Programação Dinâmica Discreta

1-Introdução

- Programação dinâmica
  - metodologia de otimização
  - problemas que requerem decisões sequenciais interelacionadas
  - decisão tem um custo imediato e afeta contexto decisões futuras

- Objetivo
  - como obter a sequência de decisões
  - minimização custo total em um número de estágios
  - compromisso entre custo imediato e futuro

Processos de decisão multiestágios

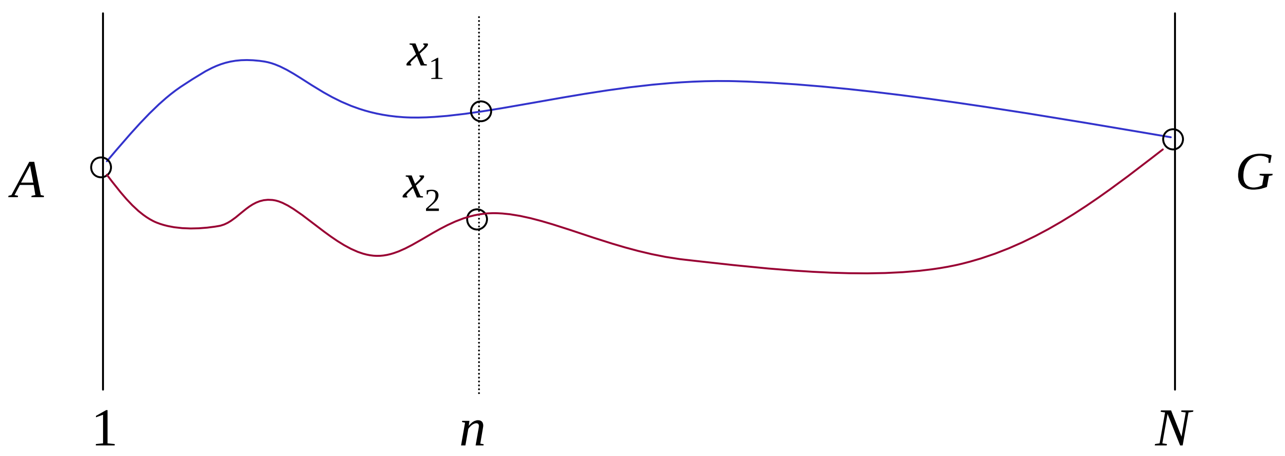
- Decisão multiestágios
  - processo que pode ser desdobrado em um número de etapas seqüenciais, ou estágios
- Estado
  - condição do processo num dado estágio é o estado neste estágio
- Decisões
  - opções que se tem em cada estágio
  - cada decisão causa uma transição do estado

- Estratégia (política)
  - uma seqüência de decisões
  - uma decisão para cada estado do processo
- Retorno
  - custo, benefício, associado a cada estágio de decisão
  - pode variar com o estágio e o estado
- Questão
  - determinar a política ótima (aquela que resulta no melhor retorno)

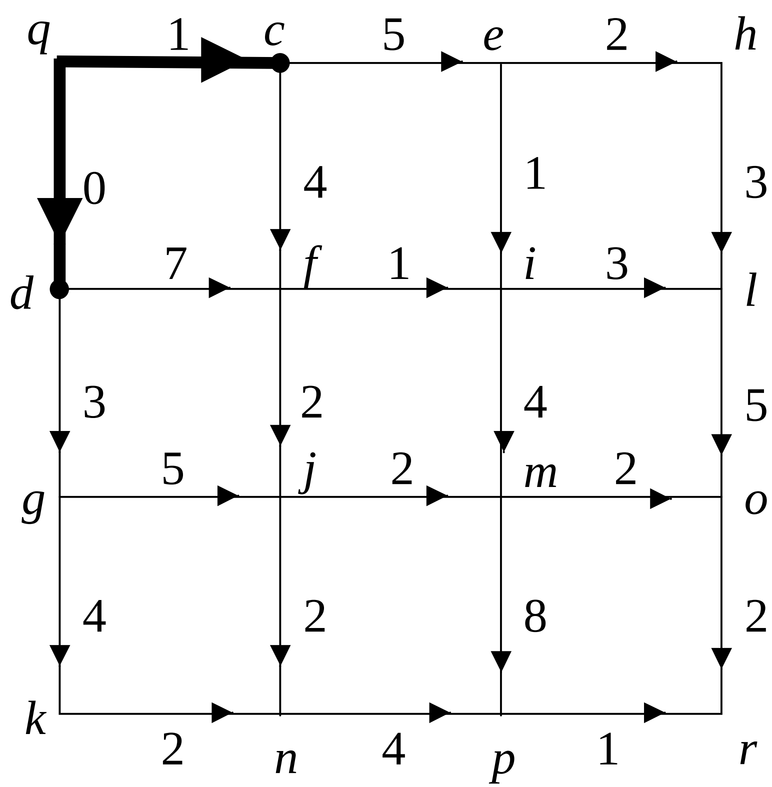
Princípio de otimalidade de Bellman

Uma estratégia ótima apresenta a propriedade segundo a qual, a despeito das decisões tomadas para se atingir um estado particular num certo estágio, as decisões restantes a partir deste estado devem constituir uma estratégia ótima.

