Grafos: caminhos mínimos
Parte 1

SCC0216 Modelagem Computacional em Grafos

Thiago A. S. Pardo
Maria Cristina F. Oliveira

O problema do menor caminho

- Um motorista deseja encontrar o caminho mais curto possível entre duas cidades do Brasil
- Caso ele receba um mapa rodoviário do Brasil, que apresenta a distância entre cada par de cidades adjacentes, como determinar uma rota mais curta entre as cidades desejadas?
- Uma maneira possível é enumerar todas as rotas possíveis que levam de uma cidade à outra, e então selecionar a mais curta

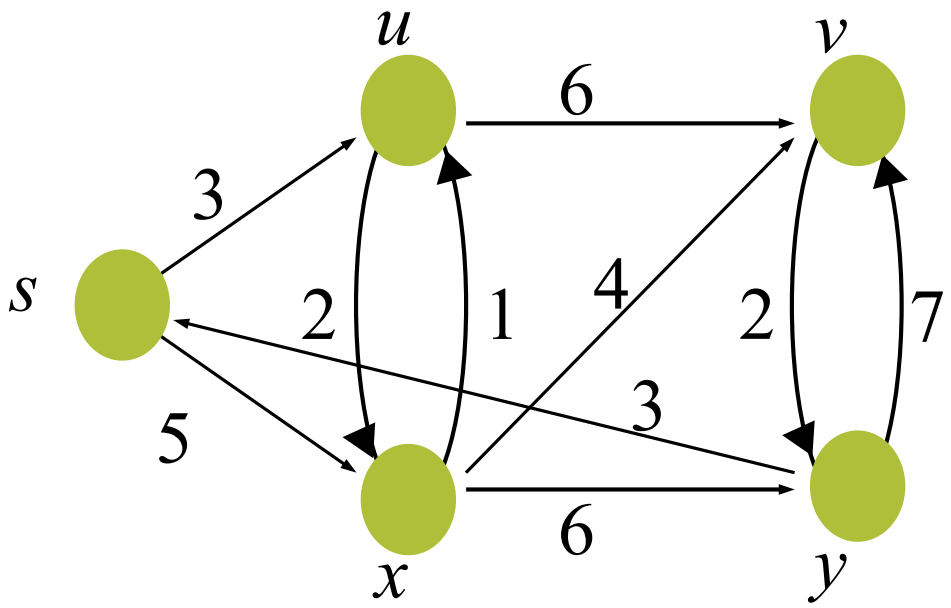
O problema do menor caminho

- Menor caminho = caminho de menor custo = melhor caminho
- Custo = distância ou qualquer outra métrica

Caminhos mínimos

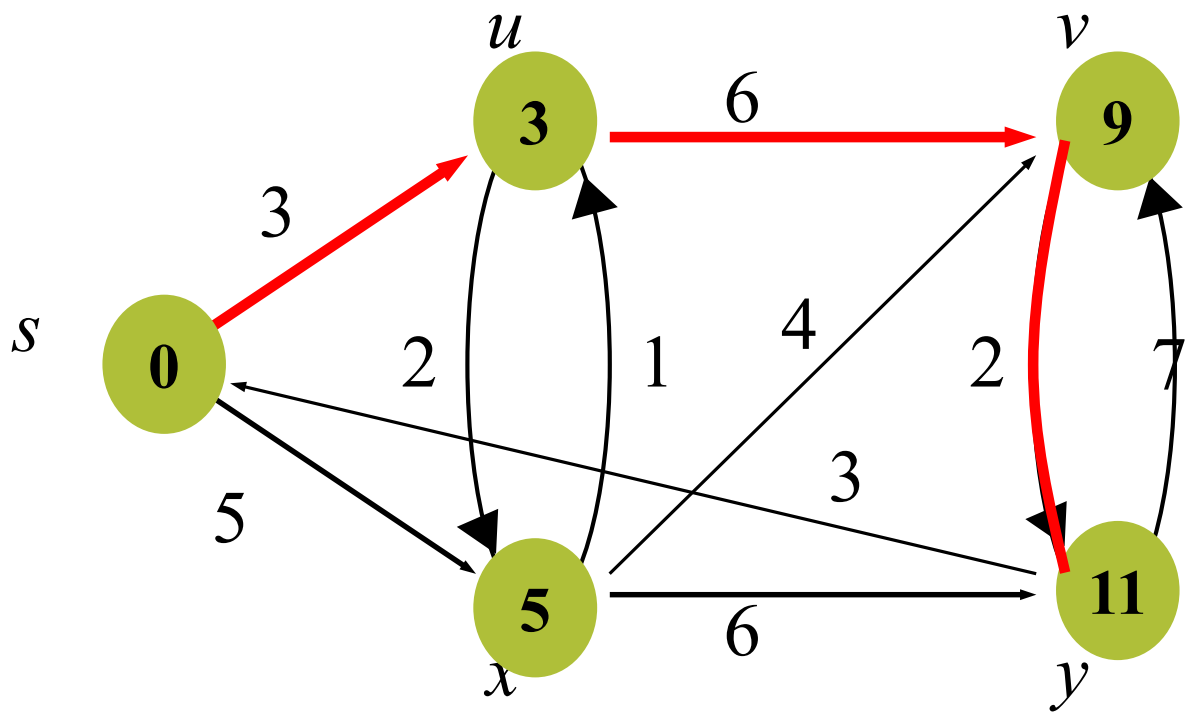
- O problema do caminho mínimo consiste em determinar um menor caminho entre um vértice de origem *u* e um vértice de destino *v*

- Qual o menor caminho entre *s* e *y*?



