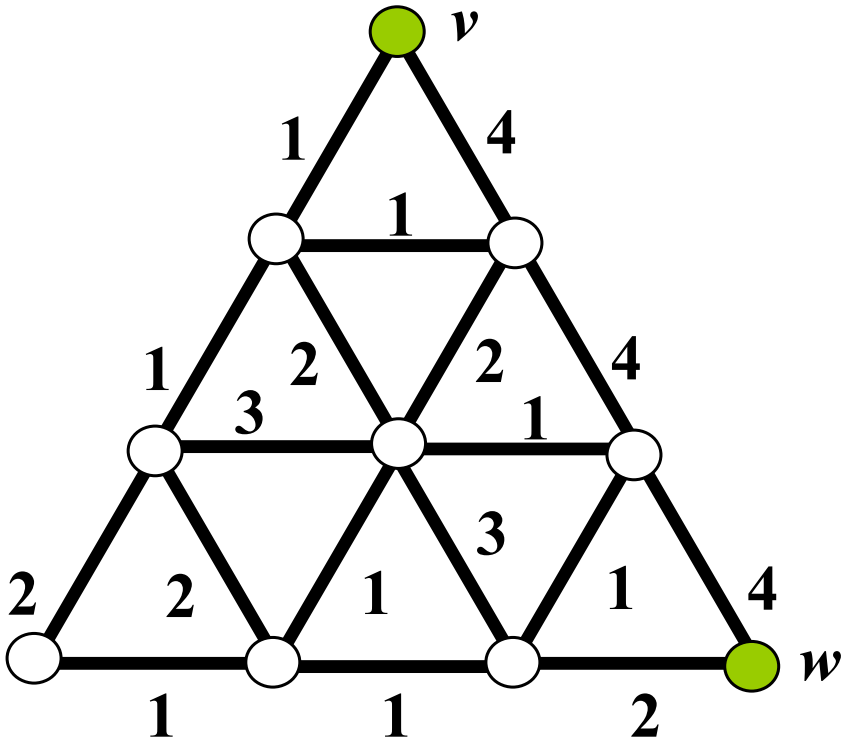
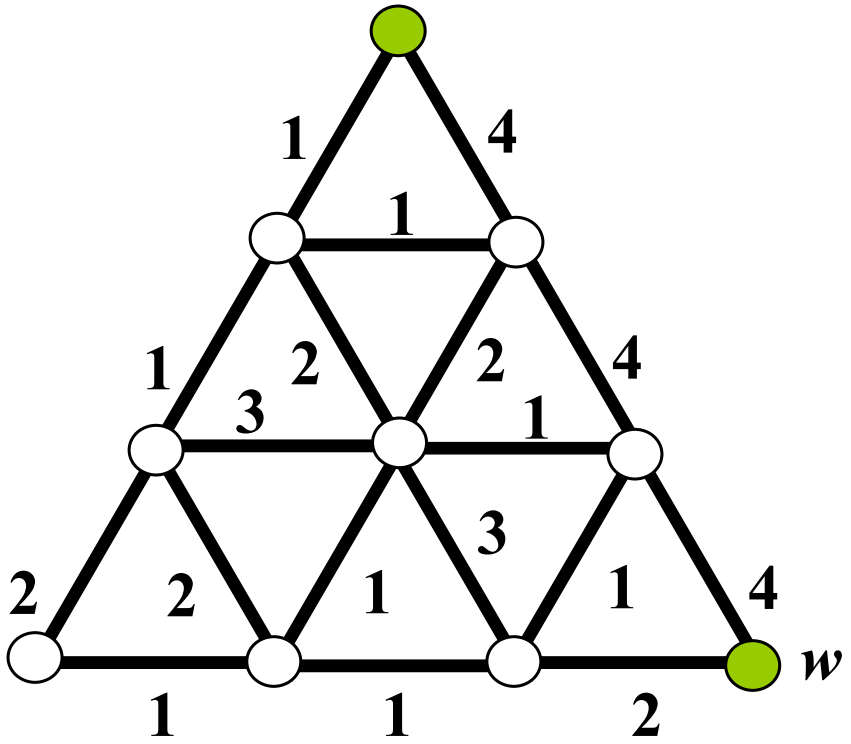
Caminho mais Curto

Caminho mais curto

O caminho mais curto **entre os vértices *v* e *w* de um grafo *G* ponderado é aquele que acumula o menor valor possível dentre todos os caminhos existentes entre *v* e *w* .**





Caminho mais curto

1. De um nó fonte para todos os demais vértices
2. De todos os nós para um destino único
3. De par único
4. Dentre todos os pares de vértices

Arestas de Peso Negativo

Se o grafo G não contém ciclo de peso negativo, a partir da origem s, então para todo vértice $v \in V$, o comprimento do caminho mais curto $\delta(s,v)$ permanece bem definido, mesmo contendo arestas de peso negativo.

