Departamento de Computação/UFS

Grafos e Algoritmos Computacionais

Single Source Shortest Path: Algoritmo de Dijkstra

Prof. André Britto

Single Source Shortest Path

- Problema: Dado um grafo *G* , com peso nas arestas, (direcionado ou não) e vértice *s* em *G*, encontrar o caminho de menor tamanho entre todos os vértices de *G* e *s.*

- Peso nas arestas :
  - considerar inteiro não negativo.
  - reflete o tamanho (distância) do caminho entre dois vértices, extremos da aresta.

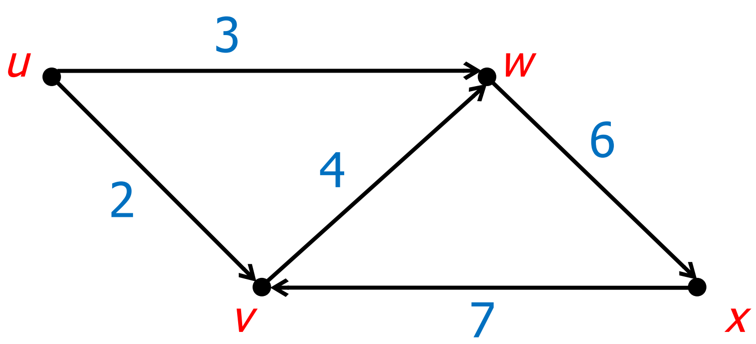
Single Source Shortest Path

Em geral, algoritmos de caminhos mínimos se baseiam na seguinte propriedade: um caminho mínimo entre dois vértices contém outros caminhos mínimos.

Single Source Shortest Path

- Simplificação: vamos na realidade determinar os comprimentos desses caminhos, ao invés do caminho explicitamente.

- Suposição: dígrafo com *u* como origem



(u)
(u,w)
(u,w,x)
(u,v)

caminhos mais curtos

Single Source Shortest Path: Algoritmo de Dijkstra

- **Guloso**
- Passos:

  - Manter um conjunto *S* de vértices para os quais já se sabe a menor distância até *s.*

Single Source Shortest Path: Algoritmo de Dijkstra

- **Guloso**
- Passos:

  - Manter um conjunto *S* de vértices para os quais já se sabe a menor distância até *s.*
    - Inicialmente que distância sabemos ?

Single Source Shortest Path: Algoritmo de Dijkstra

- **Guloso**
- Passos:

  - Manter um conjunto *S* de vértices para os quais já se sabe a menor distância até *s.*
    - Inicialmente que distância sabemos ? ($s \in S$)

Single Source Shortest Path: Algoritmo de Dijkstra

- **Guloso**
- Passos:

  - Manter um conjunto *S* de vértices para os quais já se sabe a menor distância até *s.*
    - Inicialmente que distância sabemos ? $(s \in S)$
  - a cada passo escolher um vértice $v \in VG\text{-}S$, cuja distância estimada para s é mínima, para adicionar a *S*.

Single Source Shortest Path: Algoritmo de Dijkstra

- **Guloso**
- Passos:
  - Manter um conjunto *S* de vértices para os quais já se sabe a menor distância até *s.*
    - Inicialmente que distância sabemos ? $(s \in S)$
  - a cada passo escolher um vértice $v \in VG\text{-}S$, cuja distância estimada para s é mínima, para adicionar a *S*.
  - Atualizar as estimativas de distâncias dos vértices adjacentes ao vértice *v* em relação a *s.*

Single Source Shortest Path

- **Problema:**

  - Dado um vértice s, e todo vértice w, dentre todos caminhos sendo s o início determinar:

    (i) um caminho entre s e qualquer vértice w

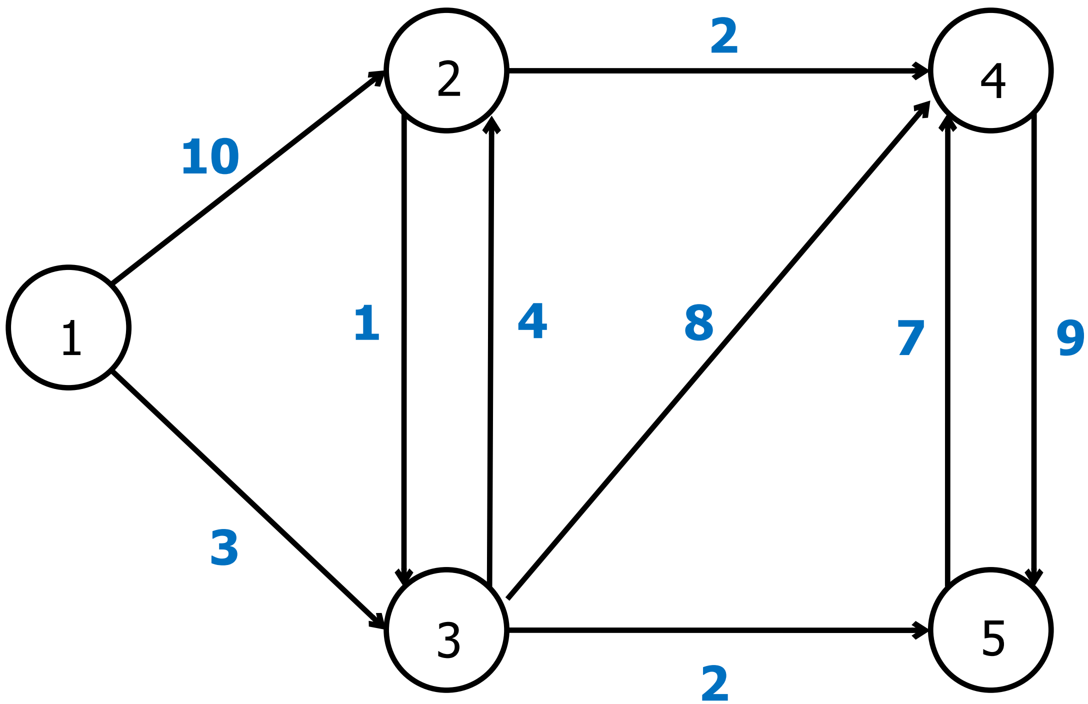
    (ii) a soma dos pesos entre s e w seja mínima.

  - Propriedade $P \Rightarrow$ (s,w) é um caminho.

Critério $\alpha \Rightarrow$ a soma dos pesos entre v e w seja mínima.

Algoritmo de Dijkstra : Exemplo

- **Exemplo (**origem no vértice 1**)**



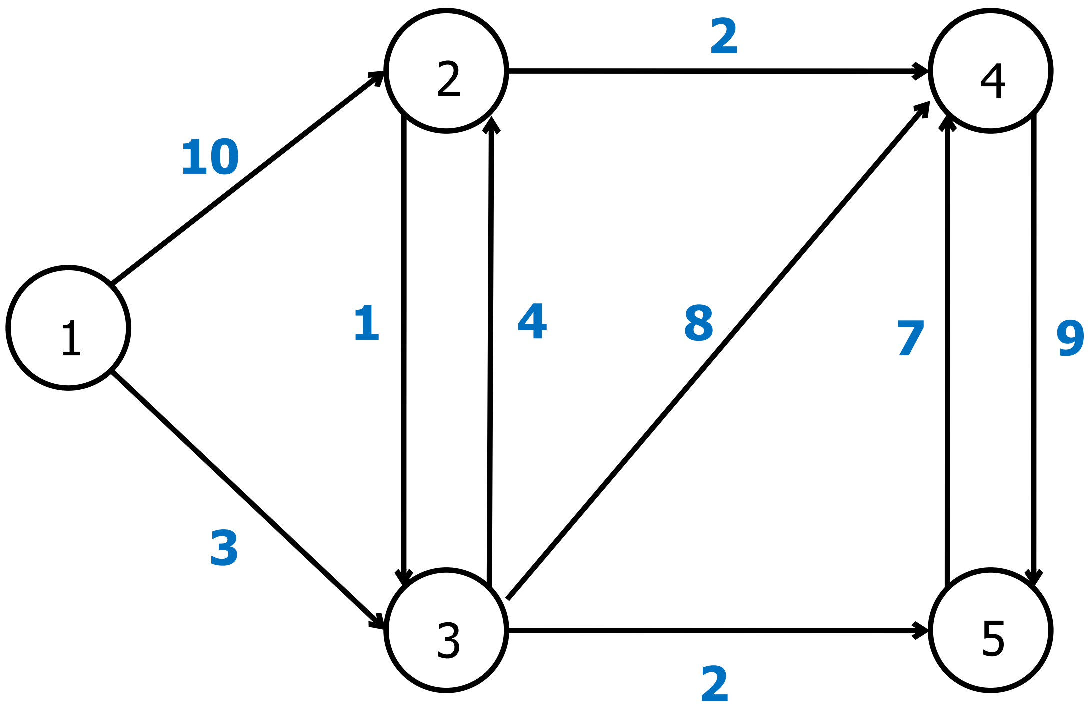
Q
1
2
3
4
5

d	v
0	1
∞	2
∞	3
∞	4
∞	5

```
S  :=  ∅;
Q  := V;
d[s]  := 0;
para para todos v ∈ V e v ≠ s
      faça
    início
      d[v]  :=  ∞;
    fim
fim
```

Algoritmo de Dijkstra : Exemplo

- **Exemplo (**origem no vértice 1**)**



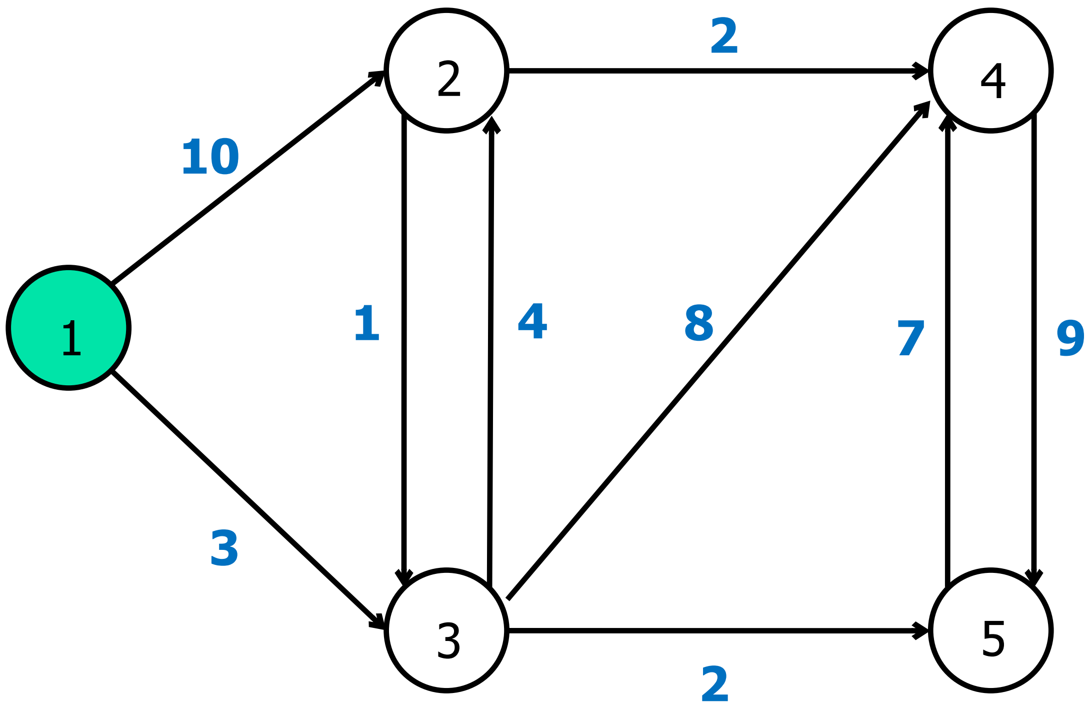
Q
1
2
3
4
5

d	v
0	1
∞	2
∞	3
∞	4
∞	5

```
enquanto Q ≠ ∅ faça
início
     v := extrairMinimo(Q);
     S := S ∪ {v};
     para todo w ∈ Adj(v)
     faça
        início
           Se d[w] > d[v] + peso(v,w)
           então
             d[w] := d[v] + peso(v,w);
        fim
fim
```

