Algoritmo de Dijkstra

```
para cada vértice v∈V hacer
   { Distancia[v] ← ∞
     Final[v] ← FALSE
     Predecesor[v] ← -1
   }
Distancia[s] ← 0
Final[s] ← TRUE
Actual ← s
mientras Final[t] = FALSE hacer
   { para cada sucesor v del vértice Actual hacer
      { si Final[v] = FALSE entonces
          { si Distancia[Actual] + w_Actual,v <
               Distancia[v] entonces
               { Distancia[v] ← Distancia[Actual] +
                                w_Actual,v
                 Predecesor[v] ← Actual
               }
          }
      }
     Sea y el vértice con marca temporal más pequeña
     Final[y] ← TRUE
     Actual ← y
   }
```

Algoritmo para calcular el camino mínimo

```
v ← t
mientras v ≠ s
   { para los vértices predecesores u de v hacer
        si Distancia[u] + w_u,v = Distancia[v] entonces
           u es un antecesor de v en el camino mínimo
           de s a t
     v ← u
   }
```

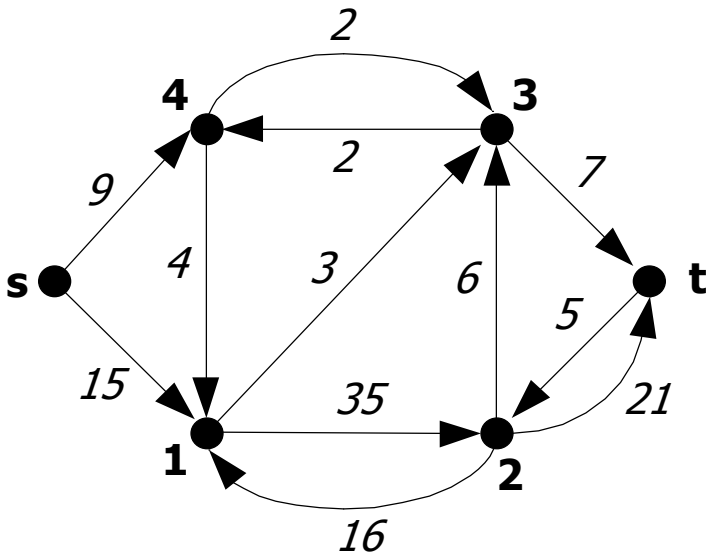


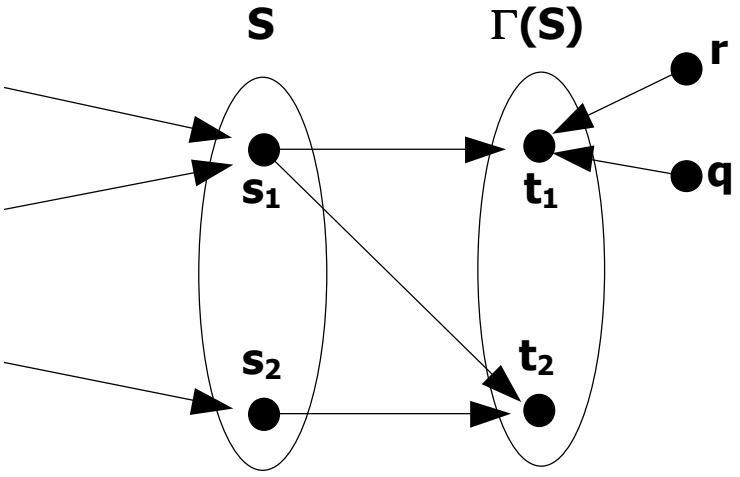
Figura 1: Grafo dirigido.

Iteración k	Distancia al vértice... s	1	2	3	4	t
0	0*	∞	∞	∞	∞	∞
1	0*	15	∞	∞	9	∞
	0*	15	∞	∞	9*	∞
2	0*	13	∞	11	9*	∞
	0*	13	∞	11*	9*	∞
3	0*	13	∞	11*	9*	18
	0*	13*	∞	11*	9*	18
4	0*	13*	48	11*	9*	18
	0*	13*	48	11*	9*	18*

Tabla 1: Dijkstra sobre el grafo anterior.