[Richard Bellman, 1957]



2-Problema do caminho mínimo

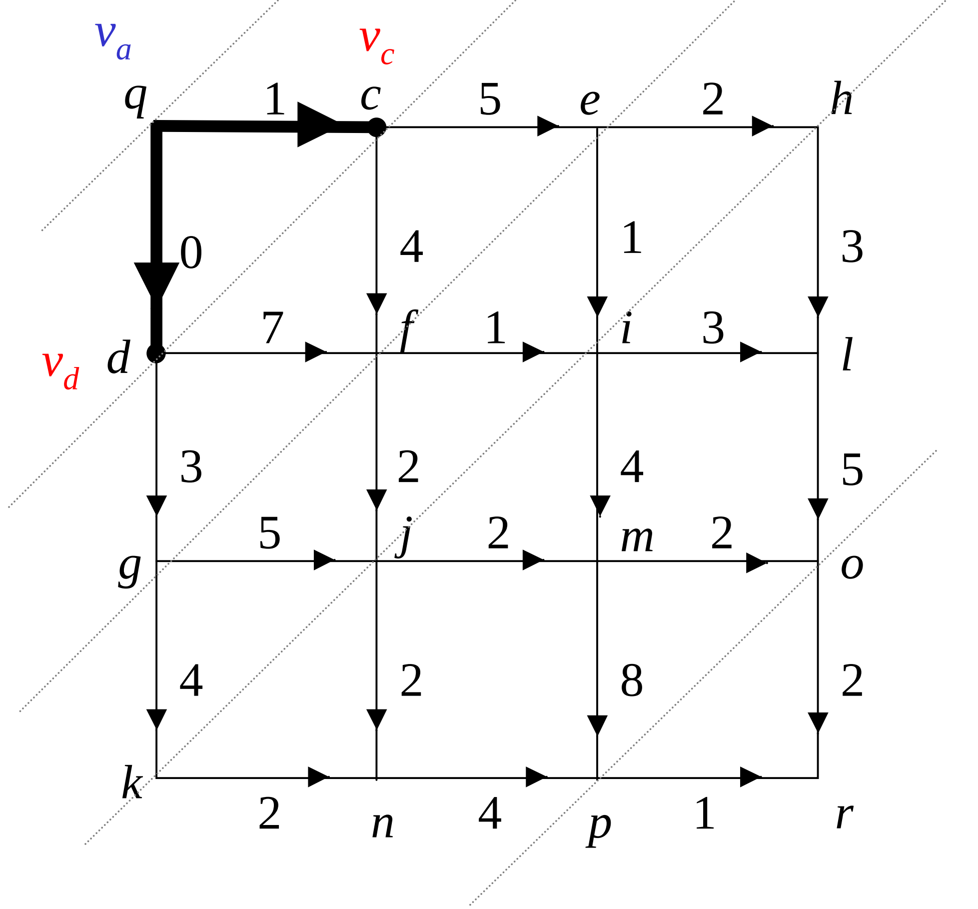


Qual é o caminho de menor esfôrço (tempo, custo, dis - tância, etc.) entre *q* e *r* ?

• 20 caminhos distinos
• 5 adições por caminho
• 19 comparações

- Programação dinâmica backward

v_i melhor caminho de q até i



$v_q = \min\{1 + v_c, 0 + v_d\}$

$v_c = \min\{5 + v_e, 4 + v_f\}$
$v_d = \min\{7 + v_f, 3 + v_g\}$
....

