Melhores momentos

AULA 16

Programação dinâmica

Propriedade (da subestrutura ótima)

Se $\mathtt{G}$ é um digrafo com custos não-negativos nos arcos e $\mathtt{v}_0$-$\mathtt{v}_1$-$\mathtt{v}_2$-$\ldots$-$\mathtt{v}_\mathtt{k}$ é um caminho mínimo então $\mathtt{v}_\mathtt{i}$-$\mathtt{v}_{\mathtt{i}+1}$-$\ldots$-$\mathtt{v}_\mathtt{j}$ é um caminho mínimo para $0 \leq \mathtt{i} \leq \mathtt{j} \leq \mathtt{k}$

$\mathtt{custo}[\mathtt{v}][\mathtt{w}]$ = menor custo de uma caminho de $\mathtt{v}$ a $\mathtt{w}$

Propriedade 1

O valor de $\mathtt{custo}[\mathtt{s}][\mathtt{t}]$ é

$$\min\{\mathtt{custo}[\mathtt{s}][\mathtt{v}] + \mathtt{custo}[\mathtt{v}][\mathtt{t}] : \mathtt{v} \text{ é vértice}\}$$

Programação dinâmica

Propriedade (da subestrutura ótima)
Se `G` é um digrafo com custos não-negativos nos arcos e $\mathtt{v}_0$-$\mathtt{v}_1$-$\mathtt{v}_2$-...-$\mathtt{v}_\mathtt{k}$ é um caminho mínimo então $\mathtt{v}_\mathtt{i}$-$\mathtt{v}_{\mathtt{i}+1}$-...-$\mathtt{v}_\mathtt{j}$ é um caminho mínimo para $0 \leq \mathtt{i} \leq \mathtt{j} \leq \mathtt{k}$

$\mathtt{custo}[\mathtt{v}][\mathtt{w}]$ = menor custo de uma caminho de `v` a `w`

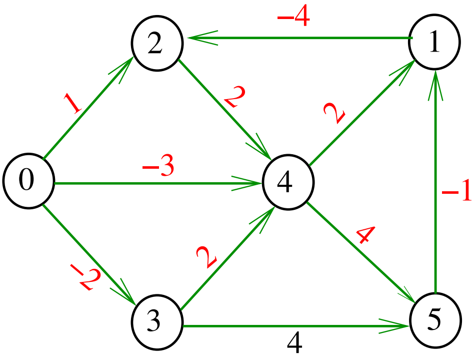
Propriedade 2
O valor de $\mathtt{custo}[\mathtt{s}][\mathtt{t}]$ é

$$\min\{\mathtt{custo}[\mathtt{s}][\mathtt{v}] + \mathtt{G\text{->}adj}[\mathtt{v}][\mathtt{t}] : \mathtt{v\text{-}t} \text{ é arco}\}$$

Problema da SPT

Problema: Dado um vértice s de um digrafo com custos (possivelmente negativos) nos arcos, encontrar uma SPT com raiz s

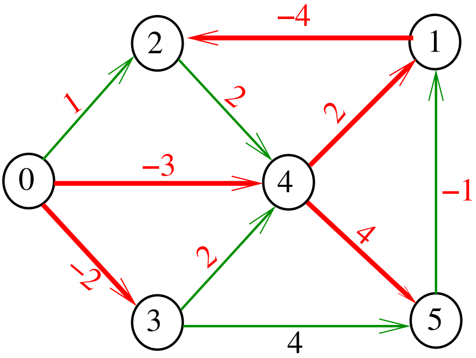
Entra:



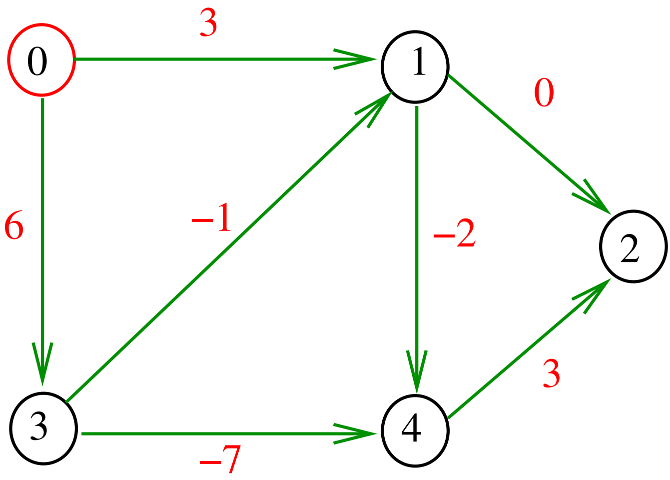
Problema da SPT

Problema: Dado um vértice s de um digrafo com custos (possivelmente negativos) nos arcos, encontrar uma SPT com raiz s

