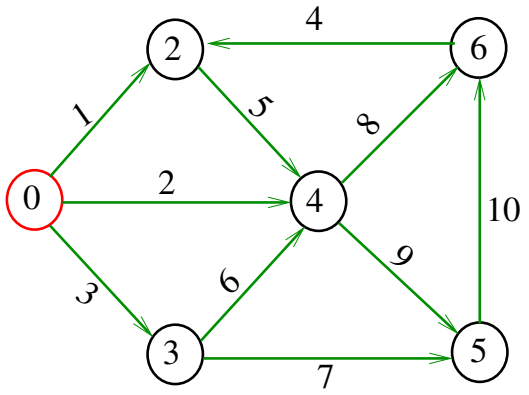
Algoritmo de Bellman-Ford

Problema da SPT

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

Entra:



Sai:

Propriedade da subestrutura ótima

Se G é um digrafo com custos não-negativos nos arcos e v_0-v_1-v_2-...-v_k é um caminho mínimo então v_i-v_{i+1}-...-v_j é um caminho mínimo para $0 \leq i \leq j \leq k$

custo[v][w] = menor custo de uma caminho de v a w

Propriedade 1

O valor de custo[s][t] é

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

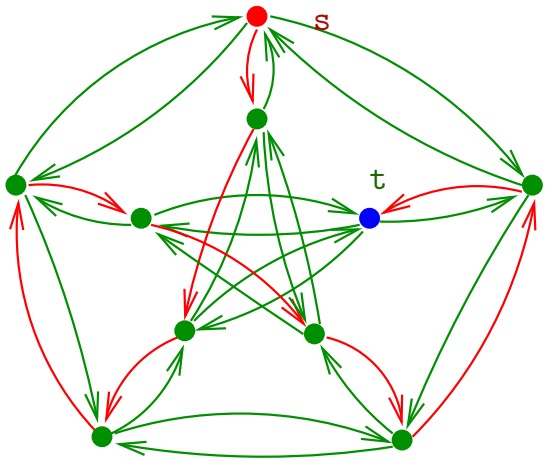
Propriedade 2

O valor de custo[s][t] é

Subestrutura ótima ...

todos custos = -1

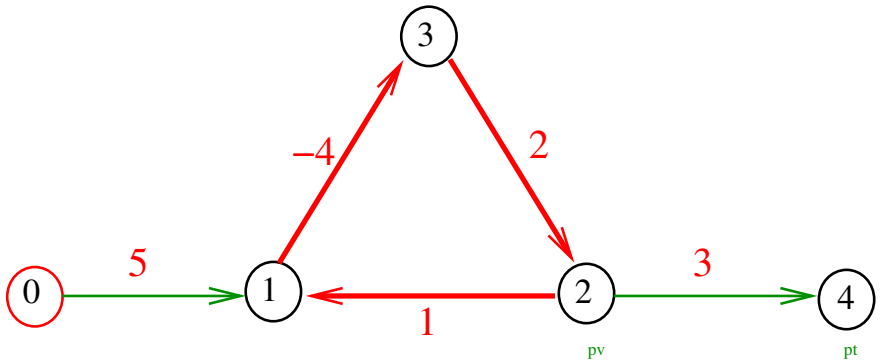
G



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

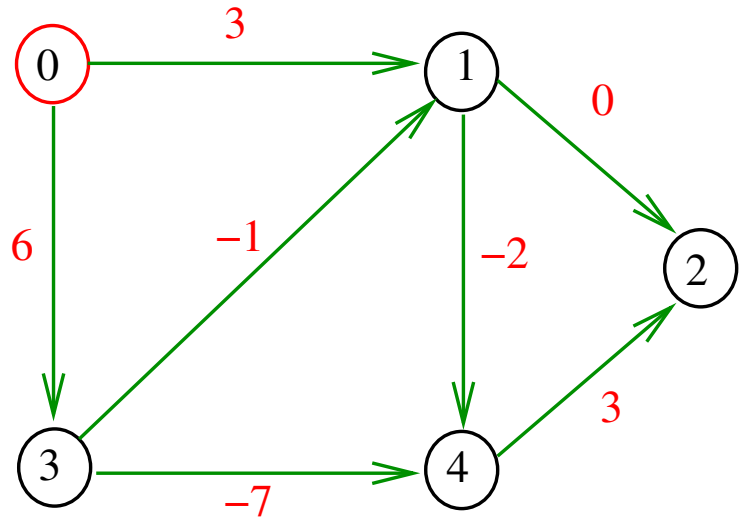
custo[k][w] = menor custo de um caminho de s a w com $\leq$ k arcos.