Contudo, se existe um ciclo de peso negativo a partir de s, os pesos de caminhos mais curtos não são bem definidos, pois sempre é possível encontrar um caminho ainda mais curto percorrendo o ciclo de peso negativo.

Subestrutura ótima do caminho mais curto

Considere $p = \langle v_1, v_2, \ldots, v_k \rangle$, um caminho mais curto entre os vértices v_1 e v_k.

Então, para quaisquer i, j, tais que $1 \leq i \leq j \leq k$, o caminho $p_{ij} = \langle v_i, v_{i+1}, \ldots, v_j \rangle$, subcaminho de p, é um caminho mais curto entre os vértices i e j.

Desigualdade Triangular

Para qualquer aresta ***(u,v)*** $\in E$, e ***s*** algum outro vértice, temos:

$$\delta(s,v) \leq \delta(s,u) + w(u,v)$$

onde

$\delta(u,v)$ – caminho mais curto de u para v

$w(u,v)$ – peso da aresta (u,v)

Algoritmo de Dijkstra

1. De um nó fonte para todos os demais vértices

2. De todos os nós para um destino único

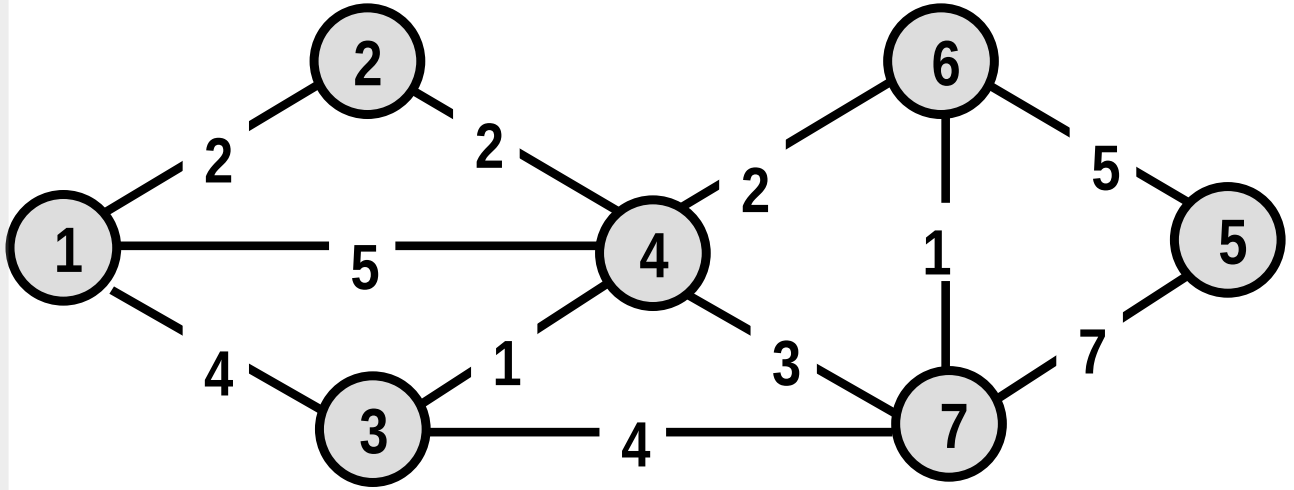
3. De par único

Algoritmo de Dijkstra

```
Dijkstra(Digrafo D, pesos w,vértice s)
  Para todo vértice v ∈ V(D)  faça
       v.visitado = 0;
       v.predecessor = NULL;
       v.distância = INF;
  s.distância= 0;
  Enquanto houver vértice u com u.visitado==0 e u.predecessor != INF  faça
   Seja x um vértice não visitado com menor  x.distância
   x.visitado = 1;
   Para todo vértice y ∈ N+(u)  faça
      Se y.visitado==0 então
          Se  x.distância + w(xy) < y.distância  então
              y.distância   = x.distância + w(xy)
              y.predecessor = x;
```

```
Dijkstra(Digrafo D, pesos w,vértice s)
  Para todo vértice v ∈ V(D)  faça
       v.visitado = 0;
       v.predecessor = NULL;
       v.distância = INF;
  s.distância= 0;
  Enquanto existir u com u.visitado==0 e u . predecessor != INF  faça
    Seja x um vértice não visitado com menor  x.distância
    x.visitado = 1;
    Para todo vértice y ∈ N+ (u)  faça
       Se y.visitado==0 então
            Se  x.distância + w(xy)  < y.distância  então
                 y.distância   = x.distância + w(xy)
                 y.predecessor = x;
```