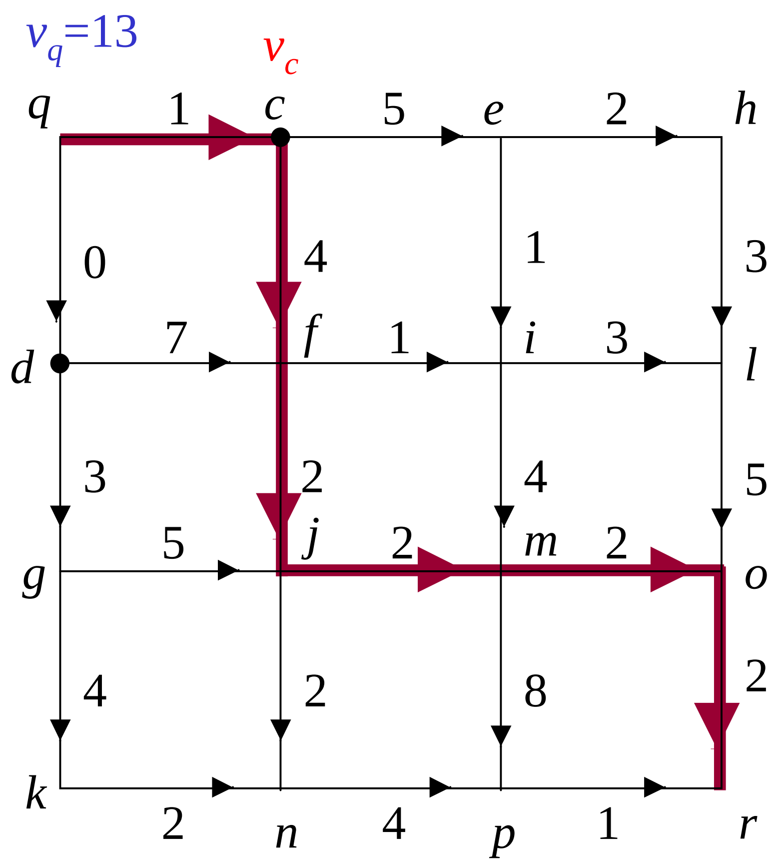
$v_l = 5 + v_o$
$v_m = \min\{2 + v_o, 8 + v_p\}$
$v_n = 4 + v_p$

$v_o = 2$
$v_p = 1$

$v_r = 0$

S estado

P_S sucessor de S no caminho ótimo de S até r



Estratégia ótima

$P_o = r$ $P_p = r$

$P_l = o$ $P_m = o$ $P_n = p$

$P_h = l$ $P_i = m$ $P_j = m$ $P_k = n$

$P_e = i$ $P_f = j$ $P_g = j$ ou k

$P_c = f$ $P_d = g$

$P_q = c$

24 adições e 9 comparações

- Complexidade

  – função objetivo (custo, utilidade, etc.): aditivamente separável

  – ambientes estocásticos: sistemas Markovianos

  – enumeração exaustiva: $O(|A|^n)$

    • $|A|$ número decisões (ações) em cada estágio (passo)

