]
dist [1][1] > dist [1][1] + dist [1][1]
0 > 0 + 0
X 0 > 0

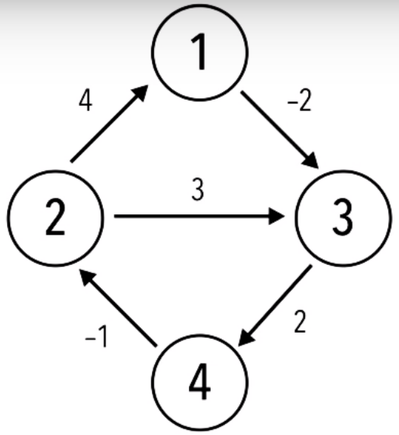


	1	2	3	4
1	0		-2	
2	4	0	3	
3			0	2
4		-1		0

Floyd - Warshall

k = 1 2 3 4
i = 1 2 3 4
j = 1 2 3 4

dist [i][j] > dist [i][k] + dist [k][j]
dist [1][2] > dist [1][1] + dist [1][2]
∞ > 0 + ∞
X ∞ > ∞

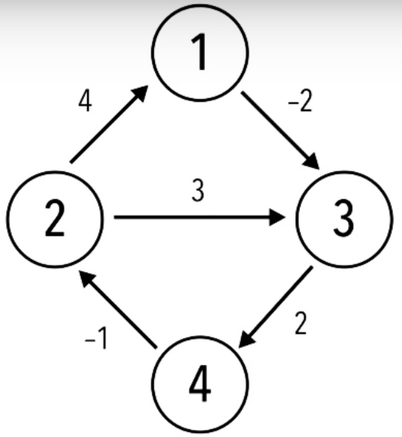


	1	2	3	4
1	0		-2	
2	4	0	3	
3			0	2
4		-1		0

Floyd - Warshall

k = 1 2 3 4
i = 1 2 3 4
j = 1 2 3 4

dist [i][j] > dist [i][k] + dist [k][j]
dist [1][3] > dist [1][1] + dist [1][3]
−2 > 0 + −2
X −2 > −2

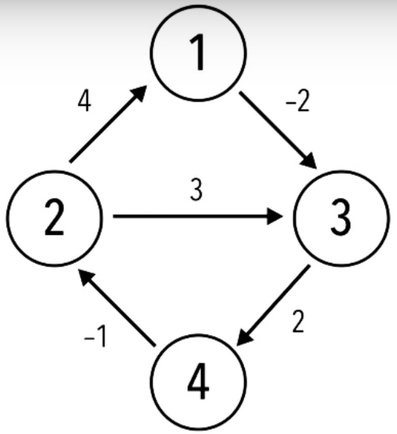


	1	2	3	4
1	0		-2	
2	4	0	3	
3			0	2
4		-1		0

Floyd - Warshall

k = 1 2 3 4
i = 1 2 3 4
j = 1 2 3 4

dist [i][j] > dist [i][k] + dist [k][j]
dist [1][4] > dist [1][1] + dist [1][4]
∞ > 0 + ∞
X ∞ > ∞

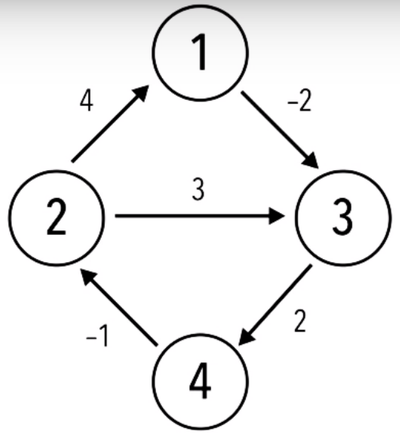


	1	2	3	4
1	0		-2	
2	4	0	3	
3			0	2
4		-1		0

Floyd - Warshall

k = 1 2 3 4
i = 1 2 3 4
j = 1 2 3 4

dist [i][j] > dist [i][k] + dist [k][j]
dist [2][3] > dist [2][1] + dist [1][3]
3 > 4 + -2
3 > 2

