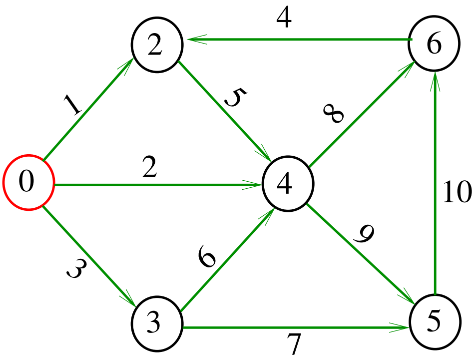
Algoritmo de Bellman-Ford

Problema da SPT

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

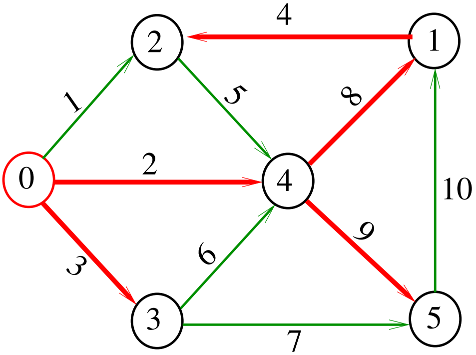
Entra:



Problema da SPT

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

Sai:



Propriedade da subestrutura ótima

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

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

Propriedade 1
O valor de $\texttt{custo}[\texttt{s}][\texttt{t}]$ é

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

Propriedade da subestrutura ótima

Se $\mathtt{G}$ é um digrafo com custos não-negativos nos arcos e $\mathtt{v}_0\text{-}\mathtt{v}_1\text{-}\mathtt{v}_2\text{-}\ldots\text{-}\mathtt{v}_\mathtt{k}$ é um caminho mínimo então $\mathtt{v}_\mathtt{i}\text{-}\mathtt{v}_{\mathtt{i}+1}\text{-}\ldots\text{-}\mathtt{v}_\mathtt{j}$ 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 2

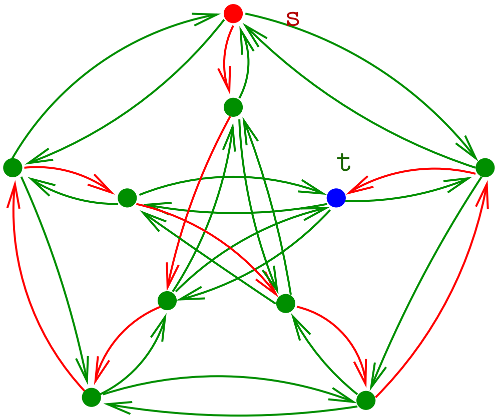
O valor de $\mathtt{custo}[\mathtt{s}][\mathtt{t}]$ é

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

Subestrutura ótima . . .

todos custos = -1

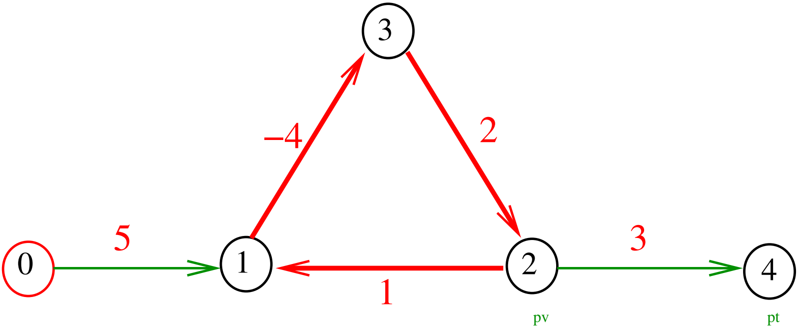
G



Não vale para digrafos com ciclos negativos

Ciclos negativos

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



Vamos supor que o digrafo não tem ciclos negativos.

Caminho de custo mínimo

Problema: Dado vértices `s` e `t` de um digrafo com custos (**possivelmente negativos**) nos arcos, encontrar uma caminho de custo mínimo de `s` a `t`.

Esse problema pode ser:

- inviável: não existe caminho de `s` a `t`;
- viável e limitado: existe caminho de `s` a `t` e nenhum caminho de `s` a `t` passa por um ciclo negativo; ou
- viável e ilimitado: existe caminho de `s` a `t` mas não existe um caminho de `s` a `t` de custo mínimo, ou seja, existe caminho de `s` a `t` que passa por um ciclo negativo.

Caminho simples de custo mínimo

Problema: Dado vértices s e t de um digrafo com custos (**possivelmente negativos**) nos arcos, encontrar uma caminho **simples** de custo mínimo de s a t.

Esse problema pode ser:

- inviável: não existe caminho de s a t; ou
- viável (e limitado): existe caminho de s a t.

Este problema é NP-difícil.
Só sabemos resolver esse problema quando o caminho de custo mínimo é **simples**. . .

Programação dinâmica

Como explorar a propriedade da subestrutura ótima

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

Recorrência

custo[0][s] = 0

custo[0][w] = INFINITO, w $\neq$ s

custo[k][w] = min{custo[k−1][w], min{custo[k−1][v] + G->adj[v][w]}}

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

Programação dinâmica

Como explorar a propriedade da subestrutura ótima

custo[k][w] = menor custo de um caminho de s a w com $\leq$ 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 s, então custo[V-1][w] é o menor custo de um caminho de s a w

Programação dinâmica

Como explorar a propriedade da subestrutura ótima

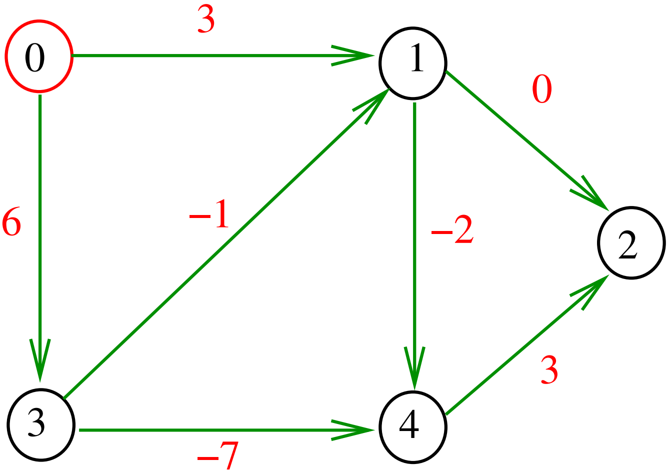
$\texttt{custo}[\texttt{k}][\texttt{w}]$ = menor custo de um caminho de $\texttt{s}$ a $\texttt{w}$ com $\leq$ $\texttt{k}$ arcos.

Recorrência

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

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 $\texttt{w}$

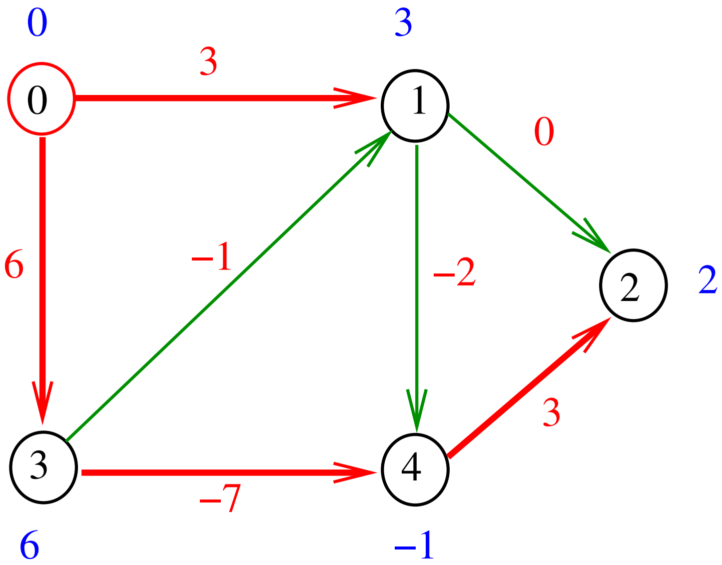
Exemplo



Exemplo

k \ v	0	1	2	3	4
0	0	*	*	*	*
1	0	3	*	6	*
2	0	3	3	6	−1
3	0	3	2	6	−1
4	0	3	2	6	−1

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 $O(V^3)$.