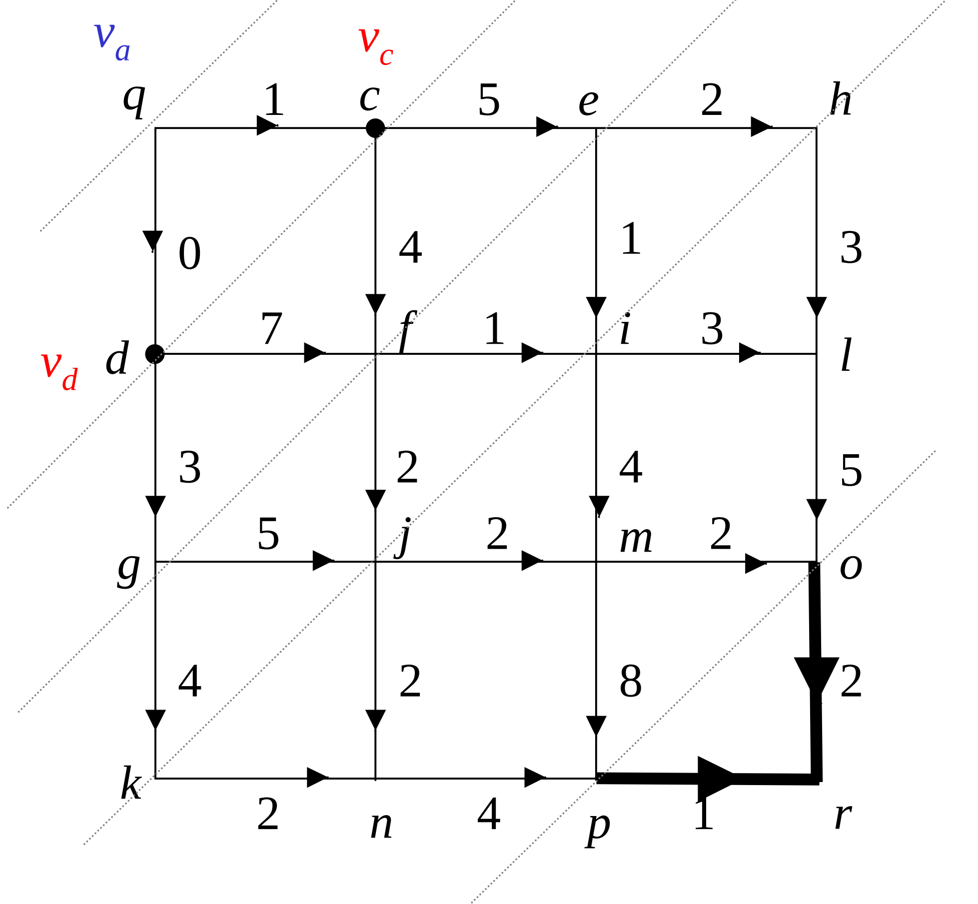
  – programação dinâmica: $O(n|A||S|)$

    • $|S|$ número de estados possíveis

    • n: número de estágios

- Programação dinâmica forward

v_i melhor caminho de q até i

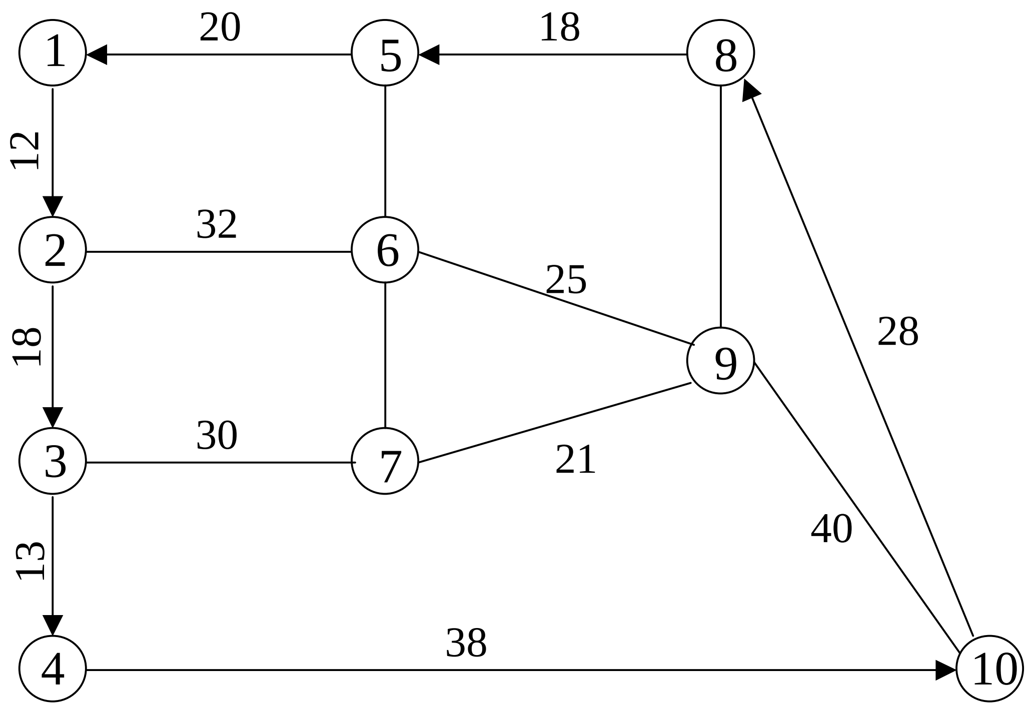


$v_r = \min\{2 + v_o, 1 + v_p\}$
$v_o = \min\{5 + v_l, 2 + v_m\}$
$v_m = \min\{4 + v_i, 2 + v_j\}$
$v_l = \min\{3 + v_h, 3 + v_i\}$
$v_n = \min\{2 + v_j, 2 + v_k\}$
....
$v_e = \min\{5 + v_c\}$
$v_f = \min\{4 + v_c, 7 + v_d\}$
$v_g = \min\{3 + v_d\}$