Recorrência

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

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

Exemplo



	0	1	2	3	4	v
0	0	∗	∗	∗	∗	


```
void bellman_ford1(Digraph G, Vertex s){
 1   Vertex v, w; double d;
 2   for (v=0; v < G->V; v++)
 3       custo[0][v] = INFINITO;
 4   custo[0][s] = 0;
 5   for (k=1; k < G->V; k++)
 6       for (w=0; w < G->V; w++){
 7           custo[k][w] = custo[k-1][w];
 8           for (v=0; v < G->V; v++){
 9               d=custo[k-1][v]+G->adj[v][w];
10               if (custo[k][w] > d)
11                   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 P′ tem no $\leq$ V-1 arcos e portanto

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

Logo, C é um ciclo de custo negativo.


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


Conclusão

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

Relação invariante

Em /* A */ na linha vale que

cst[v] $\leq$ custo[k-1][v] = o menor custo de um caminho de s a v com até 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){
1   Vertex v, w;
2   link p;
3   for (v=0; v < G->V; v++){
4       cst[v] = INFINITO;
5       parnt[v] = -1;
    }
6   cst[s] = 0;
```


Bellman-Ford

```
7   for (k=1; k < G->V; k++)
8       for (v=0;v < G->V; v++){
9           p = G->adj[v];
10          while (p != NULL) {
11              w= p->w;
12              if (cst[w] > cst[v]+p->cst){
13                  cst[w]=cst[v]+p->cst;
14                  parnt[w] = v;
                }
15              p = p->next;
            }
        }
}
```


Consumo de tempo

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

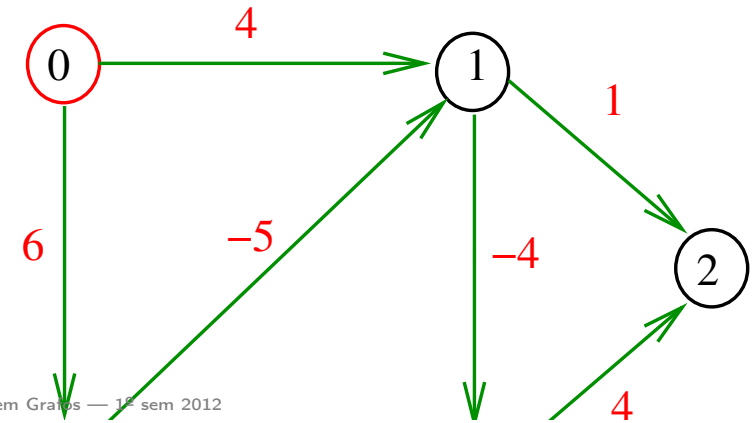
FIFO Bellman-Ford

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,...).



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)
{
1   Vertex v, w; link p; int k=0;
2   for (v = 0; v < G->V; v++) {
3       cst[v] = maxCST;
4       parnt[v] = -1;
    }
5   QUEUEinit(G->V);
6   cst[s] = 0;
7   parnt[s] = s;
8   QUEUEput(s); QUEUEput(SENTINELA);
```



```
 9  while (!QUEUEempty()) {
10      v = QUEUEget();
11      if (v == SENTINELA) {
12          if (k++ == G->V) return;
13          QUEUEput(SENTINELA);
        } else
14      for(p=G->adj[v];p!=NULL;p=p->next)
15          if(cst[w=p->w]>cst[v]+p->cst)
16              cst[w]=cst[v]+p->cst;
17              parnt[w] = v;
18              QUEUEput(w);
            }
    }
}
```


Relação invariante

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

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

Ciclos negativos