Algoritmo de Dijkstra : Exemplo

- **Exemplo (**origem no vértice 1**)**



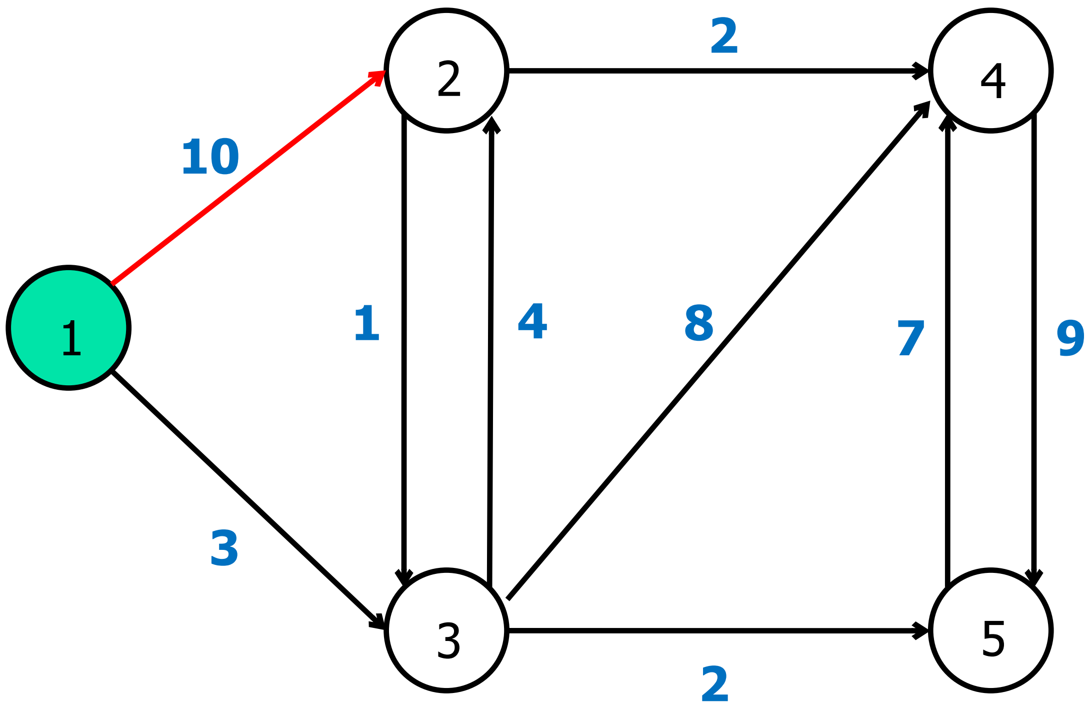
Q
2
3
4
5

d	v
0	1
∞	2
∞	3
∞	4
∞	5

```
enquanto Q ≠ ∅ faça
início
      v := extrairMinimo(Q);
      S := S ∪ {v};
      para todo w ∈ Adj(v)
      faça
          início
               Se d[w] > d[v] + peso(v,w)
               então
                  d[w] := d[v] + peso(v,w);
          fim
fim
```

Algoritmo de Dijkstra : Exemplo

- **Exemplo (**origem no vértice 1**)**



$\infty > 10+0$

Q
2
3
4
5

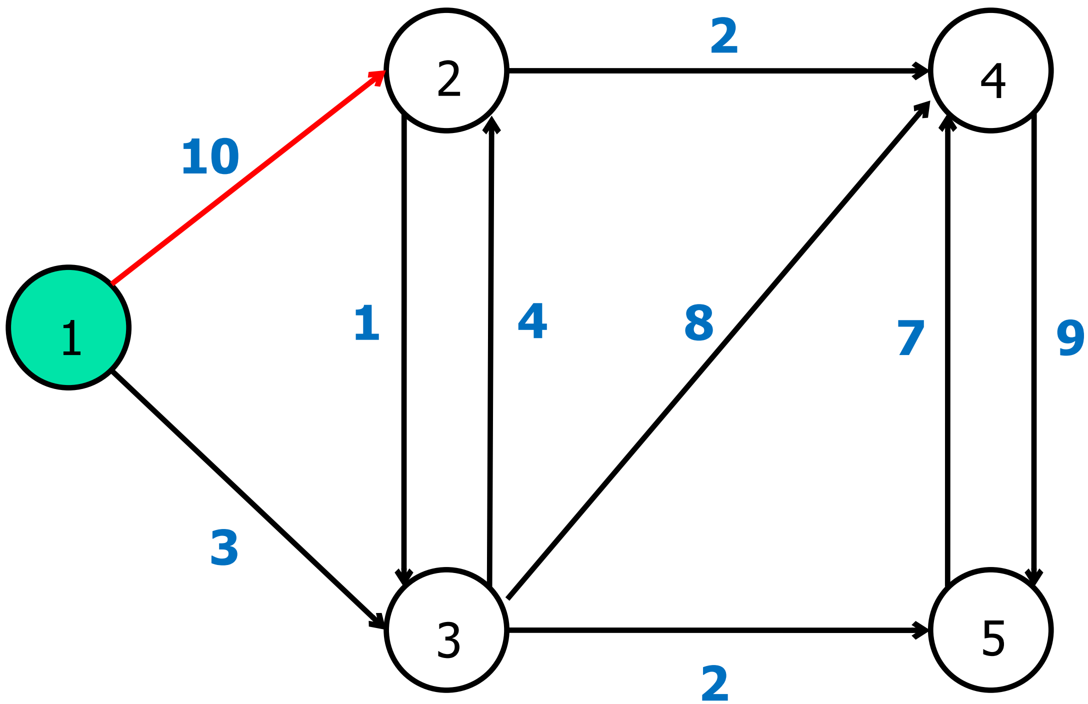
d	v
$\boxed{0}$	1
∞	2
∞	3
∞	4
∞	5

```
enquanto Q ≠ ∅ faça
início
      v := extrairMinimo(Q);
      S := S ∪ {v};
      para todo w ∈ Adj(v)
      faça
          início
               Se d[w] > d[v] + peso(v,w)
               então
                  d[w] := d[v] + peso(v,w);
          fim
fim
```

Algoritmo de Dijkstra : Exemplo

- **Exemplo (**origem no vértice 1**)**

$\infty > 10+0$



Q
2
3
4
5

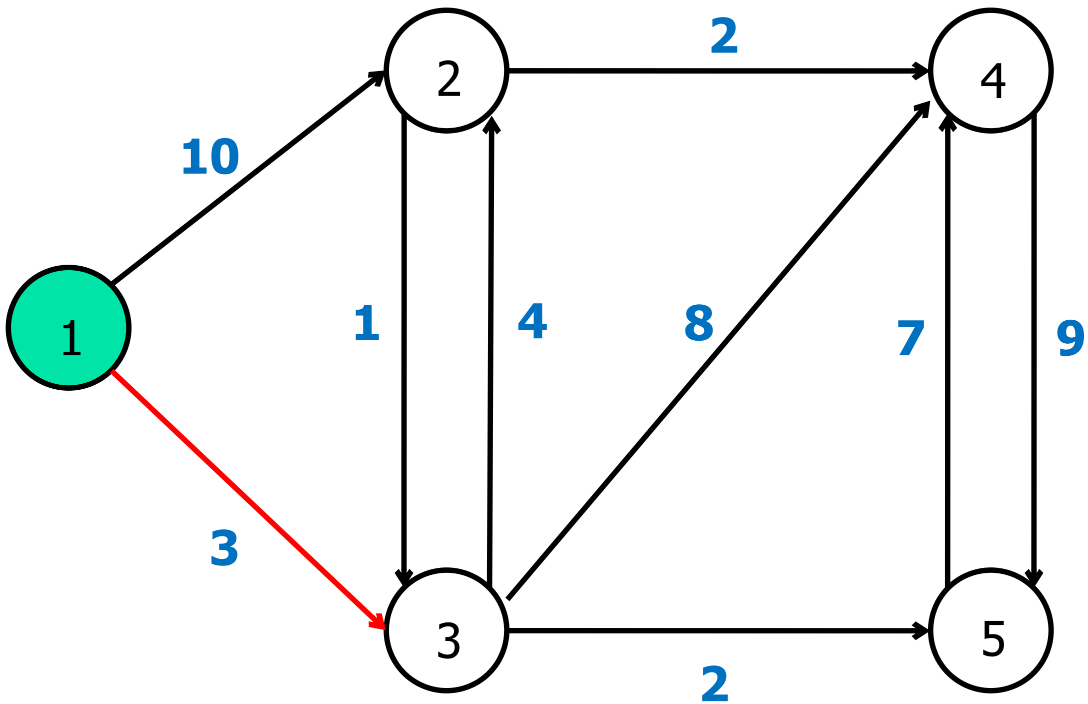
d	v
0	1
10	2
∞	3
∞	4
∞	5

```
enquanto Q ≠ ∅ faça
início
     v := extrairMinimo(Q);
     S := S ∪ {v};
     para todo w ∈ Adj(v)
     faça
        início
            Se d[w] > d[v] + peso(v,w)
            então
              d[w] := d[v] + peso(v,w);
        fim
fim
```

Algoritmo de Dijkstra : Exemplo

- **Exemplo (**origem no vértice 1**)**

$\infty > 3+0$



Q
2
3
4
5

d	v
0	1
10	2
∞	3
∞	4
∞	5

```
enquanto Q ≠ ∅ faça
início
      v := extrairMinimo(Q);
      S := S ∪ {v};
      para todo w ∈ Adj(v)
      faça
          início
               Se d[w] > d[v] + peso(v,w)
               então
                  d[w] := d[v] + peso(v,w);
          fim
fim
```