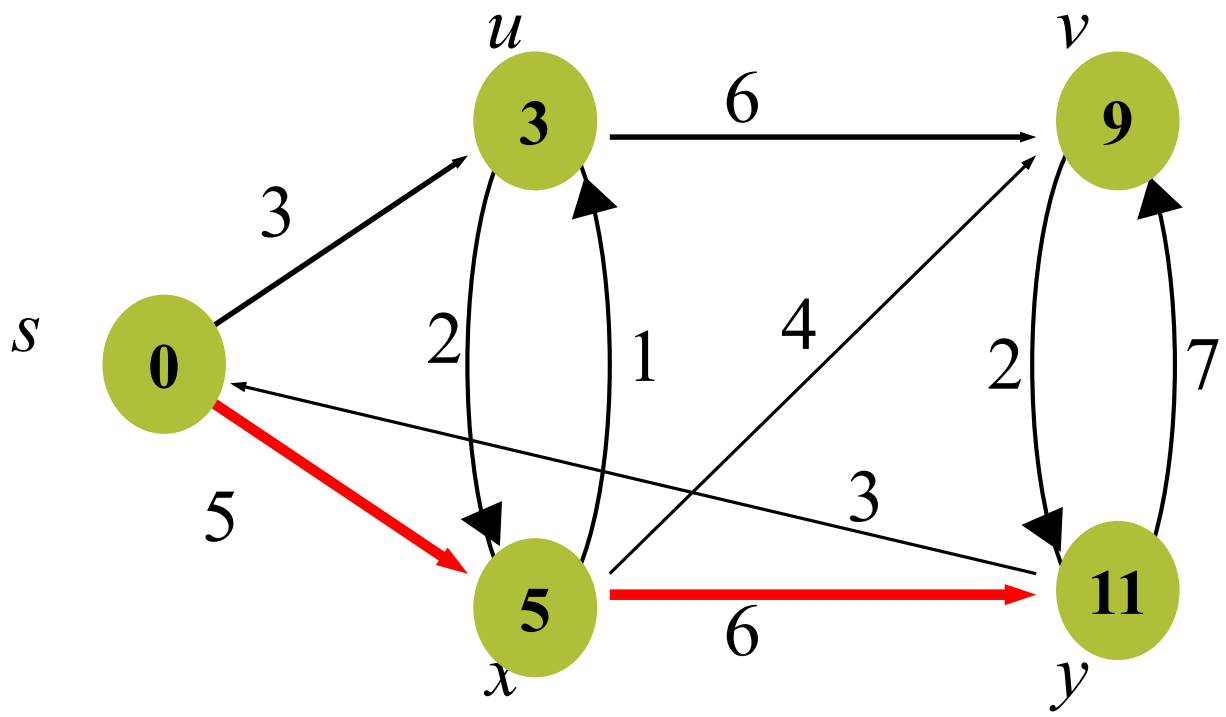
Caminhos mínimos

- Possíveis caminhos



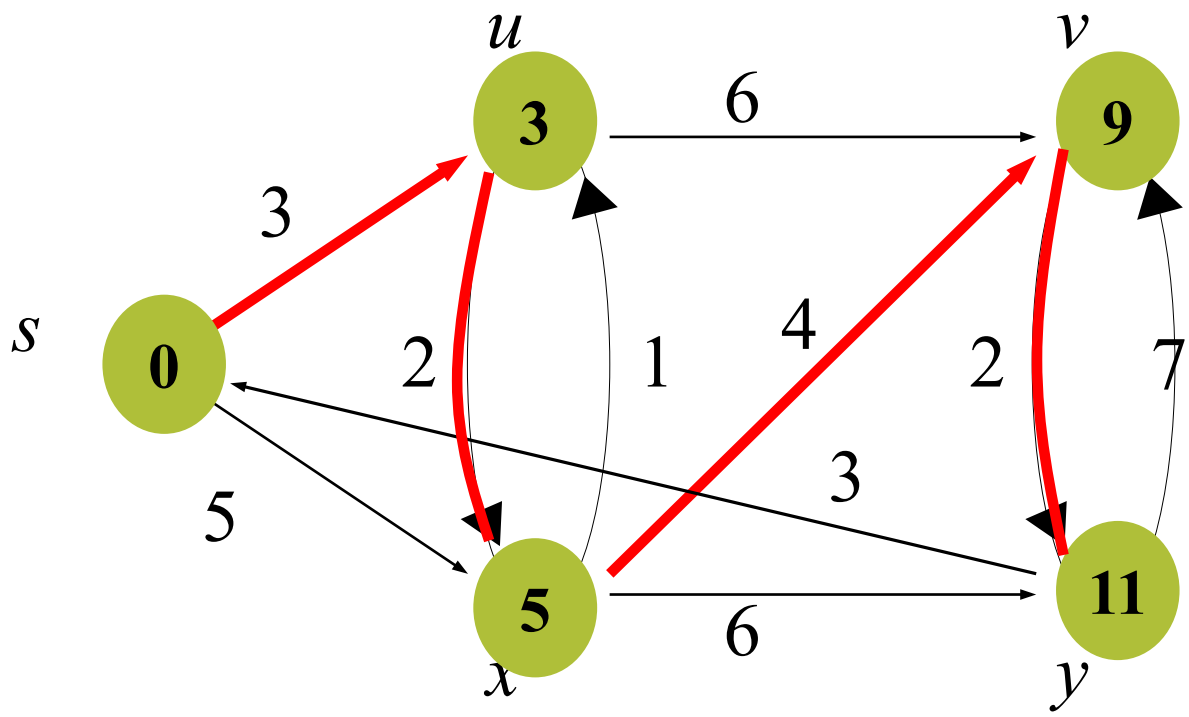
Caminhos mínimos

- Possíveis caminhos



Caminhos mínimos

- Possíveis caminhos



Caminho mínimo

- Duas abordagens para caminho mínimo
  - Se grafo não valorado (assume-se que cada aresta tem peso 1), busca em largura é uma boa opção
  - Se grafo valorado, outros algoritmos são necessários

Caminho mínimo

- Grafo dirigido G(V,A) com função peso *w*: A$\rightarrow\Re$ que associa pesos às arestas

- Peso (custo) do caminho p = <v_0, v_1, ..., v_k>

$$w(p) = \sum_{i=1}^{k} w(v_{i-1}, v_i)$$

Caminho mínimo

- Grafo dirigido G(V,A) com função peso *w*: A$\rightarrow\Re$ que associa pesos às arestas
- Peso (custo) do caminho p = <v_0, v_1, ..., v_k>

$$w(p) = \sum_{i=1}^{k} w(v_{i-1}, v_i)$$

- Caminho de menor peso entre *u* e *v*:

$$\delta(u,v) = \begin{cases} \min\{w(p) : u \overset{p}{\Rightarrow} v\} \; se \exists \; rota \; de \; u \;\; p/v \\ \infty \; cc \end{cases}$$