```
11      if (v == SENTINELA) {
12          if (k++ == G->V) {
12              if (!QUEUEempty()) {
12              /* tem ciclo negativo */
12              }
12              return;
            }
13          QUEUEput(SENTINELA);
        }
```

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

Consumo de tempo

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

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 $O(VA)$.

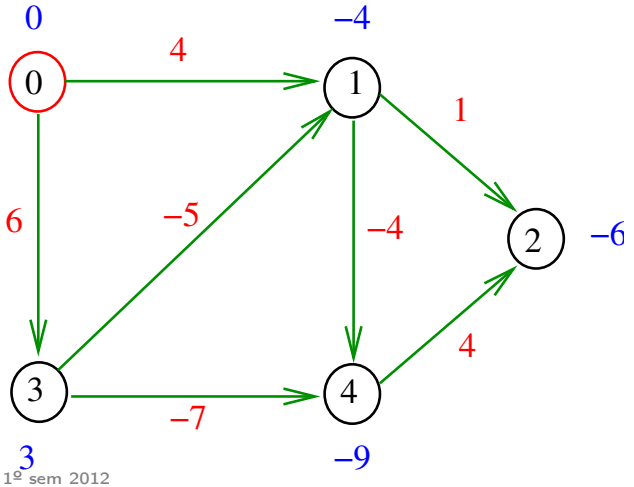
Mais Potenciais

Potenciais

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

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

Exemplo:

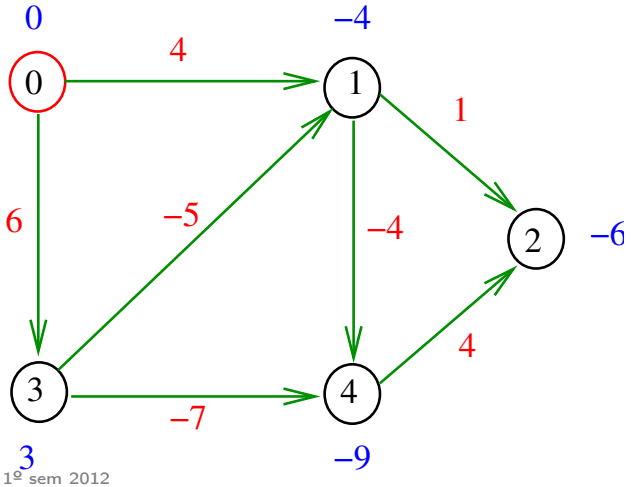


Potenciais

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

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

Exemplo:

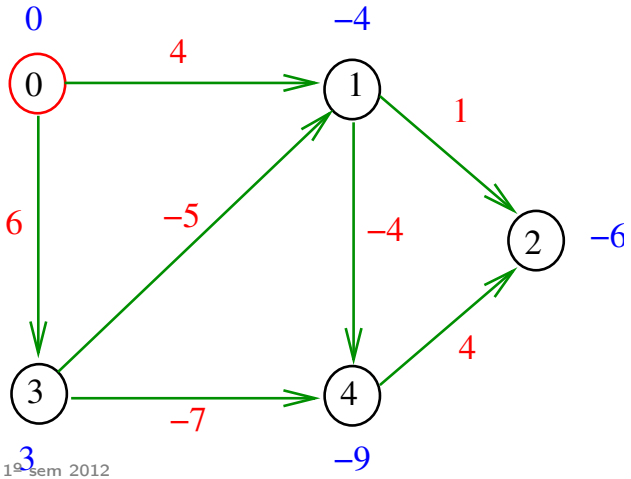


Propriedade dos potencias

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

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

Exemplo:

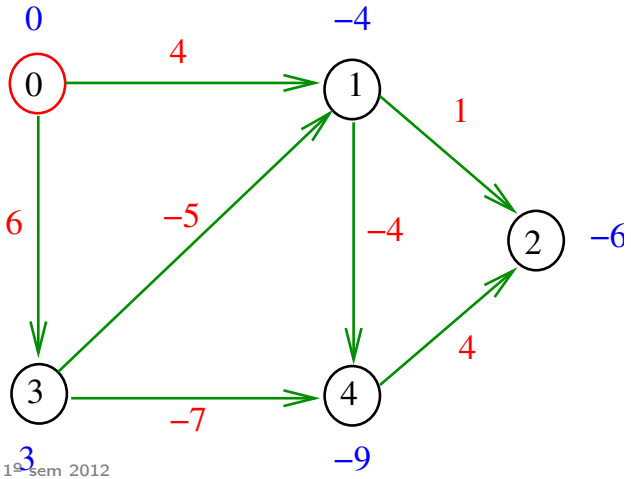


Propriedade dos potencias

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:



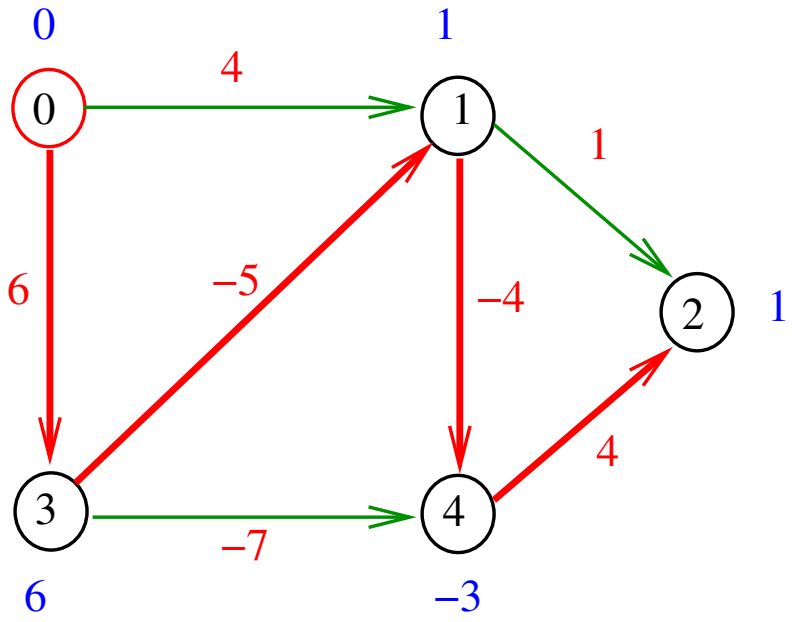
Conseqüência

Se P é um caminho de s a t e y é um potencial tais que

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

então P é um caminho **mínimo** e y é um potencial tal que $y[t] - y[s]$ 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
>
> $$\min\{\texttt{custo}(\texttt{P}) : \texttt{P} \text{ é um } \texttt{st}\text{-caminho}\}$$