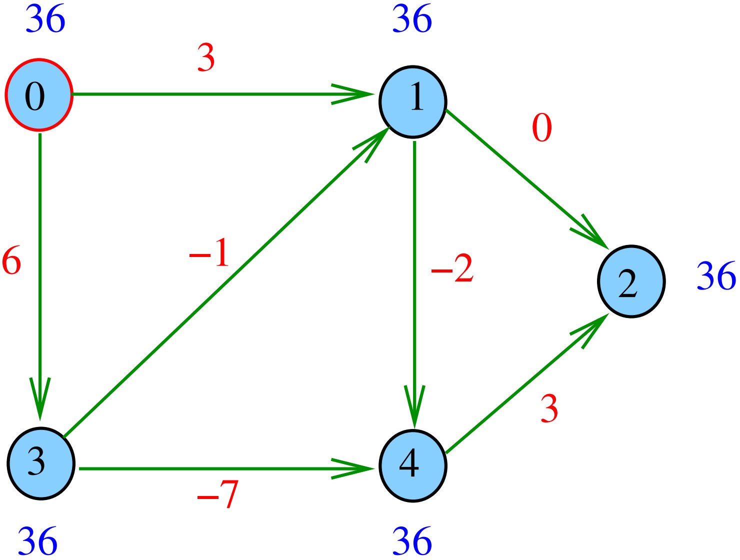
Sai:



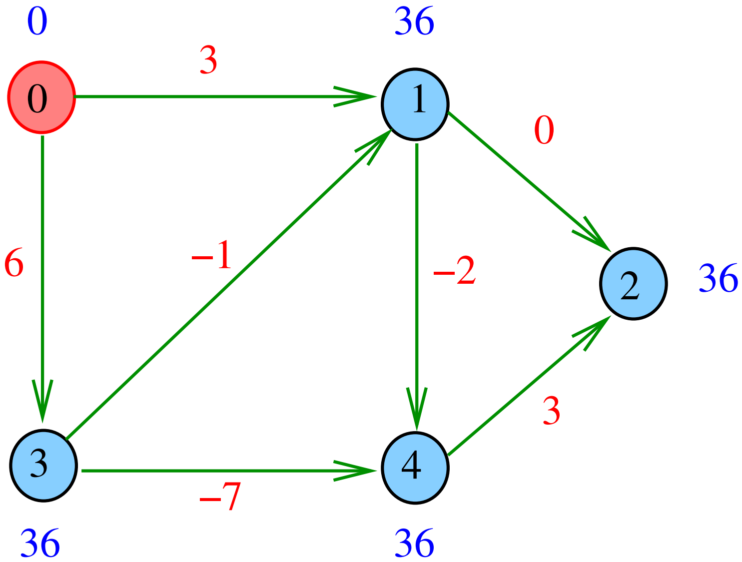
Apelemos para Dijkstra



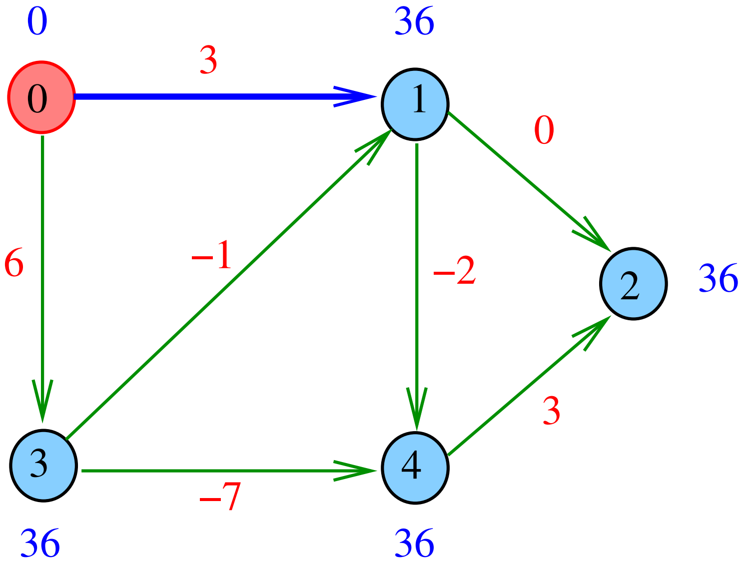
Apelemos para Dijkstra



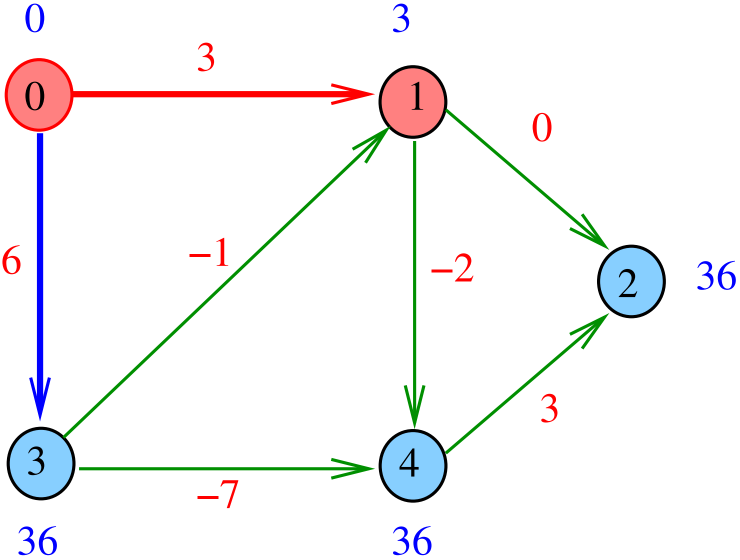
Apelemos para Dijkstra



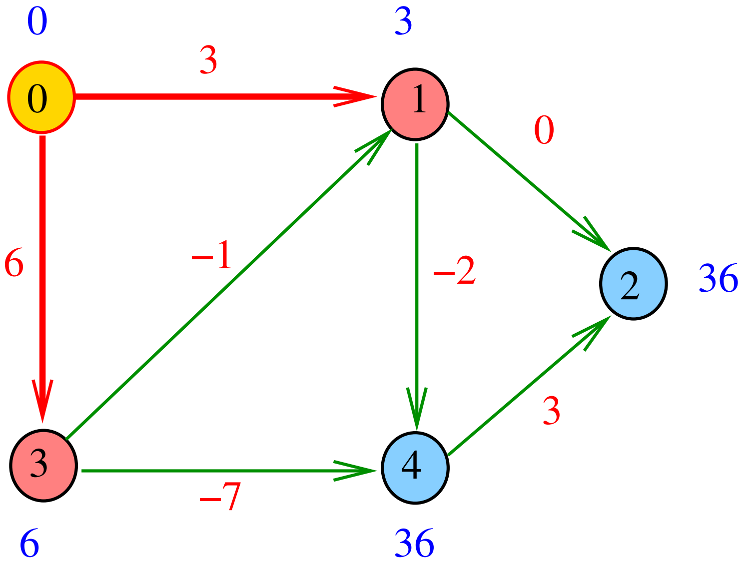
Apelemos para Dijkstra



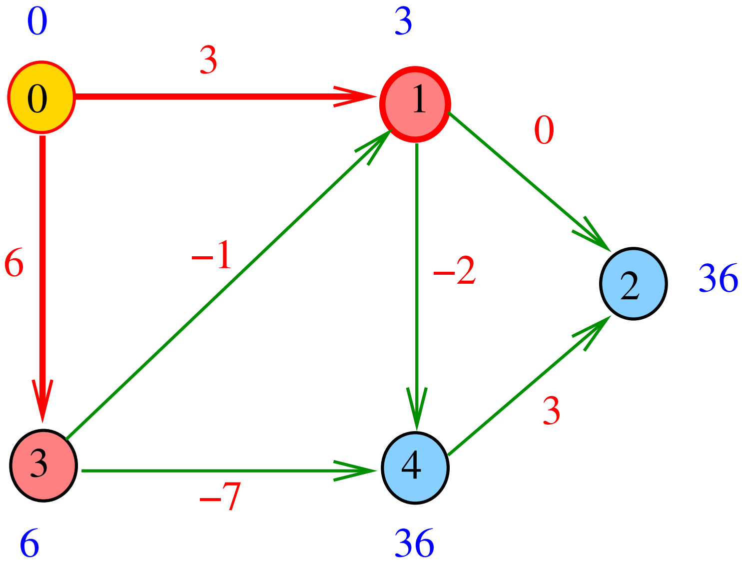
Apelemos para Dijkstra



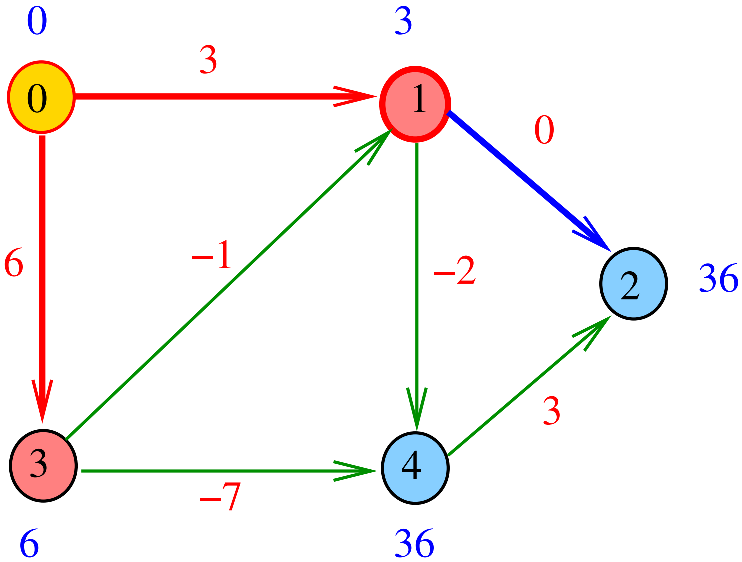
Apelemos para Dijkstra