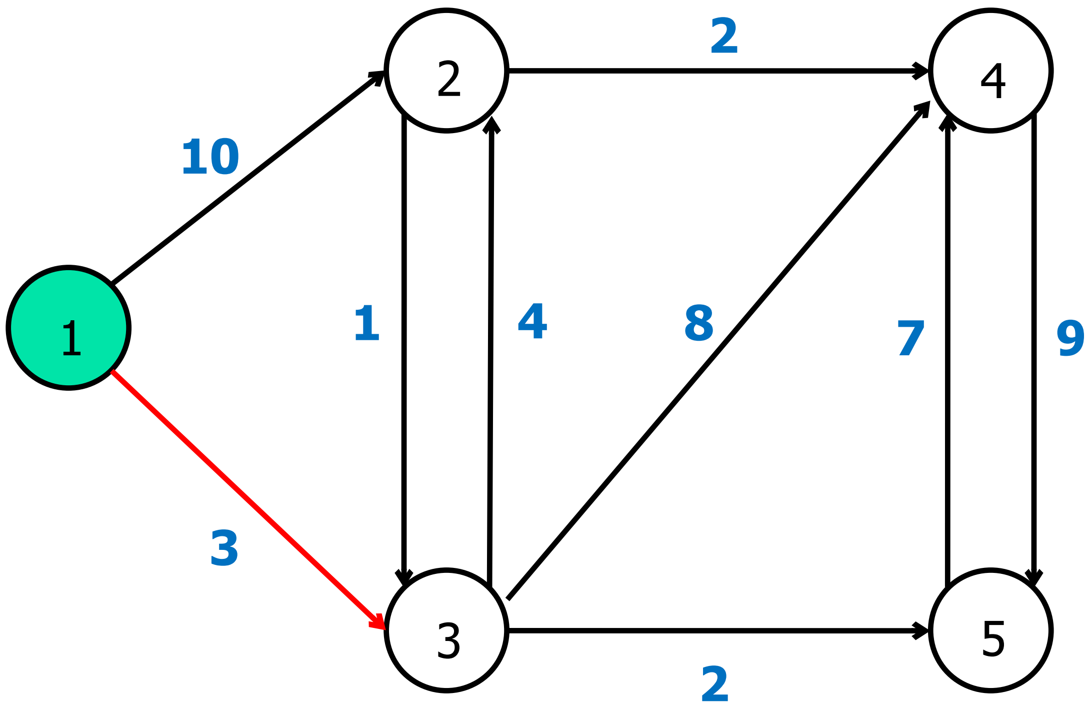
Algoritmo de Dijkstra : Exemplo

- **Exemplo (**origem no vértice 1**)**

$\infty > 3+0$

Q
2
3
4
5

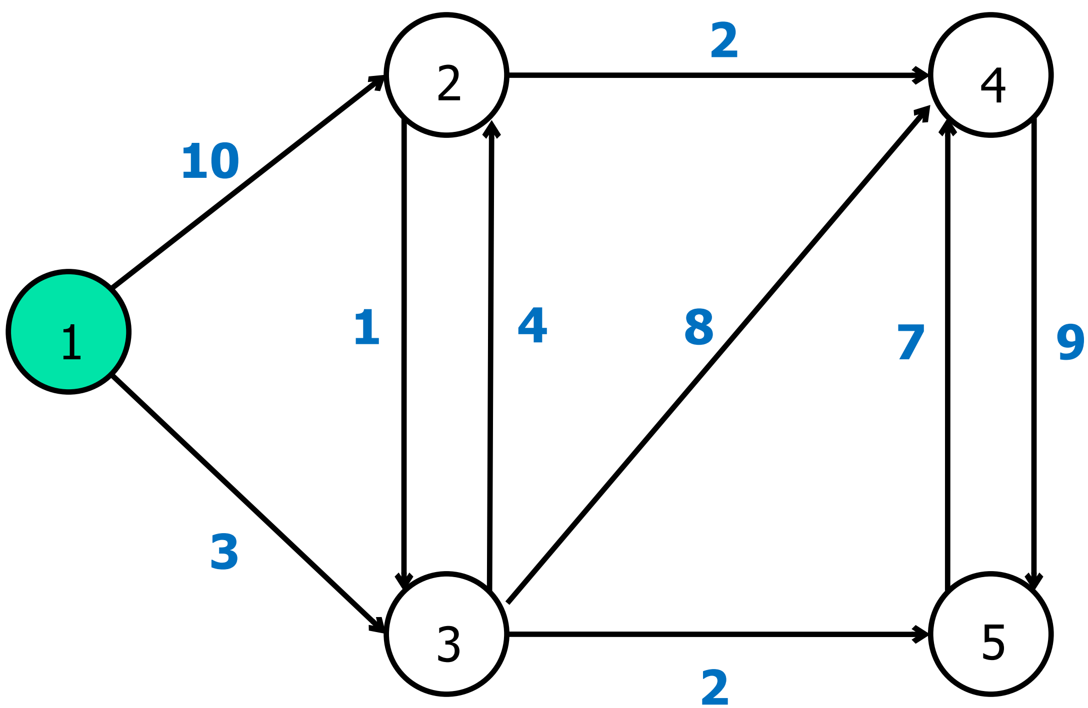
d	v
0	1
10	2
3	3
∞	4
∞	5




```
enquanto Q ≠ ∅ faça
início
      v := extrairMinimo(Q);
      S := S ∪ {v};
      para todo w ∈ Adj(v)
      faça
         início
             Se d[w] > d[v] + peso(v,w)
             então
               d[w] := d[v] + peso(v,w);
         fim
fim
```

Algoritmo de Dijkstra : Exemplo

- **Exemplo (**origem no vértice 1**)**



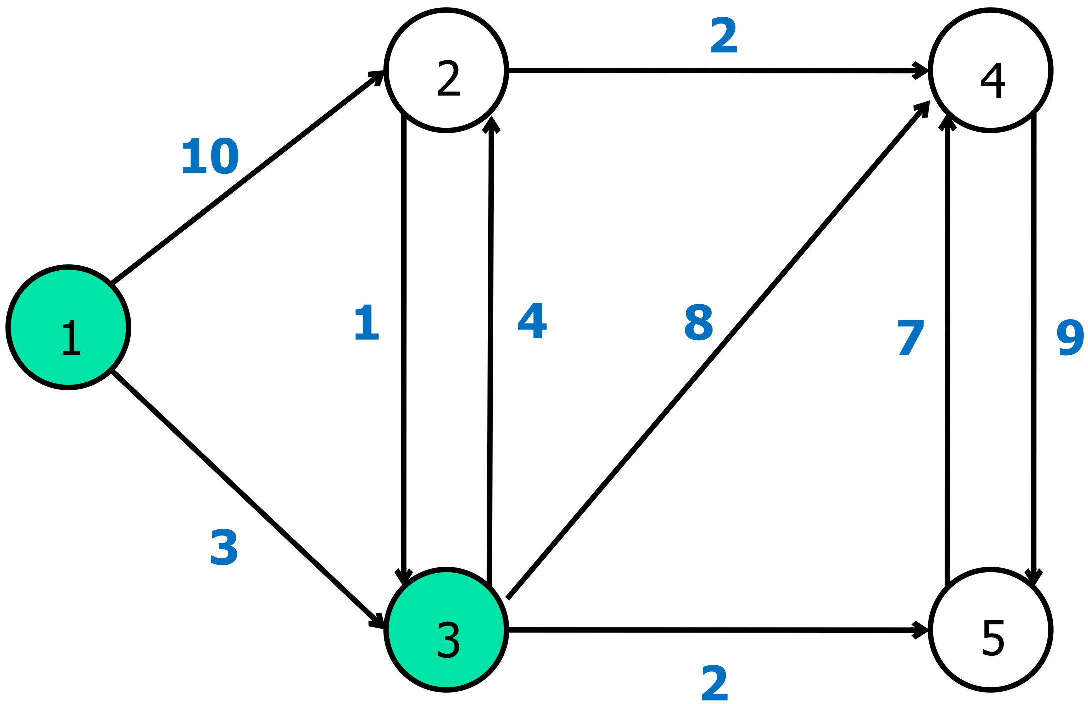
Q
2
3
4
5

d	v
0	1
10	2
3	3
∞	4
∞	5

```
enquanto Q ≠ ∅ faça
início
     v := extrairMinimo(Q);
     S := S ∪ {v};
     para todo w ∈ Adj(v)
     faça
        início
            Se d[w] > d[v] + peso(v,w)
            então
              d[w] := d[v] + peso(v,w);
        fim
fim
```

Algoritmo de Dijkstra : Exemplo

- **Exemplo (**origem no vértice 1**)**



Q
2
4
5

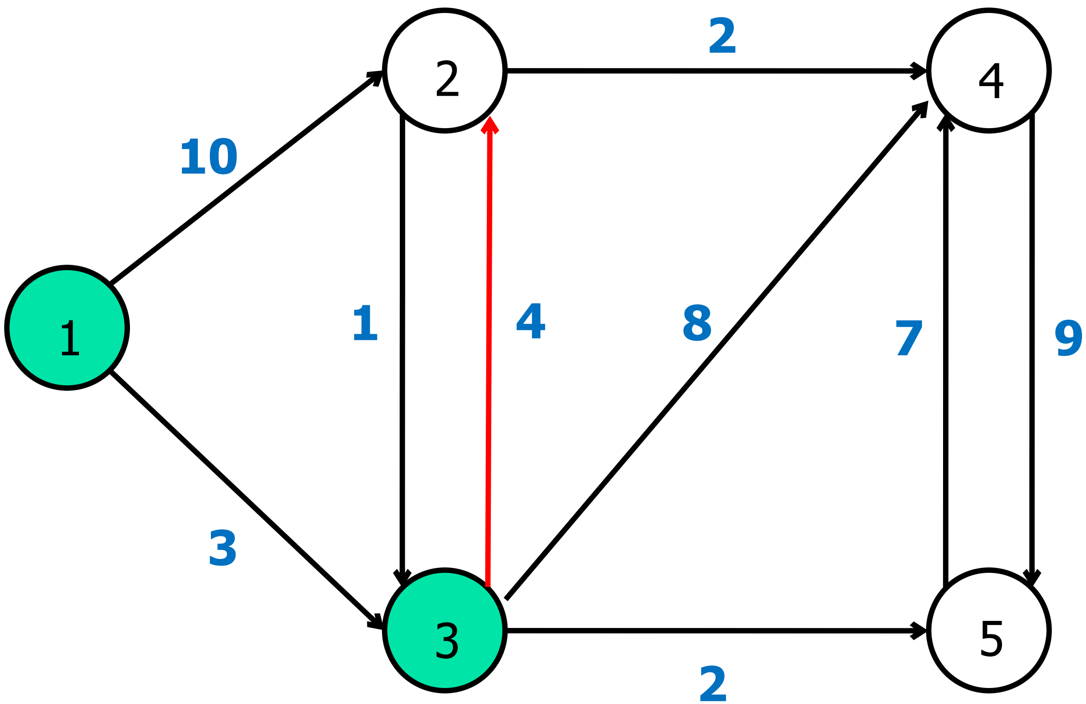
d	v
$\boxed{0}$	1
10	2
$\boxed{3}$	3
∞	4
∞	5

```
enquanto Q ≠ ∅ faça
início
      v := extrairMinimo(Q);
      S := S ∪ {v};
      para todo w ∈ Adj(v)
      faça
         início
             Se d[w] > d[v] + peso(v,w)
             então
               d[w] := d[v] + peso(v,w);
         fim
fim
```

Algoritmo de Dijkstra : Exemplo

- **Exemplo (**origem no vértice 1**)**

10 > 3+4



Q
2
4
5

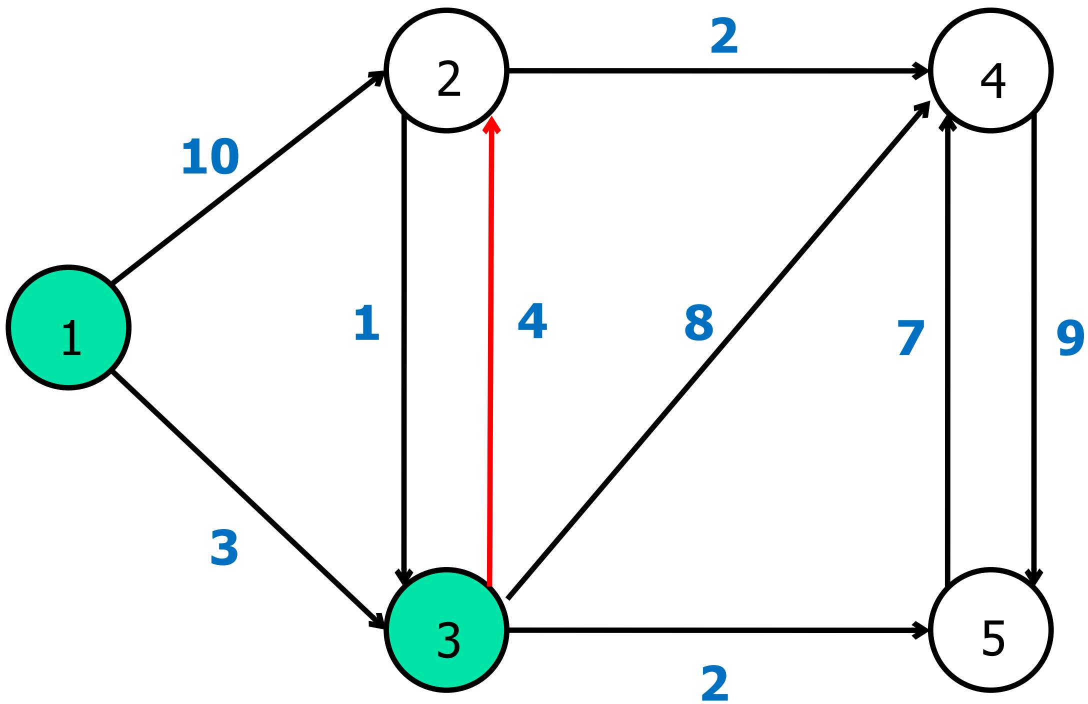
d	*v*
0	*1*
10	*2*
3	*3*
∞	*4*
∞	*5*

```
enquanto Q ≠ ∅ faça
início
      v := extrairMinimo(Q);
      S := S ∪ {v};
      para todo w ∈ Adj(v)
      faça
         início
             Se d[w] > d[v] + peso(v,w)
             então
               d[w] := d[v] + peso(v,w);
         fim
fim
```