Caminho mínimo

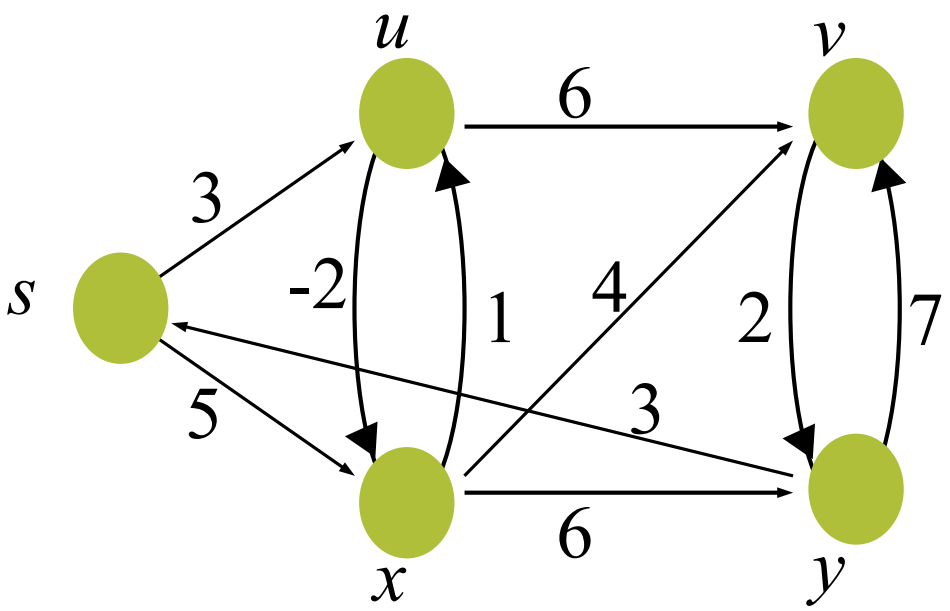
- Menor caminho entre os vértices *u* e *v* definido como qualquer rota **p** com um peso

$$w(p) = \delta(u,v)$$

- Atenção especial com ciclos e pesos negativos

Caminho mínimo

- Qual o menor caminho entre *s* e *v*?

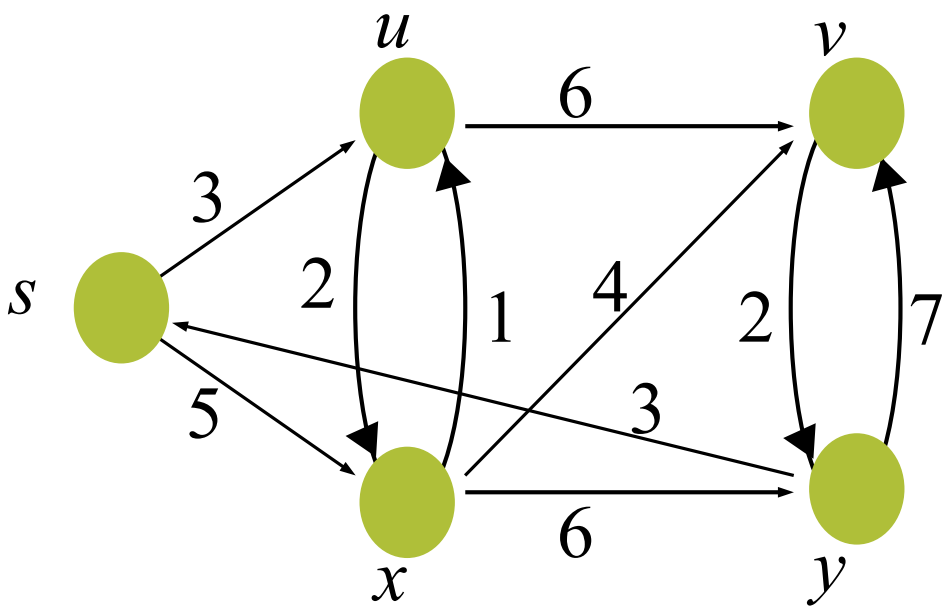


Camino mínimo

- Se há um ciclo positivo no caminho, ele não faz parte do caminho mínimo
  - Por quê?

Camino mínimo

- Se há um ciclo positivo no caminho, ele não faz parte do caminho mínimo
  - Por quê?

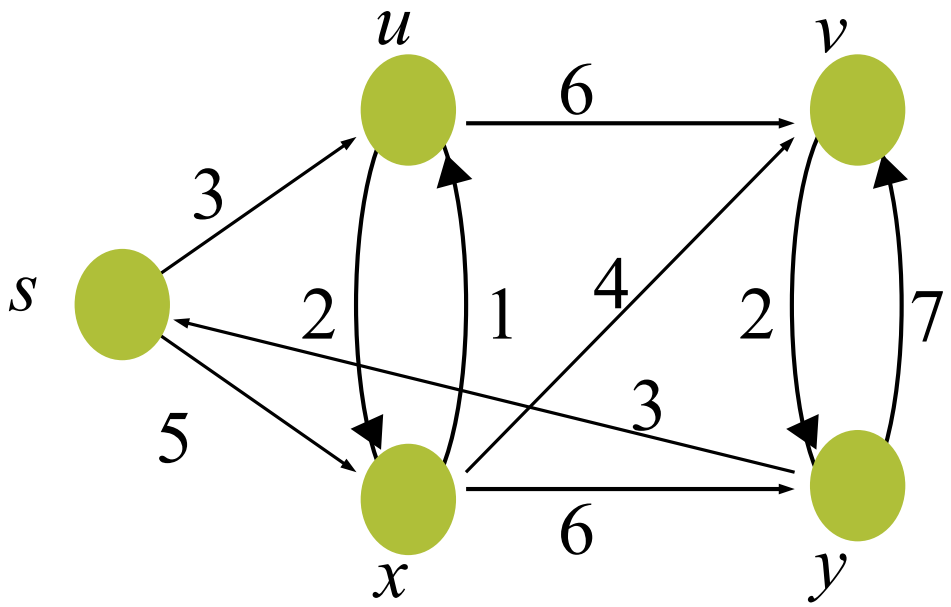


Caminhos mínimos

- Conceitos
  - Parte-se do vértice inicial *s*, associando-se a cada vértice um número d(*v*) que informa o valor (distância) do menor caminho de *s* até *v*

Caminhos mínimos

- Conceitos
  - Por exemplo, na figura abaixo, quando chegamos ao vértice *v*, d(*v*) será dado por min(d(*u*)+6, d(*x*)+4, d(*y*)+7)

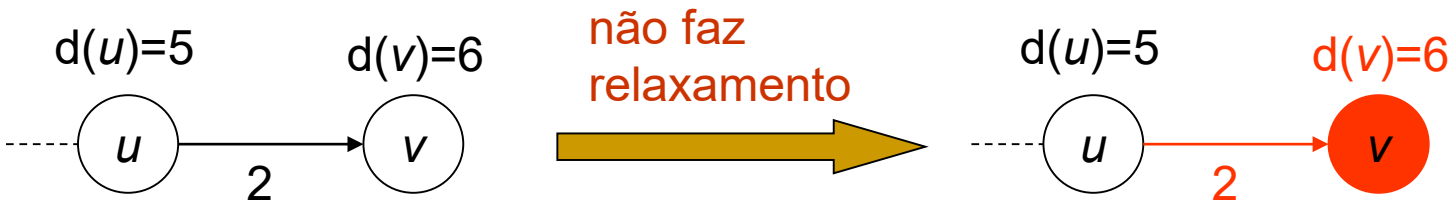


Caminhos mínimos

- Conceitos
  - *Relaxamento* de arestas
    - Faz-se uma estimativa pessimista para o valor do caminho mínimo até cada vértice: d(v) = ∞
    - O processo de 'relaxar' uma aresta consiste em verificar se é possível melhorar esta estimativa pessimista, passando-se pelo vértice u