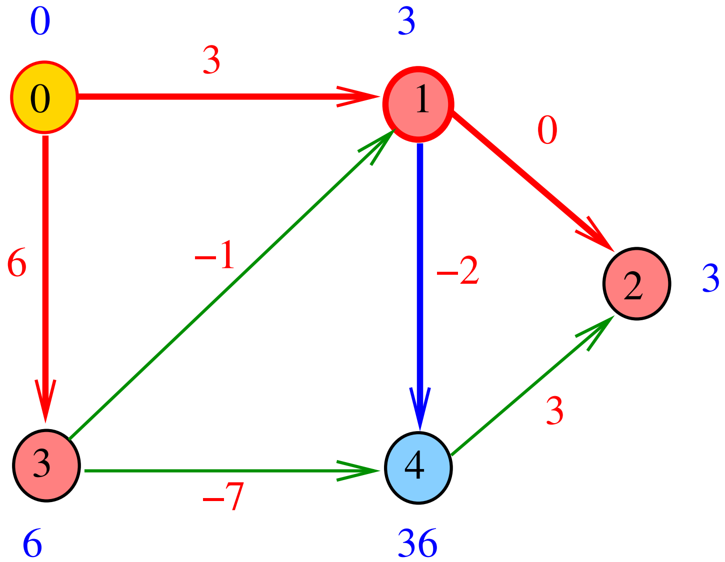
Apelemos para Dijkstra



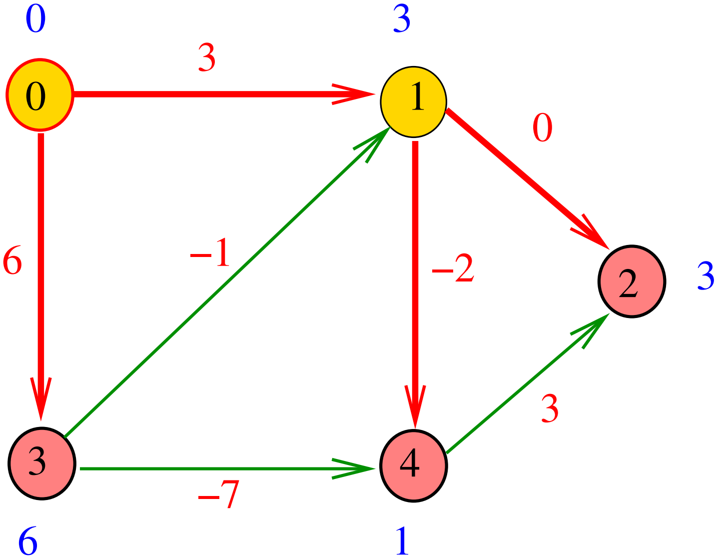
Apelemos para Dijkstra



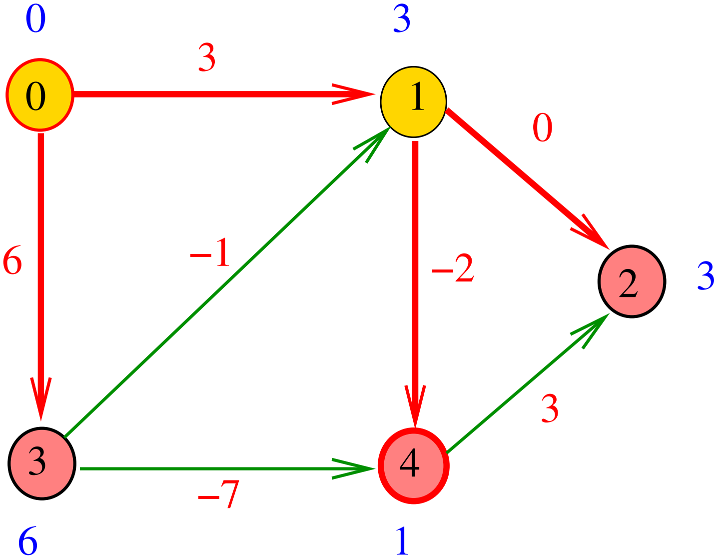
Apelemos para Dijkstra



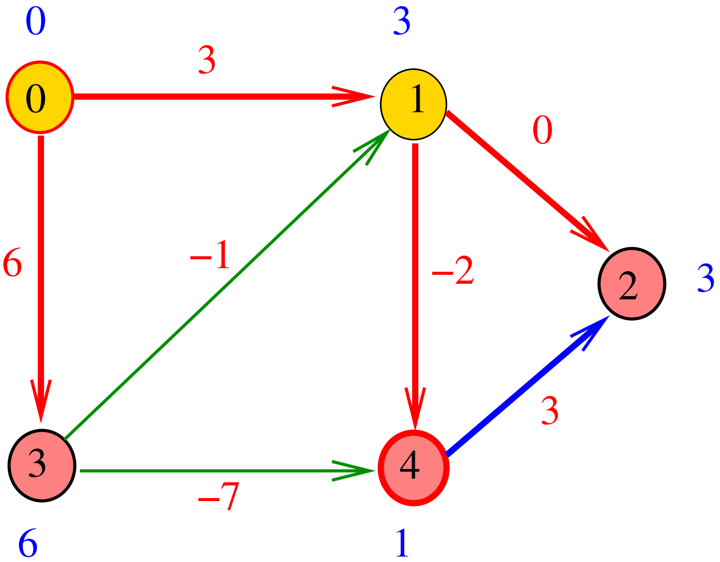
Apelemos para Dijkstra



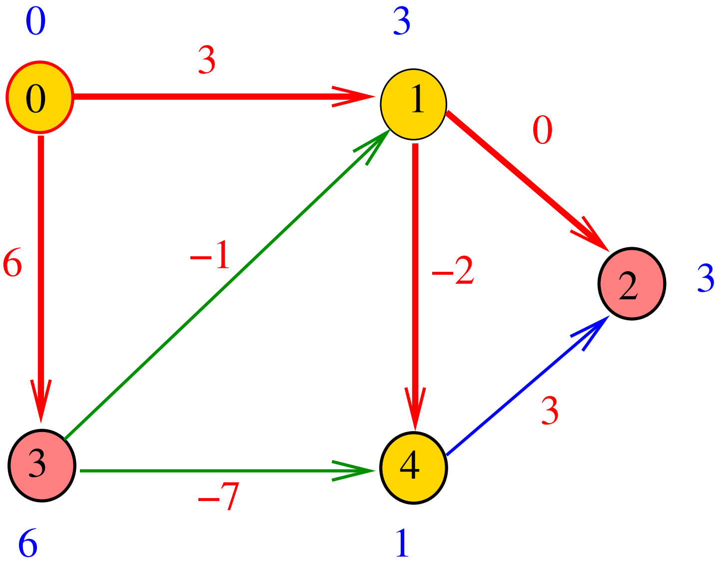
Apelemos para Dijkstra



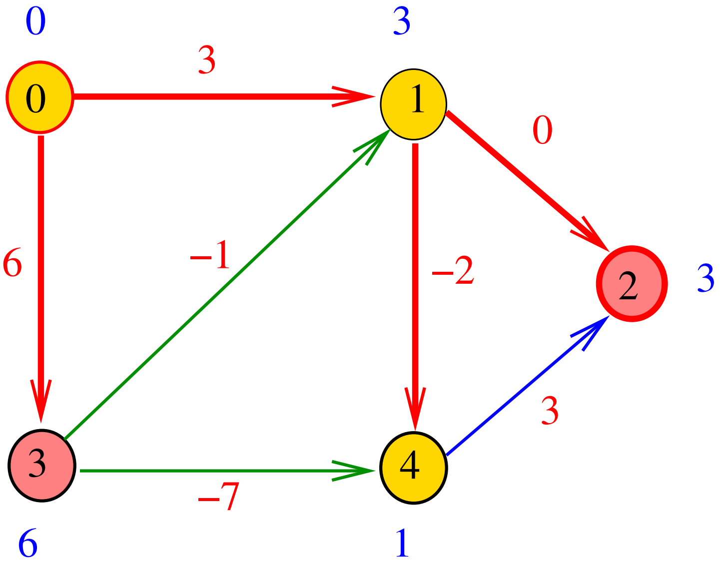
Apelemos para Dijkstra