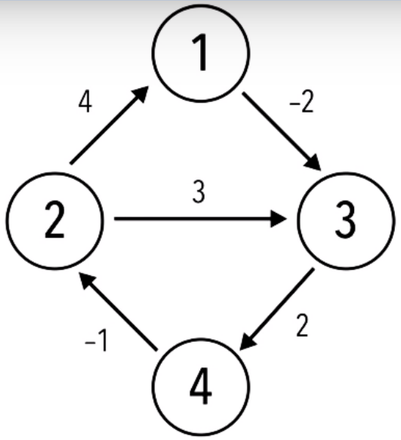


	1	2	3	4
1	0		-2	
2	4	0	2	
3			0	2
4		-1		0

Floyd - Warshall

k = 1 2 3 4
i = 1 2 3 4
j = 1 2 3 4

dist [i][j] > dist [i][k] + dist [k][j]
dist [4][1] > dist [4][2] + dist [2][1]
∞ > –1 + 4
∞ > 3

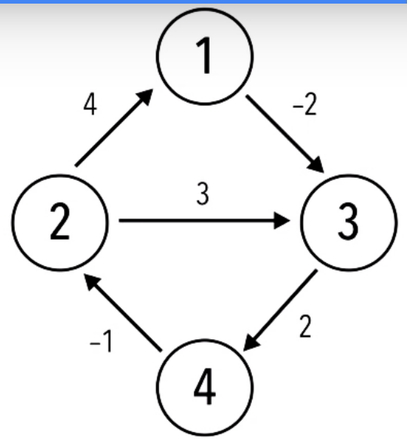


	1	2	3	4
1	0		-2	
2	4	0	2	
3			0	2
4	3	-1		0

Floyd - Warshall

k = 1 2 3 4
i = 1 2 3 4
j = 1 2 3 4

dist [i][j] > dist [i][k] + dist [k][j]
dist [4][3] > dist [4][2] + dist [2][3]
∞ > –1 + 2
∞ > 1

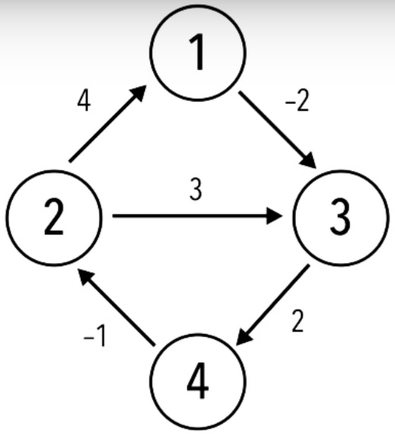


	1	2	3	4
1	0		-2	
2	4	0	2	
3			0	2
4	3	-1	1	0

Floyd - Warshall

k = 1 2 3 4
i = 1 2 3 4
j = 1 2 3 4

dist [i][j] > dist [i][k] + dist [k][j]
dist [1][4] > dist [1][3] + dist [3][4]
∞ > –2 + 2
∞ > 0

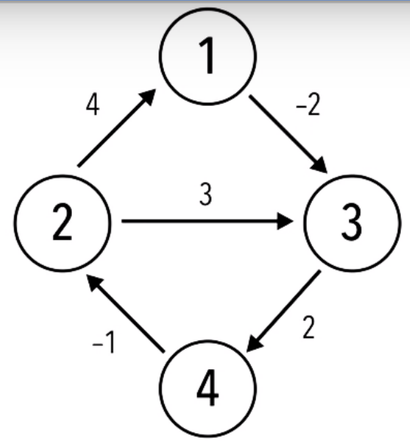


	1	2	3	4
1	0		–2	0
2	4	0	2	
3			0	2
4	3	-1	1	0

Floyd - Warshall

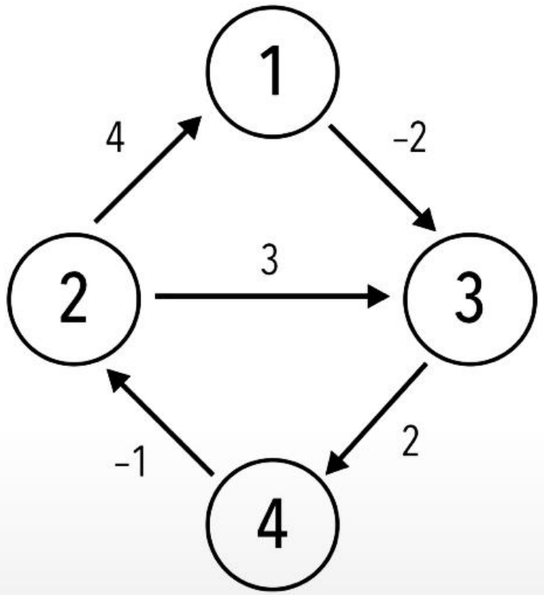
k = 1 2 3 4
i = 1 2 3 4
j = 1 2 3 4

dist [i][j] > dist [i][k] + dist [k][j]
dist [2][4] > dist [2][3] + dist [3][4]
∞ > 2 + 2
∞ > 4



