

"Caminhos Mínimos"

• Introdução

→ um motorista deseja encontrar a rota mais curta possível de Rio de Janeiro até São Paulo.

- Consulta um mapa
- checa as rodovias, interseções;
- como descobrir a melhor rota?

Solução trivial: enumerar todas as rotas
somar as distâncias de cada sub-caminho
selecionar a mais curta

* é viável? } pode ter de examinar muitas possibilidades

• Solução eficiente: modelar o problema por meio de grafos dirigidos

- "problema de caminhos mínimos"

- grafos dirigidos $G = (V, E)$
- grafos ponderados $w: E \rightarrow \mathbb{R}$

toda aresta possui um peso (número real)

• peso do caminho $p = \langle v_0, v_1, \dots, v_k \rangle$ é a soma dos pesos de seus arestas:

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

→ podemos definir o peso do caminho mínimo de u a v , por:

$$\delta(u, v) = \begin{cases} \min\{w(p); p \text{ PDSV}\} & , \text{ se há caminho de } u \text{ p/ } v \\ \infty & , \text{ caso contrário} \end{cases}$$

* Obs:

→ pesos de arestas podem representar outras medidas que não sejam distâncias:

- tempo
- custo
- multas
- prejuízos
- etc

} qualquer coisa que queremos minimizar

* Obs 2: O algoritmo de busca em largura (Bfs) é um algoritmo de caminhos mínimos que funciona em grafos não ponderados

- não-ponderado → peso unitário em todos os arestas
- muitos circuitos de busca em largura são usados aqui

• Variantes

- ① fonte única: determinamos a vértice de origem
 $s \in V$ para todo vértice $v \in V$.
- O algoritmo para o problema de fonte única pode resolver muitos outros problemas, entre os quais os demais Variantes a seguir.
- ② Caminhos mínimos com um só nó de destino
→ invertendo a direção das arestas, o problema se torna um problema de fonte única
- ③ Problema de caminho mínimo para um par
→ encontrar caminho mínimo de u a v , fornecendo quem são u e v .
→ Resolvendo o problema de fonte única u , resolve também esse problema
- ④ Problema de caminhos mínimos para todos os pares
→ encontrar um caminho de u a v , para qualquer vértice u e v .
→ executar problema de fonte única para cada u
→ há outros algoritmos mais eficientes

* Propriedades

- Um caminho mínimo entre dois vértices contém outros caminhos mínimos

→ Subestrutura ótima

* Programação dinâmica (Floyd - Warshall)

* Método guloso (Dijkstra)

- arestas de peso negativo

→ Se o grafo $G = (V, E)$ não contém nenhum ciclo de peso negativo que possa ser alcançado da fonte $s \in V$ então para todo $v \in V$, o peso do caminho mínimo $\delta(s, v)$ permanece bem definido, mesmo que tenha um valor negativo.

Semão: os pesos de caminho mínimo não são bem definidos.

→ se houver um ciclo de pesos negativos, em algum caminho de s até v , $\delta(s, v) = -\infty$.

- ciclos

↳ grafo também não pode conter ciclos com pesos positivos.

Existe ciclos de peso 0: que podem ser removidos do grafo

• Representação

→ obteremos uma árvore de caminhos mínimos com raiz em s

→ subgrafo dirigido $G' = (V', E')$, onde $V' \subseteq V$
 $E' \subseteq E$

onde:

1. V' é o conjunto de vértices que podem ser alcançados de s em G ;
2. G' forma uma árvore enraizada com raiz s e
3. para todo $v \in V'$, o único caminho simples de s a v em G' é um caminho mínimo de s a v em G .

obs: caminhos mínimos não são necessariamente únicos, nem as árvores são únicas

(exemplo)

* Relaxamento

→ algoritmos de caminho mínimo usam a técnica de relaxamento.