Algoritmo de Dijkstra : Exemplo

- **Exemplo (**origem no vértice 1**)**

10 > 3+4



Q
2
4
5

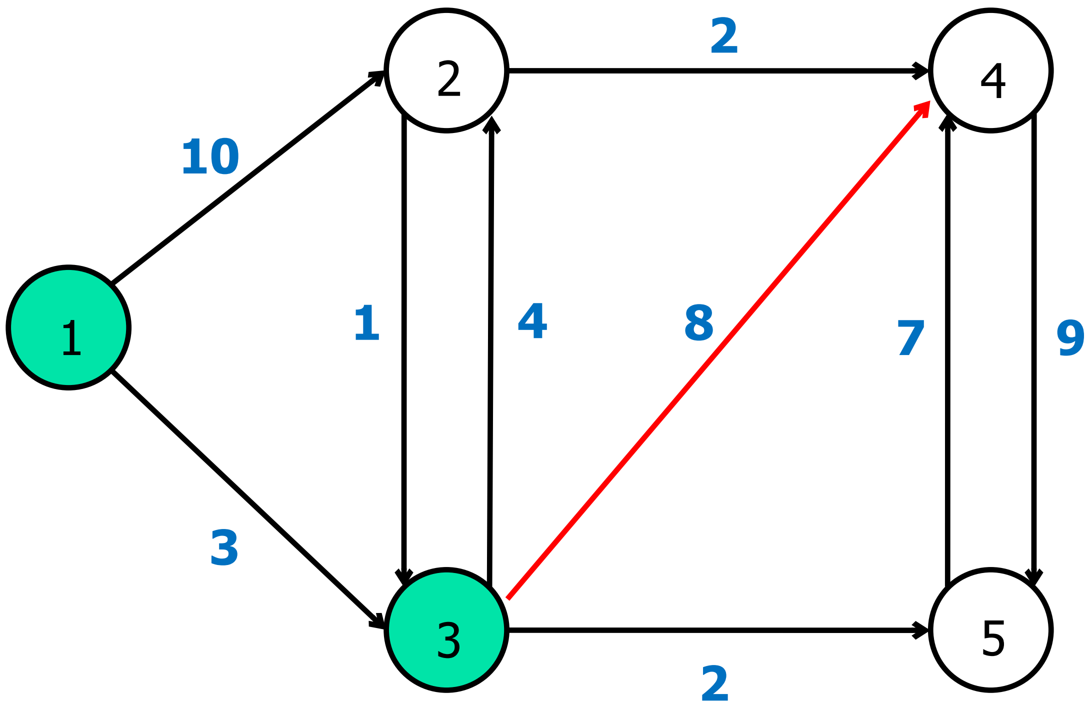
d	v
0	1
7	2
3	3
∞	4
∞	5

```
enquanto Q ≠ ∅ faça
início
      v := extrairMinimo(Q);
      S := S ∪ {v};
      para todo w ∈ Adj(v)
      faça
         início
            Se d[w] > d[v] + peso(v,w)
            então
              d[w] := d[v] + peso(v,w);
         fim
fim
```

Algoritmo de Dijkstra : Exemplo

- **Exemplo (**origem no vértice 1**)**

$\infty > 3+8$



Q
2
4
5

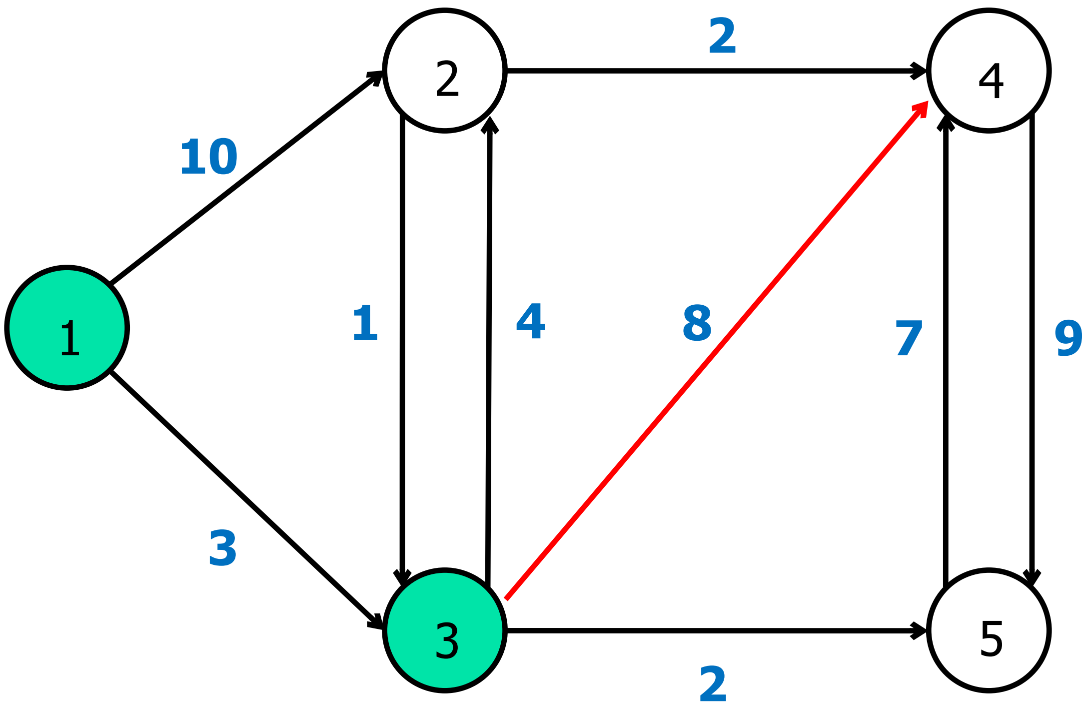
d	v
0	1
7	2
3	3
∞	4
∞	5

```
enquanto Q ≠ ∅ faça
início
      v := extrairMinimo(Q);
      S := S ∪ {v};
      para todo w ∈ Adj(v)
      faça
         início
             Se d[w] > d[v] + peso(v,w)
             então
               d[w] := d[v] + peso(v,w);
         fim
fim
```

Algoritmo de Dijkstra : Exemplo

- **Exemplo (**origem no vértice 1**)**

$\infty > 3+8$



Q
2
4
5

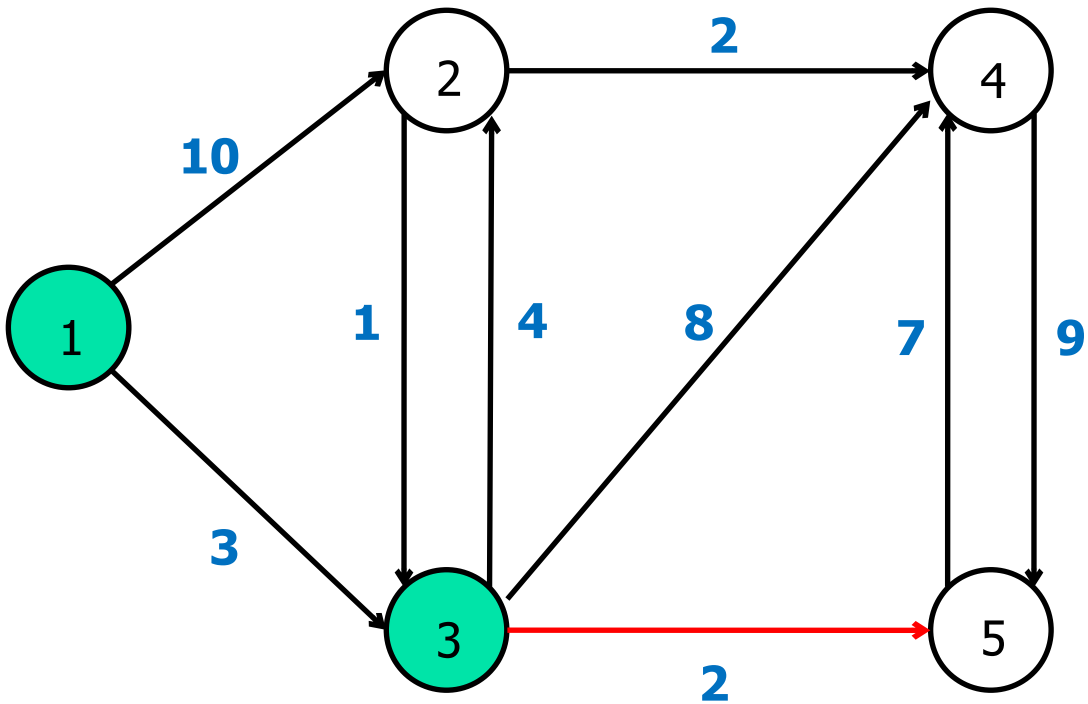
d	v
0	1
7	2
3	3
11	4
∞	5

```
enquanto Q ≠ ∅ faça
início
      v := extrairMinimo(Q);
      S := S ∪ {v};
      para todo w ∈ Adj(v)
      faça
         início
              Se d[w] > d[v] + peso(v,w)
              então
                 d[w] := d[v] + peso(v,w);
         fim
fim
```

Algoritmo de Dijkstra : Exemplo

- **Exemplo (**origem no vértice 1**)**

$\infty > 3+2$



Q
2
4
5

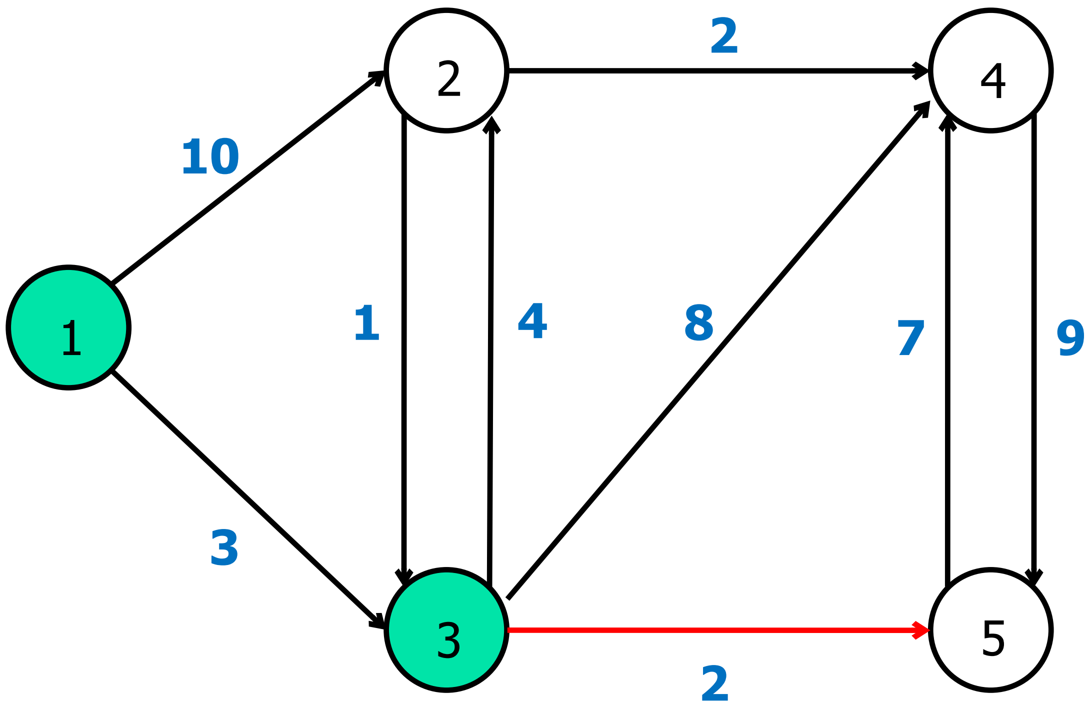
d	v
0	1
7	2
3	3
11	4
∞	5

```
enquanto Q ≠ ∅ faça
início
      v := extrairMinimo(Q);
      S := S ∪ {v};
      para todo w ∈ Adj(v)
      faça
          início
               Se d[w] > d[v] + peso(v,w)
               então
                  d[w] := d[v] + peso(v,w);
          fim
fim
```