Ciclos negativos

Se `custo[k][v]` $\neq$ `custo[k-1][v]`, então `custo[k][v]` é o custo de um caminho de `s` a `v` com **exatamente** `k` arcos.

Se `custo[V][v]` $\neq$ `custo[V-1][v]`, então

- `custo[V][v]` $<$ `custo[V-1][v]` e
- `custo[V][v]` é o custo de um caminho `P` de `s` a `v` com **exatamente** `V` arcos.

Seja `C` um ciclo em `P` e seja `P`$'$ o caminho resultante a partir de `P` após a remoção de `C`.

Ciclos negativos

Note que $\mathtt{P}'$ tem no $\leq$ $\mathtt{V}$-1 arcos e portanto

$$
\begin{aligned}
\mathtt{custo}(\mathtt{P}') &\geq \mathtt{custo}[\mathtt{V}\text{-}1][\mathtt{v}] \\
&> \mathtt{custo}[\mathtt{V}][\mathtt{v}] \\
&= \mathtt{custo}(\mathtt{P}) \\
&= \mathtt{custo}(\mathtt{P}') + \mathtt{custo}(\mathtt{C}).
\end{aligned}
$$

Logo, $\mathtt{C}$ é um ciclo de custo negativo.

Conclusão

Se `custo`[V][v] $\neq$ `custo`[V-1][v], então `G` tem um ciclo negativo alcançável a partir de `s`.

```
void bellman_ford2(Digraph G, Vertex s){
    Vertex v, w; double d;
    for (v=0; v < G->V; v++)
        cst[v] = INFINITO;
    cst[s] = 0;
    for (k=1; /* A */ k < G->V; k++)
        for (v=0; v < G->V; v++){
            for (w=0; w < G->V; w++){
                d=cst[v]+G->adj[v][w];
                if (cst[w] > d)
                    cst[w] = d;
            }
        }
}
```

Relação invariante

Em /* A */ na linha vale que

$\texttt{cst}[\texttt{v}] \leq \texttt{custo}[\texttt{k}\text{-}1][\texttt{v}] =$ o menor custo de um caminho de $\texttt{s}$ a $\texttt{v}$ com até $\texttt{k}$-1 arcos

Consumo de tempo

O consumo de tempo da função `bellman_ford2` é $O(V^3)$.

Bellman-Ford

```
void bellman_ford3(Digraph G, Vertex s){
    Vertex v, w;
    link p;
    for (v=0; v < G->V; v++){
        cst[v] = INFINITO;
        parnt[v] = -1;
    }
    cst[s] = 0;
```

Bellman-Ford

```
for (k=1; k < G->V; k++)
    for (v=0;v < G->V; v++){
        p = G->adj[v];
        while (p != NULL) {
            w= p->w;
            if (cst[w] > cst[v]+p->cst){
                cst[w]=cst[v]+p->cst;
                parnt[w] = v;
            }
            p = p->next;
        }
    }
}
```

Consumo de tempo

> O consumo de tempo da função `bellman_ford3` é $O(VA)$.

FIFO Bellman-Ford

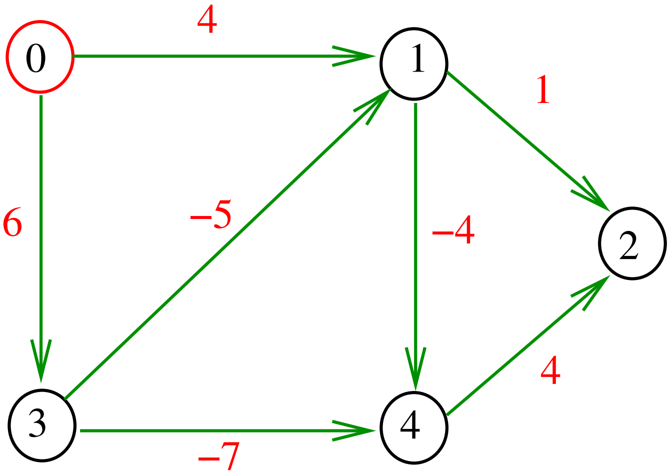
S 21.7

FIFO-Bellman-Ford

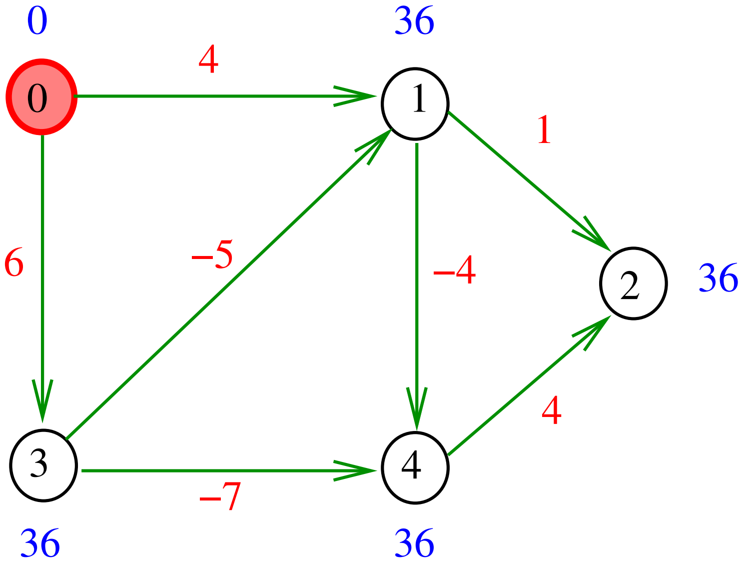
O algoritmo de Bellman e Ford pode ser divido em **passos**