Caminhos mínimos

- Conceitos
  - *Relaxamento* de arestas
    - Faz-se uma estimativa pessimista para o caminho mínimo até cada vértice: d(v) = ∞
    - O processo de relaxar uma aresta consiste em verificar se é possível melhorar esta estimativa, passando-se pelo vértice u



Caminhos mínimos

- Sub-rotina para relaxamento de arestas

relax(*u*, *v*, *w*) // (*u*,*v*) é a aresta, w é o seu peso

início

se d[*v*] > d[*u*] + w(*u*,*v*) **então**

d[*v*] = d[*u*] + w(*u*,*v*)

antecessor[*v*]=*u*

fim

Caminhos mínimos

- Sub-rotina para relaxamento de arestas

relax(*u*, *v*, *w*) // (*u*,*v*) é a aresta, w é o seu peso

início

se d[*v*] > d[*u*] + w(*u*,*v*) **então**

d[*v*] = d[*u*] + w(*u*,*v*)

antecessor[*v*]=*u*

fim

Caminhos mínimos

- Sub-rotina para relaxamento de arestas

relax(*u*, *v*, *w*) // (*u*,*v*) é a aresta, w é o seu peso

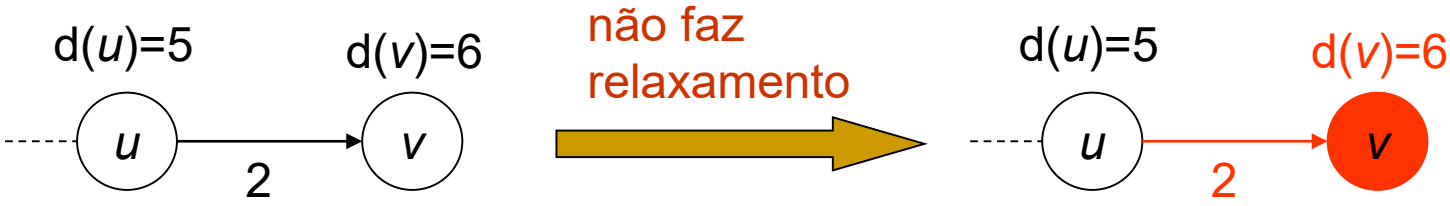
início

se d[*v*] > d[*u*] + w(*u*,*v*) **então**

d[*v*] = d[*u*] + w(*u*,*v*)

antecessor[*v*]=*u*

fim



Caminho mínimo

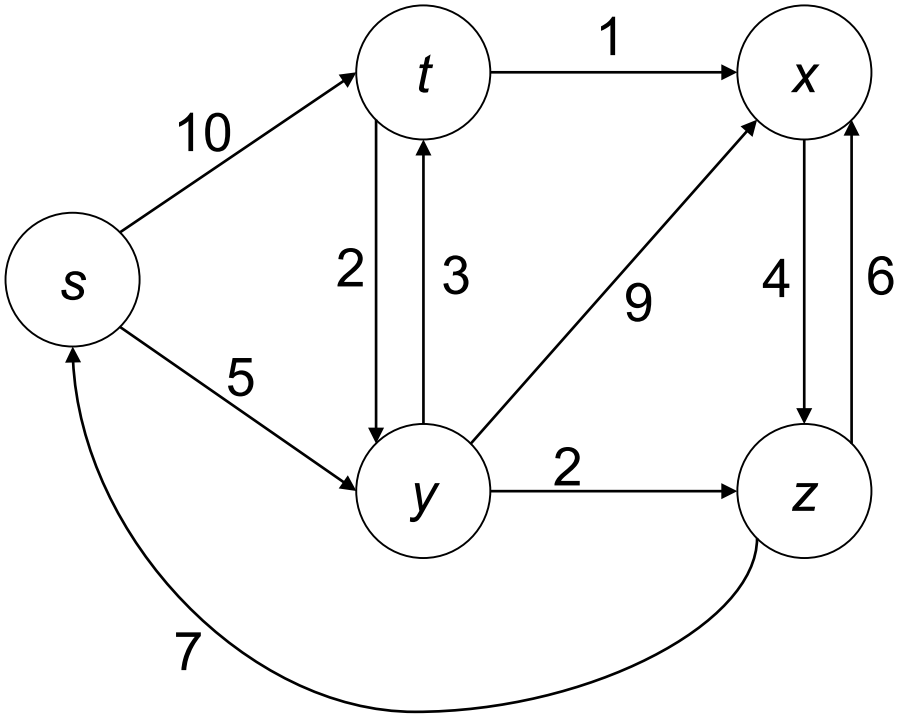
- Várias possibilidades de caminhos
  - Caminhos mais curtos de origem única
  - Caminhos mais curtos de destino único
  - Caminho mais curto de par único
  - Caminhos mais curtos de todos os pares

Algoritmo de Dijkstra

- Características
  - Caminho mais curto de origem única
  - Admite ciclos
  - Só admite pesos positivos
- Método
  - Inicia-se com estimativas pessimistas para cada vértice
  - A cada passo, adiciona um vértice *u* de menor estimativa de caminho mínimo a um conjunto S
    - S contém os vértices com caminhos mínimos desde a origem já definidos
  - Relaxam-se as arestas adjacentes a *u*