Algoritmo de Dijkstra : Exemplo

- **Exemplo (**origem no vértice 1**)**

$\infty > 3+2$



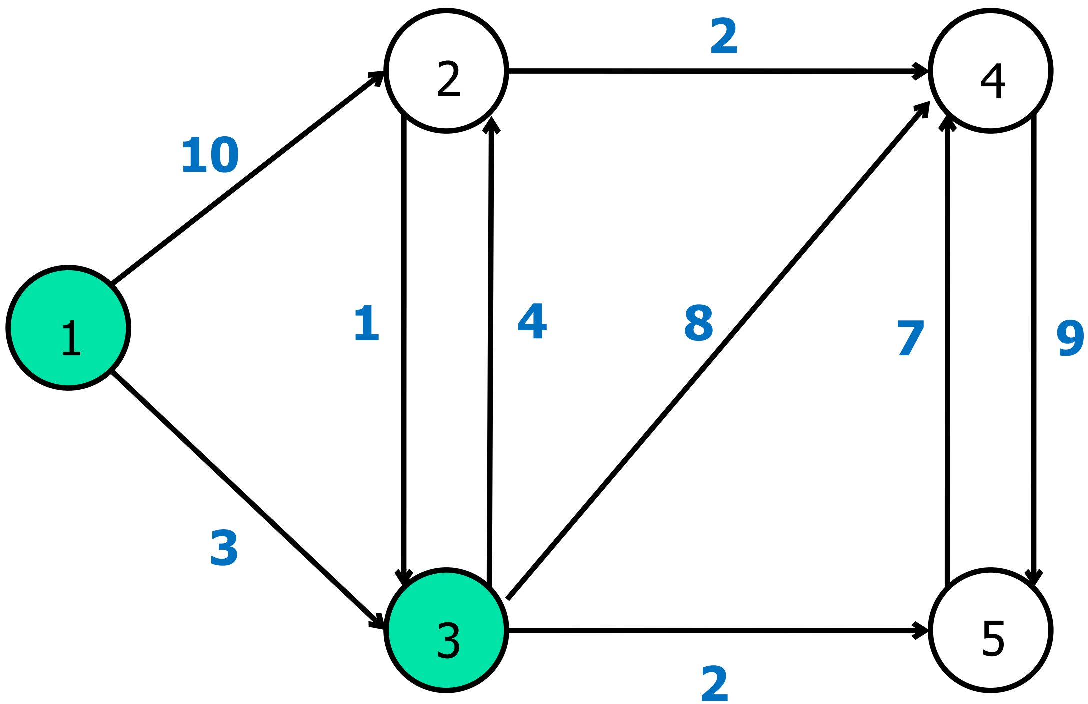
Q
2
4
5

d	v
0	1
7	2
3	3
11	4
5	5

```
enquanto Q ≠ ∅ faça
início
      v := extrairMinimo(Q);
      S := S ∪ {v};
      para todo w ∈ Adj(v)
      faça
         início
              Se d[w] > d[v] + peso(v,w)
              então
                 d[w] := d[v] + peso(v,w);
         fim
fim
```

Algoritmo de Dijkstra : Exemplo

- **Exemplo (**origem no vértice 1**)**



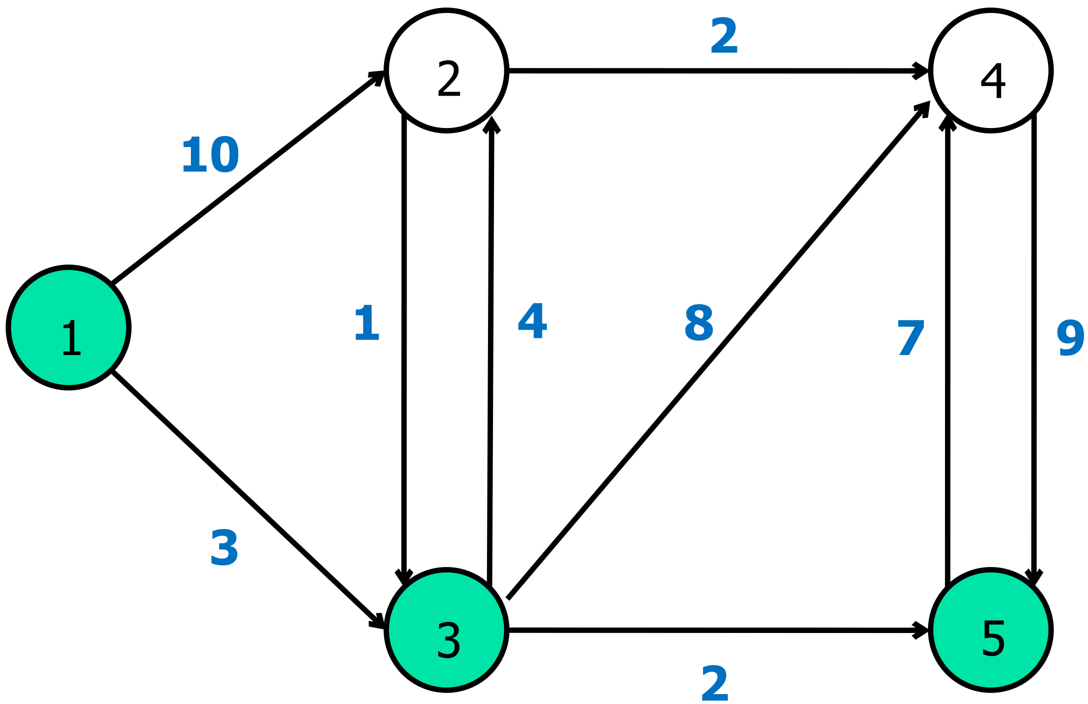
Q
2
4
5

d	v
0	1
7	2
3	3
11	4
5	5

```
enquanto Q ≠ ∅ faça
início
     v := extrairMinimo(Q);
     S := S ∪ {v};
     para todo w ∈ Adj(v)
     faça
        início
            Se d[w] > d[v] + peso(v,w)
            então
              d[w] := d[v] + peso(v,w);
        fim
fim
```

Algoritmo de Dijkstra : Exemplo

- **Exemplo (**origem no vértice 1**)**



Q
2
4

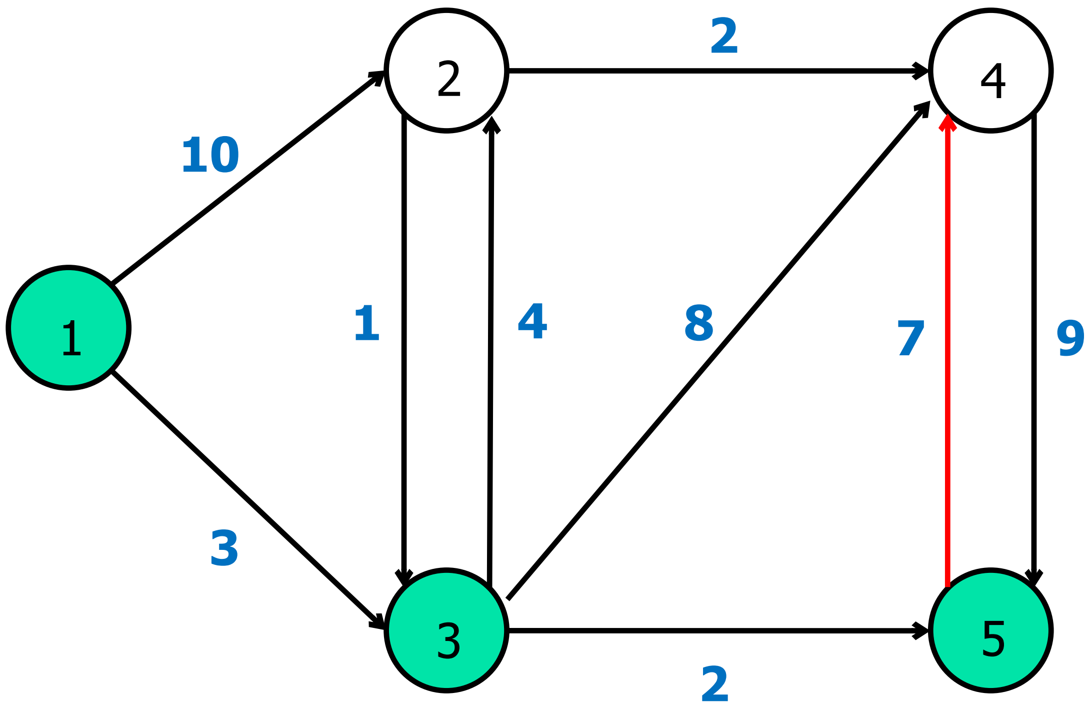
d	v
0	1
7	2
3	3
11	4
5	5

```
enquanto Q ≠ ∅ faça
início
      v := extrairMinimo(Q);
      S := S ∪ {v};
      para todo w ∈ Adj(v)
      faça
          início
               Se d[w] > d[v] + peso(v,w)
               então
                  d[w] := d[v] + peso(v,w);
          fim
fim
```

Algoritmo de Dijkstra : Exemplo

- **Exemplo (**origem no vértice 1**)**

11 < 7+5



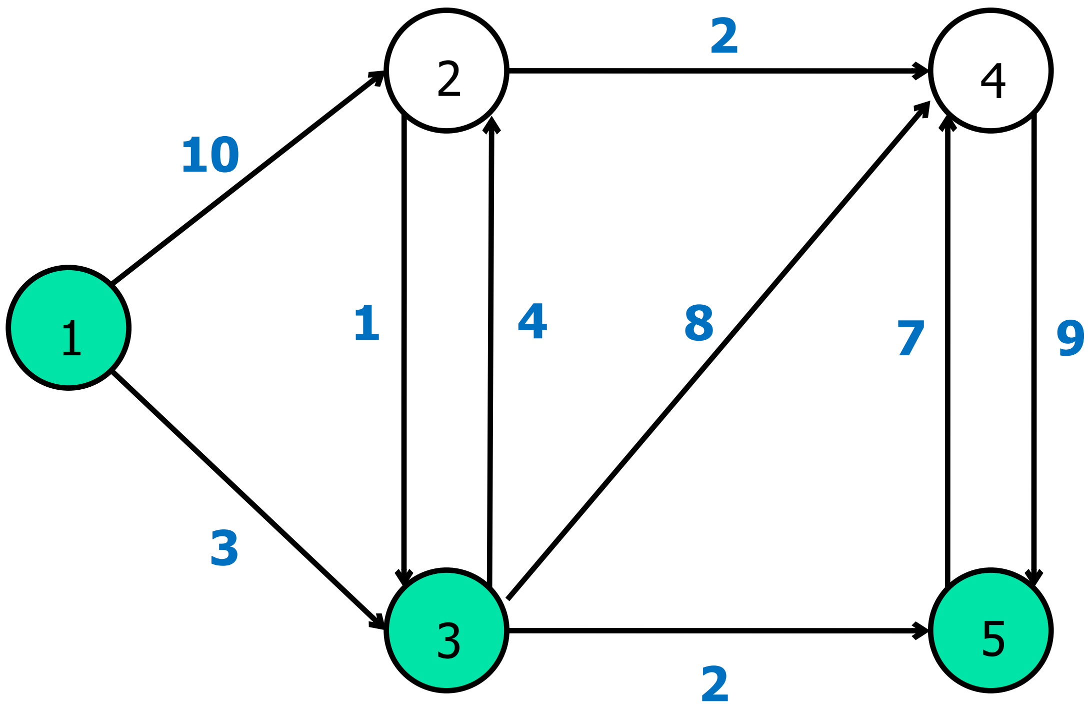
Q
2
4

d	*v*
0	*1*
7	*2*
3	*3*
11	*4*
5	*5*

```
enquanto Q ≠ ∅ faça
início
      v := extrairMinimo(Q);
      S := S ∪ {v};
      para todo w ∈ Adj(v)
      faça
          início
               Se d[w] > d[v] + peso(v,w)
               então
                  d[w] := d[v] + peso(v,w);
          fim
fim
```