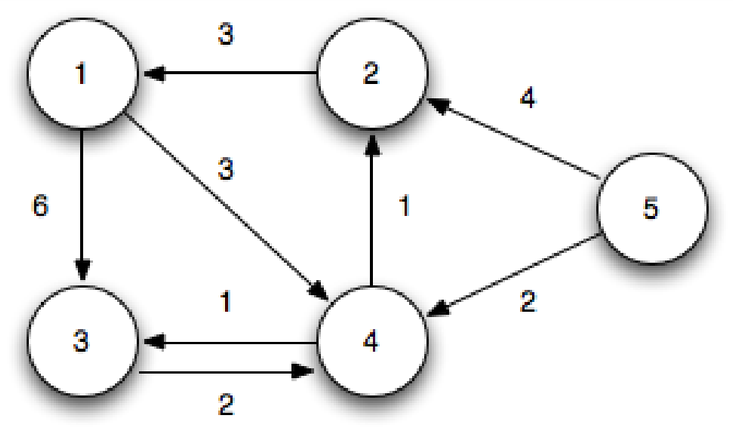
	1	2	3	4
1	0		-2	0
2	4	0	2	4
3			0	2
4	3	-1	1	0

Floyd - Warshall



	1	2	3	4
1	0	-1	-2	0
2	4	0	2	4
3	5	1	0	2
4	3	-1	1	0

Floyd Warshall



Se irá iterando sobre filas y columnas.

Durante las iteraciones, se irán operando (sumando) cada elemento de cada fila y columna y se comparará con su elemento intersección como veremos en el ejemplo.

Se reemplazará el elemento intersección por el resultado de la operación, solo cuando esta es menor.

Nota: Tomaremos **!<** como no menor

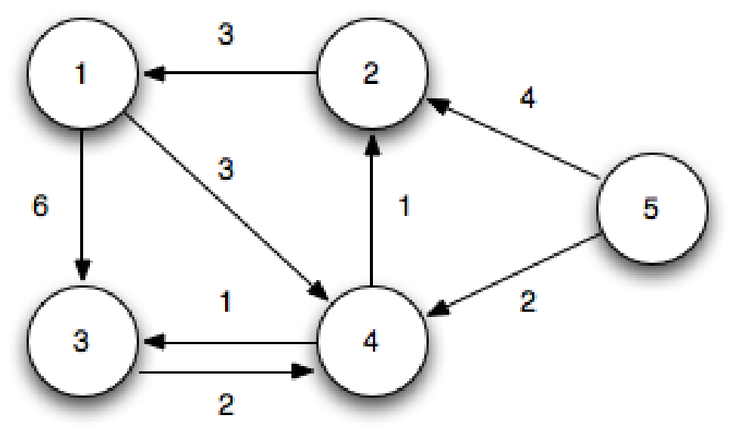
Matriz de Distancias

	1	2	3	4	5
1	0	∞	6	3	∞
2	3	0	∞	∞	∞
3	∞	∞	0	2	∞
4	∞	1	1	0	∞
5	∞	4	∞	2	0

Matriz de Recorridos

	1	2	3	4	5
1	0	2	3	4	5
2	1	0	3	4	5
3	1	2	0	4	5
4	1	2	3	0	5
5	1	2	3	4	0

Floyd Warshall



Columna 1.2 - Fila 1:
3 + ∞ !< 0
3 + 6 < ∞ (Se reemplaza)
3 + 3 < ∞ (Se reemplaza)
3 + ∞ !< ∞