> *um passo para cada valor de* k *(=0,1,2,...).*

FIFO-Bellman-Ford

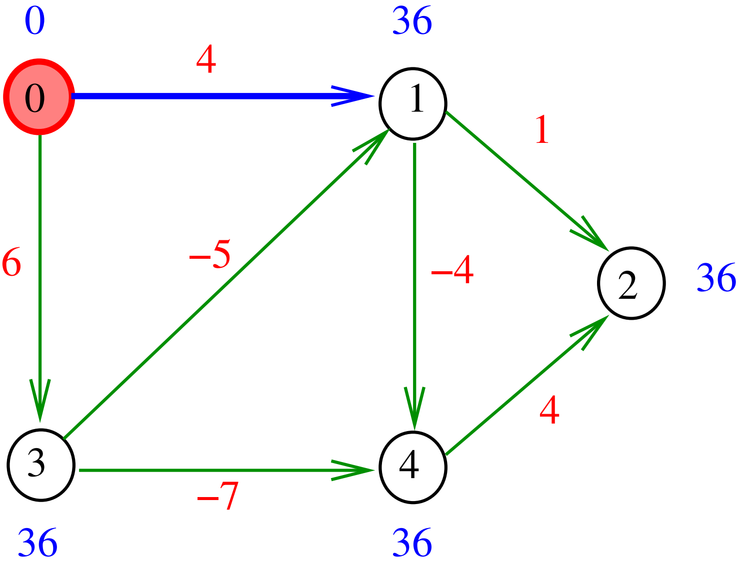


FIFO-Bellman-Ford

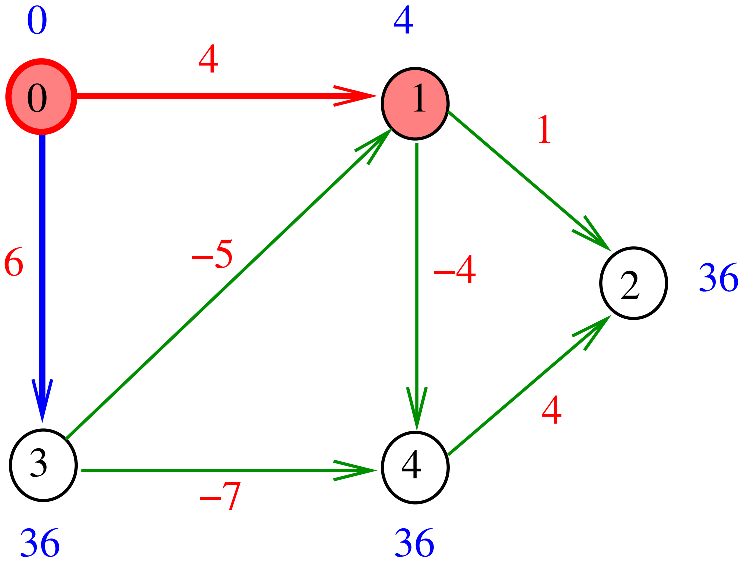


Fim do passo k=0

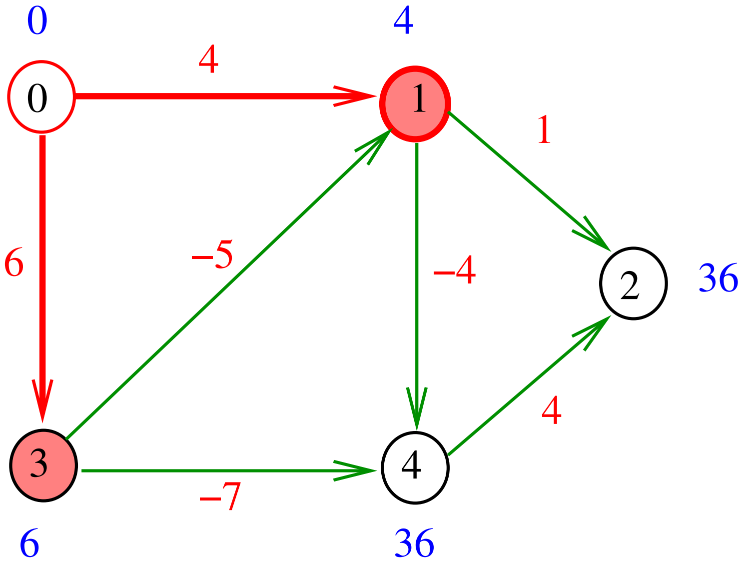
FIFO-Bellman-Ford



FIFO-Bellman-Ford

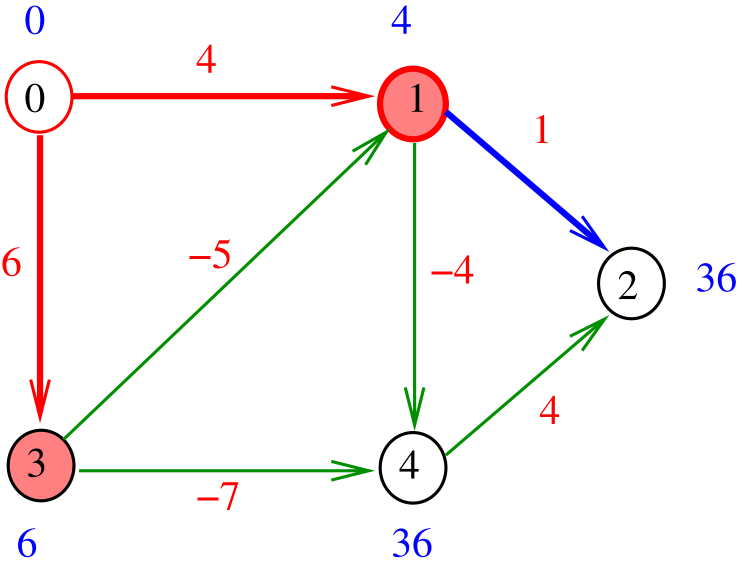


FIFO-Bellman-Ford

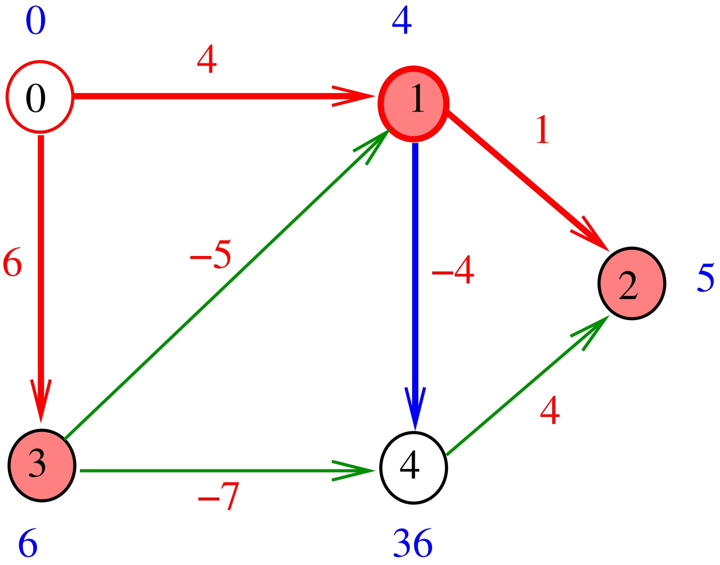


Fim do passo k=1

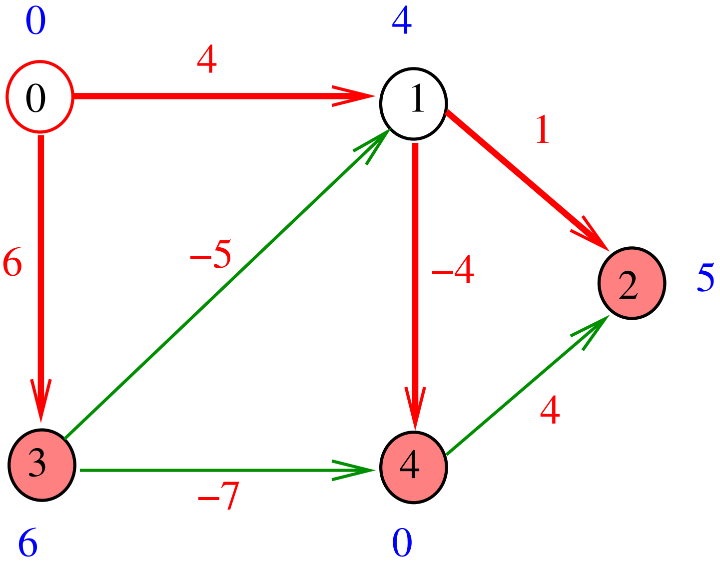
FIFO-Bellman-Ford



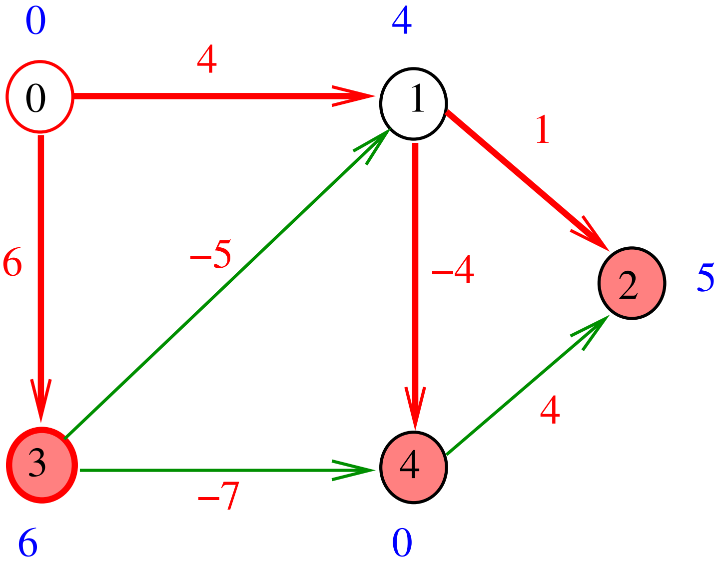
FIFO-Bellman-Ford



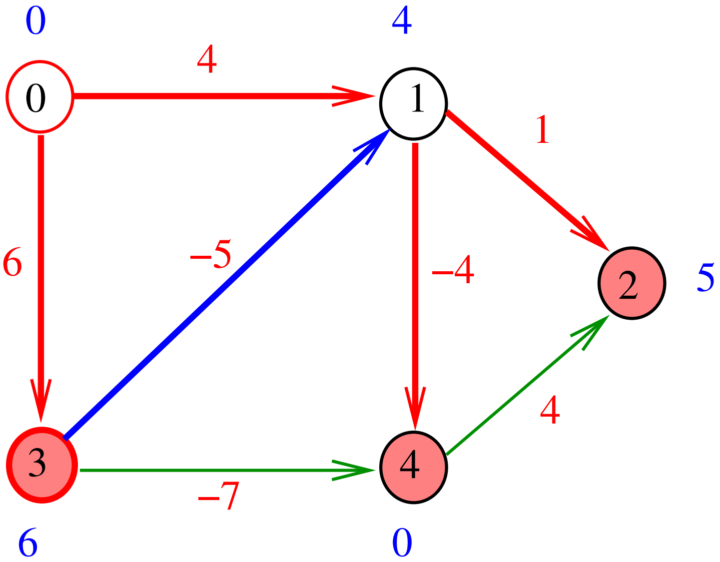
FIFO-Bellman-Ford



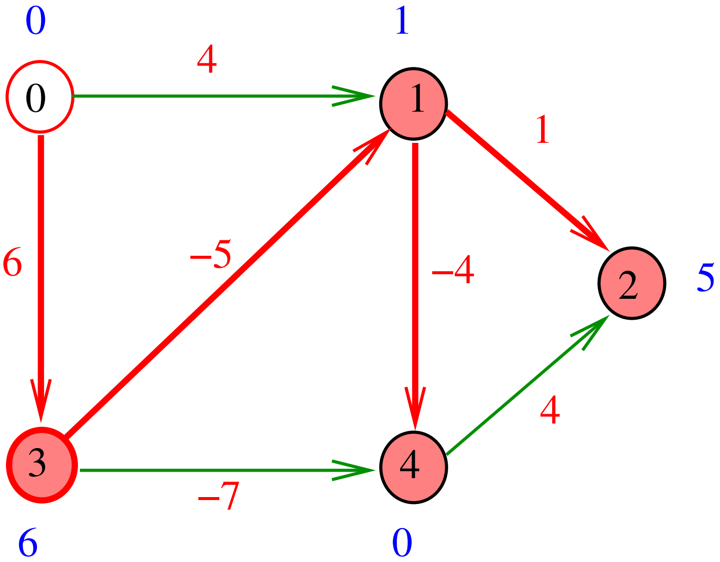
FIFO-Bellman-Ford



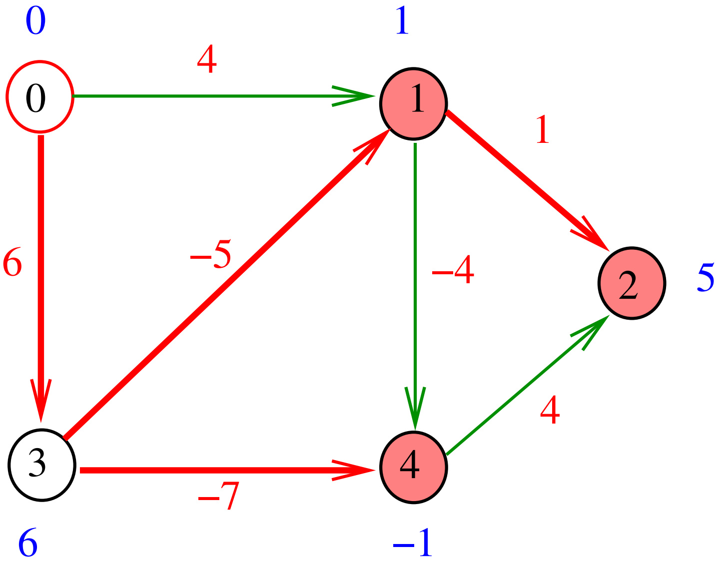
FIFO-Bellman-Ford



FIFO-Bellman-Ford