Para cada $v \in V$, mantemos um atributo $v.d$ que é um limite superior para o peso de um caminho mínimo de s a v

$v.d \Rightarrow$ estimativa de caminho mínimo

Inicializa(G, s)

1. para cada vértice $v \in V[G]$
 $v.d = \infty$
2. $v.pai = \text{NULL}$
3. $S.d = \emptyset$

→ "relaxar" uma aresta (u, v) consiste em testar se podemos melhorar o caminho até v passando por u , e se sim, atualizar $v.d$ e $v.pai$

- relaxar \Rightarrow restringir um limite superior
- uso do termo é histórico

Relaxar(u, v, w)

1. Se $v.d > u.d + w(u, v)$
2. $v.d = u.d + w(u, v)$
3. $v.pai = u$

- O relaxamento é o único meio de mudar estimativas de caminhos e predecessores.

Dijkstra \Rightarrow relaxa cada aresta exponencialmente
uma vez.

Bellman-Ford \Rightarrow relaxa cada aresta $|V| - 1$ vezes.

* Algoritmo de DIJKSTRA

→ resolve problemas de caminhos mínimos de fonte única em um grafo dirigido ponderado, para o caso onde todos os pesos de arestas são não negativos.

DIJKSTRA (G, w, s)

1. Inicializa (G, s)

2. $S = \emptyset$ // caminho, inicialmente vazio

3. $Q = V[G]$

4. enquanto $Q \neq \emptyset$

5. $u = \text{ExtraiMínimo}(Q)$

6. $S = S \cup \{u\}$

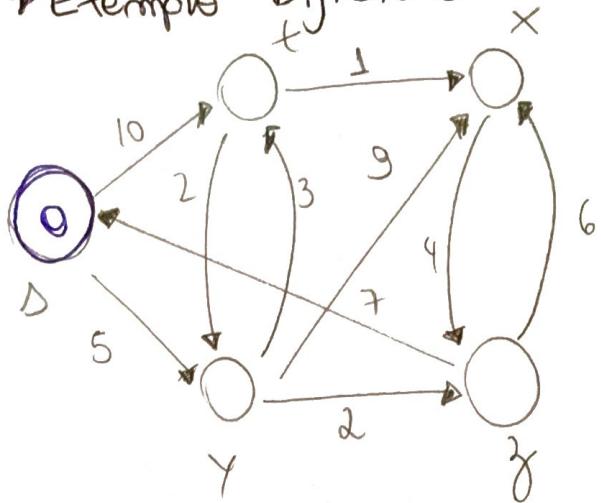
7. para cada vértice $v \in G.\text{Adj}[u]$

8. Relaxa (u, v, w)

- no primeiro loop $u = s$ ($d = \emptyset$), os demais são oo
- ExtraiMínimo sempre escolhe o vértice "mais leve" ou "mais próximo" em $V - S$ para adicionar ao conjunto S , é uma estratégia gulosa

Ideia-chave: Cada vez que o algoritmo insere um vértice u no conjunto S , temos $u.d = \delta(s, u)$

→ Exemple Dijkstra



| Vértice | d | pai | cor |
|---------|---|-----|-----|
| S | 0 | / | c |
| t | ∞ | / | b |
| x | ∞ | / | b |
| y | ∞ | / | b |
| z | ∞ | / | b |