Columna 1.3 - Fila 1:
∞ + ∞ !< ∞
∞ + 6 !< 0
∞ + 3 !< 2
∞ + ∞ !< ∞

Columna 1.4 - Fila 1:
∞ + ∞ !< 1
∞ + 6 !< 1
∞ + 3 !< 0
∞ + ∞ !< ∞

Columna 1.5 - Fila 1:
∞ + ∞ !< 4
∞ + 6 !< ∞
∞ + 3 !< 2
∞ + ∞ !< 0

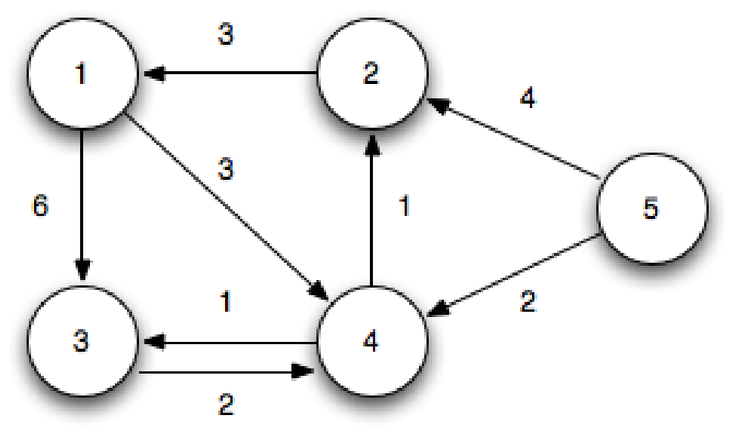
Matriz de Distancias

	1	2	3	4	5
1	0	∞	6	3	∞
2	3	0	∞	∞	∞
3	∞	∞	0	2	∞
4	∞	1	1	0	∞
5	∞	4	∞	2	0

Matriz de Recorridos

	1	2	3	4	5
1	0	2	3	4	5
2	1	0	3	4	5
3	1	2	0	4	5
4	1	2	3	0	5
5	1	2	3	4	0

Floyd Warshall



Columna 2.1 - Fila 2:
∞ + 3 !< 0
∞ + 9 !< 6
∞ + 6 !< 3
∞ + ∞ !< ∞

Columna 2.3 - Fila 2:
∞ + 3 !< ∞
∞ + 9 !< 0
∞ + 6 !< 2
∞ + ∞ !< ∞

Columna 2.4 - Fila 2:
1 + 3 < ∞ (Se reemplaza)
1 + 9 !< 1
1 + 6 !< 0
1 + ∞ !< ∞

Columna 2.5 - Fila 2:
4 + 3 < ∞ (Se reemplaza)
4 + 9 < ∞ (Se reemplaza)
4 + 6 !< 2
4 + ∞ !< 0

