

Análise de Algoritmos

Aula 20

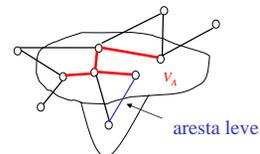
Edson Norberto Cáceres
Dept. de Computação e Estatística
UFMS
edson@dct.ufms.br

30/09/2005

1

Algoritmo de Prim

- Constrói uma árvore, desta forma A é sempre uma árvore.
- Começa com uma raiz arbitrária " r ".
- A cada passo, encontre uma aresta leve atravessando o corte $(V_A, V - V_A)$. Adicione esta aresta em A .



30/09/2005

3

Aula de Hoje

- Árvore Geradora Mínima
 - Algoritmo de Prim
- Caminho mais curtos usando em digrafos ponderados
 - Propriedades de Caminhos mais Curtos, Relaxação
 - Algoritmo de Bellman-Ford
 - Caminhos mais curtos em DAG
 - Algoritmo de Dijkstra

30/09/2005

2

Algoritmo de Prim

- Como encontrar uma aresta leve rapidamente?
- Use uma fila de prioridade Q :
 - Cada objeto é um vértice em $V - V_A$.
 - Rotule os vértices v fora da nuvem ($key[v]$) com o peso mínimo de qualquer aresta (u, v) , onde $u \in V_A$.
 - Então o vértice retornado por Extract-Min é v tal que existe $u \in V_A$ e (u, v) é uma aresta leve atravessando $(V_A, V - V_A)$.
 - Chave de v é ∞ ($key[v] = \infty$) se v não é adjacente a qualquer vértice em V_A .

30/09/2005

4

Algoritmo de Prim

MST-Prim(G, w, r)

```

01  $Q \leftarrow V[G]$ 
02 for each  $u \in Q$ 
03    $key[u] \leftarrow \infty$ 
04  $key[r] \leftarrow 0$ 
05  $\pi[r] \leftarrow NIL$ 
06 while  $Q \neq \emptyset$  do
07    $u \leftarrow ExtractMin(Q)$ 
08   for each  $v \in Adj[u]$  do
09     if  $v \in Q$  and  $w(u,v) < key[v]$  then
10        $\pi[v] \leftarrow u$ 
11        $key[v] \leftarrow w(u,v)$ 

```

Alterando
as chaves

30/09/2005

5

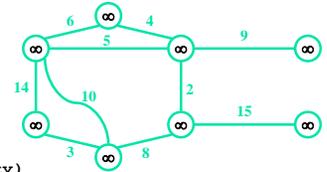
Algoritmo de Prim

MST-Prim(G, w, r)

```

 $Q = V[G];$ 
for each  $u \in Q$ 
   $key[u] = \infty;$ 
 $key[r] = 0;$ 
 $p[r] = NULL;$ 
while ( $Q$  not empty)
   $u = ExtractMin(Q);$ 
  for each  $v \in Adj[u]$ 
    if ( $v \in Q$  and  $w(u,v) < key[v]$ )
       $p[v] = u;$ 
       $key[v] = w(u,v);$ 

```



30/09/2005

7

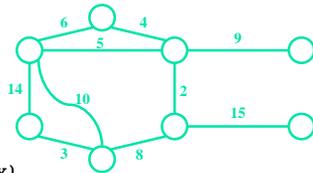
Algoritmo de Prim

MST-Prim(G, w, r)

```

 $Q = V[G];$ 
for each  $u \in Q$ 
   $key[u] = \infty;$ 
 $key[r] = 0;$ 
 $p[r] = NULL;$ 
while ( $Q$  not empty)
   $u = ExtractMin(Q);$ 
  for each  $v \in Adj[u]$ 
    if ( $v \in Q$  and  $w(u,v) < key[v]$ )
       $p[v] = u;$ 
       $key[v] = w(u,v);$ 

```



30/09/2005

6

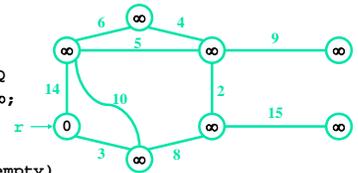
Algoritmo de Prim

MST-Prim(G, w, r)

```

 $Q = V[G];$ 
for each  $u \in Q$ 
   $key[u] = \infty;$ 
 $key[r] = 0;$ 
 $p[r] = NULL;$ 
while ( $Q$  not empty)
   $u = ExtractMin(Q);$  Pegue um vértice inicial r
  for each  $v \in Adj[u]$ 
    if ( $v \in Q$  and  $w(u,v) < key[v]$ )
       $p[v] = u;$ 
       $key[v] = w(u,v);$ 

```



30/09/2005

8

Algoritmo de Prim

MST-Prim(G, w, r)

$Q = V[G];$

for each $u \in Q$

key[u] = ∞ ;

key[r] = 0;

p[r] = NULL;

while (Q not empty)

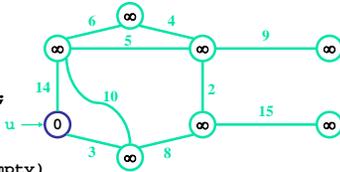
u = ExtractMin(Q);

for each $v \in \text{Adj}[u]$

if ($v \in Q$ and $w(u,v) < \text{key}[v]$)

p[v] = u;

key[v] = $w(u,v)$;



30/09/2005

9

Algoritmo de Prim

MST-Prim(G, w, r)

$Q = V[G];$

for each $u \in Q$

key[u] = ∞ ;

key[r] = 0;

p[r] = NULL;

while (Q not empty)