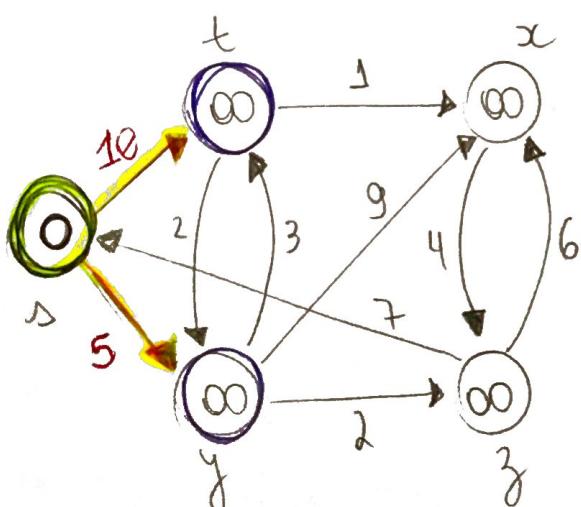
• Inicializa G

$$S = \{ \}$$

* iteração 01

$$S = \{ \}$$

$$Q = \{ S, t, x, y, z \}$$



| Vértice | d | pai | cor |
|---------|----|-----|-----|
| S | 0 | / | c/p |
| t | 10 | S | b/c |
| x | ∞ | / | b |
| y | 5 | S | b/c |
| z | ∞ | / | b |

$S = \{ S \} // \text{remove de } Q \text{ (menor d)}$

$$Q = \{ t, x, y, z \}$$

* $v = S, u = t$

$$t.d > u.d + w(u, t)$$

$$\rightarrow \infty > 0 + 10 ?$$

$$t.d = 10$$

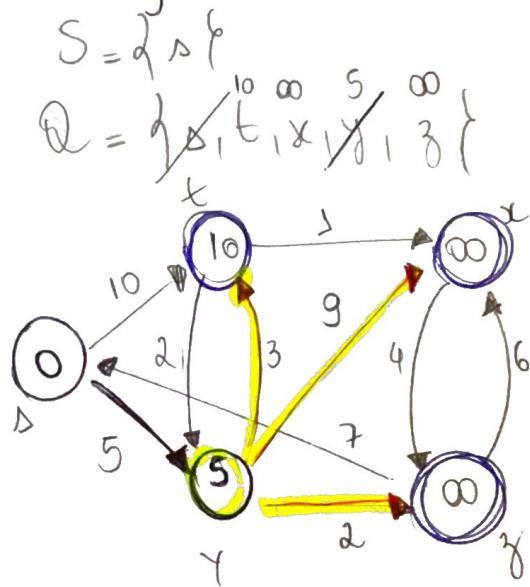
* $v = S, u = y$

$$\infty > 0 + 5 ?$$

$$t.d = 5$$



• Iteração 02:



| Vértice | d | pai | cor |
|---------|----|-----|------|
| s | 0 | n | gp |
| t | 8 | y | bc |
| x | 14 | y | bc |
| y | 5 | s | bcgp |
| z | 7 | y | bc |

$$S = \{s, y\}$$

$$Q = \{\cancel{s}, \cancel{t}, \cancel{x}, \cancel{y}, \cancel{z}\}$$

* (y, t)

$$v.d > u.d + w(u, v) ?$$

$$t.d > y.d + w(y, t)$$

$$10 > 5 + 3 ? \quad \text{Sim!}$$

(update)

$$t.d = 8$$

$$t.pai = y$$

* (y, x)

$$x.d > y.d + w(y, x) ?$$

$$\infty > 5 + 9 ? \quad \text{Sim!}$$

$$\left. \begin{array}{l} x.d = 14 \\ x.pai = y \end{array} \right\}$$

* (y, z)

$$z.d > y.d + w(y, z) \quad \left. \begin{array}{l} z.d = 7 \\ z.pai = y \end{array} \right\}$$