$v_d = 0$
$v_c = 1$

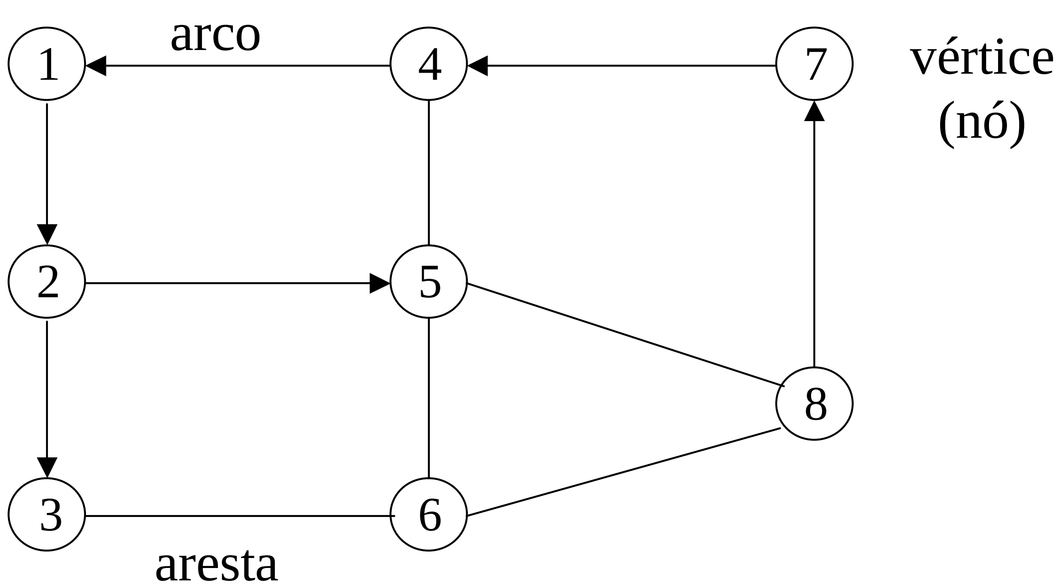
$v_q = 0$

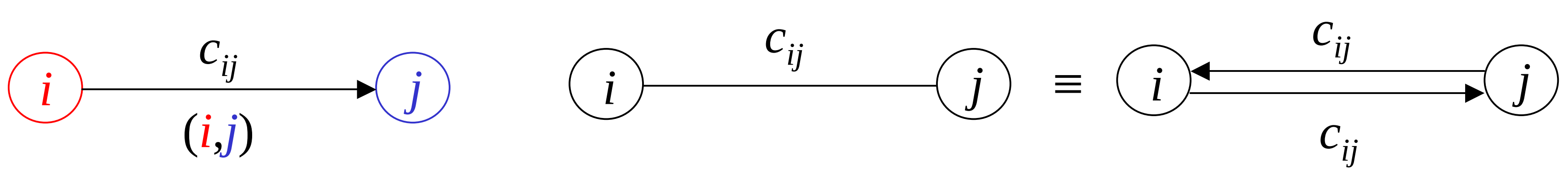
Modelos de caminhos mínimos



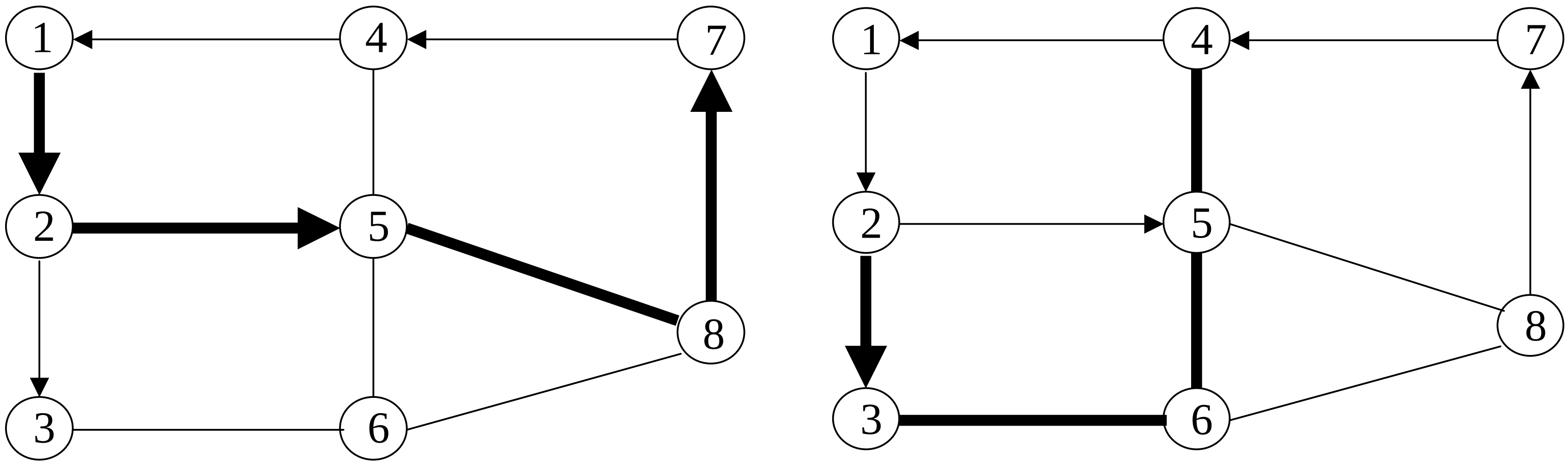
$$s \rightarrow t$$
$$s \rightarrow k, \ \forall k$$
$$\mathrm{k} \rightarrow l, \ \forall k,l$$