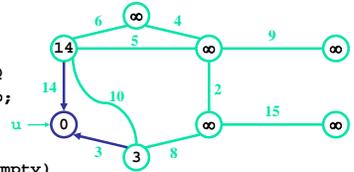
u = ExtractMin(Q);

for each $v \in \text{Adj}[u]$

if ($v \in Q$ and $w(u,v) < \text{key}[v]$)

p[v] = u;

key[v] = $w(u,v)$;



30/09/2005

11

Algoritmo de Prim

MST-Prim(G, w, r)

$Q = V[G];$

for each $u \in Q$

key[u] = ∞ ;

key[r] = 0;

p[r] = NULL;

while (Q not empty)

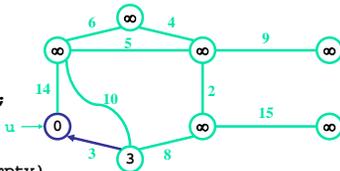
u = ExtractMin(Q); *setas azuis indicam ponteiros para os pais*

for each $v \in \text{Adj}[u]$

if ($v \in Q$ and $w(u,v) < \text{key}[v]$)

p[v] = u;

key[v] = $w(u,v)$;



30/09/2005

10

Algoritmo de Prim

MST-Prim(G, w, r)

$Q = V[G];$

for each $u \in Q$

key[u] = ∞ ;

key[r] = 0;

p[r] = NULL;

while (Q not empty)

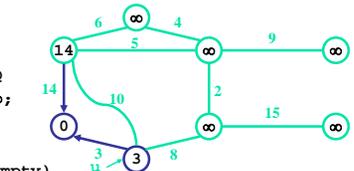
u = ExtractMin(Q);

for each $v \in \text{Adj}[u]$

if ($v \in Q$ and $w(u,v) < \text{key}[v]$)

p[v] = u;

key[v] = $w(u,v)$;



30/09/2005

12

Algoritmo de Prim

MST-Prim(G, w, r)

$Q = V[G];$

for each $u \in Q$

$key[u] = \infty;$

$key[r] = 0;$

$p[r] = NULL;$

while (Q not empty)

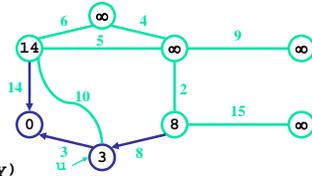
$u = \text{ExtractMin}(Q);$

 for each $v \in \text{Adj}[u]$

 if ($v \in Q$ and $w(u,v) < key[v]$)

$p[v] = u;$

$key[v] = w(u,v);$



30/09/2005

13

Algoritmo de Prim

MST-Prim(G, w, r)

$Q = V[G];$

for each $u \in Q$

$key[u] = \infty;$

$key[r] = 0;$

$p[r] = NULL;$

while (Q not empty)

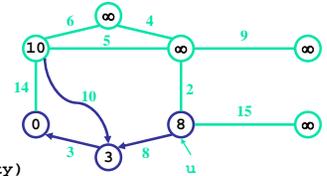
$u = \text{ExtractMin}(Q);$

 for each $v \in \text{Adj}[u]$

 if ($v \in Q$ and $w(u,v) < key[v]$)

$p[v] = u;$

$key[v] = w(u,v);$



30/09/2005

15

Algoritmo de Prim

MST-Prim(G, w, r)

$Q = V[G];$

for each $u \in Q$

$key[u] = \infty;$

$key[r] = 0;$

$p[r] = NULL;$

while (Q not empty)

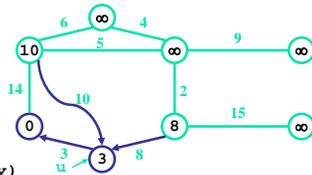
$u = \text{ExtractMin}(Q);$

 for each $v \in \text{Adj}[u]$

 if ($v \in Q$ and $w(u,v) < key[v]$)

$p[v] = u;$

$key[v] = w(u,v);$



30/09/2005

14

Algoritmo de Prim

MST-Prim(G, w, r)

$Q = V[G];$

for each $u \in Q$

$key[u] = \infty;$

$key[r] = 0;$

$p[r] = NULL;$

while (Q not empty)

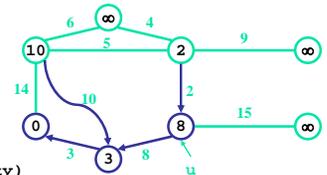
$u = \text{ExtractMin}(Q);$

 for each $v \in \text{Adj}[u]$

 if ($v \in Q$ and $w(u,v) < key[v]$)

$p[v] = u;$

$key[v] = w(u,v);$



30/09/2005

16

Algoritmo de Prim

```
MST-Prim(G, w, r)
```

```
Q = V[G];
```

```
for each u ∈ Q
```

```
    key[u] = ∞;
```

```
key[r] = 0;
```

```
p[r] = NULL;
```

```
while (Q not empty)
```

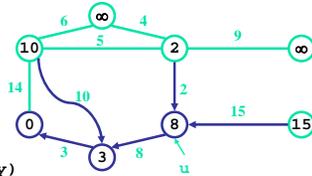
```
    u = ExtractMin(Q);
```

```
    for each v ∈ Adj[u]
```

```
        if (v ∈ Q and w(u,v) < key[v])
```

```
            p[v] = u;
```

```
            key[v] = w(u,v);
```



30/09/2005

17

Algoritmo de Prim

```
MST-Prim(G, w, r)
```

```
Q = V[G];
```

```
for each u ∈ Q
```

```
    key[u] = ∞;
```

```
key[r] = 0;
```

```
p[r] = NULL;
```

```
while (Q not empty)
```