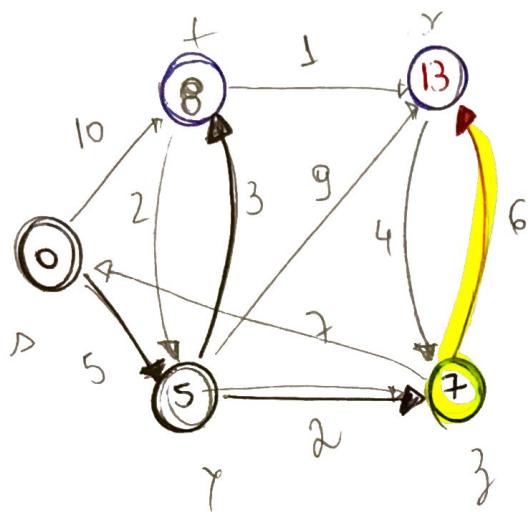
$$\infty > 5 + 2 ? \quad \text{Sim!}$$

• iteração 03

$$S = \{s, y, z\}$$

$$Q = \{t, x, z\}$$

8 14 7



| Vértice | d | pai | cor |
|---------|----|-----|-----|
| s | 0 | z | kp |
| t | 8 | y | bc |
| x | 13 | z | bc |
| y | 5 | s | kp |
| z | 7 | y | kp |

obs: s não é checado, pois já é preto

$$*(z, x) \Rightarrow z.d > z.d + w(z, x)$$

14 > 7 + 6 ? Sim

$$\left. \begin{array}{l} z.d = 13 \\ z.pai = x \end{array} \right\}$$

$$S = \{s, y, z\}$$

$$Q = \{t, x\}$$

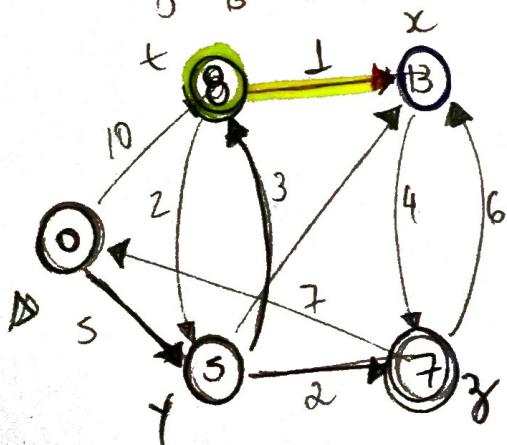
8 13

• iteração 04:

$$S = \{s, y, z, t\}$$

$$Q = \{x\}$$

8 13



| Vértice | d | pai | cor |
|---------|---|-----|-----|
| s | 0 | z | kp |
| t | 8 | y | kp |
| x | 9 | t | kp |
| y | 5 | s | kp |
| z | 7 | y | kp |

(10)

$$*(t, x)$$

$$x.d > t.d + w(t, x)$$

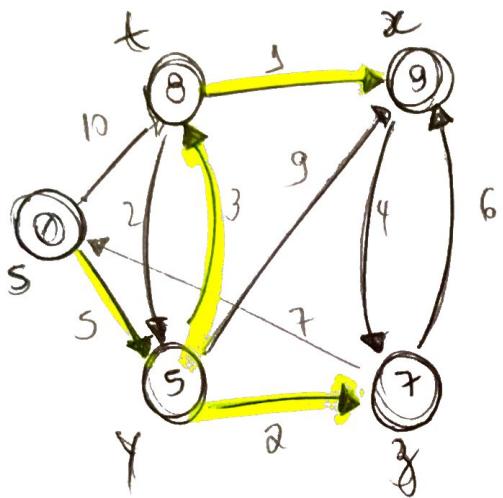
$$14 > 8 + 5 \quad ? \text{ Sim}$$

$$\left\{ \begin{array}{l} x.d = 9 \\ x.pai = t \end{array} \right.$$

• Iterações OS:

$$S = \{s, y, t, x\}$$

$$Q = \cancel{\{x\}}$$

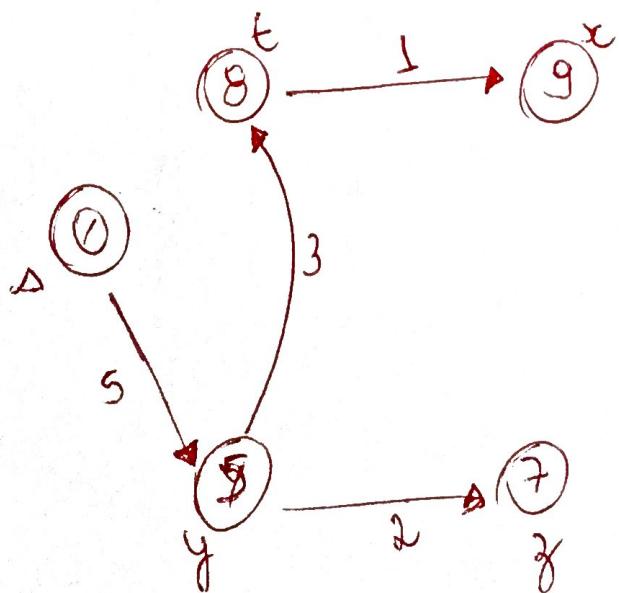


| Vértice | d | pai | cor |
|---------|---|-----|-----|
| s | 0 | — | Xp |
| t | 8 | y | Kcp |
| x | 9 | t | Kcp |
| y | 5 | s | Kcp |
| z | 7 | y | Xcp |

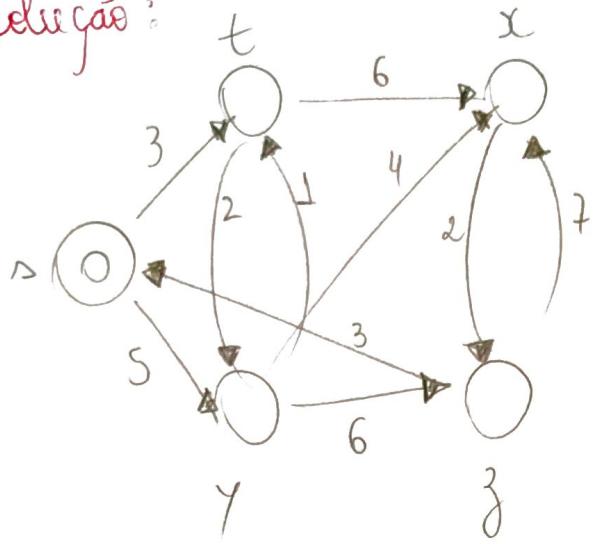
$$S = \{s, y, t, y, x\}$$

$$Q = \emptyset$$

Saída: subgrafo $G' = (V', E')$



soluções:



| Vértice | d | pai | cor |
|---------|----|-----|-------|
| s | 0 | ? | exp |
| t | 3 | s | black |
| x | 9 | t | black |
| y | 5 | s | black |
| z | 11 | y | black |

→ inicialização

$$S = \{s\}$$

$$Q = \{s, t, x, y, z\}$$

→ iteração 1

$$S = \{s\}$$

$$Q = \{\cancel{s}, t, x, y, z\}$$

$$*(s, t) \Rightarrow \infty > 0 + 3 ? \begin{cases} t.d = 3 \\ t.pai = s \end{cases}$$

$$*(s, y) \Rightarrow \infty > 0 + 5 ? \begin{cases} y.d = 5 \\ y.pai = s \end{cases}$$

→ iteração 2

$$S = \{s, t\}$$

$$Q = \{\cancel{s}, \cancel{t}, x, y, z\}$$

$$*(t, x) \Rightarrow \infty > 3 + 6 ? \begin{cases} x.d = 9 \\ x.pai = t \end{cases}$$