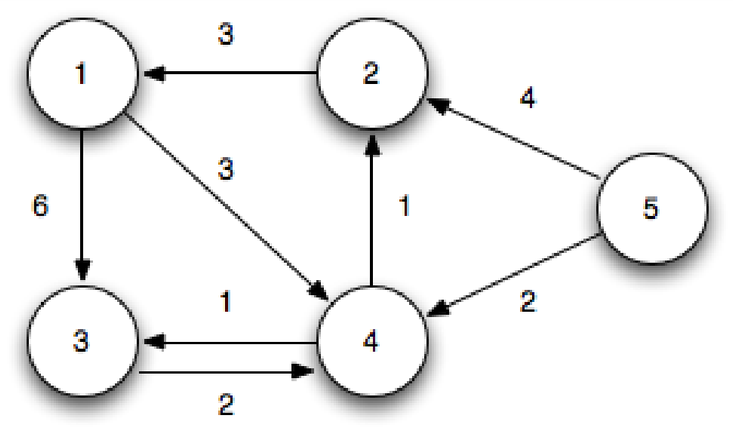
Matriz de Distancias

	1	2	3	4	5
1	0	∞	6	3	∞
2	3	0	9	6	∞
3	∞	∞	0	2	∞
4	∞	1	1	0	∞
5	∞	4	∞	2	0

Matriz de Recorridos

	1	2	3	4	5
1	-	2	3	4	5
2	1	-	1	1	5
3	1	2	-	4	5
4	1	2	3	-	5
5	1	2	3	4	-

Floyd Warshall



Columna 3.1 - Fila 3:
6 + ∞ !< 0
6 + ∞ !< 0
6 + 2 !< 3
6 + ∞ !< ∞

Columna 3.2 - Fila 3:
9 + ∞ !< 3
9 + ∞ !< 0
9 + 2 !< 6
9 + ∞ !< ∞

Columna 3.4 - Fila 3:
1 + ∞ <! 4
1 + ∞ !< 1
1 + 2 !< 0
1 + ∞ !< ∞

Columna 3.5 - Fila 3:
13 + ∞ !< 7
13 + ∞ !< 4
13 + 2 !< 2
13 + ∞ !< 0

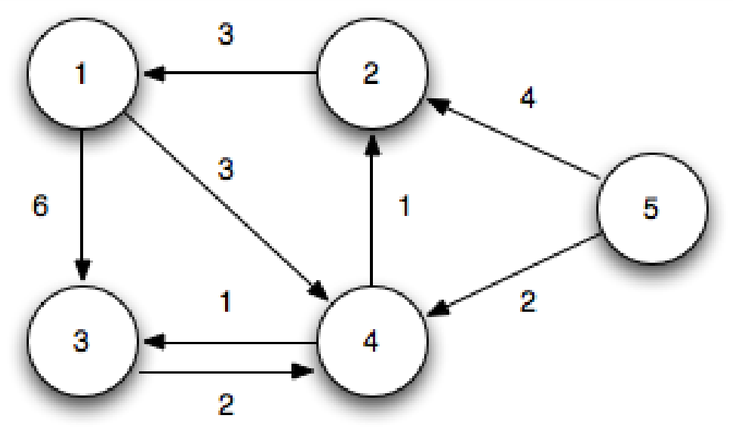
Matriz de Distancias

	1	2	3	4	5
1	0	∞	6	3	∞
2	3	0	9	6	∞
3	∞	∞	0	2	∞
4	4	1	1	0	∞
5	7	4	13	2	0

Matriz de Recorridos

	1	2	3	4	5
1	-	2	3	4	5
2	1	-	1	1	5
3	1	2	-	4	5
4	2	2	3	-	5
5	2	2	2	4	-

Floyd Warshall



Columna 4.1 - Fila 4:
3 + 4 !< 0
3 + 1 < ∞ (Se reemplaza)
3 + 1 < 6 (Se reemplaza)
3 + ∞ !< ∞

Columna 4.2 - Fila 4:
6+ 4 !< 3
6 + 1 !< 0
6 + 1 < 9 (Se reemplaza)
6 + ∞ !< ∞

Columna 4.3 - Fila 4:
2 + 4 < ∞ (Se reemplaza)
2 + 1 < ∞ (Se reemplaza)
2 + 1 !< 0
2 + ∞ !< ∞

Columna 4.5 - Fila 4:
2 + 4 < 7 (Se reemplaza)
2 + 1 < 4 (Se reemplaza)
2 + 1 < 13 (Se reemplaza)
2 + ∞ !< 0