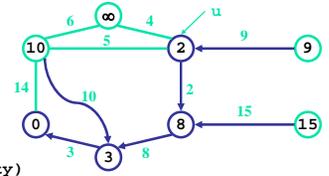
```
    u = ExtractMin(Q);
```

```
    for each v ∈ Adj[u]
```

```
        if (v ∈ Q and w(u,v) < key[v])
```

```
            p[v] = u;
```

```
            key[v] = w(u,v);
```



30/09/2005

19

Algoritmo de Prim

```
MST-Prim(G, w, r)
```

```
Q = V[G];
```

```
for each u ∈ Q
```

```
    key[u] = ∞;
```

```
key[r] = 0;
```

```
p[r] = NULL;
```

```
while (Q not empty)
```

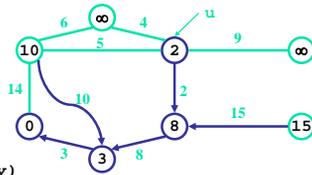
```
    u = ExtractMin(Q);
```

```
    for each v ∈ Adj[u]
```

```
        if (v ∈ Q and w(u,v) < key[v])
```

```
            p[v] = u;
```

```
            key[v] = w(u,v);
```



30/09/2005

18

Algoritmo de Prim

```
MST-Prim(G, w, r)
```

```
Q = V[G];
```

```
for each u ∈ Q
```

```
    key[u] = ∞;
```

```
key[r] = 0;
```

```
p[r] = NULL;
```

```
while (Q not empty)
```

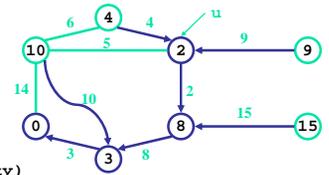
```
    u = ExtractMin(Q);
```

```
    for each v ∈ Adj[u]
```

```
        if (v ∈ Q and w(u,v) < key[v])
```

```
            p[v] = u;
```

```
            key[v] = w(u,v);
```



30/09/2005

20

Algoritmo de Prim

MST-Prim(G, w, r)

$Q = V[G];$

for each $u \in Q$

$key[u] = \infty;$

$key[r] = 0;$

$p[r] = NULL;$

while (Q not empty)

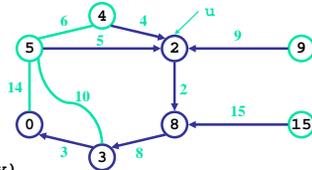
$u = \text{ExtractMin}(Q);$

 for each $v \in \text{Adj}[u]$

 if ($v \in Q$ and $w(u,v) < key[v]$)

$p[v] = u;$

$key[v] = w(u,v);$



30/09/2005

21

Algoritmo de Prim

MST-Prim(G, w, r)

$Q = V[G];$

for each $u \in Q$

$key[u] = \infty;$

$key[r] = 0;$

$p[r] = NULL;$

while (Q not empty)

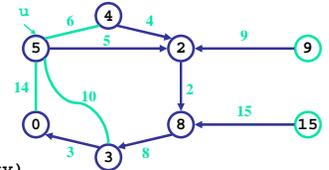
$u = \text{ExtractMin}(Q);$

 for each $v \in \text{Adj}[u]$

 if ($v \in Q$ and $w(u,v) < key[v]$)

$p[v] = u;$

$key[v] = w(u,v);$



30/09/2005

23

Algoritmo de Prim

MST-Prim(G, w, r)

$Q = V[G];$

for each $u \in Q$

$key[u] = \infty;$

$key[r] = 0;$

$p[r] = NULL;$

while (Q not empty)

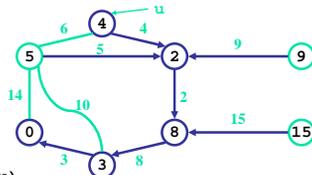
$u = \text{ExtractMin}(Q);$

 for each $v \in \text{Adj}[u]$

 if ($v \in Q$ and $w(u,v) < key[v]$)

$p[v] = u;$

$key[v] = w(u,v);$



30/09/2005

22

Algoritmo de Prim

MST-Prim(G, w, r)

$Q = V[G];$

for each $u \in Q$

$key[u] = \infty;$

$key[r] = 0;$

$p[r] = NULL;$

while (Q not empty)

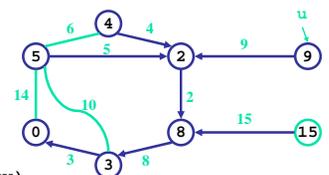
$u = \text{ExtractMin}(Q);$

 for each $v \in \text{Adj}[u]$

 if ($v \in Q$ and $w(u,v) < key[v]$)

$p[v] = u;$

$key[v] = w(u,v);$



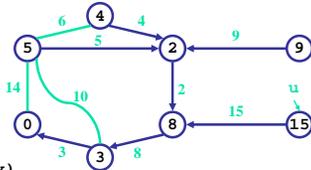
30/09/2005

24

Algoritmo de Prim

```

MST-Prim(G, w, r)
  Q = V[G];
  for each u ∈ Q
    key[u] = ∞;
  key[r] = 0;
  p[r] = NULL;
  while (Q not empty)
    u = ExtractMin(Q);
    for each v ∈ Adj[u]
      if (v ∈ Q and w(u,v) < key[v])
        p[v] = u;
        key[v] = w(u,v);
    
```