Algoritmo de Dijkstra : Exemplo

- **Exemplo (**origem no vértice 1**)**



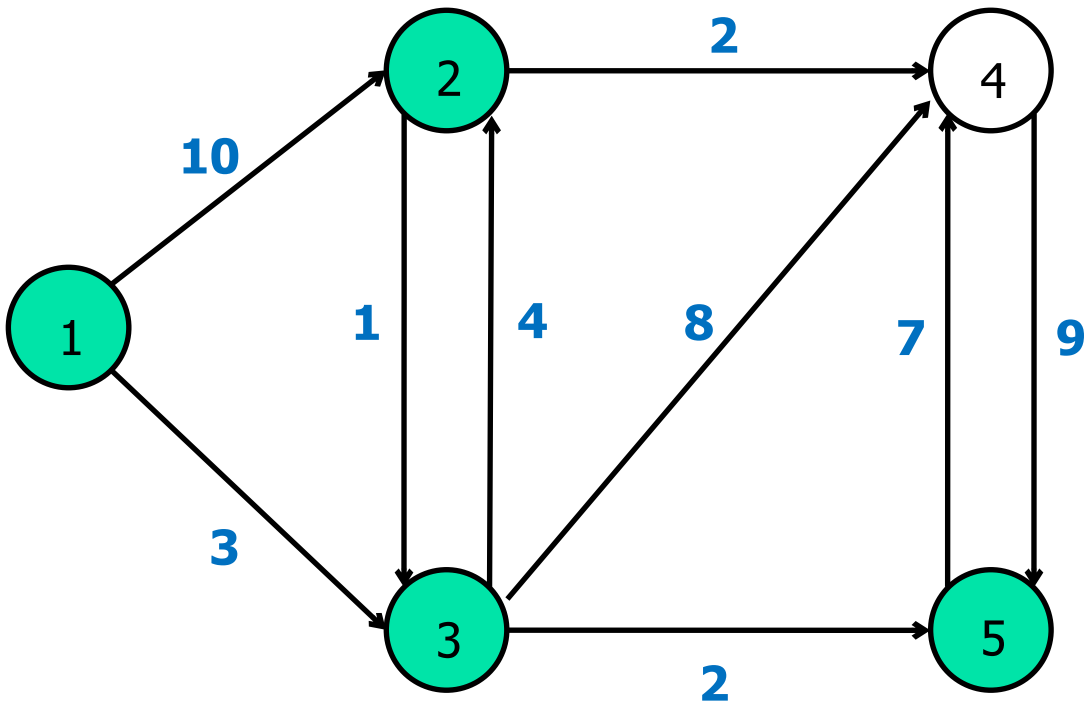
Q
2
4

d	v
0	1
7	2
3	3
11	4
5	5

```
enquanto Q ≠ ∅ faça
início
      v := extrairMinimo(Q);
      S := S ∪ {v};
      para todo w ∈ Adj(v)
      faça
          início
               Se d[w] > d[v] + peso(v,w)
               então
                  d[w] := d[v] + peso(v,w);
          fim
fim
```

Algoritmo de Dijkstra : Exemplo

- **Exemplo (**origem no vértice 1**)**



Q
4

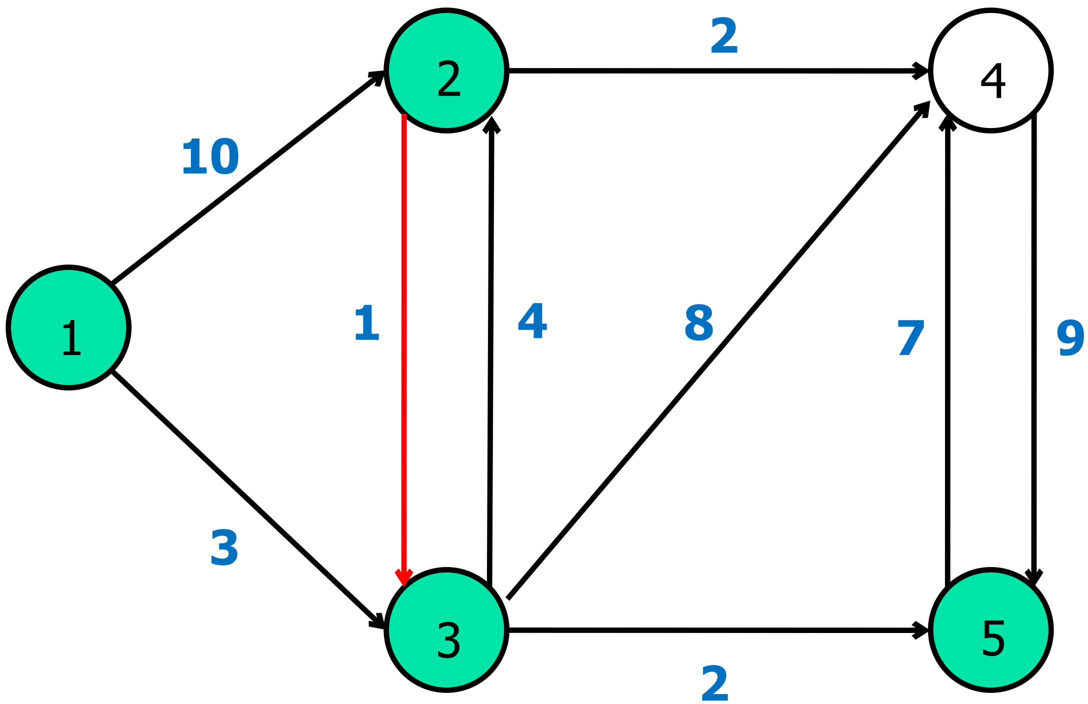
d	v
0	1
7	2
3	3
11	4
5	5

```
enquanto Q ≠ ∅ faça
início
     v := extrairMinimo(Q);
     S := S ∪ {v};
     para todo w ∈ Adj(v)
     faça
        início
            Se d[w] > d[v] + peso(v,w)
            então
              d[w] := d[v] + peso(v,w);
        fim
fim
```

Algoritmo de Dijkstra : Exemplo

- **Exemplo (**origem no vértice 1**)**

3 < 7+1



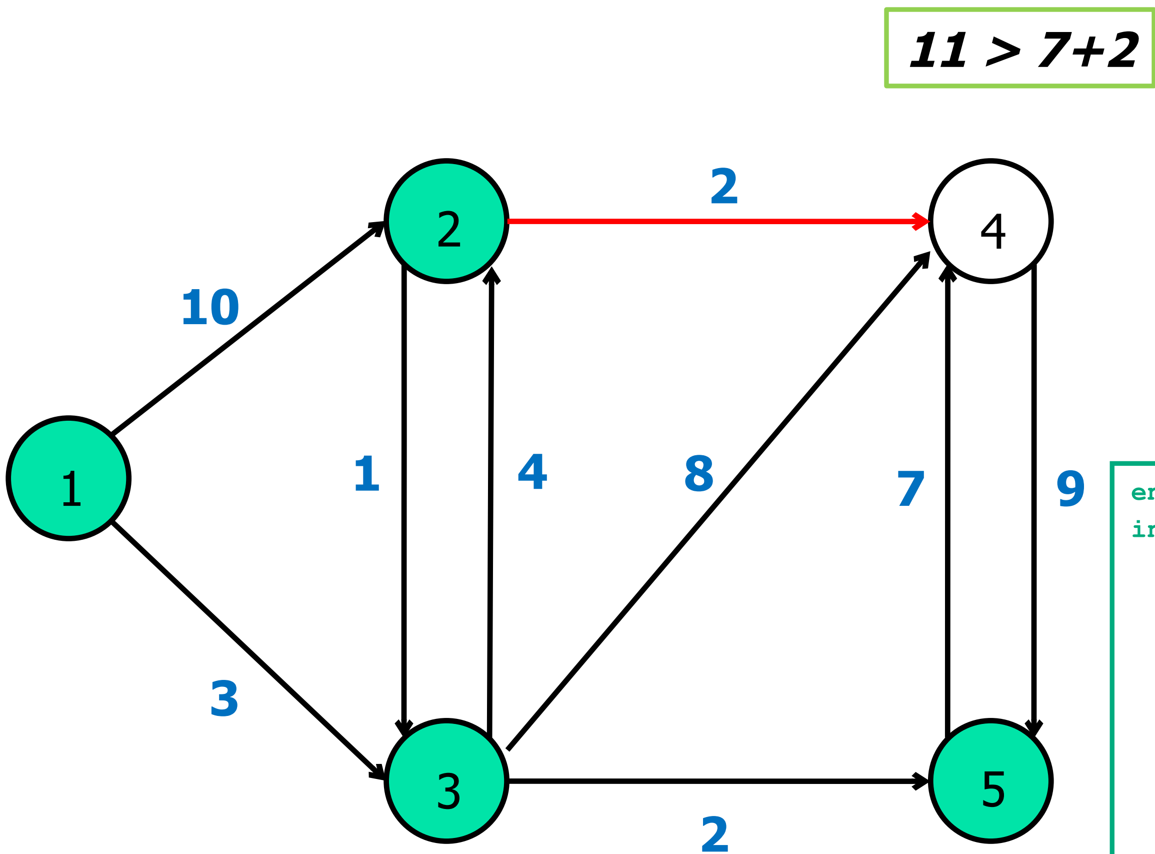
Q
4

d	v
0	*1*
7	*2*
3	*3*
11	*4*
5	*5*

```
enquanto Q ≠ ∅ faça
início
      v := extrairMinimo(Q);
      S := S ∪ {v};
      para todo w ∈ Adj(v)
      faça
          início
               Se d[w] > d[v] + peso(v,w)
               então
                  d[w] := d[v] + peso(v,w);
          fim
fim
```