$$*(t, y) \Rightarrow 5 > 3 + 2 ? \text{Não!}$$

→ iteração 3

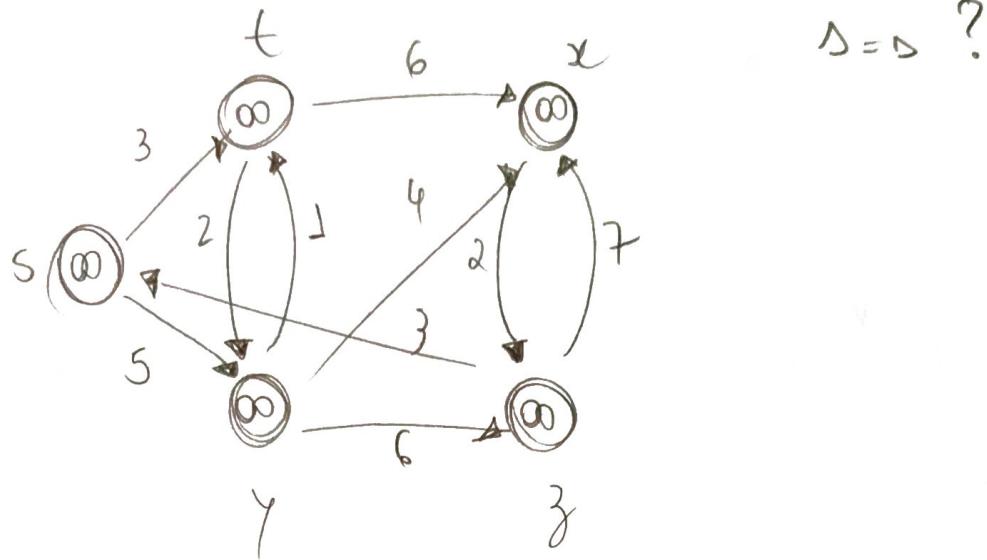
$$S = \{s, t, y\}$$

$$Q = \{x, z\}$$

$$*(y, x) \Rightarrow 9 > 5 + 4 ? \text{Não!}$$

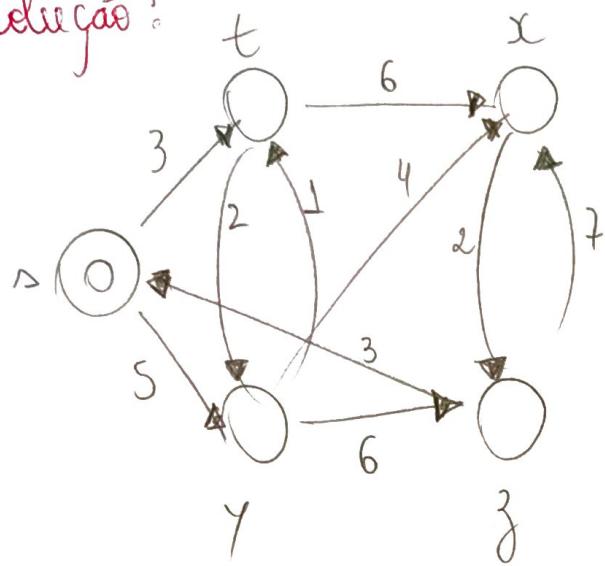
$$*(y, z) \Rightarrow \infty > 5 + 6 ? \text{Sim} \begin{cases} z.d = 11 \\ z.pai = y \end{cases}$$

Exercício 01



$\Delta = \Delta ?$

seleção:



| Vértice | d | pai | cor |
|---------|----|-----|-----|
| s | 0 | ? | EP |
| t | 3 | s | NEP |
| x | 9 | t | NEP |
| y | 5 | s | NEP |
| z | 11 | y | NEP |

→ inicialização

$$S = \{s\}$$

$$Q = \{s, t, x, y, z\}$$

→ iteração 1

$$S = \{s\}$$

$$Q = \{\cancel{s}, \cancel{t}, x, \cancel{y}, \cancel{z}\}$$

$$*(s, t) \Rightarrow \infty > 0 + 3 ? \begin{cases} t.d = 3 \\ t.pai = s \end{cases}$$

$$*(s, y) \Rightarrow \infty > 0 + 5 ? \begin{cases} y.d = 5 \\ y.pai = s \end{cases}$$