30/09/2005

25

Tempo de Execução do Alg. Prim

- Tempo = $|V|T(\text{ExtractMin}) + O(E)T(\text{ModifyKey})$
- Tempo = $O(V \lg V + E \lg V) = O(E \lg V)$

| Q | $T(\text{ExtractMin})$ | $T(\text{ModifyKey})$ | Total |
|----------------|------------------------|-----------------------|------------------|
| array | $O(V)$ | $O(1)$ | $O(V^2)$ |
| binary heap | $O(\lg V)$ | $O(\lg V)$ | $O(E \lg V)$ |
| Fibonacci heap | $O(\lg V)$ | $O(1)$ amortizado | $O(V \lg V + E)$ |

30/09/2005

27

Filas de Prioridade (PQ)

- Uma fila de prioridade é uma ADT para a manutenção de um conjunto S de elementos, cada um com um valor associado denominado chave
- Necessitamos que PQ realize as seguintes operações:
 - BuildPQ(S) – inicializa PQ para conter elementos de S
 - ExtractMin(S) devolve e remove o elemento de S com a menor chave
 - ModifyKey($S, x, newkey$) – modifica a chave de x em S
- Um heap binário implementa uma PQ
 - BuildPQ – $O(n)$
 - ExtractMin e ModifyKey – $O(\lg n)$

30/09/2005

26

Caminhos mais Curtos

- Digrafo $G = (V, E)$ com uma função de pesos $W: E \rightarrow R$
- Peso do caminho** $p = \langle v_0, v_1, \dots, v_k \rangle$ é

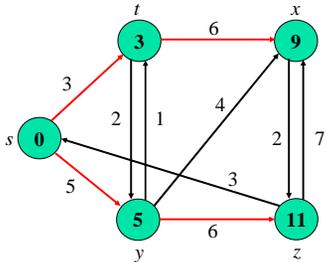
$$= \sum_{i=1}^k w(v_{i-1}, v_i) = \text{soma dos pesos das arestas do caminho } p$$
- Caminho mais curto**

$$\delta(u, v) = \begin{cases} \min\{w(p): u \xrightarrow{p} v\} & \text{se existe um caminho } u \rightarrow v \\ \infty & \text{caso contrário} \end{cases}$$
- Aplicações: **static/dynamic network routing, robot motion planning, map/route generation in traffic**

30/09/2005

28

Exemplo



30/09/2005

29

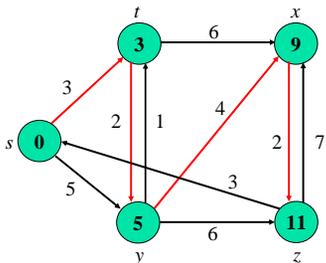
Caminhos mais Curtos

- Os pesos podem representar qualquer medida que
 - Seja linear ao longo do caminho
 - Algo que desejamos minimizar
- Problema: dado um grafo direcionado ponderado G , encontre o caminho mais curto de um vértice fonte s a outro vértice v
 - "Caminho mais curto" = peso mínimo
 - Ex., num mapa rodoviário: qual o caminho mais curto de Barra Mansa a Paty do Alferes?

30/09/2005

31

Exemplo



30/09/2005

30

Caminhos mais Curtos

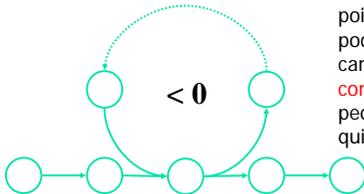
- Problemas do caminhos mais curtos
 - Single-source (single-destination)**. Encontrar um caminho mais curto de uma dada fonte (vértice s) a cada um dos demais vértices.
 - Single-pair**. Dados dois vértices, encontrar um caminho mais curto entre eles. A solução do problema single-source também resolve esse problema de forma eficiente.
 - All-pairs**. Encontre o caminho mais curto para todos os pares de vértices. Algoritmo de Programação Dinâmica.

30/09/2005

32

Arestas com Pesos negativos e ciclos?

- Podemos ter arestas com peso negativo, desde que não tenhamos *ciclos com peso negativo*
- Em grafos com ciclos de peso negativo, alguns caminhos mais curtos podem não existir (*Porque?*):



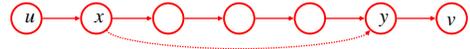
pois nesse caso poderíamos ter caminhos com **comprimento** tão pequeno quanto quiséssemos

30/09/2005

33

Subestrutura Ótima

- Lema: Qualquer subcaminhos de um caminho mais curto é um caminho mais curto.



- Demonstração: Seja p um caminho mais curto entre u e v. Suponha que algum subcaminho não seja um menor caminho
 - Então existe um subcaminho menor
 - Substitua o menor subcaminho pelo subcaminho menor
 - Então o menor caminho não é um menor caminho. Contradição

30/09/2005

35

Arestas com Pesos negativos e ciclos?

- Podemos ter ciclos negativos desde que eles não sejam atingíveis a partir da fonte.
- Alguns algoritmos só funcionam se o digrafo não tiver arestas negativas.

30/09/2005

34

Ciclos

- Caminhos mais curtos não podem conter ciclos.
 - Já comentamos a respeito de ciclos negativos.
 - Se tivermos um ciclo com peso positivo, podemos obter um caminho melhor retirando o ciclo.
 - Se os ciclos tiverem peso 0, podemos simplesmente omiti-los.

30/09/2005

36

Saída para um Caminho mais Curto

- Para cada vértice $v \in V$:
 - $d[v] = \delta(s,v)$
 - Inicialmente, $d[v] = \infty$
 - Reduz na medida em que o algoritmo avança.
 - Mas sempre mantém $d[v] \geq \delta(s,v)$.
 - Denominamos $d[v]$ uma estimativa do caminho mais curto.
 - $\pi[v]$ = predecessor de v em um caminho mais curto a partir de s .
 - Se não há predecessor, $\pi[v]=NIL$.
 - π induz uma árvore – árvore do caminho mais curto.
 - As demonstrações das propriedades de π podem ser vistas no texto.