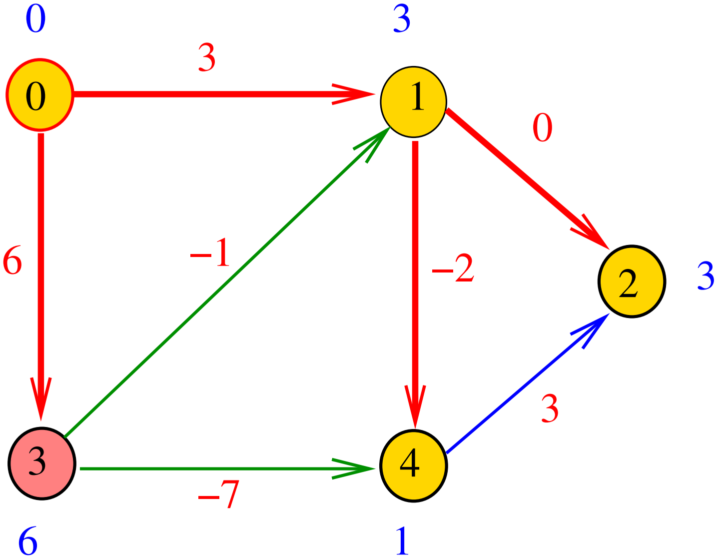
Apelemos para Dijkstra



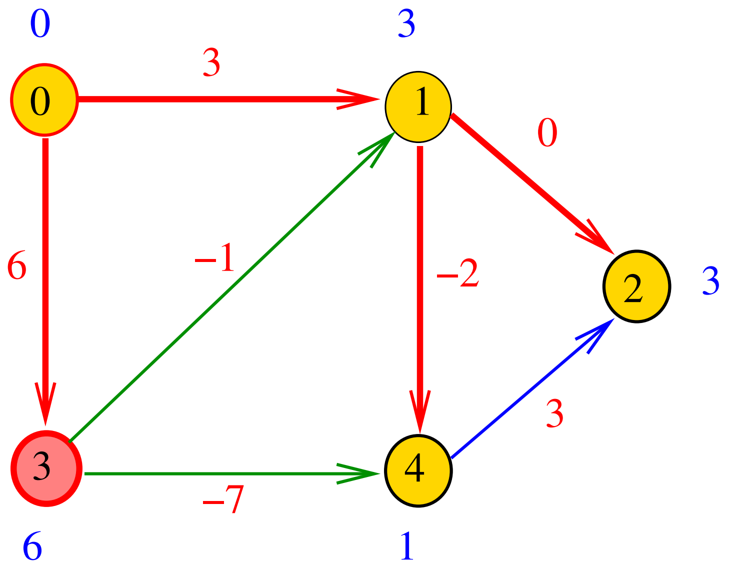
Apelemos para Dijkstra



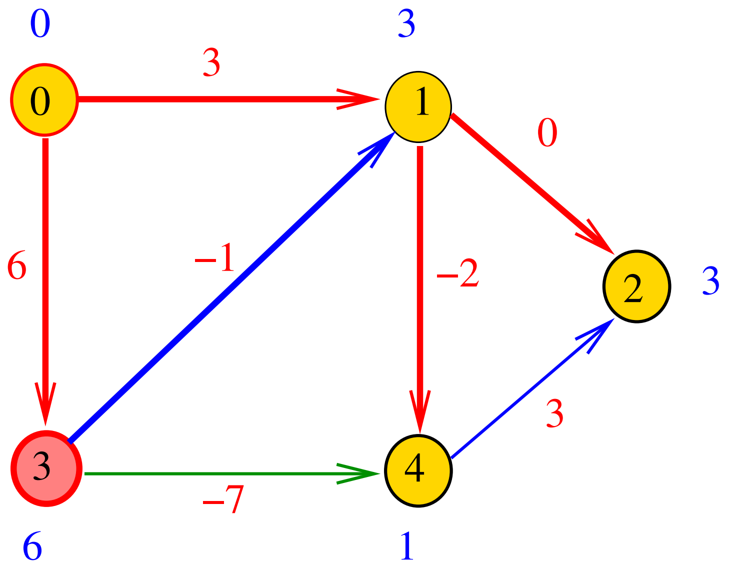
Apelemos para Dijkstra



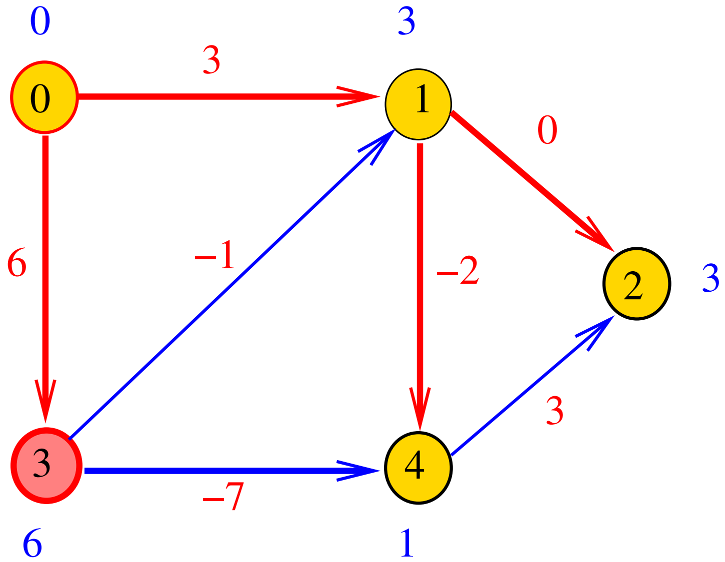
Apelemos para Dijkstra



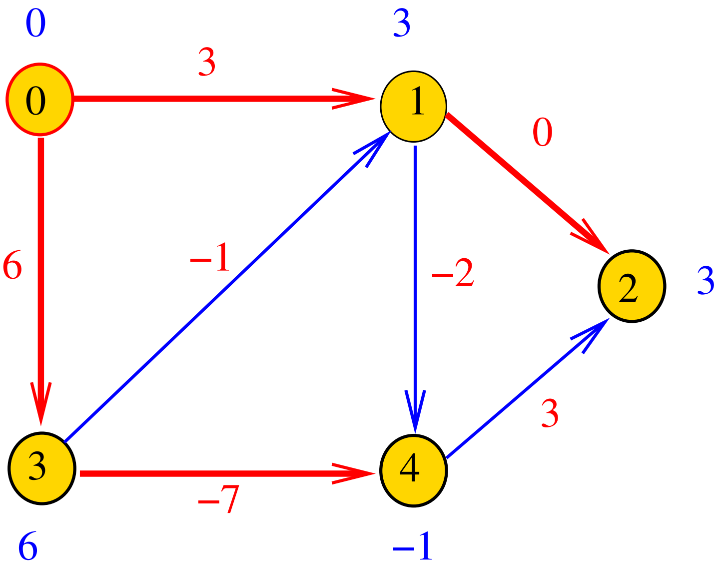
Apelemos para Dijkstra



Apelemos para Dijkstra



Opsss



O caminho mínimo de 0 a 2 tem custo 2 e não 3...