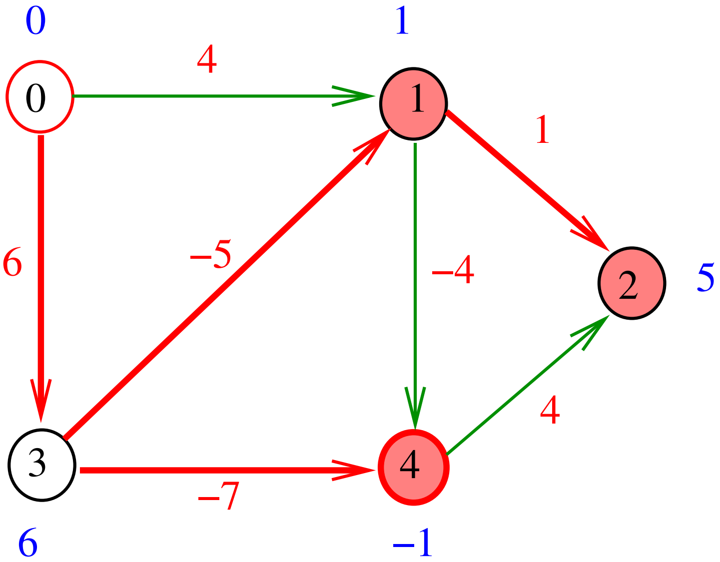


FIFO-Bellman-Ford

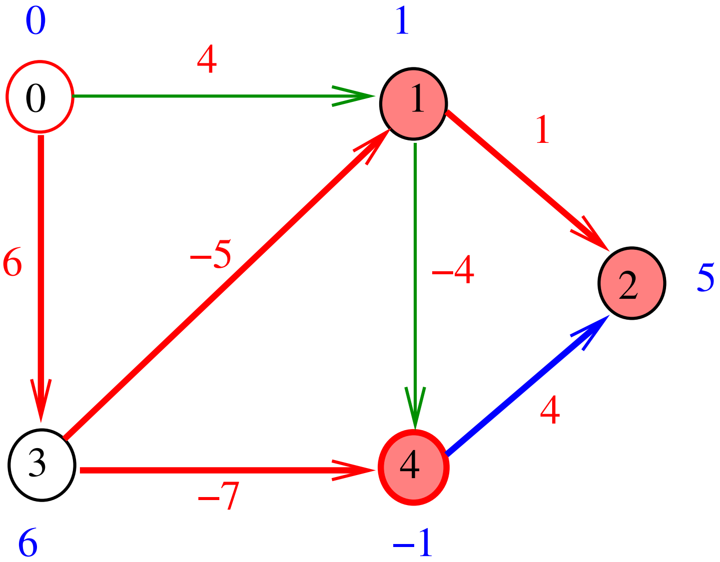


Fim do passo k=2

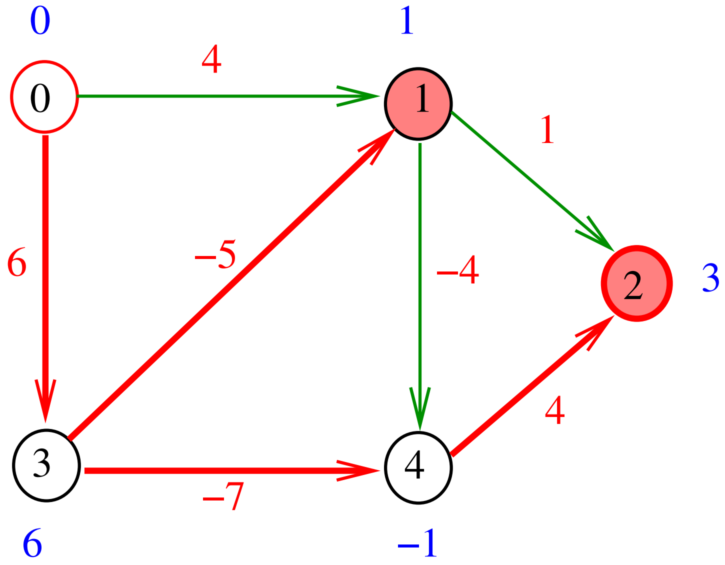
FIFO-Bellman-Ford



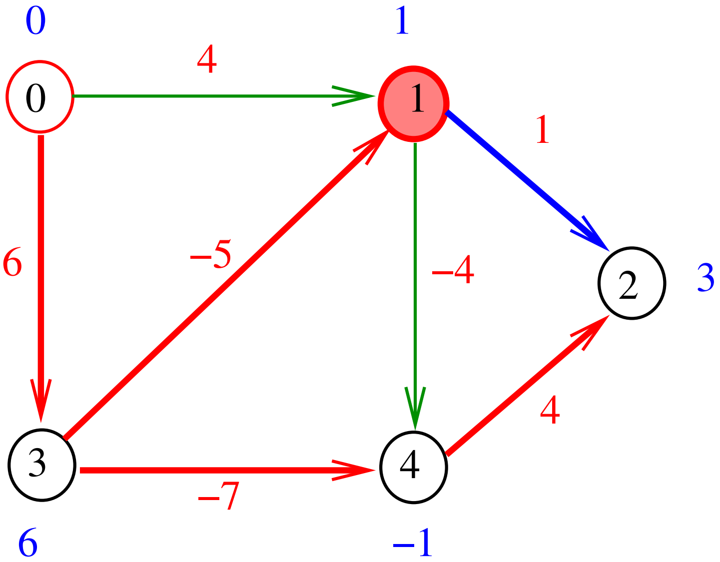
FIFO-Bellman-Ford



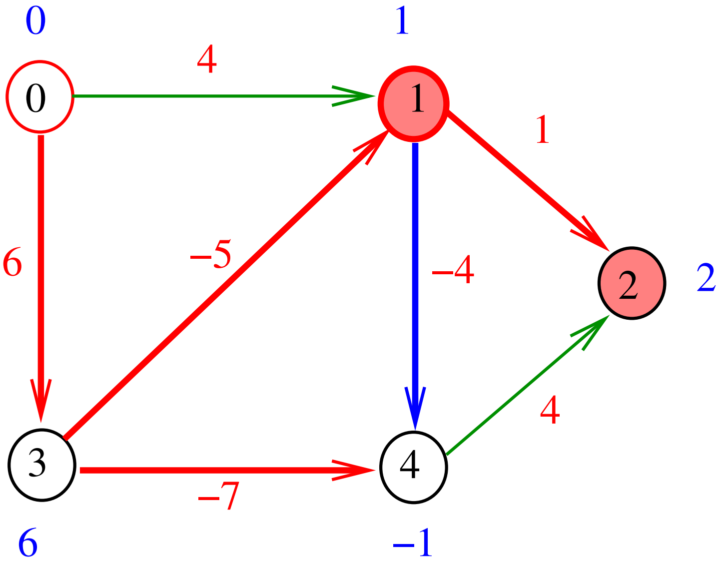
FIFO-Bellman-Ford



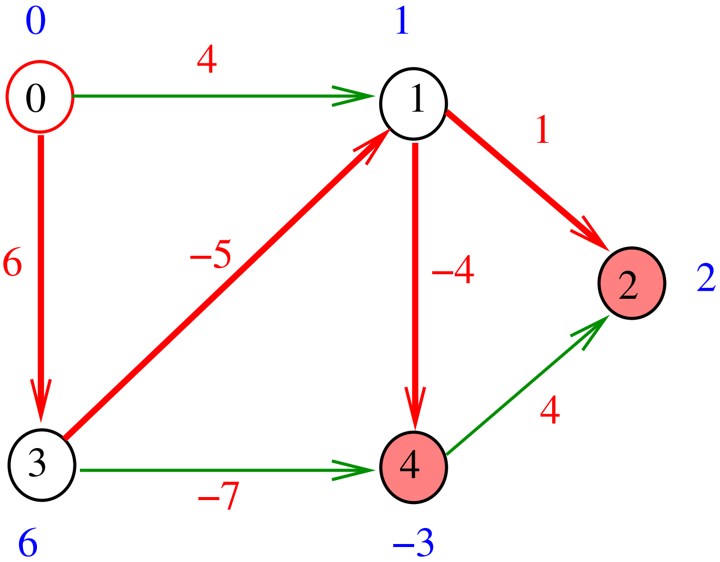
FIFO-Bellman-Ford



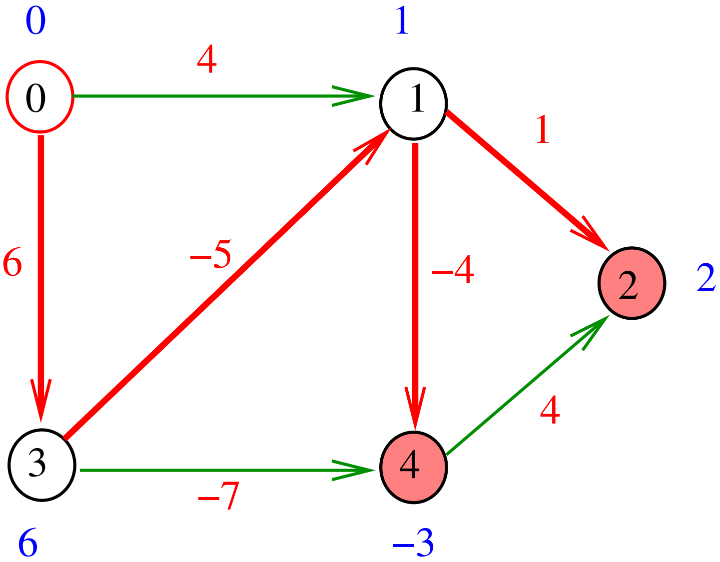
FIFO-Bellman-Ford



FIFO-Bellman-Ford

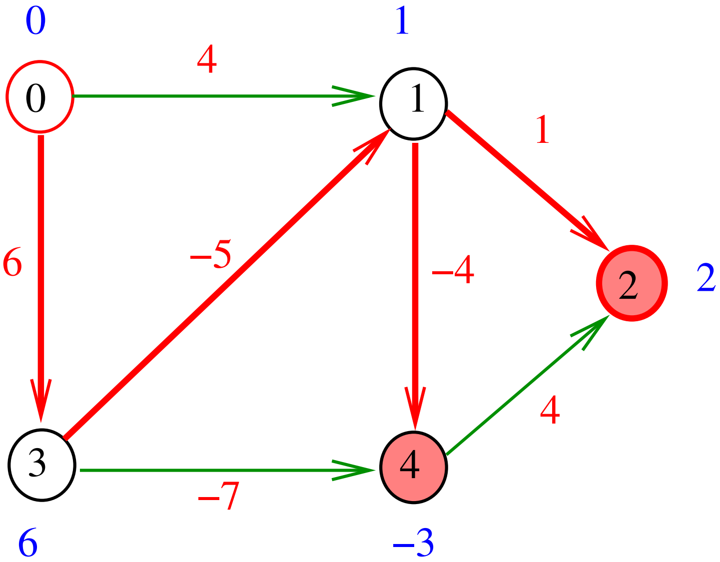


FIFO-Bellman-Ford

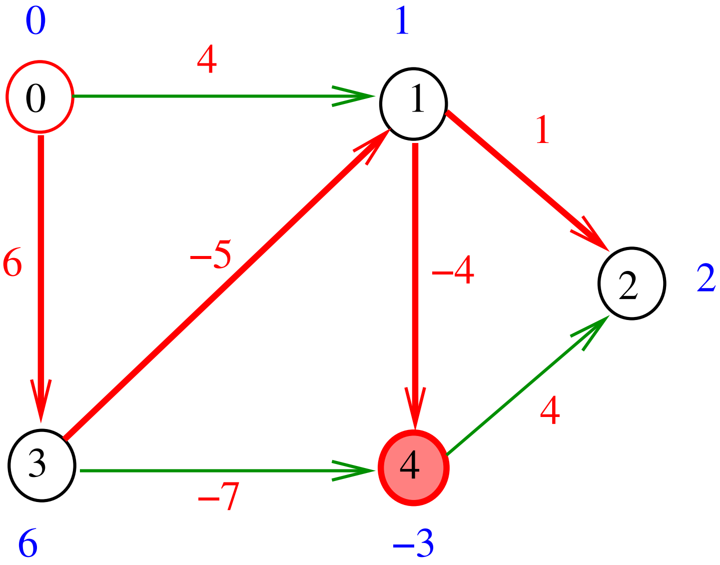


Fim do passo k=3

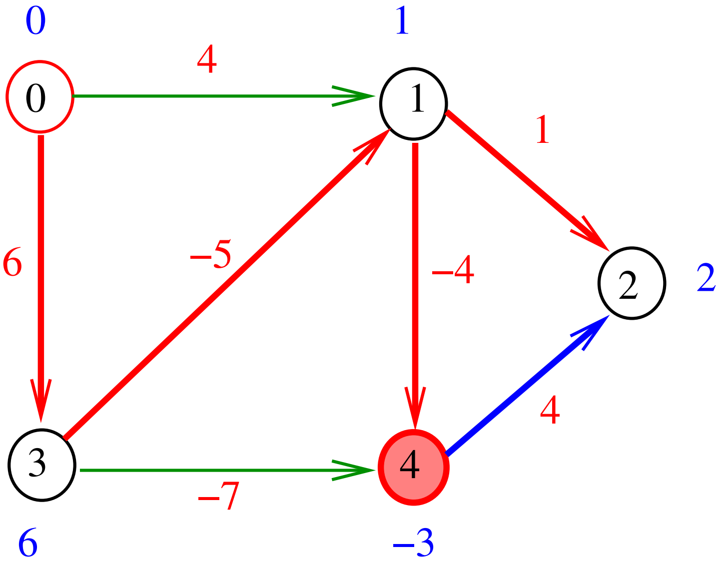
FIFO-Bellman-Ford



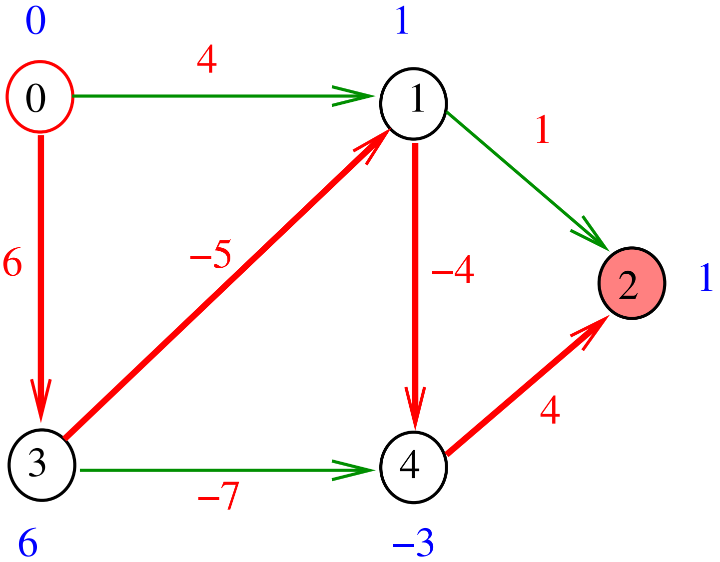
FIFO-Bellman-Ford



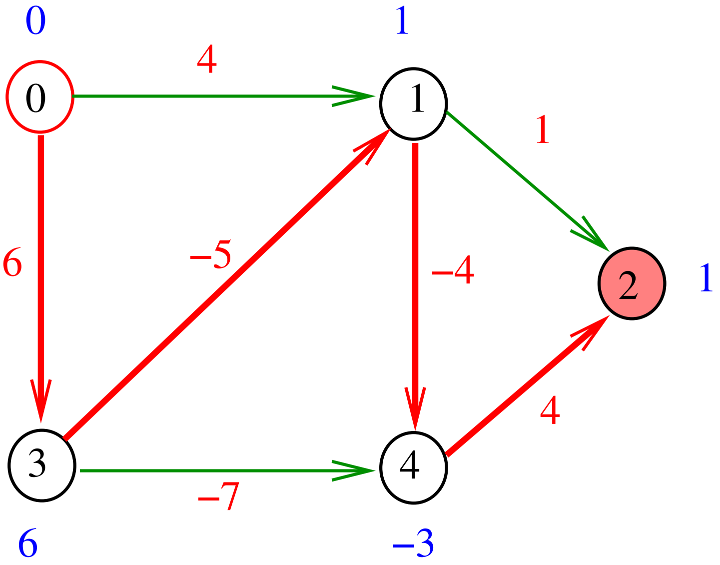
FIFO-Bellman-Ford