30/09/2005

37

Relaxação

- Como podemos melhorar a estimativa do caminho mais curto para v passando através de u e usando a aresta (u,v) ?
- Uma técnica chave no algoritmos de menor caminho é a *relaxação*
 - Idéia: para todo v , $d[v]$ armazena um limite superior para o peso do caminho mais curto de s a v , $\delta(s,v)$.
 - $d[v]$ é uma estimativa do caminho mais curto.
- O passo da relaxação pode diminuir o valor da estimativa do caminho mais curto e alterar o predecessor de v , $\pi[v]$.

30/09/2005

39

Inicialização

- Todos os algoritmos de caminhos mais começam com Init-Single-Source.

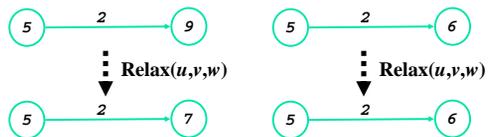
```
Init-Single-Source(V,s)
01 for each v ∈ V
02   do d[v] ← ∞
03   π[v] ← NIL
04 d[s] ← 0
```

30/09/2005

38

Relaxação

```
Relax(u,v,w)
01 if d[v] > d[u]+w(u,v)
02   then d[v]=d[u]+w(u,v)
03   π[v]← u
```



30/09/2005

40

Desigualdade Triangular

- $\delta(u,v) \equiv$ peso do caminho mais curto de u a v .
- **Teorema:**
Para qualquer $(u,v) \in E$, temos $\delta(s,v) \leq \delta(s,u) + w(u,v)$.

- **Demonstração:**
O peso do caminho mais curto de s a v não pode ser maior que qualquer outro caminho de s a v . O caminho de s a u adicionado da aresta (u,v) é um caminho de s a v , e se usamos o menor caminho de s a u , seu peso é $\delta(s,u) + w(u,v)$.



30/09/2005

41

Propried. dos caminhos mais curtos

- Propriedade da falta de caminho.
 - Se $\delta(s,v) = \infty$, então $d[v] = \infty$ sempre.
- Propriedade da convergência
 - Se $s \rightarrow^n u \rightarrow v$ é um caminho mais curto, $d[u] = \delta(s,u)$, e fizemos a chamada Relax(u,v,w), então a partir desta iteração $d[v] = \delta(s,v)$.
- Propriedade da Relaxação do Caminho.
 - Seja $p = \langle v_0, v_1, \dots, v_k \rangle$ o caminho mais curto a partir de $s = v_0$ to v_k . Se fazemos uma relaxação *in order*, $(v_0, v_1), (v_1, v_2), \dots, (v_{k-1}, v_k)$, mesmo intercalada com outras relaxações, então $d[v_k] = \delta(s, v_k)$.
- As demonstrações estão no livro texto.

30/09/2005

43

Propriedade do Limite Inferior

- **Teorema:**
Para todo $v \in V$ temos que $d[v] \geq \delta(s,v)$. Além disso, uma vez que $d[v] = \delta(s,v)$ seu valor não muda mais.
- **Demonstração:**
Suponha que exista um vértice tal que $d[v] < \delta(s,v)$. Seja u o vértice que (no relaxamento) que faz com que $d[v]$ seja alterado. Então $d[v] = d[u] + w(u,v)$.
 $d[v] < \delta(s,v) \leq \delta(s,u) + w(u,v)$ (Des. Triangular).
 $\leq d[u] + w(u,v) \Rightarrow$
 $d[v] < d[u] + w(u,v)$. **Contradição.**
 Uma vez que $d[v] = \delta(s,v)$, o valor não aumenta e nem diminui.

30/09/2005

42

Algoritmo de Bellman-Ford

- Permite arestas com pesos negativos.
- Computa $d[v]$ e $\pi[v]$ para todo $v \in V$.
- O algoritmo de Bellman-Ford detecta ciclos negativos (retorna *false*) ou retorna a árvore dos caminhos mais curto.

30/09/2005

44

Algoritmo de Bellman-Ford

```

Bellman-Ford(V, E, w, s)
01 for each v ∈ V[G]
02   do d[v] ← ∞
03   π[v] ← NIL
04 d[s] ← 0
05 for i ← 1 to |V[G]|-1
06   do for each edge (u,v) ∈ E[G]
07     do Relax (u,v,w)
08 for each edge (u,v) ∈ E[G] do
09   do if d[v] > d[u] + w(u,v)
10     then return false
11 return true
    
```

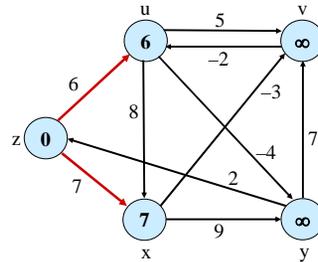
Inicialização

O laço 5 a 7
relaxa todas
as arestas ($|E|$)
 $|V|-1$ vezes.
Tempo: $\Theta(VE)$.