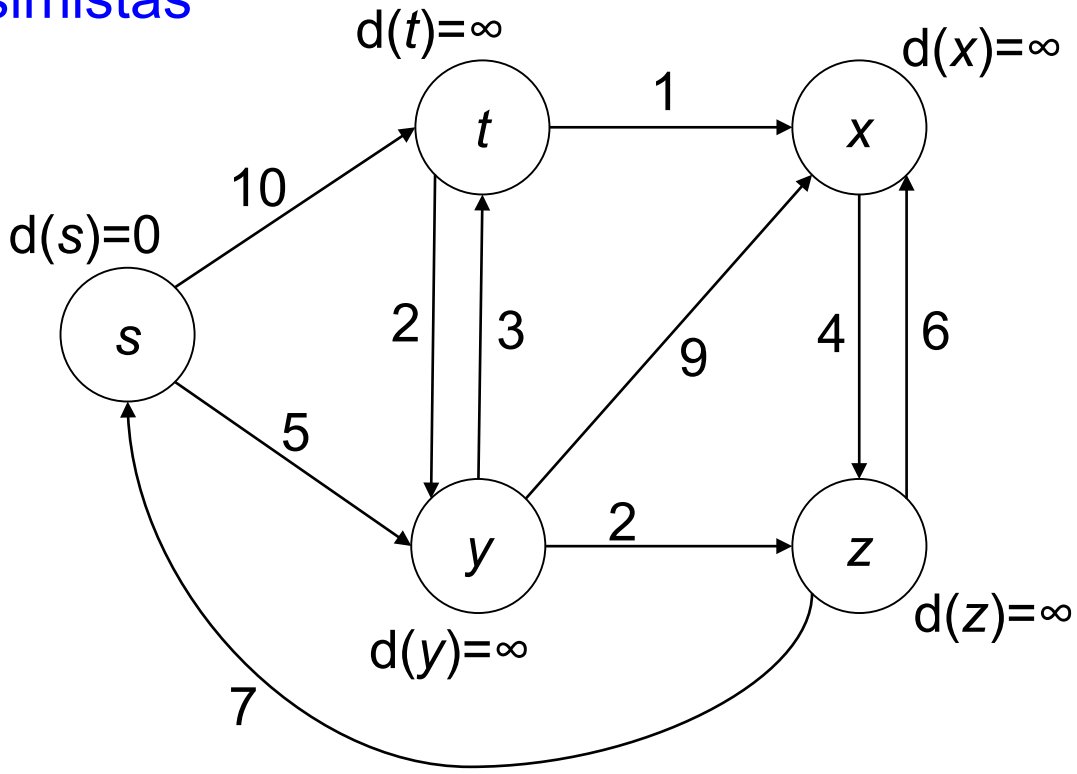
Algoritmo de Dijkstra

- Exemplo (a partir de *s*)