Algoritmo de Dijkstra : Exemplo

- **Exemplo (**origem no vértice 1**)**



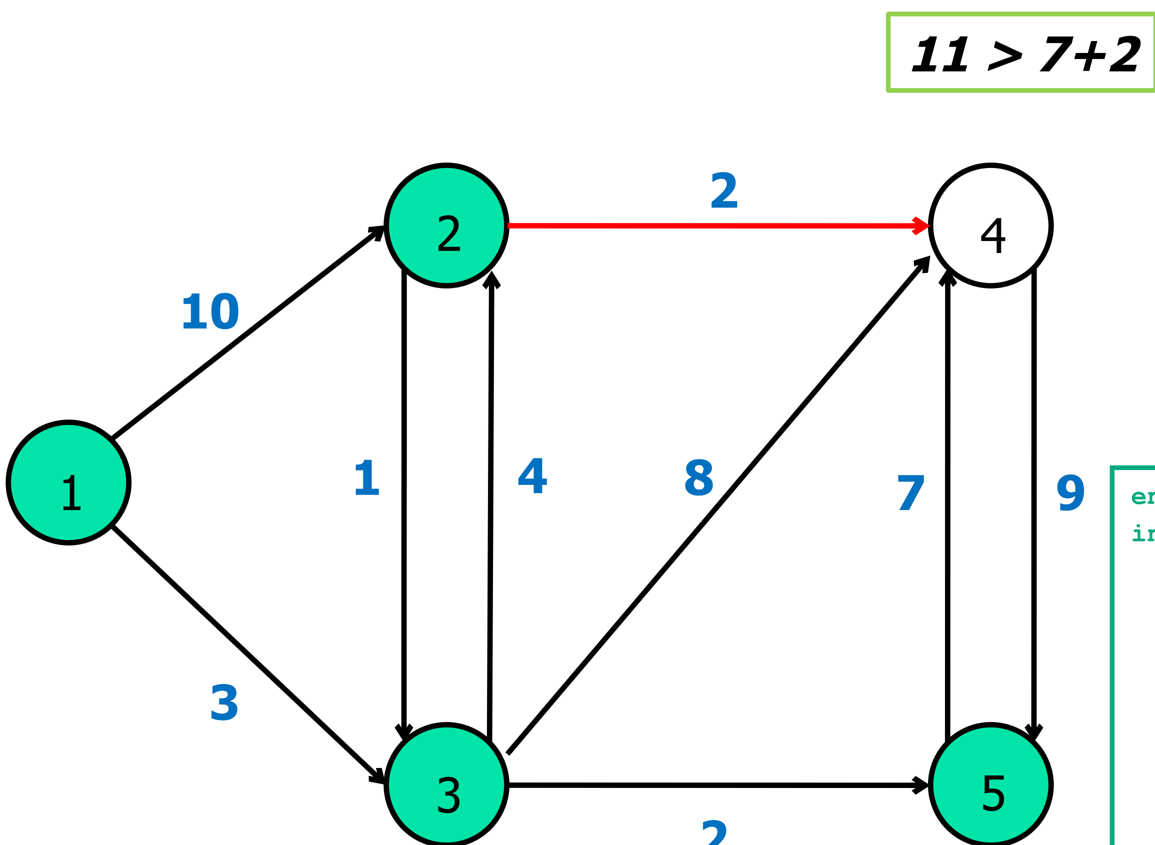
Q
4

d	v
0	1
7	2
3	3
11	4
5	5

```
enquanto Q ≠ ∅ faça
início
      v := extrairMinimo(Q);
      S := S ∪ {v};
      para todo w ∈ Adj(v)
      faça
          início
               Se d[w] > d[v] + peso(v,w)
               então
                  d[w] := d[v] + peso(v,w);
          fim
fim
```

Algoritmo de Dijkstra : Exemplo

- **Exemplo (**origem no vértice 1**)**



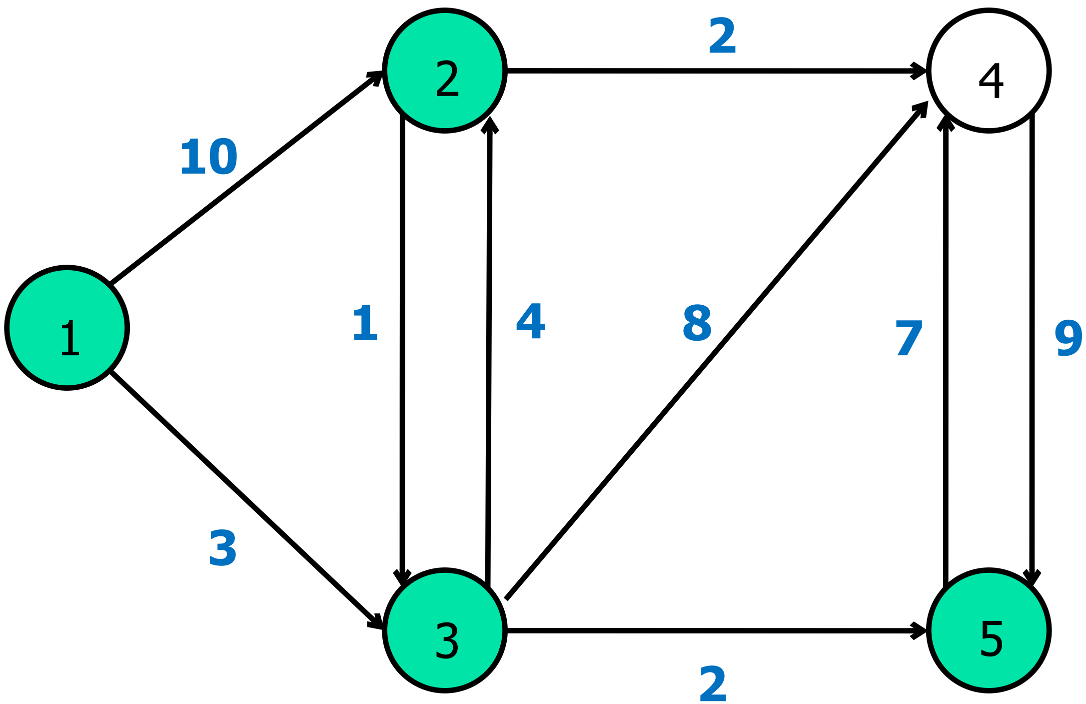
Q
4

d	v
0	1
7	2
3	3
9	4
5	5

```
enquanto Q ≠ ∅ faça
início
     v := extrairMinimo(Q);
     S := S ∪ {v};
     para todo w ∈ Adj(v)
     faça
        início
            Se d[w] > d[v] + peso(v,w)
            então
              d[w] := d[v] + peso(v,w);
        fim
fim
```

Algoritmo de Dijkstra : Exemplo

- **Exemplo (**origem no vértice 1**)**



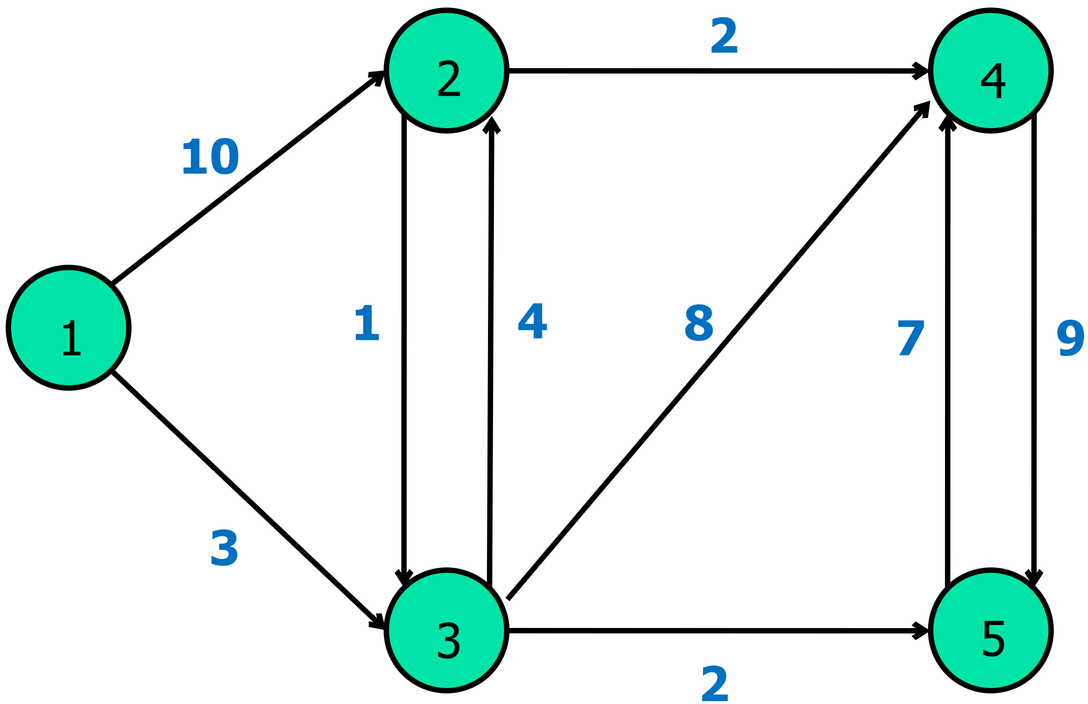
Q
4

d	v
0	1
7	2
3	3
9	4
5	5

```
enquanto Q ≠ ∅ faça
início
      v := extrairMinimo(Q);
      S := S ∪ {v};
      para todo w ∈ Adj(v)
      faça
          início
               Se d[w] > d[v] + peso(v,w)
               então
                  d[w] := d[v] + peso(v,w);
          fim
fim
```

Algoritmo de Dijkstra : Exemplo

- **Exemplo (**origem no vértice 1**)**



Q

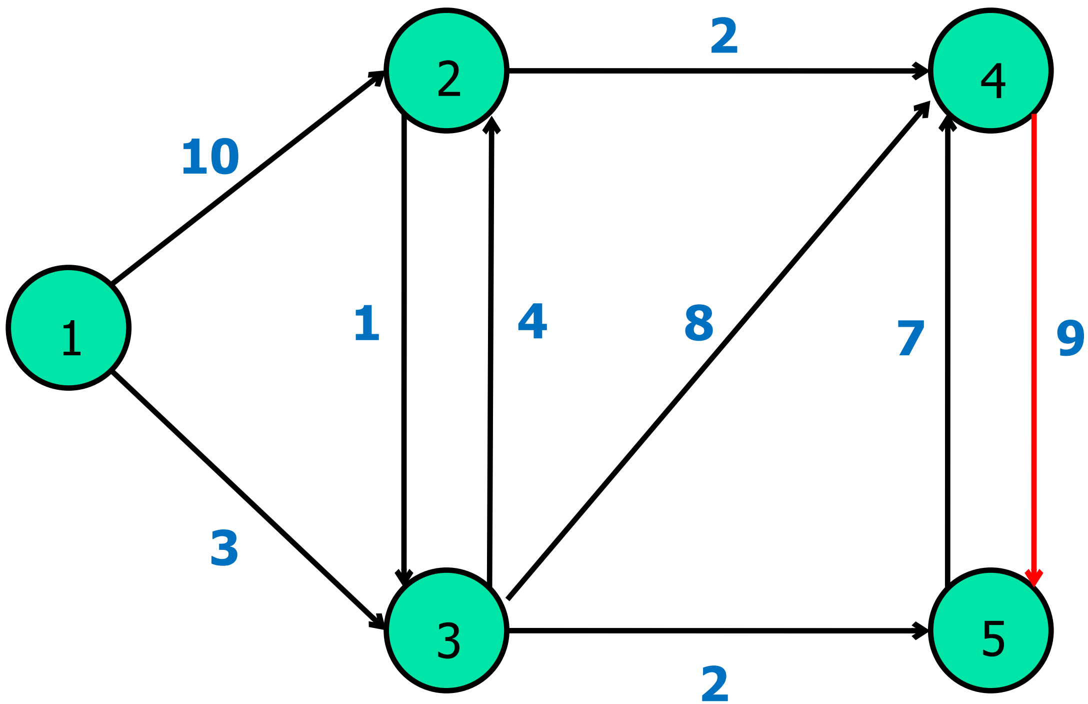
d	v
0	1
7	2
3	3
9	4
5	5

```
enquanto Q ≠ ∅ faça
início
      v := extrairMinimo(Q);
      S := S ∪ {v};
      para todo w ∈ Adj(v)
      faça
          início
               Se d[w] > d[v] + peso(v,w)
               então
                  d[w] := d[v] + peso(v,w);
          fim
fim
```

