Sumários

- Caminhos mais curtos de origem única
  - Propriedades
  - Algoritmo de Dijkstra

Definição [Grafo Pesado]

- Um grafo pesado $G = (V, E, w)$ é um triplo constituído por:
  - um conjunto de vértices V
  - um conjunto de arcos $E \subseteq V \times V$
  - uma função de peso $w : E \to \mathbb{R}$

- Seja $G = (V, E, w)$ um grafo pesado, o peso do caminho $p = \langle v_0, \ldots, v_n \rangle$ é dado por:

$$w(p) = \sum_{i=0}^{n-1} w(v_i, v_{i+1})$$

- Seja $\delta : V \times V \to \mathbb{R}$ a função q mapeia cada par de vértices no peso do caminho mais curto que os liga.

$$\delta(u, v) = \min \{ w(p) \mid u \overset{p}{\rightsquigarrow} v \}$$

↳ v é atingível a partir de u pelo caminho p

Definição [Problemas de Caminhos Mais Curtos]

- Caminhos mais curtos de origem única
  Dado um vértice s determinar para todo o vértice $v \in V$ o caminho p tal que: $s \overset{p}{\leadsto} v$ e $\delta(s, v) = w(p)$.

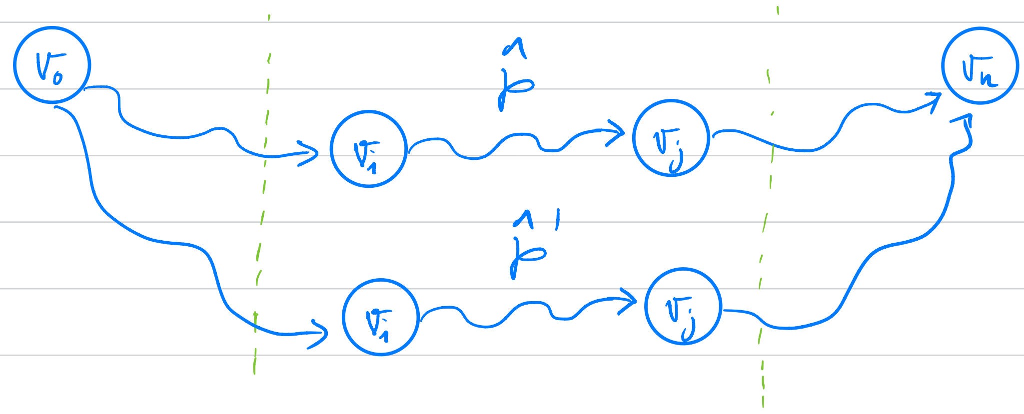
- Caminhos mais curtos de origem única e fonte única
  Dados dois vértices s e u, determinar o caminho p tal que: $s \overset{p}{\leadsto} u$ e $\delta(s, u) = w(p)$.

- Caminhos mais curtos entre todos os pares
  Para todos os vértices $u, v \in V$, determinar o caminho p tal que: $u \overset{p}{\leadsto} v$ e $\delta(u, v) = w(p)$.

Lema [Caminhos Mais Curtos - Sub-estrutura Óptima]

Seja $G = (V, E, w)$ um grafo pesado e $p = \langle v_0, \dots, v_n \rangle$ um caminho mais curto em G, temos que:

$$\forall_{0 \le i \le j \le n} .\ w(\langle v_i, \dots, v_j \rangle) = \delta(v_i, v_j)$$

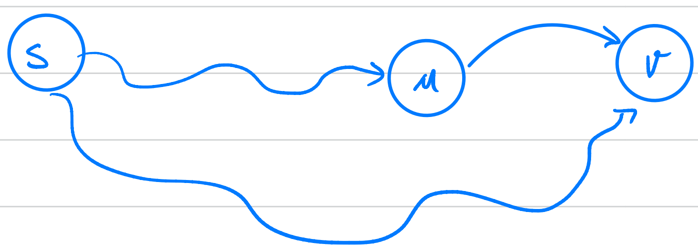


- Se $\hat{p}'$ fosse mais curto que $\hat{p}$, então o caminho de baixo seria mais curto que o caminho de cima

contradição

Lema [Desigualdade Triangular]

$$(u, v) \in E \Rightarrow \delta(s, v) \le \delta(s, u) + w(u, v)$$



Operação de Relaxação

- Os algoritmos de caminhos mais curtos estudados na disciplina funcionam através do cálculo de estimativas de distância.
  Associam cada vértice u a uma distância $u.d$.