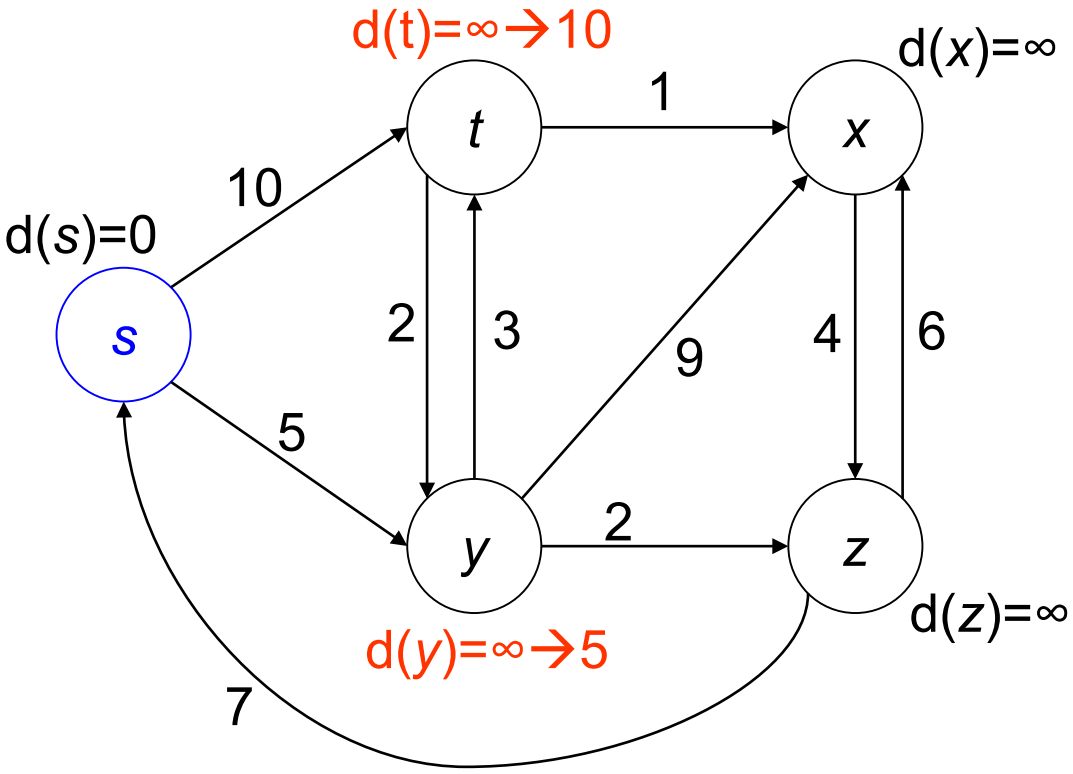
Algoritmo de Dijkstra

- Exemplo (a partir de *s*)
  - Estimativas pessimistas
  - S = Ø



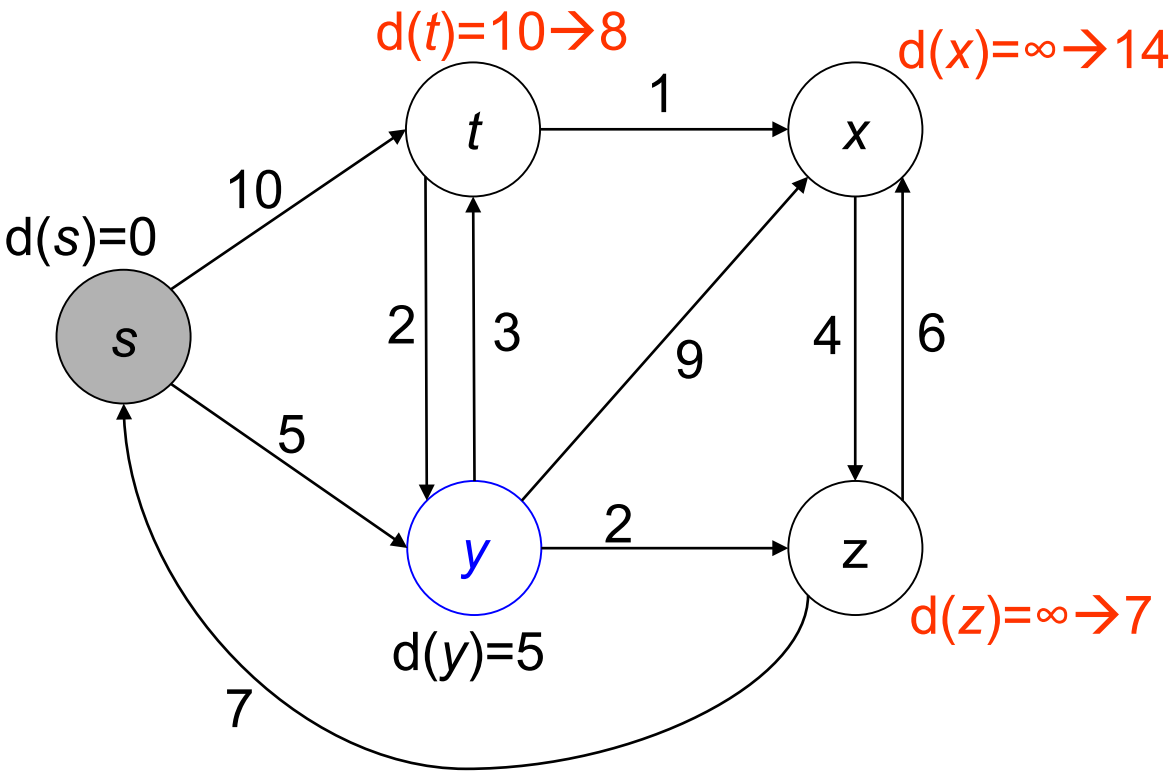
Algoritmo de Dijkstra

- Exemplo (a partir de s)
  - Adiciona s a S
  - S={**s**}



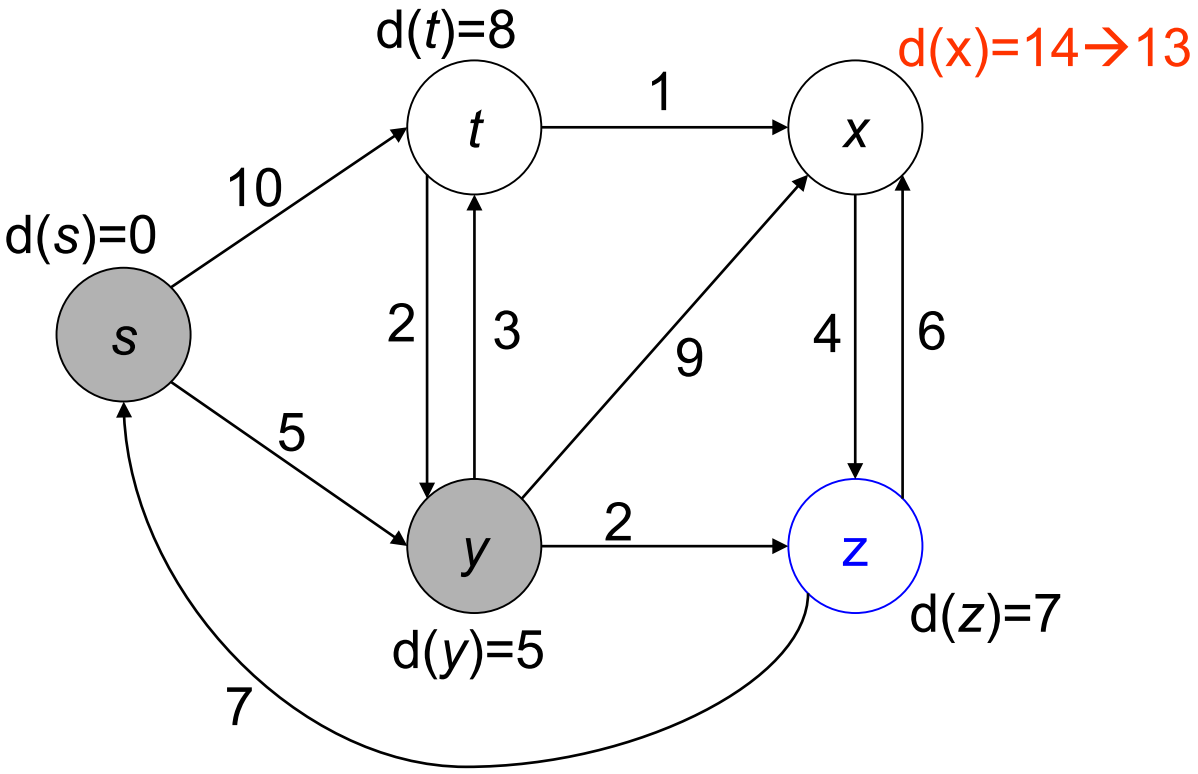
Algoritmo de Dijkstra

- Exemplo (a partir de s)
  - Adiciona y a S
  - S={s,**y**}



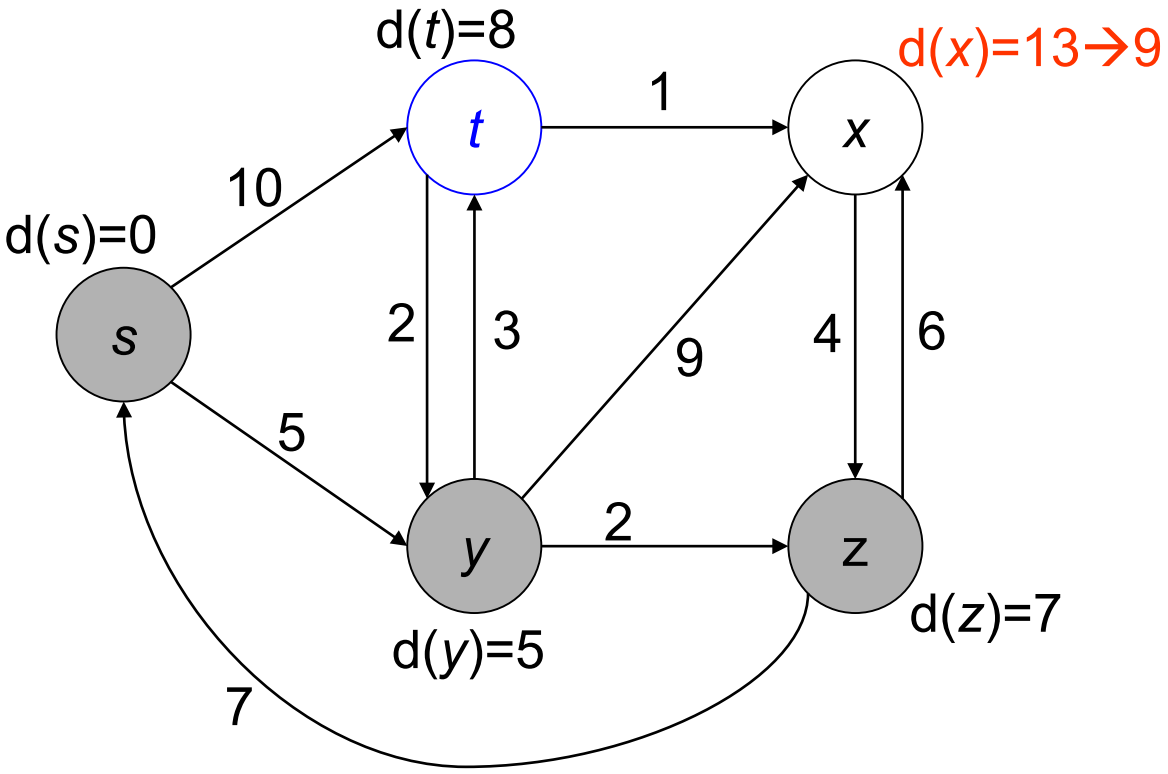
Algoritmo de Dijkstra

- Exemplo (a partir de s)