Algoritmo de Ford

```
S ← conjunto de sucesores del vértice origen s
para todos los vértices v∈S hacer
   D0[v] ← ws,v
para todos los vértices v∈(V−S) hacer
   D0[v] ← ∞
D0[s] ← 0
k ← 1
Solución ← FALSE
CicloNegativo ← FALSE
mientras (Solución = FALSE) y (CicloNegativo = FALSE)
hacer
   { para todos los sucesores v del conjunto S hacer
        { m ← ∞
          para (todos los predecesores u de v) y
               (que estén en el conjunto S) hacer
             m ← min(m, Dk-1[u] + wu,v)
          Dk[v] ← min(Dk-1[v], m)
        }
     si (k ≤ (N − 1)) y (Dk[v] = Dk-1[v], ∀v∈V) entonces
        Solución ← TRUE
     si (k < (N − 1)) y
        (Dk[v] ≠ Dk-1[v], para algún v∈V) entonces
        { S ← { v tal que Dk[v] ≠ Dk-1[v] }
          k ← k + 1
        }
     si (k = (N − 1)) y
        (Dk[v] ≠ Dk-1[v], para algún v∈V) entonces
        CicloNegativo ← TRUE
   }
```



$$S = \{s_1, s_2\}$$

$$T = \Gamma(S) = \{t_1, t_2\}$$

$$\Gamma^{-1}(t_1) = \{s_1, q, r\}$$

$$\Gamma^{-1}(t_1) \quad S = \{s_1\}$$

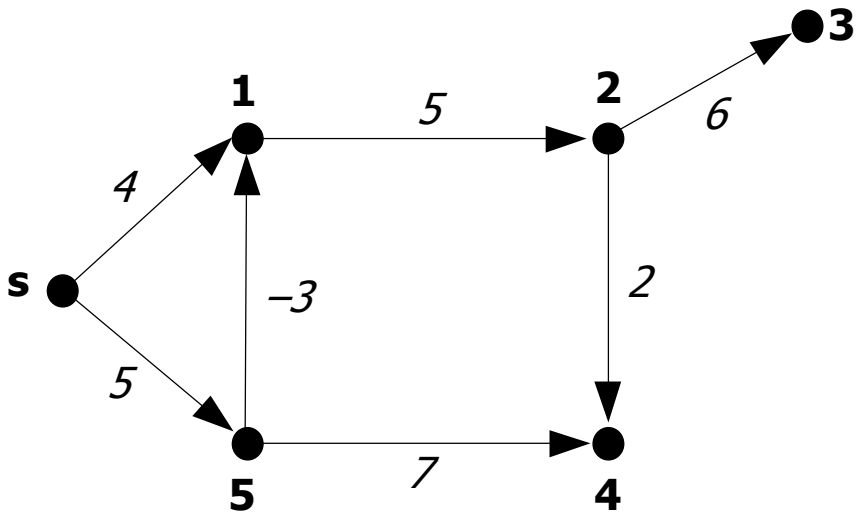


Figura 2: Grafo dirigido.

Iteración	Conjuntos		Distancia al vértice...						Solución
k	S	Γ(S)	s	1	2	3	4	5	
0	{1, 5}	—	0	4	∞	∞	∞	5	FALSE
1	{1, 5}	{2, 1, 4}	0	2	9	∞	12	5	FALSE
2	{1, 2, 4}	{2, 3, 4}	0	2	7	13	9	5	FALSE
3	{2, 3, 4}	{3, 4}	0	2	7	13	9	5	TRUE

Tabla 2: Ford sobre el grafo anterior.