FIFO-Bellman-Ford

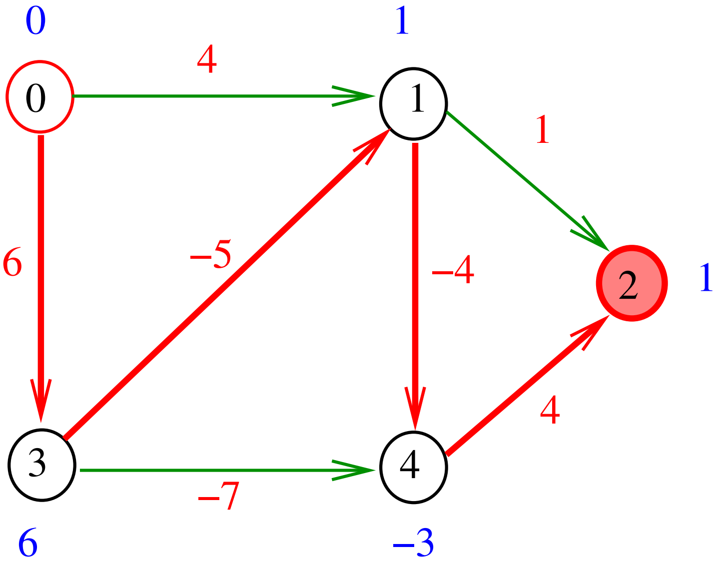


FIFO-Bellman-Ford

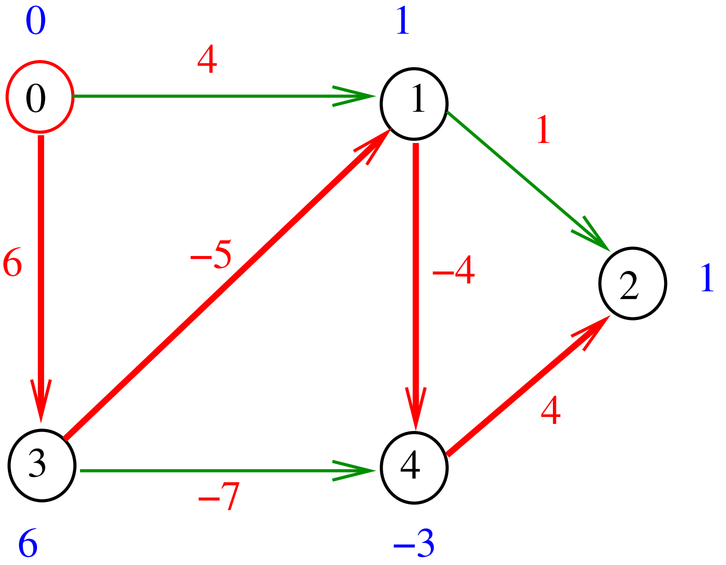


Fim do passo k=4

FIFO-Bellman-Ford



FIFO-Bellman-Ford



Fim do passo k=5

bellman-ford

Recebe digrafo G com custos (possivelmente negativos) nos arcos e um vértice s

Se o digrafo não tem ciclo negativo alcançável a partir de s, calcula uma arborescência de caminhos mínimos com raiz s.

A arborescência é armazenada no vetor parnt

As distâncias em relação a s são armazenadas no vetor cst

bellman-ford

A implementação utiliza uma fila.

Supomos que em cada instante haja no máximo uma cópia de cada vértice na fila:

> *um vértice deve ser inserido na fila apenas se já não estiver na fila.*

```
#define SENTINELA G->V
#define maxV 10000;
double cst[maxV];
Vertex parnt[maxV];
void bellman-ford(Digraph G, Vertex s)
```

bellman-ford

```
void bellman-ford(Digraph G, Vertex s)
{
    Vertex v, w; link p; int k=0;
    for (v = 0; v < G->V; v++) {
        cst[v] = maxCST;
        parnt[v] = -1;
    }
    QUEUEinit(G->V);
    cst[s] = 0;
    parnt[s] = s;
    QUEUEput(s); QUEUEput(SENTINELA);
```

```
   while (!QUEUEempty()) {
      v = QUEUEget();
      if (v == SENTINELA) {
         if (k++ == G->V) return;
         QUEUEput(SENTINELA);
      } else
      for(p=G->adj[v];p!=NULL;p=p->next)
         if(cst[w=p->w]>cst[v]+p->cst)
            cst[w]=cst[v]+p->cst;
            parnt[w] = v;
            QUEUEput(w);
         }
   }
}
```