→ iteração 2

$$S = \{s, t\}$$

$$Q = \{\cancel{s}, \cancel{t}, x, y, z\}$$

$$*(t, x) \Rightarrow \infty > 3 + 6 ? \begin{cases} x.d = 9 \\ x.pai = t \end{cases}$$

$$*(t, y) \Rightarrow 5 > 3 + 2 ? \text{Não!}$$

→ iteração 3

$$S = \{s, t, y\}$$

$$Q = \{x, z\}$$

$$*(y, x) \Rightarrow 9 > 5 + 4 ? \text{Não!}$$

$$*(y, z) \Rightarrow \infty > 5 + 6 ? \text{Sim} \begin{cases} z.d = 11 \\ z.pai = y \end{cases}$$

→ iteração 4

$$S = \{s, t, y, x\}$$

$$Q = \{y\}$$

↓

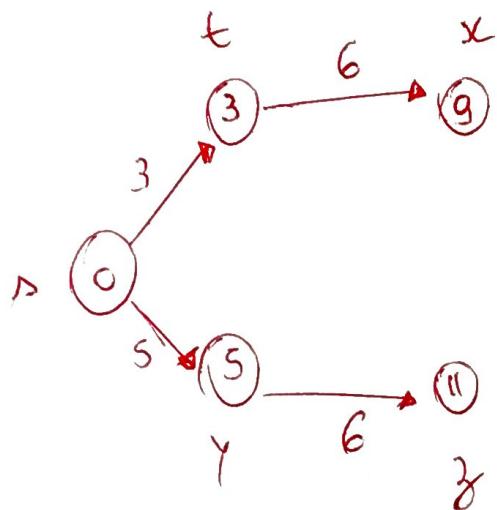
$$*(x, y) \Rightarrow 11 > 9+2 ? \text{ Não}$$

→ iteração 5

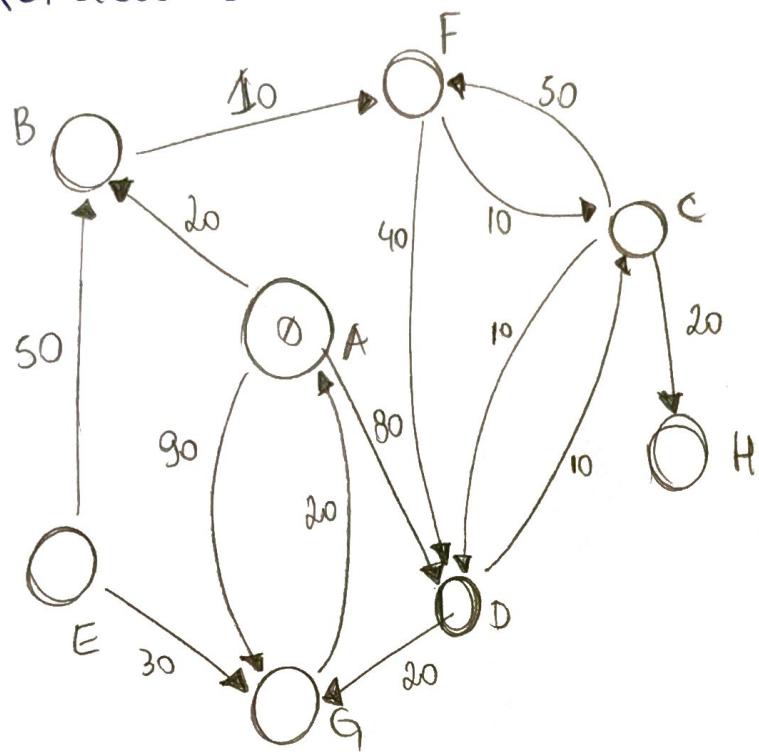
$$S = \{s, t, y, x\} \quad * y \text{ vai para } x_1 \text{ mas } x \text{ já é preto}$$

$$Q = \emptyset$$

final:



Exemplo 02 / Exercício 02



*Vértice inicial
= A