Algoritmo de Dijkstra : Exemplo

- **Exemplo (**origem no vértice 1**)**

5 < 9+9



Q

d	v
0	1
7	2
3	3
9	4
5	5

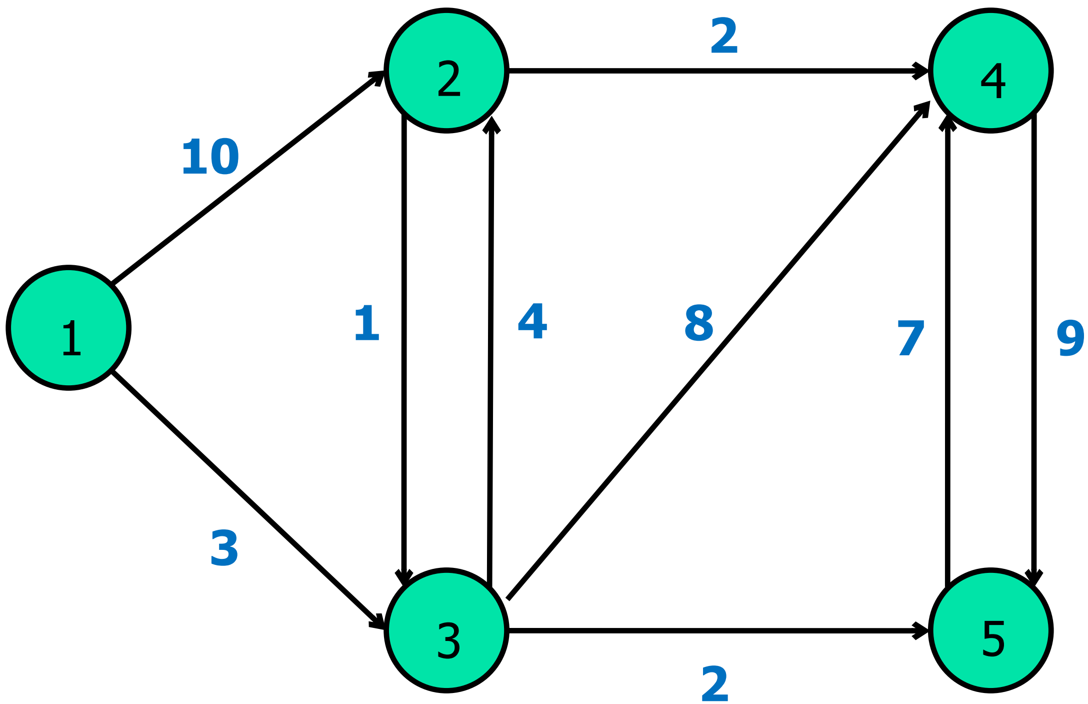
```
enquanto Q ≠ ∅ faça
início
      v := extrairMinimo(Q);
      S := S ∪ {v};
      para todo w ∈ Adj(v)
      faça
         início
              Se d[w] > d[v] + peso(v,w)
              então
                 d[w] := d[v] + peso(v,w);
         fim
fim
```

Algoritmo de Dijkstra : Exemplo

- **Exemplo (**origem no vértice 1**)**

5 < 9+9

Q	d	v
	0	1
	7	2
	3	3
	9	4
	5	5




```
enquanto Q ≠ ∅ faça
início
      v := extrairMinimo(Q);
      S := S ∪ {v};
      para todo w ∈ Adj(v)
      faça
          início
               Se d[w] > d[v] + peso(v,w)
               então
                  d[w] := d[v] + peso(v,w);
          fim
fim
```

Algoritmo de Dijkstra

```
algoritmo dijkstra(G,V,s)
{dados: um grafo G, onde V é seu conjunto de vértices, um
vértice s como origem, d[v] denota distância estimada de
um vértice v ao vértice origem s}
Procedimento inicialização(G,V,s)
início
   S := ∅;
   Q := V;
   d[s] := 0;
   para para todos v ∈ V e v ≠ s faça
      início
         d[v] := ∞;
      fim
fim
```

Algoritmo de Dijkstra

```
algoritmo dijkstra(G,V,s)
{dados: um grafo G, onde V é seu conjunto de vértices, um
vértice s como origem, d[v] denota distãncia estimada de
um vértice v ao vértice origem s}
Procedimento inicialização(G,V,s)
início
   S := ∅;
   Q := V;   ←  Fila de prioridade com chave d[v]
   d[s] := 0;
   para para todos v ∈ V e v ≠ s faça
      início

          d[v] := ∞;
      fim
fim
```

Algoritmo de Dijkstra

```
início
   inicialização(G,V,s);
   enquanto Q ≠ ∅ faça
      início
         v := extrairMinimo(Q);
         S := S ∪ {v};
         para todo w ∈ Adj(v) faça
            início
               Se d[w] > d[v] + peso(v,w) então
                  d[w] := d[v] + peso(v,w);
            fim
      fim
fim
```

Algoritmo de Dijkstra

```
início
    inicialização(G,V,s);
    enquanto Q ≠ ∅ faça
        início
            v := extrairMinimo(Q);
            S := S ∪ {v};
            para todo w ∈ Adj(v) faça
                início
                    Se d[w] > d[v] + peso(v,w) então
                        d[w] := d[v] + peso(v,w);
                fim
        fim
fim
```

O processo de ***relaxar*** uma aresta (***v***, ***w***) consiste em testar se podemos melhorar o caminho mínimo até w que encontramos até agora passando por v e, em caso positivo, atualizar ***d[w]***. Uma etapa de relaxamento pode diminuir o valor da estimativa do caminho mínimo d[w] e atualizar o atributo predecessor de ***w***.

Relaxamento

Algoritmo de Dijkstra: Complexidade

- A complexidade do algoritmo depende diretamente de que estrutura de dados será utilizada para realizar as funções de inserção, mudança de chave, extrair mínimo da fila de prioridade.

```
Procedimento inicialização(G,V,s)
início
    S := ∅;
    Q := V;
    d[s] := 0;
    para para todos v ∈ V e v ≠ s faça
        início
            d[v] := ∞;
        fim
fim
```

Algoritmo de Dijkstra: Complexidade

- Complexidade

```
Procedimento inicialização(G,V,s)
início
    S := ∅;        →    O(1)
    Q := V;
    d[s] := 0;
    para para todos v ∈ V e v ≠ s faça
        início
            d[v] := ∞;
        fim
fim
```

Algoritmo de Dijkstra: Complexidade

- Complexidade

```
Procedimento inicialização(G,V,s)
início
    S = ∅;        →    O(1)
    Q = V;        →    O(|V|)
    d[s] = 0;
    para para todos v ∈ V e v ≠ s faça
        início
            d[v] = ∞;
        fim
fim
```

Algoritmo de Dijkstra: Complexidade

- Complexidade

```
Procedimento inicialização(G,V,s)
início
    S = ∅;      →    O(1)
    Q = V;      →    O(|V|)
    d[s] = 0;   →    O(1)
    para para todos v ∈ V e v ≠ s faça
        início
            d[v] = ∞;
        fim
fim
```

Algoritmo de Dijkstra: Complexidade

- Complexidade

```
Procedimento inicialização(G,V,s)
início
    S = ∅;          →    O(1)
    Q = V;          →    O(|V|)
    d[s] = 0;       →    O(1)
    para para todos v ∈ V e v ≠ s faça  ┐
        início                          │
            d[v] = ∞;                   ├  O(|V|)
        fim                             ┘
fim
```

Algoritmo de Dijkstra: Complexidade

- Complexidade $O(|V|)$

```
Procedimento inicialização(G,V,s)
início
    S := ∅;          →   O(1)
    Q := V;          →   O(|V|)
    d[s] := 0;       →   O(1)
    para para todos v ∈ V e v ≠ s faça ┐
        início                        │
            d[v] := ∞;                ├  O(|V|)
        fim                           ┘
fim
```

Algoritmo de Dijkstra: Complexidade

- Complexidade $O(|V|)$

```
enquanto Q ≠ ∅ faça
      início
         v := extrairMinimo(Q);
         S := S ∪ {v};
         para todo w ∈ Adj(v) faça
            início
               Se d[w] > d[v] + peso(v,w) então
                  d[w] := d[v] + peso(v,w);
            fim
      fim
```

Algoritmo de Dijkstra: Complexidade

- Complexidade $O(|V|)$

```
enquanto Q ≠ ∅ faça
       início
          v := extrairMinimo(Q);  → O(|V||Extrair Mínimo|)
          S := S ∪ {v};
          para todo w ∈ Adj(v) faça
             início
                 Se d[w] > d[v] + peso(v,w) então
                    d[w] := d[v] + peso(v,w);
             fim
       fim
```

Algoritmo de Dijkstra: Complexidade

- Complexidade $O(|V|)$

```
enquanto Q ≠ ∅ faça
      início
         v := extrairMinimo(Q);   → O(|V||Extrair Mínimo|)
         S := S ∪ {v};            → O(|V|)
         para todo w ∈ Adj(v) faça
            início
               Se d[w] > d[v] + peso(v,w) então
                  d[w] := d[v] + peso(v,w);
            fim
      fim
```

Algoritmo de Dijkstra: Complexidade

- Complexidade *O(|V|)*

```
enquanto Q ≠ ∅ faça
      início
         v := extrairMinimo(Q);   → O(|V||Extrair Mínimo|)
         S := S ∪ {v};              → O(|V|)
         para todo w ∈ Adj(v) faça
            início
               Se d[w] > d[v] + peso(v,w) então
                  d[w] := d[v] + peso(v,w);
            fim
      fim
```

$$\left.\begin{array}{l} \text{Se } d[w] > d[v] + peso(v,w) \text{ então} \\ \quad d[w] := d[v] + peso(v,w); \end{array}\right\} O(|E||\text{Mudança de Chave}|)$$

Algoritmo de Dijkstra: Complexidade

- Complexidade

$O(|V|) + O(|V||\text{Extrair Mínimo}|) + O(|E||\text{Mudança de Chave}|)$

```
enquanto Q ≠ ∅ faça
      início
         v := extrairMinimo(Q);     → O(|V||Extrair Mínimo|)
         S := S ∪ {v};              → O(|V|)
         para todo w ∈ Adj(v) faça
            início
               Se d[w] > d[v] + peso(v,w) então
                  d[w] := d[v] + peso(v,w);
            fim
      fim
```

$\left.\begin{array}{l}\text{início} \\ \text{Se } d[w] > d[v] + peso(v,w) \text{ então} \\ \quad d[w] := d[v] + peso(v,w); \\ \text{fim}\end{array}\right\} O(|E||\text{Mudança de Chave}|)$

Algoritmo de Dijkstra: Complexidade

- Ex.: Utilizando um Heap

$O(|V|) + O(|V| \log|V|) + O(|E| \log|V|) = O(\ (|V|+|E|) \log|V|\ )$

```
enquanto Q ≠ ∅ faça
      início
         v := extrairMinimo(Q);   → O(|V||Extrair Mínimo|)
         S := S ∪ {v};            → O(|V|)
         para todo w ∈ Adj(v) faça
            início
               Se d[w] > d[v] + peso(v,w) então
                  d[w] := d[v] + peso(v,w);
            fim
      fim
```

O(|E||Mudança de Chave|)

Single Source Shortest Path

- E se todos os pesos forem 1 ? Existe uma estratégia melhor ?

Single Source Shortest Path

- E se todos os pesos forem 1 ? Existe uma estratégia melhor ?
  - Sim! Trocar a Fila de Prioridade por uma **Fila.**
  - Que algoritmo é este e qual sua complexidade ?

Single Source Shortest Path

- E se todos os pesos forem 1 ? Existe uma estratégia melhor ?
  - Sim! Trocar a Fila de Prioridade por uma **Fila**.
  - Que algoritmo é este e qual sua complexidade ?
    - Busca em largura.
    - Complexidade $O(|V|+|E|)$

Referências

- Seção 24.3 do Cormen, *Introduction to Algorithms,* MIT Press, 2001.

Adaptado do material da Profa. Leila Silva.

Seção 3.3 do Grafos: conceitos, algoritmos e aplicações. Goldbarg, E. e Goldbarg M. Elsevier, 2012