Relação invariante

No início de cada iteração do **while** da linha 9 vale que

$$\texttt{cst}[\texttt{v}] \leq \texttt{custo}[\texttt{k}][\texttt{v}] = \text{o menor custo de um caminho de } \texttt{s} \text{ a } \texttt{v} \text{ com} \leq \texttt{k} \text{ arcos}$$

Ciclos negativos

```
if (v == SENTINELA) {
    if (k++ == G->V) {
        if (!QUEUEempty()) {
        /* tem ciclo negativo */
        }
        return;
    }
    QUEUEput(SENTINELA);
}
```

O **ciclo negativo** pode ser encontrado no digrafo representado por parnt

Consumo de tempo

linha	número de execuções da linha
2–4	$\Theta(\texttt{V})$
5	$= 1$ `QUEUEinit`
6–7	$= 1$
8	$= 2$ `QUEUEput`
9–10	$\mathrm{O}(\texttt{V}^2)$ `QUEUEempty` e `QUEUEget`
11	$\mathrm{O}(\texttt{V}^2)$
12	$\mathrm{O}(\texttt{V})$
13	$\leq \texttt{V}$ `QUEUEput`
14–17	$\mathrm{O}(\texttt{VA})$
18	$\mathrm{O}(\texttt{VA})$ `QUEUEput`
total	$= \mathrm{O}(\texttt{VA}) + ???$

Conclusão

O consumo de tempo da função `bellman-ford` é $O(VA)$ mais o consumo de tempo de

1	execução de	`QUEUEinit` e `QUEUEget`,
$O(VA)$	execuções de	`QUEUEput`,
$O(V^2)$	execuções de	`QUEUEempty`, e
$O(V^2)$	execuções de	`QUEUEget`

Conclusão

Se implementarmos a fila de tal forma que cada operação consuma tempo constante teremos:

> O consumo de tempo da função `bellman_ford` é $O(VA)$.

Conclusão

Para todo grafo digrafo G com custo nos arcos e todo par de vértices s e t, vale uma e apenas umas das seguintes afirmações:

- não existe caminho de s a t;
- existe um caminho mínimo de s a t; ou
- existe um caminho de s a t que contém um ciclo negativo

Ciclos negativos

Problema: Dado um digrafo com custos nos arcos, decidir se o digrafo possui algum ciclo negativo.

Uma adaptação da função `bellman_ford` decide se um dado digrafo com custos nos arcos possui algum ciclo negativo. O consumo de tempo dessa função adaptada é $O(VA)$.

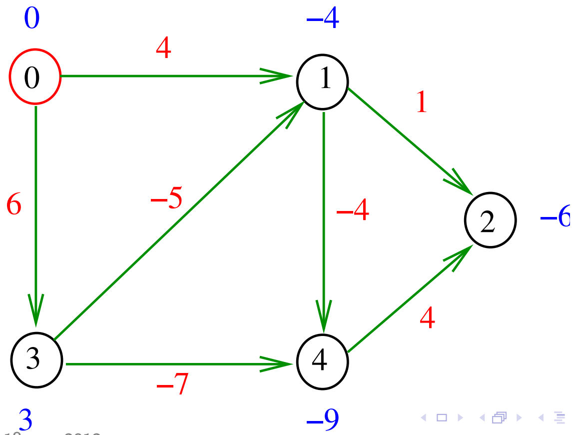
Mais Potenciais

Potenciais

Um **potencial** é um vetor y indexado pelos vértices do digrafo tal que

$y[\texttt{w}] - y[\texttt{v}] \leq \texttt{G->adj}[\texttt{v}][\texttt{w}]$ para todo arco `v-w`

Exemplo:

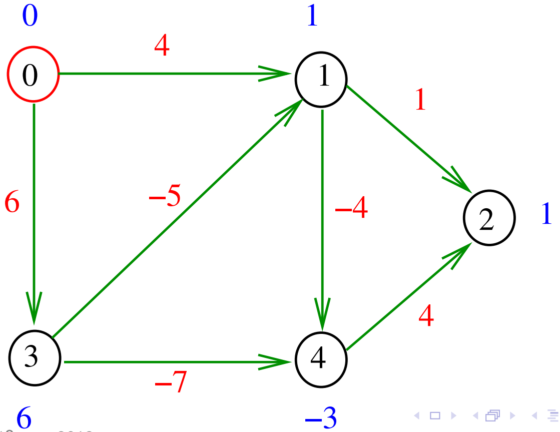


Potenciais

Um **potencial** é um vetor y indexado pelos vértices do digrafo tal que

$$y[\texttt{w}] - y[\texttt{v}] \leq \texttt{G->adj}[\texttt{v}][\texttt{w}] \text{ para todo arco } \texttt{v-w}$$

Exemplo:

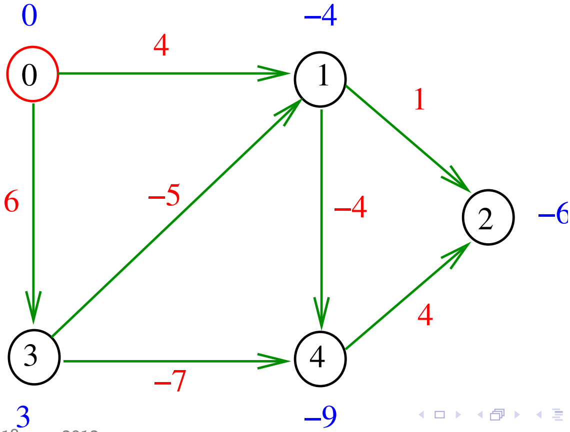


Potenciais

Um **potencial** é um vetor y indexado pelos vértices do digrafo tal que

$$y[\texttt{w}] \leq y[\texttt{v}] + \texttt{G->adj}[\texttt{v}][\texttt{w}] \text{ para todo arco } \texttt{v-w}$$

Exemplo:

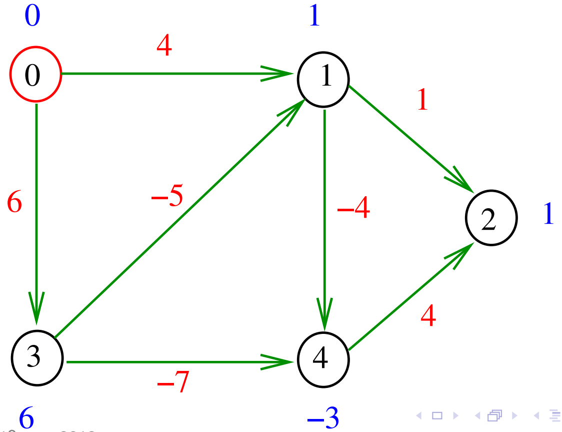


Potenciais

Um **potencial** é um vetor y indexado pelos vértices do digrafo tal que

$$y[\texttt{w}] \leq y[\texttt{v}] + \texttt{G->adj}[\texttt{v}][\texttt{w}] \text{ para todo arco } \texttt{v-w}$$

Exemplo:

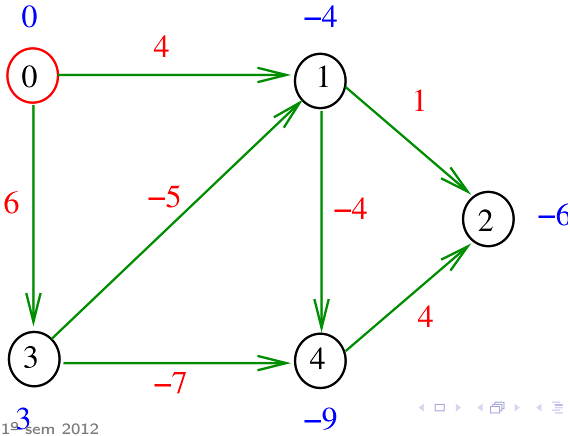


Propriedade dos potencias

Lema da dualidade. Se P é um caminho de s a t e y é um potencial, então

$$\mathtt{custo}(\mathtt{P}) \geq \mathtt{y}[\mathtt{t}] - \mathtt{y}[\mathtt{s}]$$

Exemplo:

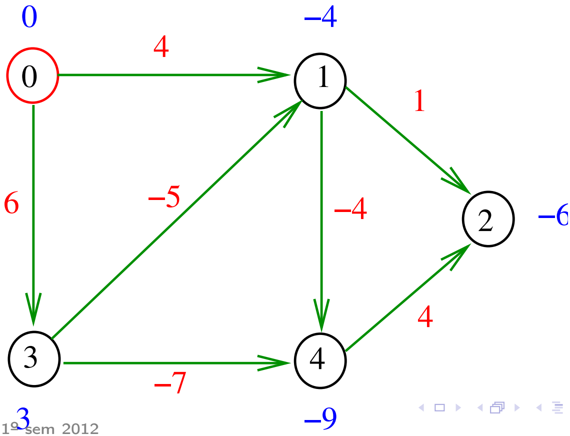


Propriedade dos potencias

Lema da dualidade. Se P é um caminho de s a t e y é um potencial, então

$$\texttt{custo}(P) + y[s] \geq y[t]$$

Exemplo:



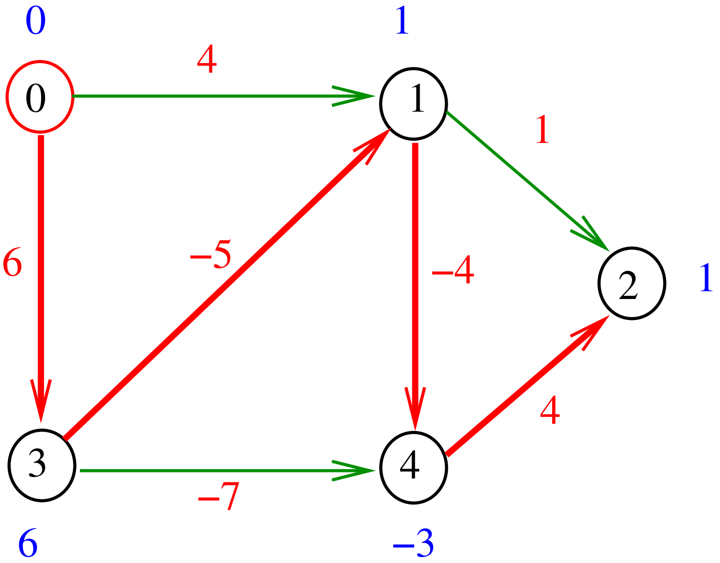
Conseqüência

Se P é um caminho de $\mathtt{s}$ a $\mathtt{t}$ e y é um potencial tais que

$$\mathtt{custo}(\mathrm{P}) = \mathrm{y}[\mathtt{t}] - \mathrm{y}[\mathtt{s}],$$

então P é um caminho **mínimo** e y é um potencial tal que $\mathrm{y}[\mathtt{t}] - \mathrm{y}[\mathtt{s}]$ é **máximo**

Exemplo



Teorema da dualidade

Da propriedade dos -potenciais (lema da dualidade) e da correção de `dijkstra` concluímos o seguinte:

Se s e t são vértices de um digrafo com custo **não-negativos** nos arcos e t está ao alcance de s então

$$\min\{\texttt{custo}(\texttt{P}) : \texttt{P} \text{ é um } \texttt{st}\text{-caminho}\}$$
$$= \max\{y[\texttt{t}] - y[\texttt{s}] : y \text{ é um potencial}\}.$$

Algoritmo de Floyd-Warshall

S 21.3

Problema dos caminhos mínimos entre todos os pares

Problema: Dado um digrafo com custo nos arcos, determinar, para cada par de vértices $\mathtt{s}$, $\mathtt{t}$ o custo de um caminho mínimo de $\mathtt{s}$ a $\mathtt{t}$

Esse problema pode ser resolvido aplicando-se $\mathtt{V}$ vezes o algoritmo de Bellman-Ford

O consumo de tempo dessa solução é $O(\mathtt{V}^2\mathtt{A})$.

Um algoritmo mais eficiente foi descrito por Floyd, baseado em uma idéia de Warshall.

O algoritmo supõe que o digrafo não tem ciclo negativo

Problema dos caminhos mínimos entre todos os pares

Problema: Dado um digrafo com custo nos arcos, determinar, para cada par de vértices `s`, `t` o custo de um caminho mínimo de `s` a `t`

Esse problema pode ser resolvido aplicando-se `V` vezes o algoritmo de Bellman-Ford

O consumo de tempo dessa solução é $O(V^2 A)$.

Um algoritmo mais eficiente foi descrito por Floyd, baseado em uma idéia de Warshall.

O algoritmo supõe que o digrafo não tem ciclo negativo

Problema dos caminhos mínimos entre todos os pares

Problema: Dado um digrafo com custo nos arcos, determinar, para cada par de vértices s, t o custo de um caminho mínimo de s a t

Esse problema pode ser resolvido aplicando-se V vezes o algoritmo de Bellman-Ford

O consumo de tempo dessa solução é $O(V^2 A)$.

Um algoritmo mais eficiente foi descrito por Floyd, baseado em uma idéia de Warshall.

O algoritmo supõe que o digrafo não tem ciclo negativo

Programação dinâmica

$0, 1, 2, \ldots, \texttt{V}-1$ = lista dos vértices do digrafo

$\texttt{custo}[\texttt{k}][\texttt{s}][\texttt{t}]$ = menor custo de um caminho de $\texttt{s}$ a $\texttt{t}$ usando vértices em $\{\texttt{s}, \texttt{t}, 0, 1, \ldots, \texttt{k}-1\}$

Recorrência:

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

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

Programação dinâmica

$0, 1, 2, \ldots, \mathtt{V}{-}1$ = lista dos vértices do digrafo

$\mathtt{custo}[\mathtt{k}][\mathtt{s}][\mathtt{t}]$ = menor custo de um caminho de $\mathtt{s}$ a $\mathtt{t}$ usando vértices em $\{\mathtt{s}, \mathtt{t}, 0, 1, \ldots, \mathtt{k}{-}1\}$

Recorrência:

$$\begin{aligned}
\mathtt{custo}[0][\mathtt{s}][\mathtt{t}] &= \mathtt{G\text{->}adj}[\mathtt{s}][\mathtt{t}] \\
\mathtt{custo}[\mathtt{k}][\mathtt{s}][\mathtt{t}] &= \min\{\mathtt{custo}[\mathtt{k}\text{-}1][\mathtt{s}][\mathtt{t}], \\
&\quad \mathtt{custo}[\mathtt{k}\text{-}1][\mathtt{s}][\mathtt{k}\text{-}1] + \mathtt{custo}[\mathtt{k}\text{-}1][\mathtt{k}\text{-}1][\mathtt{t}]\}
\end{aligned}$$

Se o digrafo não tem ciclo negativo acessível a partir de $\mathtt{s}$, então $\mathtt{custo}[\mathtt{V}][\mathtt{s}][\mathtt{t}]$ é o menor custo de um **caminho simples** de $\mathtt{s}$ a $\mathtt{t}$

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

Consumo de tempo

O consumo de tempo da função `floyd_warshall1` é $O(V^3)$.

```
void floyd_warshall (Digraph G){
    Vertex s, t; double d;
    for (s=0; s < G->V; s++)
        for (t=0; t < G->V; t++)
            cst[s][t] = G->adj[s][t];
    for (k=1; k <= G->V; k++)
        for (s=0; s < G->V; s++)
            for (t=0; t < G->V; t++){
                d=cst[s][k-1]+cst[k-1][t];
                if (cst[s][t] > d)
                    cst[s][t] = d;
            }
        }
}
```

Relação invariante

No início de cada iteração da linha 5 vale que

$\texttt{cst}[\texttt{s}][\texttt{t}] = \texttt{custo}[\texttt{k}][\texttt{s}][\texttt{t}] =$ o menor custo de um caminho de $\texttt{s}$ a $\texttt{t}$ usando vértices em $\{\texttt{s}, \texttt{t}, 0, 1, \ldots, \texttt{k}{-}1\}$

Novo resumo

função	consumo de tempo	observação
DAGmin	$O(V + A)$	digrafos acíclicos custos arbitrários
dijkstra	$O(A \lg V)$	custos ≥ 0, min-heap
	$O(V^2)$	custos ≥ 0, fila
bellman-ford	$O(V^3)$ $O(VA)$	digrafos densos digrafos esparsos
floyd-warshall	$O(V^3)$	digrafos sem ciclos negativos

O problema SPT em digrafos com ciclos negativos é NP-difícil.