s=1

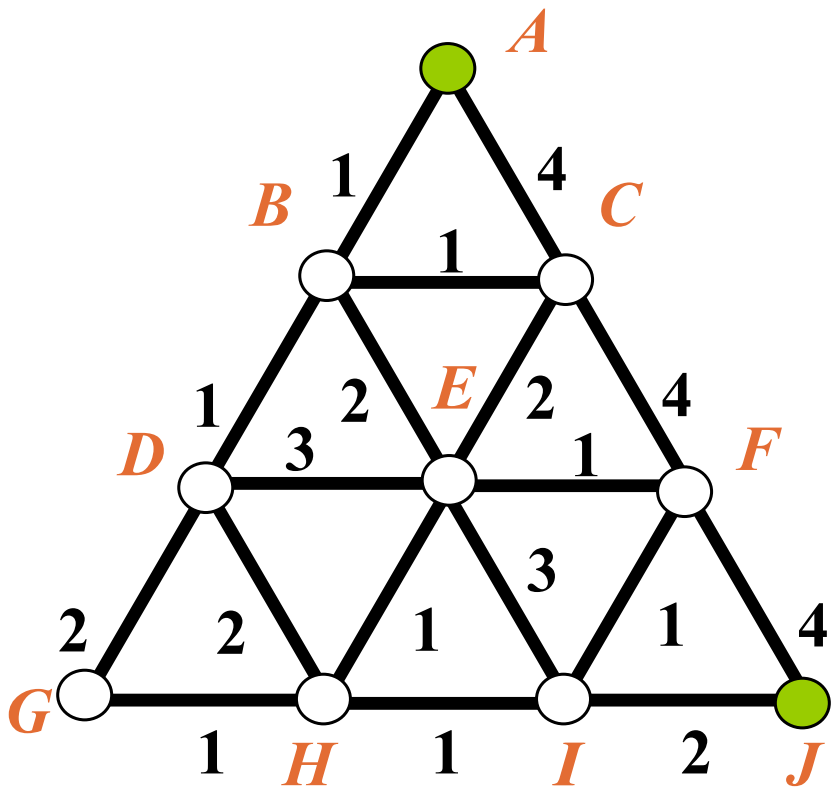


Vértices	1	2	3	4	5	6	7
Visitado							
Predecessor							
Distância							

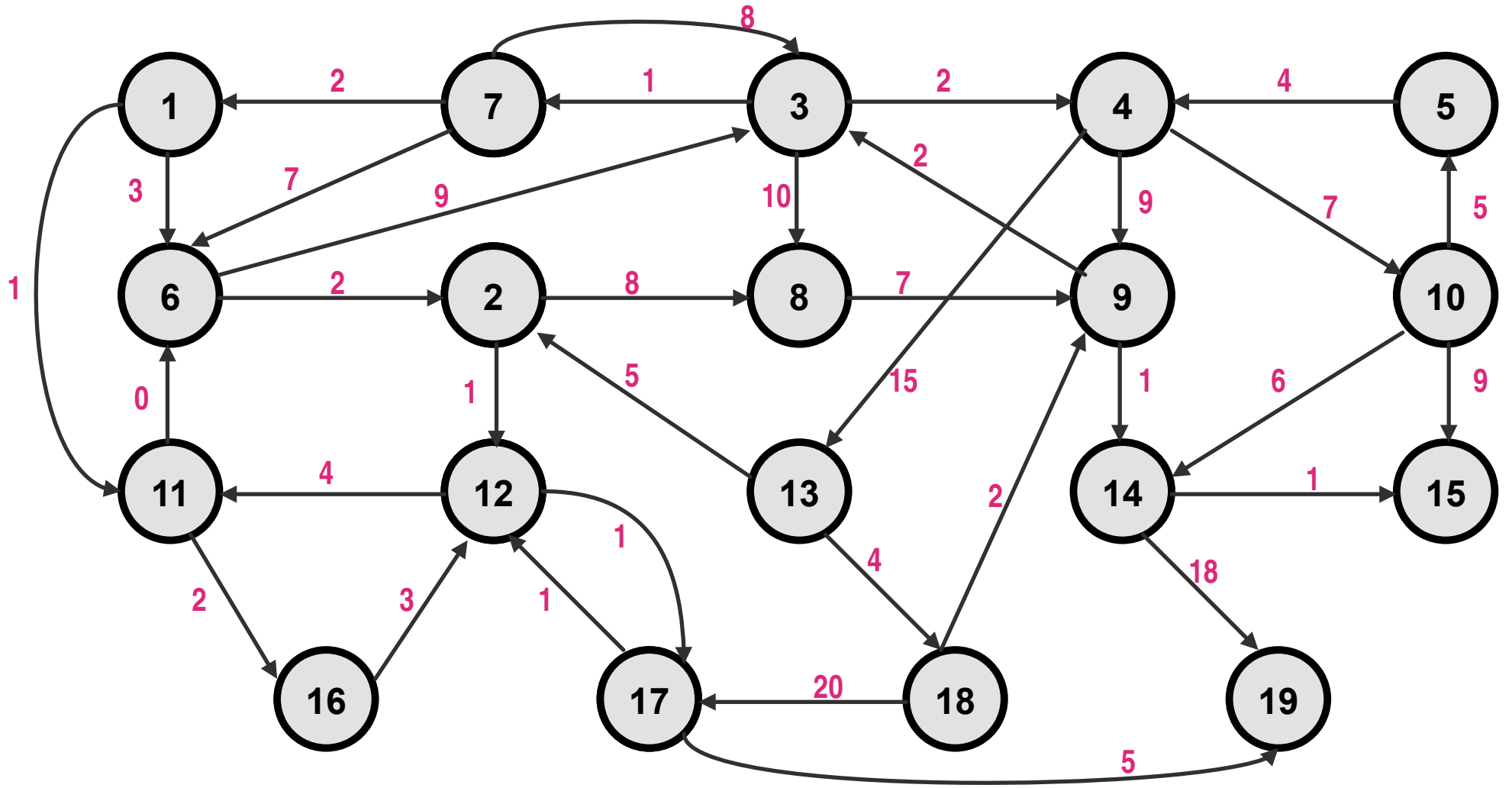
	1	2	3	4	5	6	7
1	---	2	4	5	---	---	---
2	2	---	---	2	---	---	---
3	4	---	---	1	---	---	4
4	5	2	1	---	---	2	3
5	---	---	---	---	---	5	7
6	---	---	---	2	5	---	1
7	---	---	4	3	7	1	---

```
Dijkstra(Digrafo D, pesos w,vértice s)
  Para todo vértice v ∈ V(D)  faça
       v.visitado = 0;
       v.predecessor = NULL;
       v.distância = INF;
  s.distância= 0;
  Enquanto existir u com u.visitado==0 e u . predecessor != INF  faça
   Seja x um vértice não visitado com menor  x.distância
   x.visitado = 1;
   Para todo vértice y ∈ N+ (u)  faça
      Se y.visitado==0 então
          Se  x.distância + w(xy)  < y.distância  então
              y.distância   = x.distância + w(xy)
              y.predecessor = x;
```

s=A



Vértices	A	B	C	D	E	F	G	H	I	J
Visitado										
Predecessor										
Distância										



s=4

Vértices	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Visitado																			
Predecessor																			
Distância																			

Algoritmo de Dijkstra

Arestas com custo negativo

Como o algoritmo emprega um critério guloso para fechar a cada iteração o vértice aberto e de menor caminho acumulado, torna-se incapaz de calcular o caminho mais curto em grafos com **arestas de custo negativo**.