Algoritmo PDM

```
para cada vértice v∈V hacer
   { Distancia[v] ← ∞
     Predecesor[v] ← -1
   }
Distancia[s] ← 0
Inicializar la cola Q e insertar s al final de la misma
mientras Q no esté vacía hacer
   { Sacar de la cabeza de la cola el vértice v
     para cada sucesor inmediato u del vértice v hacer
       { si Distancia[v] + w_{v,u} < Distancia[u] entonces
           { Distancia[u] ← Distancia[v] + w_{v,u}
             Predecesor[u] ← v
             si u no ha estado antes en Q entonces
                Meter u al final de la cola Q
             si u estuvo en Q pero no está actualmente
             en Q entonces
                Insertar u al principio de la cola Q
           }
       }
   }
```

Q			Distancia					
←			s	1	2	3	4	5
s			0	∞	∞	∞	∞	∞
1	5		0	4	∞	∞	∞	5
5	2		0	4	9	∞	∞	5
1	2	4	0	2	9	∞	12	5
2	4		0	2	7	∞	12	5
4	3		0	2	7	13	9	5
3			0	2	7	13	9	5
			0	2	7	13	9	5

Tabla 3: PDM sobre el grafo de la Figura 2.

Algoritmo de Glover et al. (1985)

```
para cada vértice v∈V hacer
   { Distancia[v] ← ∞
     Predecesor[v] ← -1
   }
Distancia[s] ← 0
Añadir s a la ListaActual
hacer
  { ListaSiguiente ← Vacia
    mientras ListaActual no vacía hacer
      { Actual ← Extraer un nodo de ListaActual
        para cada sucesor v del vértice Actual hacer
          { si Distancia[Actual]+w_Actual,v < Distancia[v]
            entonces
              { Distancia[v] ←Distancia[Actual]+w_Actual,v
                Predecesor[v] ← Actual
                si v no está en ListaActual
                   ni en ListaSiguiente entonces
                  Añadir v a ListaSiguente
              }
          }
      }
    ListaActual ← ListaSiguiente
  } mientras ListaSiguiente no vacía
```

Algoritmo de Floyd

```
para k ← 1 hasta N hacer
   para i ← 1 hasta N hacer
      si w_ik ≠ ∞ entonces
         { para j ← 1 hasta N hacer
              { si w_kj ≠ ∞ entonces
                   w_ij ← min(w_ij , w_ik + w_kj)
              }
         }
```

$$p_{ij} = \begin{cases} i & \text{si } w_{ij} \neq \infty \\ 0 & \text{si } w_{ij} = \infty \end{cases}$$

$$v_q = p_{ij}$$

$$v_{q-1} = p_{i,v_q}$$

$$v_{q-2} = p_{i,v_{q-1}}$$

$$i = p_{i,v_1}$$

$$W = \begin{pmatrix} 0 & 13 & 23 & 11 & 9 & 18 \\ \infty & 0 & 15 & 3 & 5 & 10 \\ \infty & 12 & 0 & 6 & 8 & 13 \\ \infty & 6 & 12 & 0 & 2 & 7 \\ \infty & 4 & 14 & 2 & 0 & 9 \\ \infty & 17 & 5 & 11 & 13 & 0 \end{pmatrix}$$

```
procedimiento Caminos(i, j)
{ Meter en la pila el vértice j
  si i = j entonces
     Sacar de la pila todo el camino almacenado
  en otro caso
     { para todos los predecesores v de j hacer
          { Comprobar que el vértice v no está ya en
            la pila para evitar los ciclos de costo 0
            si Distancia[i, v] + w_v,j = Distancia[i, j]
            entonces
               { Caminos(i, v)
                 Sacar de la pila el último elemento
               }
          }
     }
}
```

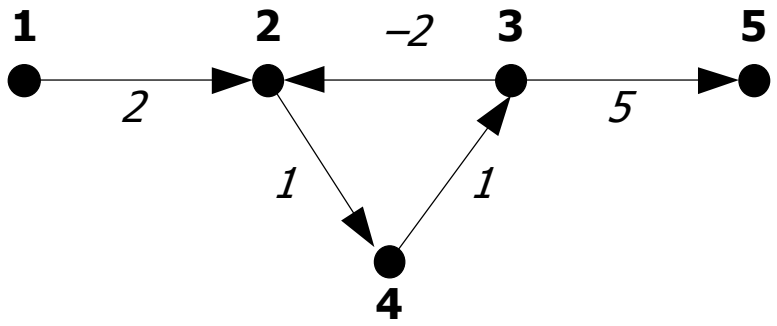


Figura 3: Grafo dirigido con ciclo de costo 0.