Grafos


arco
1
4
7
vértice
(nó)
2
5
8
3
6
aresta

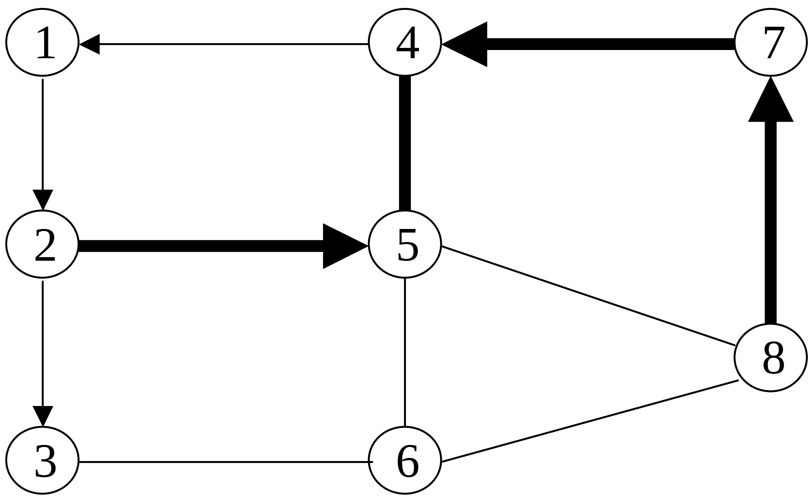

c_{ij}
i
j
(i,j)
c_{ij}
i
j
$\equiv$
c_{ij}
i
j
c_{ij}

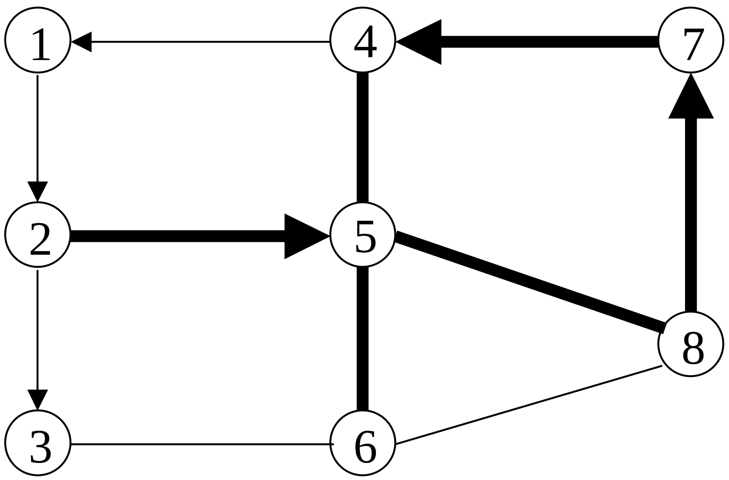
Caminhos



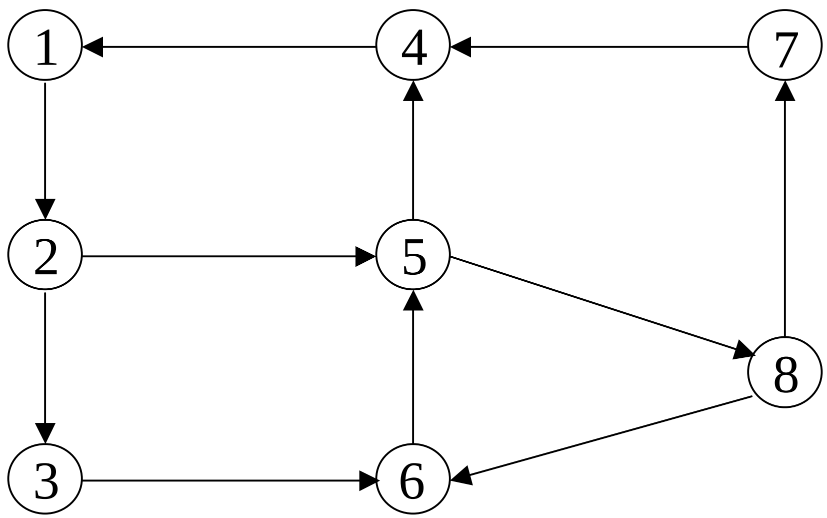
- Caminho mínimo
  - menor caminho entre dois vértices de um grafo