Estimativa de distância entre s e u.

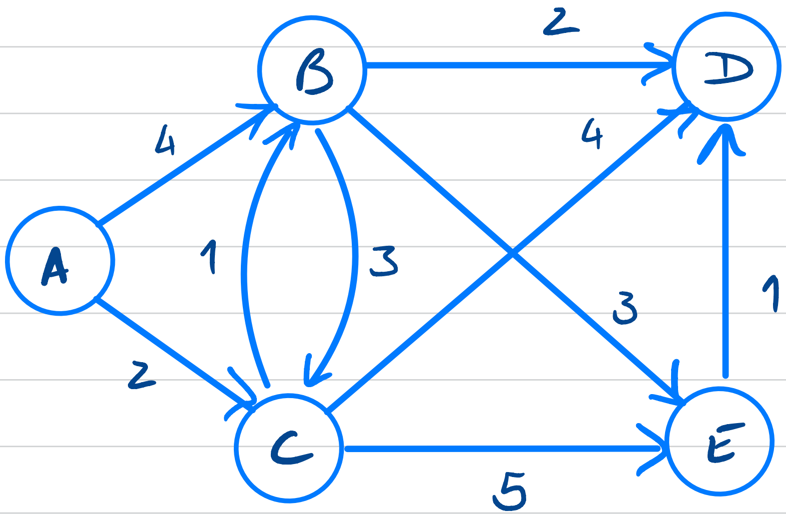
- As distâncias são actualizadas pela operação de relaxação:

```
Relax (w, u, v)
    if (v.d > u.d + w(u,v))
        v.d = u.d + w(u,v)
        v.π = u
```

```
InitializeSingleSource (G, s)
    for v ∈ G.V
        v.d := ∞;  v.π := Nil
    s.d := 0
```

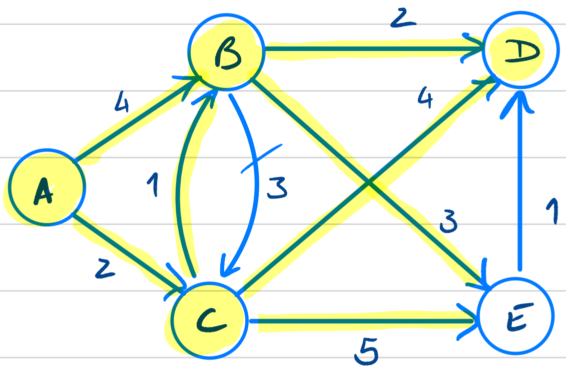
- Diferentes algoritmos usam estratégias diferentes para escolher a ordem pela qual devem efectuar as operações de relaxação.