Conclusão

O algoritmo de Dijkstra não funciona para digrafos com custos negativos, mesmo que o digrafo seja acíclico.

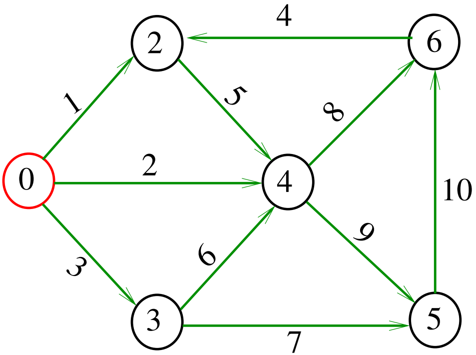
AULA 17

Algoritmo de Bellman-Ford

Problema da SPT

Problema: Dado um vértice **s** de um digrafo com custos (possivelmente negativos) nos arcos, encontrar uma SPT com raiz **s**

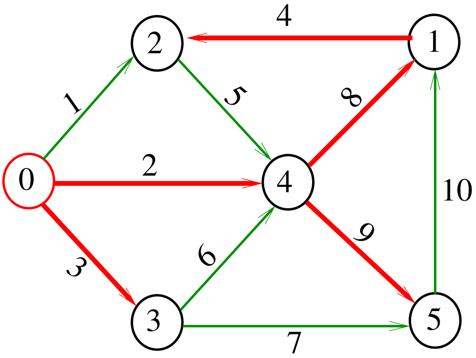
Entra:



Problema da SPT

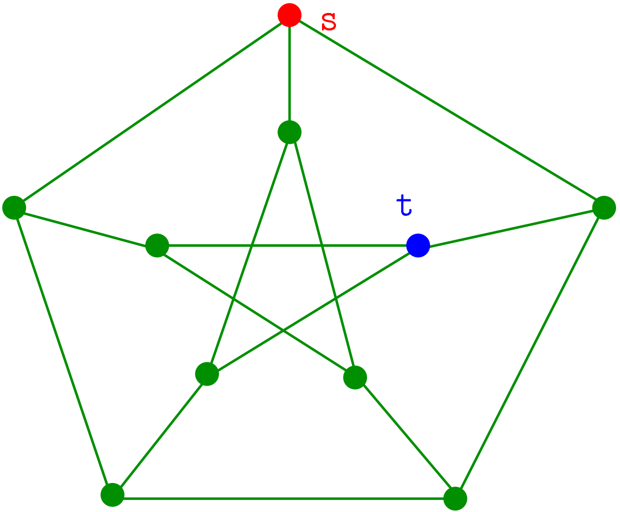
Problema: Dado um vértice s de um digrafo com custos (possivelmente negativos) nos arcos, encontrar uma SPT com raiz s

Sai:



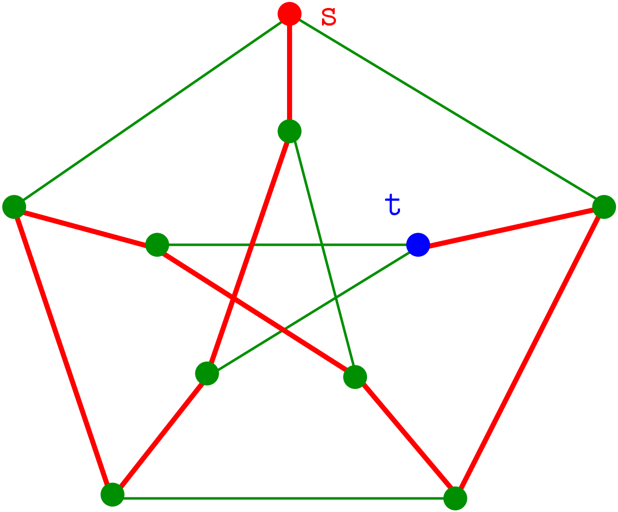
Caminhos hamiltonianos

Problema: Dados vértices s e t de um grafo encontrar um caminho hamiltoniano de s e t



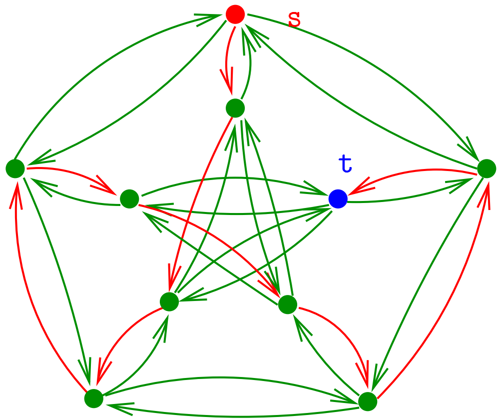
Caminhos hamiltonianos

Problema: Dados vértices s e t de um grafo encontrar um caminho hamiltoniano de s e t



Redução polinomial

todos custos $= -1$



G

G possui um st-caminho hamiltoniano $\Leftrightarrow$

G possui um st-caminho **simples** de custo $-(V - 1)$.

Conclusão

O problema do caminho simples mínimo com custos negativos é tão difícil quanto o problema do caminho hamiltoniano.

> O problema do caminho simples de custo mínimo é NP-difícil.

Complexidade computacional

O problema do caminho simples de custo mínimo é NP-difícil.

NP-difícil = **não se conhece** algoritmo de consumo de 'tempo polinomial'

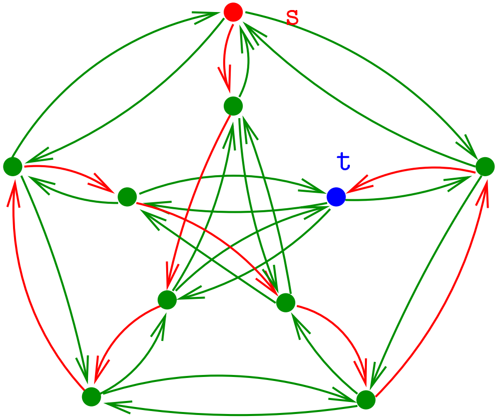
Em outras palavras: ninguém conhece um algoritmo eficiente para o problema ...

Se alguém conhece, não contou para ninguém ...

Subestrutura ótima . . .

todos custos = -1

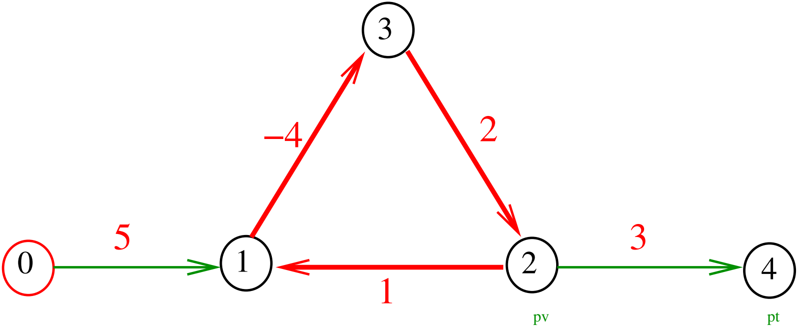
G



Não vale para digrafos com ciclos negativos

Ciclos negativos

Se o digrafo possui um **ciclo (de custo) negativo** alcançavel a patir de s, então não existe caminho mínimo de s a alguns vértices



Vamos supor que o digrafo não tem ciclos negativos.