30/09/2005

45

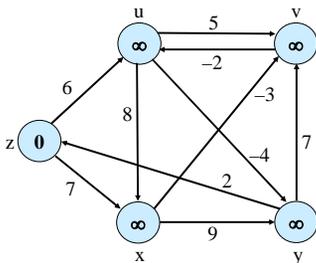
Exemplo



30/09/2005

47

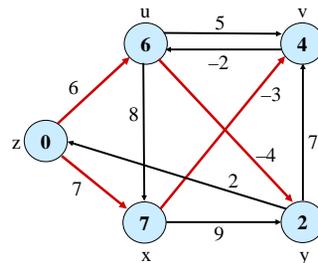
Exemplo



30/09/2005

46

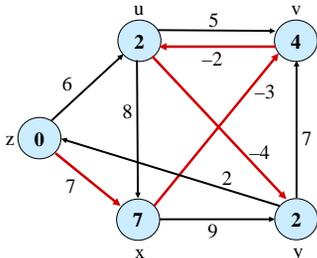
Exemplo



30/09/2005

48

Exemplo



30/09/2005

49

Correção do Alg. de Bellman-Ford

Teorema:

Assumindo que o digrafo não possui ciclos com peso negativo, o algoritmo converge após $|V|-1$ passos.

Demonstração:

Seja v atingível a partir de s , e seja $p = \langle v_0, v_1, \dots, v_k \rangle$ o caminho mais curto de s a v , onde $v_0 = s$ e $v_k = v$. Visto que p é acíclico, ele tem no máximo $|V|-1$ arestas, e desta forma $k \leq |V|-1$.

Cada iteração do laço for relaxa todas as arestas;

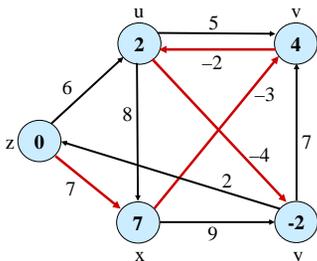
- Primeira iteração relaxa (v_0, v_1) .
- Segunda iteração relaxa (v_1, v_2) .
- k -ésima iteração relaxa (v_{k-1}, v_k) .

Pela propriedade da relaxação do caminho, $d[v] = d[v_k] = \delta(s, v_k) = \delta(s, v)$.

30/09/2005

51

Exemplo



30/09/2005

50

Correção do Alg. de Bellman-Ford

Demonstração (cont.):

Valor retornado (true/false)?

- Suponha que não exista ciclo com peso negativo atingível a partir de s . Ao terminar, para todo $(u, v) \in E$,

$$\begin{aligned} d[v] &= \delta(s, v) \\ &\leq \delta(s, u) + w(u, v) \quad (\text{Des. Triangular}) \\ &= d[u] + w(u, v). \end{aligned}$$

Desta forma, Bellman-Ford retorna true.

- Agora vamos supor que exista um ciclo com peso negativo $c = \langle v_0, v_1, \dots, v_k \rangle$, onde $v_0 = v_k$, atingível a partir de s .

$$\text{Then } \sum_{i=1}^k (v_{i-1}, v_i) < 0$$

- Suponha (por contradição) que Bellman-Ford retorna true.

30/09/2005

52

Correção do Alg. de Bellman-Ford

- Demonstração (cont.):**

Então $d[v_i] \leq d[v_{i-1}] + w(v_{i-1}, v_i)$ para $i = 1, 2, \dots, k$.

Somando o ciclo c :

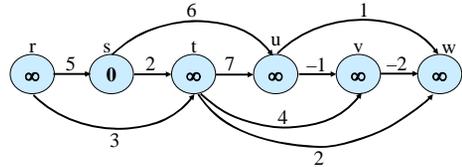
$$\begin{aligned} \sum_{i=1}^k d[v_i] &\leq \sum_{i=1}^k (d[v_{i-1}] + w(v_{i-1}, v_i)) \\ &= \sum_{i=1}^k d[v_{i-1}] + \sum_{i=1}^k w(v_{i-1}, v_i) \end{aligned}$$

Visto que $v_0 = v_k$, cada vértice em c aparece uma única vez em cada Somatória.

$$\sum_{i=1}^k d[v_i] = \sum_{i=1}^k d[v_{k-1}], \text{ desta forma } 0 \leq \sum_{i=1}^k w(v_{i-1}, v_i)$$

Isso contradiz o fato de c ser um ciclo com peso negativo.

Exemplo



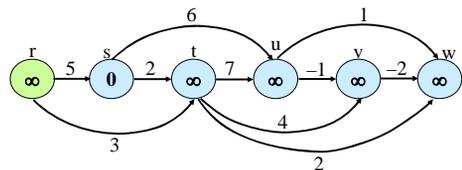
Caminhos mais curtos em DAGs

- Visto que temos um DAG (digrafo acíclico), não temos ciclos, e com isso podemos encontrar uma ordem para fazer as relaxações com uma ordenação topológica dos vértices.

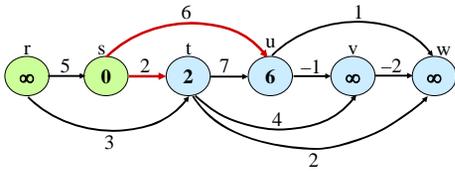
```

DAG-Shortest-Paths(G, w, s)
01 faça uma ordenação topológica dos vértices V[G]
02 Init-Single-Source(V, s)
03 for cada vértice u, pego em ordem topológ.
04   do for cada vértice v ∈ Adj[u]
05     do Relax(u, v, w)
    
```