Algoritmo de Dijkstra

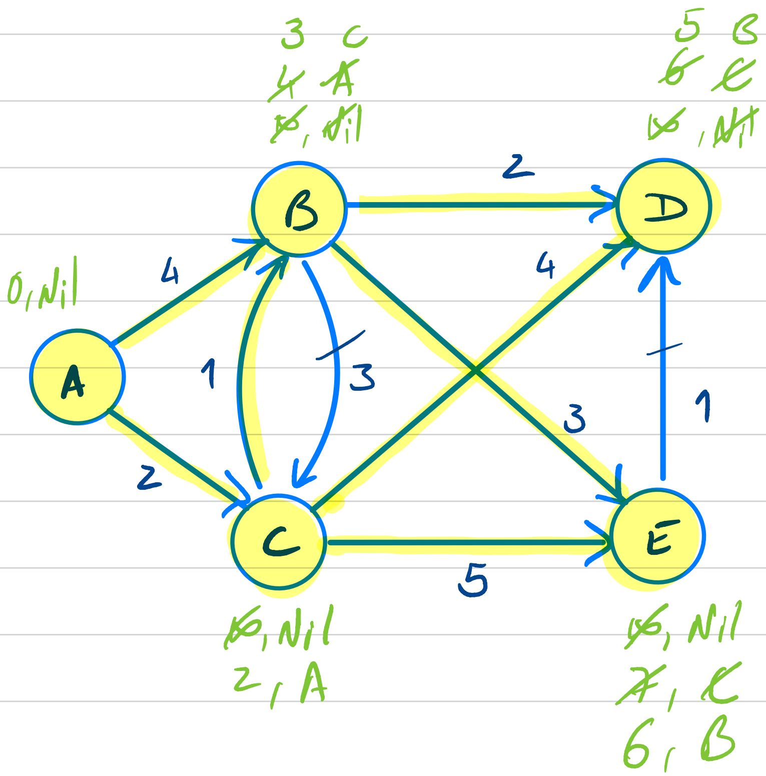


A		
B		
C		
D		
E		

Algoritmo de Dijkstra



A	0				
B	∞	4	3		
C	∞	2			
D	∞	∞	6	5	
E	∞	∞	7	6	6



Algoritmo de Dijkstra

```
Dijkstra(G, s)
  ① InitializeSingleSource(G, s)
  ② Q := new MinQueue(G.V)
     S = { }
  ③ while (! Q.empty()) {
       u := Q.extractMin();
       S := S ∪ {u};
       for each v ∈ G.Adj[u]
           Relax(G.w, u, v)
     } ④
```

fila de prioridade mínima c/ conteúdo G.V

As chaves são as distâncias!

Algoritmo de Dijkstra

```
Dijkstra(G, s)
  ① Initialize Single Source(G, s)
  ② Q := new MinQueue(G.V)
     S = { }
  ③ while (! Q.empty()) {
        u := Q.extractMin();
        S := S ∪ {u};
        for each v ∈ G.Adj[u]
            Relax(G.w, u, v)
  } ④
```

Análise de Complexidade

- Análise agregada

① $O(V)$

② $O(V)$ → build Min Heap

③ $O(V)$ iterações → minHeapify
↳ cada iteração custa $O(\lg V)$ } $O(V \cdot \lg V)$

④ $O(E)$ iterações
↳ cada iteração custa $O(\lg V)$ } $O(E \cdot \lg V)$
↳ decrease key

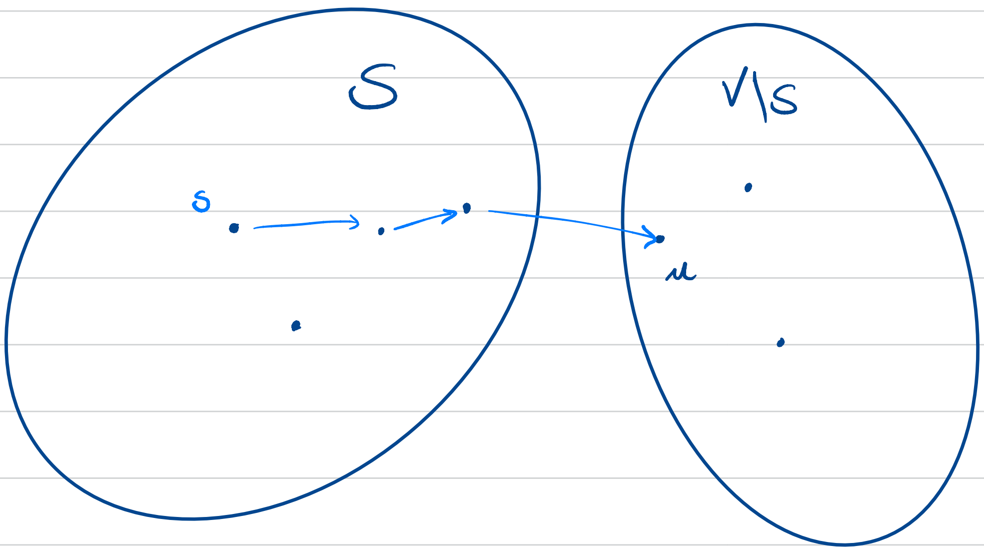
$+$

$$O((E+V) \lg V)$$

Algoritmo de Dijkstra - Invariante

$\forall v \in S . \ \delta(s,v) = v.d$

$\wedge \ S = V \setminus Q$



- u é tal que:

  $u.d = \min\{x.d \mid x \in V\setminus Q\}$

- Há que provar que:

  $u.d = \delta(s,u)$

- Suponhamos que $u.d \neq \delta(s,u)$

  - Existe um caminho p q̄ liga s a u tal que $s \overset{p}{\leadsto} u \wedge w(p) = u.d$

  - Seja x o predecessor de u no caminho mais curto entre s e u.

Algoritmo de Dijkstra - Invariante

$$\forall v \in S \,.\, \delta(s,v) = v.d$$
$$\wedge \; S = V \setminus Q$$

• u é tal que:
$$u.d = \min\{x.d \mid x \in V \setminus Q\}$$

• Há que provar que:
$$u.d = \delta(s,u)$$

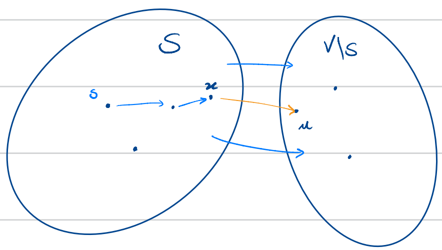
• Suponhamos que $u.d \neq \delta(s,u)$

- Existe um caminho p q̄ liga s a u tal que $s \overset{p}{\leadsto} u \wedge w(p) = u.d$

- Seja x o predecessor de u no caminho mais curto entre s e u.

• 2 casos a considerar:

(I) $x \in S$

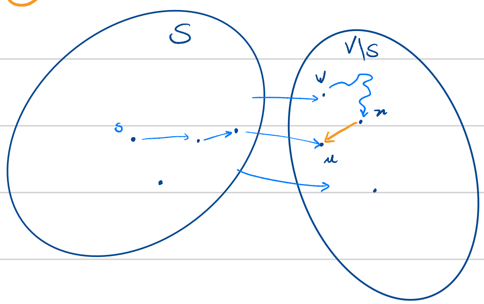


• $x.d = \delta(s,x)$ (invariante)

• O arco (x,u) já foi relaxado

$$u.d = x.d + w(x,u)$$
$$= \delta(s,u)$$

(II) $x \notin S$



• $\delta(s,u) = \delta(s,x) + w(x,u)$

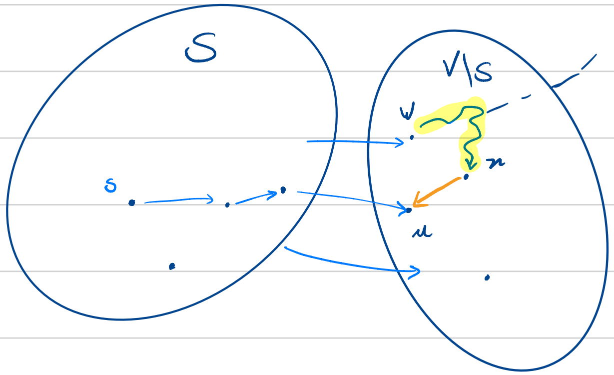
• Seja w o 1º vértice de p em $V \setminus S$
$$w.d = \delta(s,w)$$

• $w.d = \delta(s,w) \leq \delta(s,u) < u.d$

$$\underbrace{w.d < u.d}_{\text{Mas } u \text{ é o vértice com distância mais baixa}}$$

Algoritmo de Dijkstra - Pesos Negativos

(II) $x \notin S$



E se o caminho entre w e x tiver peso negativo?