  - Adiciona *z* a S
  - S={*s*,*y*,***z***}



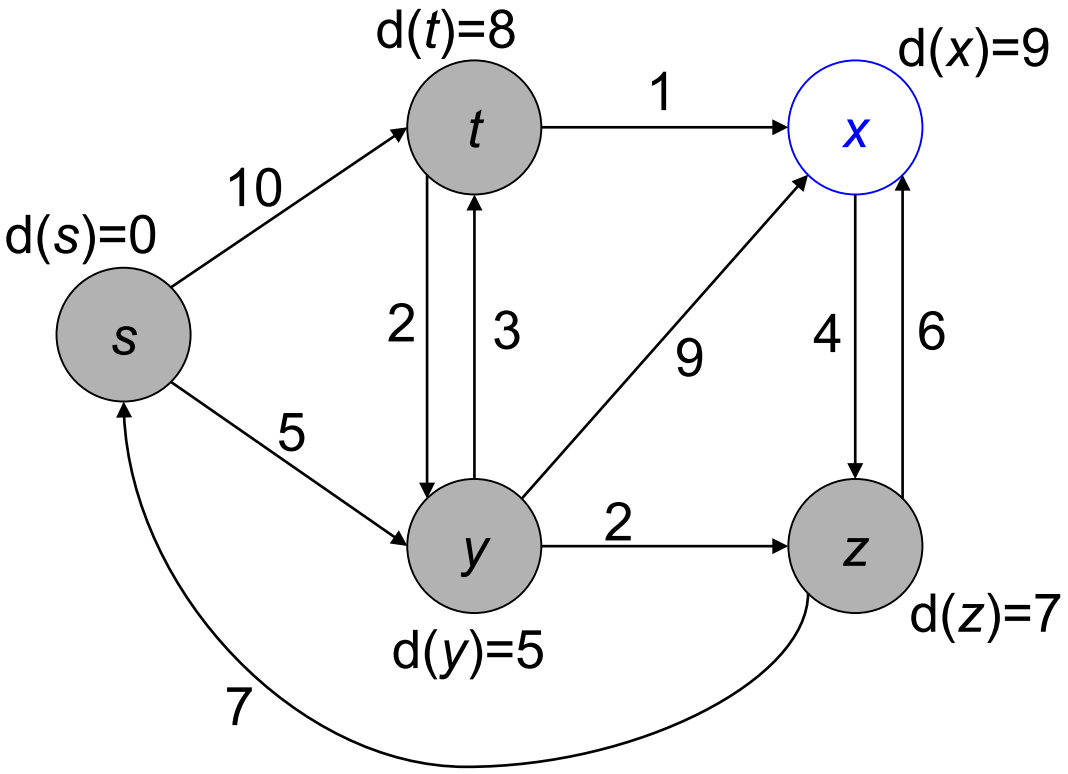
Algoritmo de Dijkstra

- Exemplo (a partir de *s*)
  - Adiciona *t* a S
  - S={*s*,*y*,*z*,***t***}



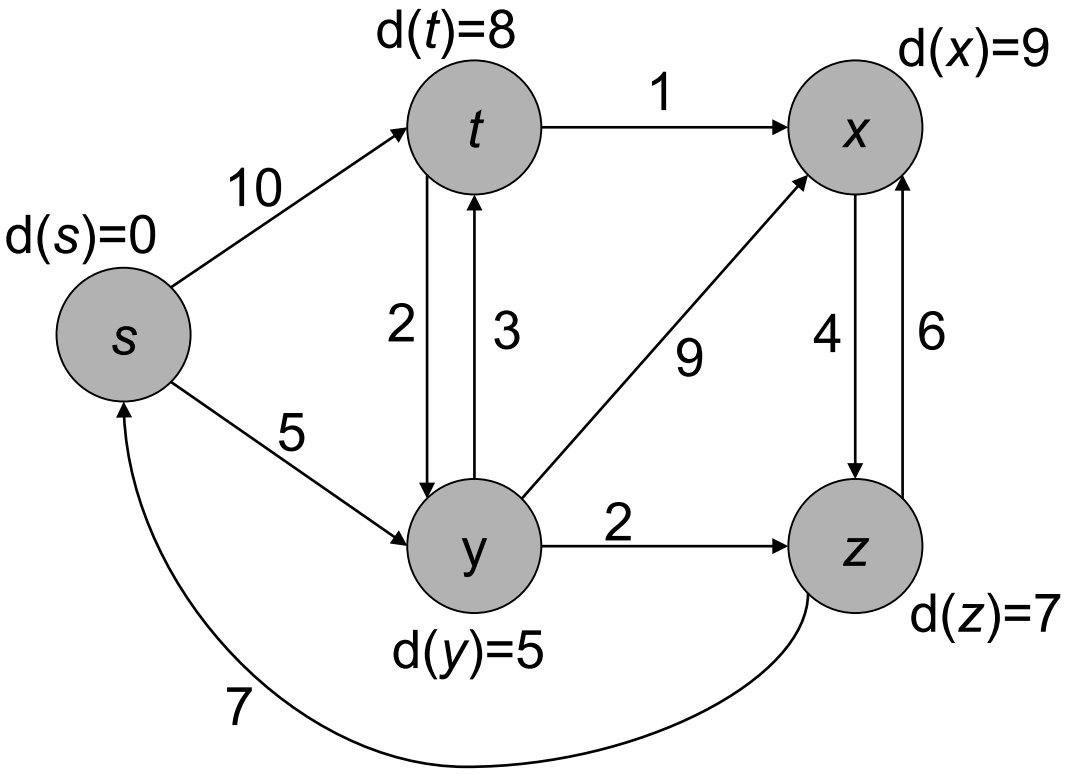
Algoritmo de Dijkstra

- Exemplo (a partir de *s*)
  - Adiciona *x* a S
  - S={*s*,*y*,*z*,*t*,***x***}



Algoritmo de Dijkstra

- Exemplo (a partir de s)
  - S={*s*,*y*,*z*,*t*,*x*}



Algoritmo de Dijkstra

- Por que funciona? Solução ótima?