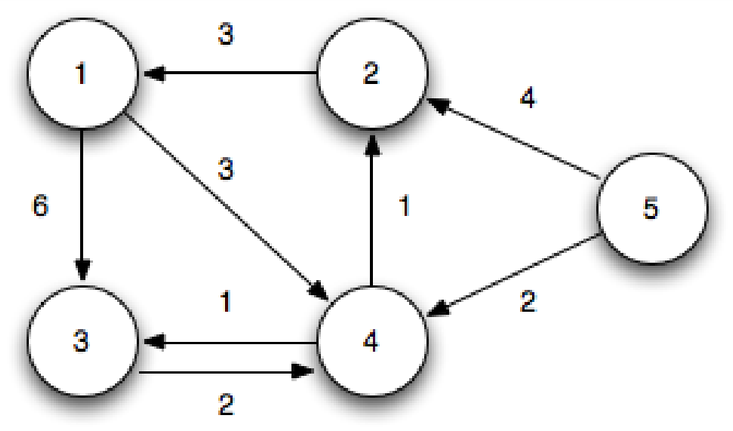
Matriz de Distancias

	1	2	3	4	5
1	0	∞	6	3	∞
2	3	0	9	6	∞
3	∞	∞	0	2	∞
4	4	1	1	0	∞
5	7	4	13	2	0

Matriz de Recorridos

	1	2	3	4	5
1	-	2	3	4	5
2	1	-	1	1	5
3	1	2	-	4	5
4	2	2	3	-	5
5	2	2	2	4	-

Floyd Warshall



Columna 5.1 - Fila 5:
∞ + 6 !< 0
∞ + 3 !< 4
∞ + 3 !< 4
∞ + 2 !< 3

Columna 5.2 - Fila 5:
∞ + 6 !< 3
∞ + 6 !< 0
∞ + 3 !< 7
∞ + 2 !< 6

Columna 5.3 - Fila 5:
∞ + 6 !< 6
∞ + 3 !< 3
∞ + 3 !< 0
∞ + 2 !< 2

Columna 5.4 - Fila 5:
∞ + 6 !< 4
∞ + 3 !< 1
∞ + 3 !< 1
∞ + 2 !< 0

Matriz de Distancias

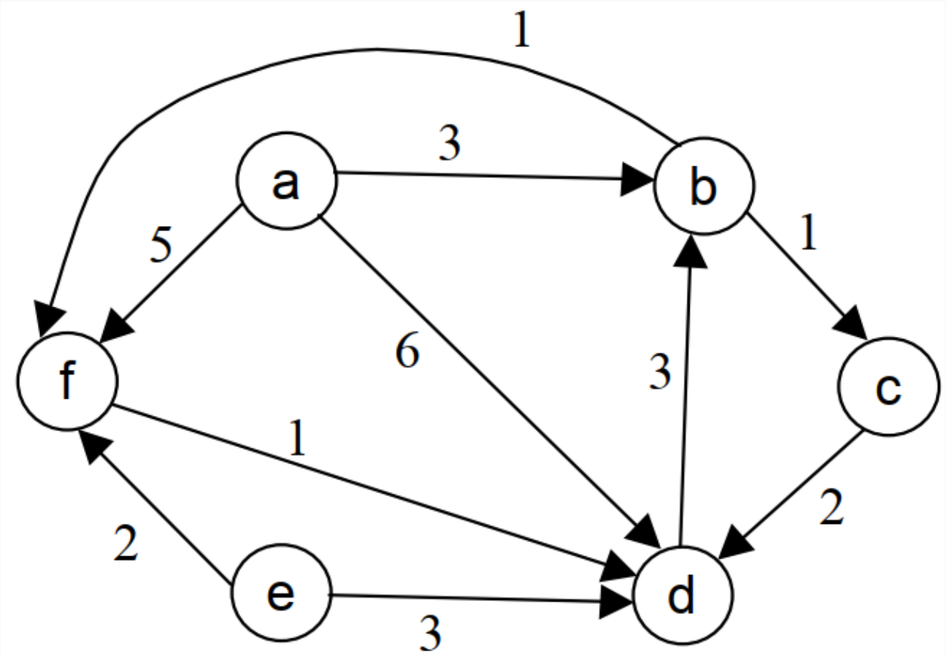
	1	2	3	4	5
1	0	4	4	3	∞
2	3	0	7	6	∞
3	6	3	0	2	∞
4	4	1	1	0	∞
5	6	3	3	2	0