- Não são caminhos

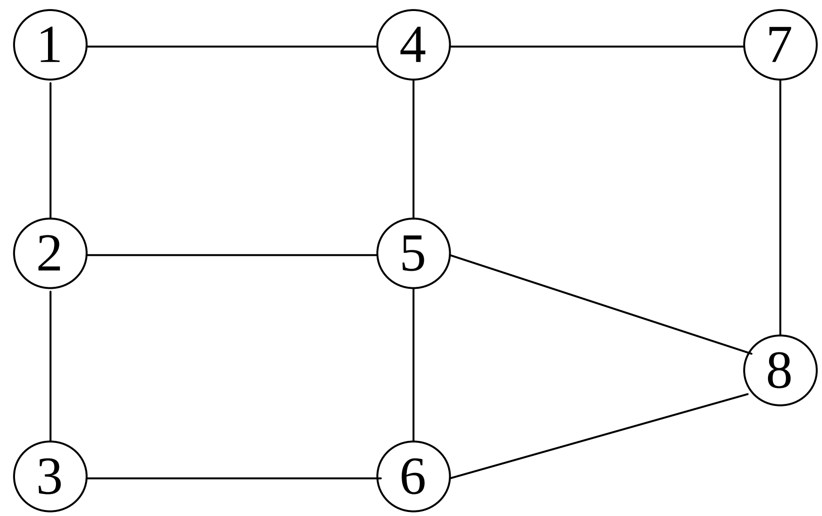


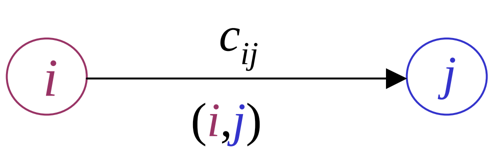


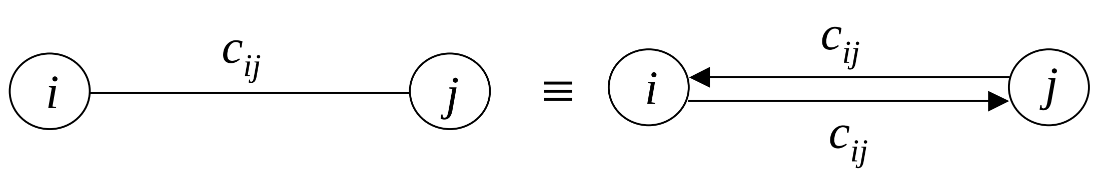
Grafos dirigidos (dígrafos)