> $$= \max\{\texttt{y[t]} - \texttt{y[s]} : \texttt{y} \text{ é um potencial}\}.$$

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 `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 $O(\texttt{V}^2\texttt{A})$.

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

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

Recorrência:

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

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

```
void floyd_warshall (Digraph G){
 1  Vertex s, t; double d;
 2  for (s=0; s < G->V; s++)
 3      for (t=0; t < G->V; t++)
 4          custo[0][s][t] = G->adj[s][t];
 5  for (k=1; k <=G->V; k++)
 6      for (s=0; s < G->V; s++)
 7          for (t=0; t < G->V; t++){
 8              custo[k][s][t]=custo[k-1][s][t];
 9              d=custo[k-1][s][k-1]
                          +custo[k-1][k-1][t];
10              if (custo[k][s][t] > d)
11                  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){
 1  Vertex s, t; double d;
 2  for (s=0; s < G->V; s++)
 3      for (t=0; t < G->V; t++)
 4          cst[s][t] = G->adj[s][t];
 5  for (k=1; k <= G->V; k++)
 6      for (s=0; s < G->V; s++)
 7          for (t=0; t < G->V; t++){
 8              d=cst[s][k-1]+cst[k-1][t];
10              if (cst[s][t] > d)
11                  cst[s][t] = d;
            }
        }
}
```


Relação invariante

No início de cada iteração da linha 5 vale que

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

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.