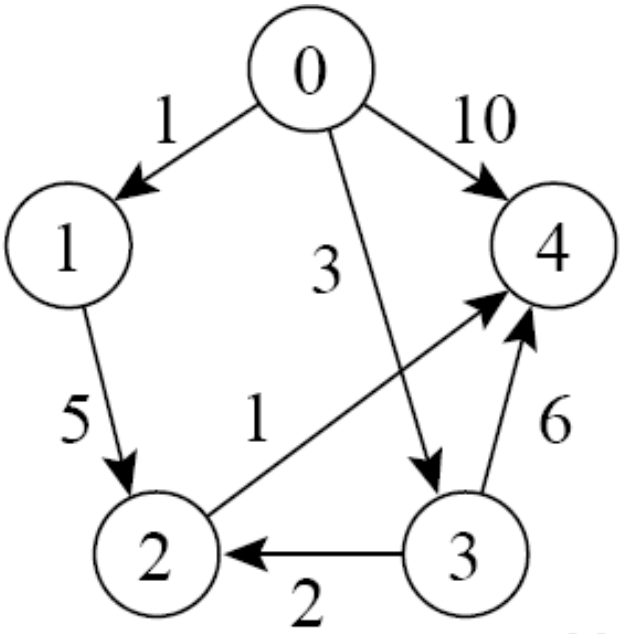
Algoritmo de Dijkstra

- Por que funciona? Solução ótima?
  - Estratégia gulosa
  - Solução ótima, pois sempre busca pelo vértice mais leve

Exercícios

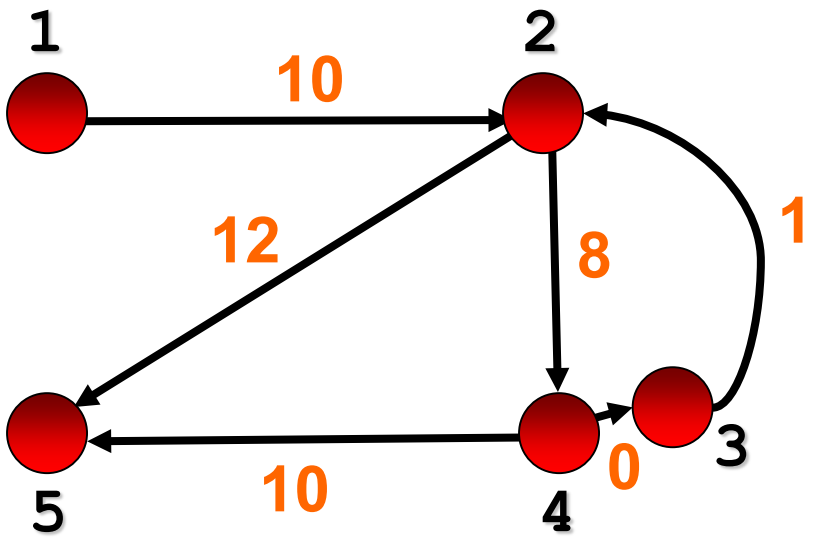
Exercício

- Calcule os caminhos mínimos para o grafo abaixo a partir do vértice 0 aplicando o algoritmo de Dijkstra



Exercício

- Calcule os caminhos mínimos para o grafo abaixo a partir do vértice 1 aplicando o algoritmo de Dijkstra



Algoritmo de Dijkstra

- Implementação do algoritmo de Dijkstra
  - Insere os vértices em uma fila de prioridades, organizada em função da atual estimativa do custo do caminho mínimo

Algoritmo de Dijkstra

```
DIJKSTRA(G, w, s)
    início
    //inicializa variáveis
    para cada vértice v faça
            d[v]= ∞
            antecessor[v]= -1
    d[s]= 0
    S= Ø
    cria fila de prioridade F com vértices do grafo
    //insere vértice u em S e faz relaxamento a partir das arestas adjacentes
    enquanto F ≠ Ø faça
            u= retirar vértice de F
            S= S + {u}
            para cada vértice v adjacente a u faça
                    relax(u,v,w) //atualizando fila de prioridade, se necessário
    fim
```

Algoritmo de Dijkstra

- Complexidade de tempo (se fila de prioridade bem implementada sobre *heap*) em função da somatória de:
  - O(|V|) para inicializar variáveis
  - O(|V| log|V|) para criar fila de prioridade sobre heap
  - O(|V|) para processar todos os vértices e, para cada um, atualizar a fila na ordem de O(log|V|)
    - O(|V| log|V|), portanto
  - O(|A|) para processar todas as arestas e, para cada relaxamento, atualizar a fila na ordem de O(log|V|)
    - O(|A| log|V|), portanto

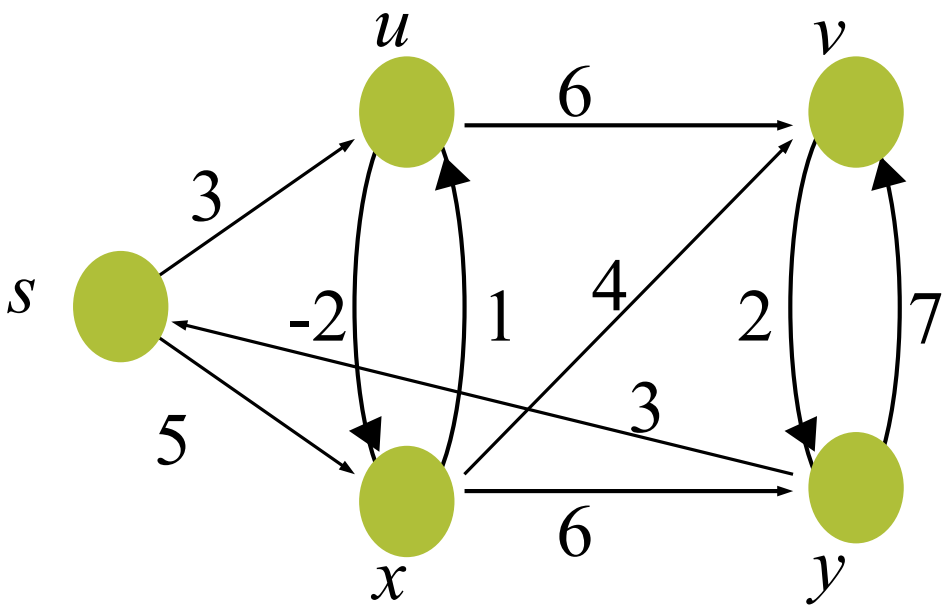
- O(|A| log|V|), no pior caso, considerando que |A|>|V|

Questão

- Como lidar com os demais casos abaixo?
  - Caminhos mais curtos de origem única → **ok**
  - Caminhos mais curtos de destino único?
  - Caminho mais curto de par único?
  - Caminhos mais curtos de todos os pares?

Exercício

- Propor algoritmo de caminho mínimo para grafo com ciclo e peso negativo



Questão

- Para pensar
  - Como a busca pelo caminho mínimo pode se beneficiar da ordenação topológica?