Algoritmo de Dijkstra

Complexidade?

```
Dijkstra(Digrafo D, pesos w,vértice s)
  Para todo vértice v ∈ V(D)  faça
       v.visitado = 0;
       v.predecessor = NULL;
       v.distância = INF;
  s.distância= 0;
  Enquanto houver vértice u com u.visitado==0 e u.predecessor != INF  faça
   Seja x um vértice não visitado com menor  x.distância
   x.visitado = 1;
   Para todo vértice y ∈ N+(u)  faça
      Se y.visitado==0 então
          Se  x.distância + w(xy) < y.distância  então
              y.distância   = x.distância + w(xy)
              y.predecessor = x;
```

Algoritmo de Dijkstra

Caminho mais curto entre todos os pares de vértices

O algoritmo pode ser utilizado para encontrar o caminho mais curto entre todos os pares de vértices do grafo desde que seja executado uma vez para cada vértice como origem.

Nessa situação a complexidade do método seria multiplicada por n, tornando-se $O(n^3)$ na implementação sem estruturas de dados especiais.

Algoritmo Dijkstra - Implementações

Ano	Autores	Complexidade
1987	Fredman e Tarjan	$O(m + n\log n)$
1990	Ahuja *et al.*	$O(m + n\sqrt{\log c})$
1996	Raman	$O(m + n\sqrt{\log n})$
1997	Raman	$O(m + n(\log c \log\log c)^{1/3})$
1999	Thorup	$O(m)$ - grafo não direcionado
2000	Hagerup	$O(m \log\log c)$
2000	Thorup	$O(m \log\log n)$

Obs. A constante *c* é um limite superior para o peso das arestas