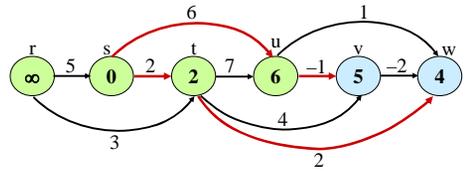
Exemplo



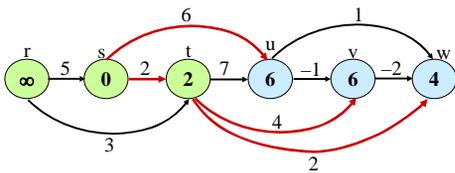
Exemplo



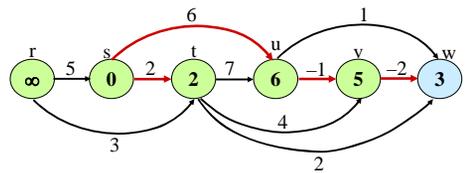
Exemplo



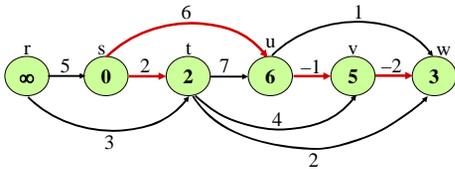
Exemplo



Exemplo



Exemplo



30/09/2005

61

Algoritmo de Dijkstra

- Sem arestas com pesos negativos.
- Estratégia Gulosa, semelhante ao algoritmo de Prim para MST (sempre escolhe o vértice mais leve "próximo" em $V-S$ para adicionar a S).
- Como a busca em largura (se todos os pesos = 1, simplesmente usamos BFS)
- Use Q , fila de prioridade cujas chaves são dadas por $d[v]$ (BFS usa fila FIFO, aqui utilizaremos uma PQ , a qual é reorganizada sempre que algum d decresce)
- Idéia Básica
 - Mantenha um conjunto S de vértices resolvidos.
 - A cada passo seleccione o vértice u mais próximo, adicione-o a S , e faça uma relaxação de todas as arestas de u .

30/09/2005

63

Caminhos mais curtos em DAGs

- Tempo de Execução:
 - $\Theta(V+E)$ – necessitamos fazer apenas uma relaxação para cada aresta, $|V|$ vezes mais rápido que o Bellman-Ford.

30/09/2005

62

Algoritmo de Dijkstra

```
Dijkstra(V, E, w, s)
01 Init-Single-Source(V, s)
02 S ← ∅
03 Q ← V // insira todos os vértices em Q
04 while Q ≠ ∅
05   do u ← Extract-Min(Q)
06   S ← S ∪ {u}
07   for cada vértice u ∈ Adj[u]
08     do Relax(u, v, w)
```

Parece muito com o algoritmo de Prim, mas computando $d[v]$, e usando os caminhos mais curtos como chave.

30/09/2005

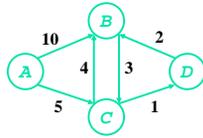
64

Algoritmo de Dijkstra

```

Dijkstra(G)
d[s] = 0;
s = ∅;
Q = V;
while (Q ≠ ∅)
    u = ExtractMin(Q);
    s = s ∪ {u};
    for each v ∈ u->Adj[]
        if (d[v] > d[u]+w(u,v))
            d[v] = d[u]+w(u,v);

```



Passo de Relaxação

Note: na realidade estamos fazendo uma chamada para DecreaseKey(Q)

30/09/2005

65

Algoritmo de Dijkstra

```

Dijkstra(G)
d[s] = 0;
s = ∅;
Q = V;
while (Q ≠ ∅)
    u = ExtractMin(Q);
    s = s ∪ {u};
    for each v ∈ u->Adj[]
        if (d[v] > d[u]+w(u,v))
            d[v] = d[u]+w(u,v);

```

Quantas vezes

ExtractMin() é chamada?

Quantas vezes

DecreaseKey() é chamada?

$O(E \lg V)$ usando heap binário para Q .
Podemos obter $O(V \lg V + E)$ com heaps de Fibonacci.

30/09/2005

67

Algoritmo de Dijkstra

```

Dijkstra(G)
d[s] = 0;
s = ∅;
Q = V;
while (Q ≠ ∅)
    u = ExtractMin(Q);
    s = s ∪ {u};
    for each v ∈ u->Adj[]
        if (d[v] > d[u]+w(u,v))
            d[v] = d[u]+w(u,v);

```

Quantas vezes

ExtractMin() é chamada?

Quantas vezes

DecreaseKey() é chamada?

Qual é o tempo de execução total?

30/09/2005

66

Algoritmo de Dijkstra

```

Dijkstra(G)
d[s] = 0;
s = ∅;
Q = V;
while (Q ≠ ∅)
    u = ExtractMin(Q);
    s = s ∪ {u};
    for each v ∈ u->Adj[]
        if (d[v] > d[u]+w(u,v))
            d[v] = d[u]+w(u,v);

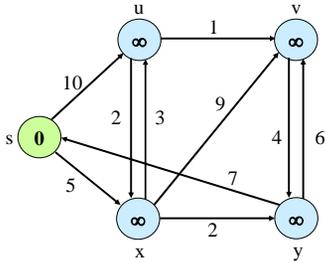
```

Correção: devemos mostrar que quando u é retirado de Q , ele já convergiu.