Programação dinâmica

$\texttt{custo}[\texttt{k}][\texttt{w}]$ = menor custo de um caminho de $\texttt{s}$ a $\texttt{w}$ com $\leq$ $\texttt{k}$ arcos.

Recorrência

Programação dinâmica

$\texttt{custo}[\texttt{k}][\texttt{w}]$ = menor custo de um caminho de $\texttt{s}$ a $\texttt{w}$ com $\leq$ $\texttt{k}$ arcos.

Recorrência

$$
\begin{aligned}
\texttt{custo}[0][\texttt{s}] &= 0 \\
\texttt{custo}[0][\texttt{w}] &= \texttt{INFINITO},\ \texttt{w} \neq \texttt{s} \\
\texttt{custo}[\texttt{k}][\texttt{w}] &= \min\{\texttt{custo}[\texttt{k}-1][\texttt{w}], \\
&\quad \min\{\texttt{custo}[\texttt{k}-1][\texttt{v}] + \texttt{G->adj}[\texttt{v}][\texttt{w}]\}\}
\end{aligned}
$$

Programação dinâmica

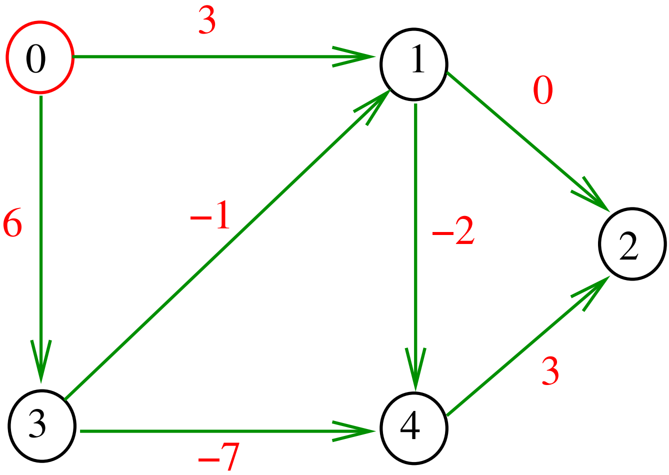
$\texttt{custo}[\texttt{k}][\texttt{w}]$ = menor custo de um caminho de $\texttt{s}$ a $\texttt{w}$ com $\leq \texttt{k}$ arcos.

Recorrência

$$
\begin{aligned}
\texttt{custo}[0][\texttt{s}] &= 0 \\
\texttt{custo}[0][\texttt{w}] &= \texttt{INFINITO},\ \texttt{w} \neq \texttt{s} \\
\texttt{custo}[\texttt{k}][\texttt{w}] &= \min\{\texttt{custo}[\texttt{k}-1][\texttt{w}], \\
&\qquad \min\{\texttt{custo}[\texttt{k}-1][\texttt{v}] + \texttt{G->adj}[\texttt{v}][\texttt{w}]\}\}
\end{aligned}
$$

Se o digrafo não tem ciclo negativo acessível a partir de $\texttt{s}$, então $\texttt{custo}[\texttt{V-1}][\texttt{w}]$ é o menor custo de um caminho de $\texttt{s}$ a $\texttt{w}$

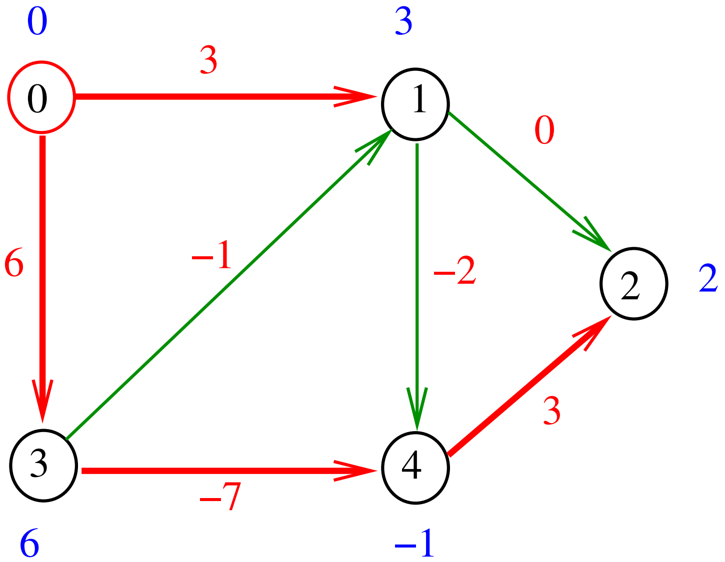
Exemplo



Exemplo

	0	1	2	3	4	v
0	0	$*$	$*$	$*$	$*$	
1	0	3	$*$	6	$*$	
2	0	3	3	6	-1	
3	0	3	2	6	-1	
4	0	3	2	6	-1	
k						

Exemplo



```
void bellman_ford1(Digraph G, Vertex s){
    Vertex v, w; double d;
    for (v=0; v < G->V; v++)
        custo[0][v] = INFINITO;
    custo[0][s] = 0;
    for (k=1; k < G->V; k++)
        for (w=0; w < G->V; w++){
            custo[k][w] = custo[k-1][w];
            for (v=0; v < G->V; v++){
                d=custo[k-1][v]+G->adj[v][w];
                if (custo[k][w] > d)
                    custo[k][w] = d;
            }
        }
}
```

Consumo de tempo

O consumo de tempo da função `bellman_ford1` é $O(V^3)$.