Matriz de Recorridos

	1	2	3	4	5
1	-	4	4	4	5
2	1	-	4	1	5
3	4	4	-	4	5
4	2	2	3	-	5
5	4	4	4	4	-

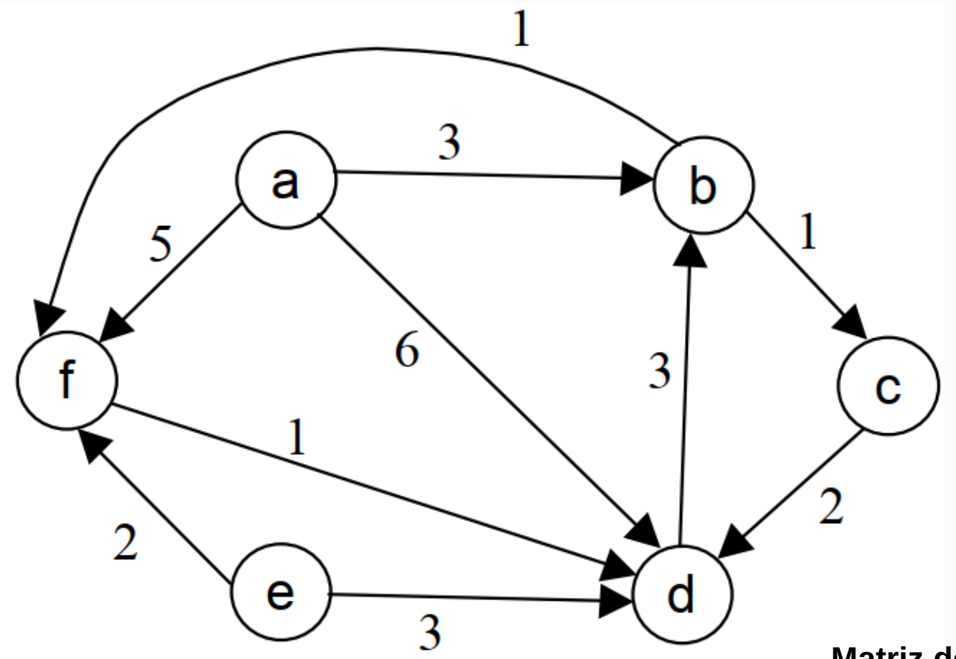
Ejercicios

Floyd Warshall:



Ejercicios

Floyd Warshall:



Matriz de Distancias

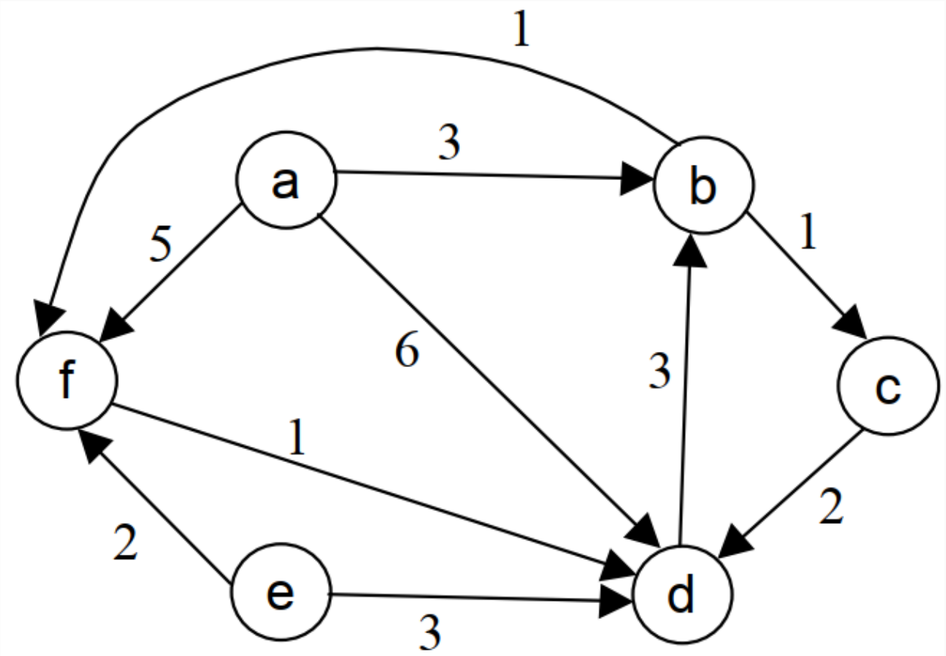
	a	b	c	d	e	f
A	0	3	∞	6	∞	5
B	∞	0	1	∞	∞	1
C	∞	∞	0	2	∞	∞
D	∞	3	∞	0	∞	∞
E	∞	∞	∞	3	0	2
F	∞	∞	∞	1	∞	0

Matriz de Recorridos

	a	b	c	d	e	f
A	-	b	c	d	e	f
B	a	-	c	d	e	f
C	a	b	-	d	e	f
D	a	b	c	-	e	f
E	a	b	c	d	-	f
F	a	b	c	d	e	-

Ejercicios

Floyd Warshall:



Matriz de Distancias

	a	b	c	d	e	f
A	0	3	4	5	∞	4
B	∞	0	1	2	∞	1
C	∞	5	0	2	∞	6
D	∞	3	4	0	∞	4
E	∞	6	7	3	0	2
F	∞	4	5	1	∞	0

Matriz de Recorridos

	a	b	c	d	e	f
A	-	b	b	f	e	b
B	a	-	c	f	e	f
C	a	d	-	d	e	d
D	a	b	b	-	e	b
E	a	d	d	d	-	f
F	a	d	d	d	e	-

UTEC
UNIVERSIDAD DE INGENIERÍA
Y TECNOLOGÍA

Welcome to Algorithms and Data Structures! - CS2100