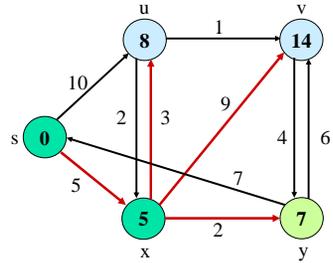
30/09/2005

68

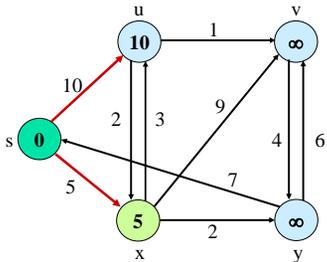
Exemplo



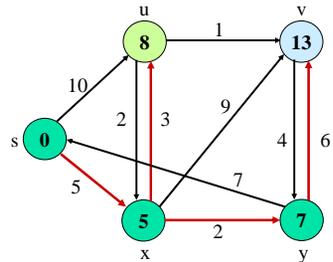
Exemplo



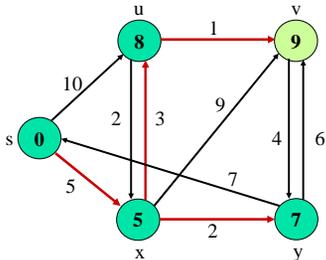
Exemplo



Exemplo



Exemplo



30/09/2005

73

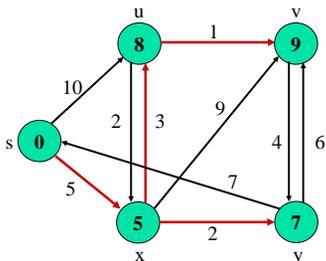
Correção do Alg. Dijkstra

- Basta demonstrar o **laço invariante** : Ao início de cada iteração do laço **while**, $d[v] = \delta(s,v)$ para todo $v \in S$.
- **Inicialização**:
 - Inicialmente, $S = \emptyset$, desta forma é trivialmente verdadeira.
- **Terminação**:
 - Ao final, $Q = \emptyset \Rightarrow S = V \Rightarrow d[v] = \delta(s,v)$ para todo $v \in V$.
- **Manutenção**:
 - Temos que provar que $d[u] = \delta(s,u)$ quando u é adicionado a S em cada iteração. Isso será feito por contradição.
 - Suponha que exista u tal que $d[u] \neq \delta(s,u)$. Sem perda de generalidade, seja u o primeiro vértice para o qual $d[u] \neq \delta(s,u)$ quando u é adicionado a S .

30/09/2005

75

Exemplo



30/09/2005

74

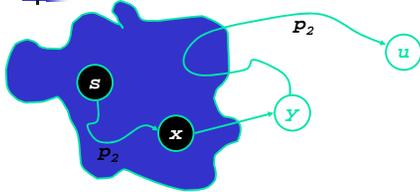
Correção do Alg. Dijkstra

- **Observações**:
 - $u \neq s$, visto que $d[s] = \delta(s,s) = 0$.
 - Portanto, $s \in S$, desta forma $S \neq \emptyset$.
 - Deve existir algum caminho $s \rightarrow^p u$, pois de outra forma $d[u] = \delta(s,u) = \infty$ pela propriedade da ausência de caminho.
 - Desta forma, existe um caminho mais curto $s \rightarrow^p u$.
 - Antes que u seja adicionado a S , o caminho p conecta um vértice em S (i.e., s) a um vértice em $V-S$ (i.e., u).

30/09/2005

76

Correção do Alg. Dijkstra



- Seja y o primeiro vértice ao longo de $p \in V-S$, e seja $x \in S$ o predecessor de y .
- Decomponha p em $s \rightarrow^{p_1} x \rightarrow^{p_2} u$ (poderíamos ter $x = s$ ou $y = u$, de forma que p_1 ou p_2 não tivessem arestas).

30/09/2005

77

Correção do Alg. Dijkstra

- $d[y] = \delta(s,y)$
 $\leq \delta(s,u)$
 $\leq d[u]$ (*propriedade do limite superior*)
- Além disso, ambos y e u estavam em Q quando escolhemos u , desta forma
- $d[u] \leq d[y] \Rightarrow d[u] = d[y]$.
- Portanto, $d[y] = d(s,y) = \delta(s,u) = d[u]$.
- Isso contradiz a hipótese que $d[u] \neq \delta(s,u)$.
- Portanto o algoritmo de Dijkstra está correto.

30/09/2005

79

Correção do Alg. Dijkstra

- $d[y] = \delta(s,y)$ quando u é adicionado a S .
- Demonstração:**
 - $x \in S$ e u é o primeiro vértice tal que $d[u] \neq \delta(s,u)$ quando u é adicionado a $S \Rightarrow d[x] = \delta(s,x)$ quando x é adicionado a S . Nesse passo é feito a relaxação de (x,y) , desta forma, pela propriedade da convergência, $d[y] = \delta(s,y)$.
- Podemos voltar a demonstração da correção e obter uma contradição para $d[u] \neq \delta(s,u)$.
 - y está no caminho mais curto $s \rightarrow^p u$, e todos os pesos das arestas são não negativos $\Rightarrow \delta(s,y) \leq \delta(s,u) \Rightarrow$

30/09/2005

78

Algoritmo de Dijkstra

- Observe
 - Passo de relaxação (linhas 10-11)
 - Atribuição de $d[v]$ modifica Q (necessita Decrease-Key)
 - Semelhante ao algoritmo de Prim para MST.

30/09/2005

80

Tempo de Execução do Alg. Dijkstra

- Tempo = $|V|T(\text{Extract-Min}) + O(E)T(\text{Modify-Key})$
- Tempo = $O(V \lg V + E \lg V) = O(E \lg V)$

| Q | $T(\text{ExtractMin})$ | $T(\text{ModifyKey})$ | Total |
|----------------|------------------------|-----------------------|------------------|
| array | $O(V)$ | $O(1)$ | $O(V^2)$ |
| binary heap | $O(\lg V)$ | $O(\lg V)$ | $O(E \lg V)$ |
| Fibonacci heap | $O(\lg V)$ | $O(1)$ amortizado | $O(V \lg V + E)$ |

Próxima Aula

- Fluxo Máximo