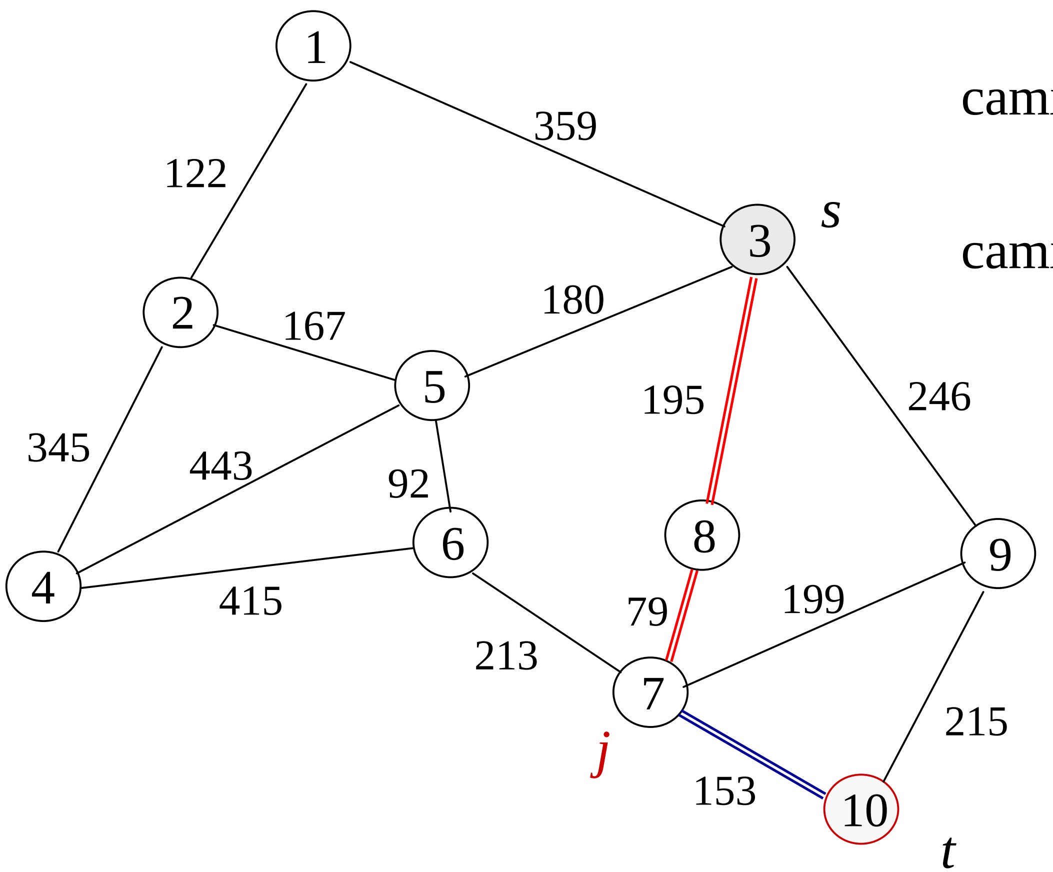
Grafos não dirigidos







Princípio de otimalidade de Bellman



caminho mínimo 3 ⟶ 10: 3–8 – 7 – 10

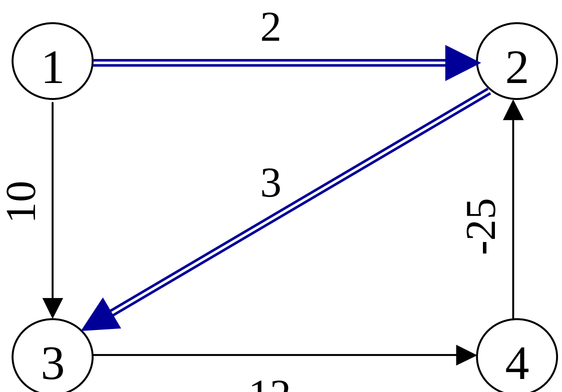
⇓

caminho mínimo 3 ⟶ 7:   3 – 8 – 7

- Ciclo
  - – caminho que inicia e termina no mesmo vértice
  - – ciclo positivo: ciclo com comprimento positivo
  - – ciclo negativo: ciclo com comprimento negativo

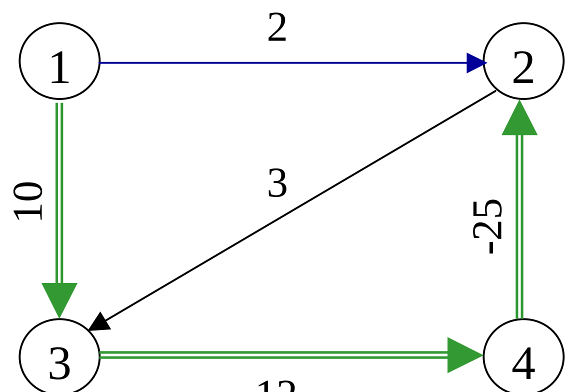
- Ciclos negativos criam problemas para sub-caminhos

ótimo de 1 $\rightarrow$ 3



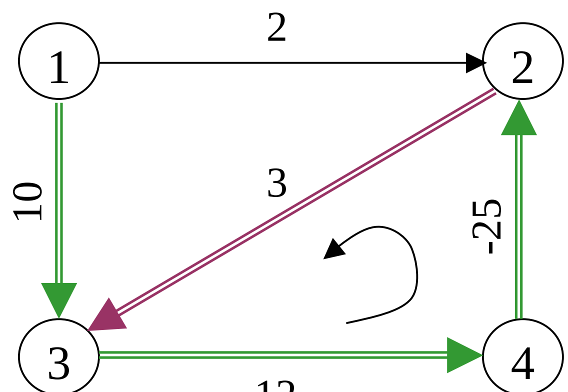
1 – 2 – 3

ótimo de 1 $\rightarrow$ 2



1 – 3 – 4 – 2

extendendo $\rightarrow$ 3
ótimo de 1 $\rightarrow$ 2



1 – 3 – 4 – 2 – 3

- Princípio de otimalidade de Bellman
  - caminhos ótimos em grafos sem ciclos negativos possuem subcaminhos ótimos

- Equações funcionais: $s \rightarrow k, \quad \forall k \neq s$

$v[s] = 0$

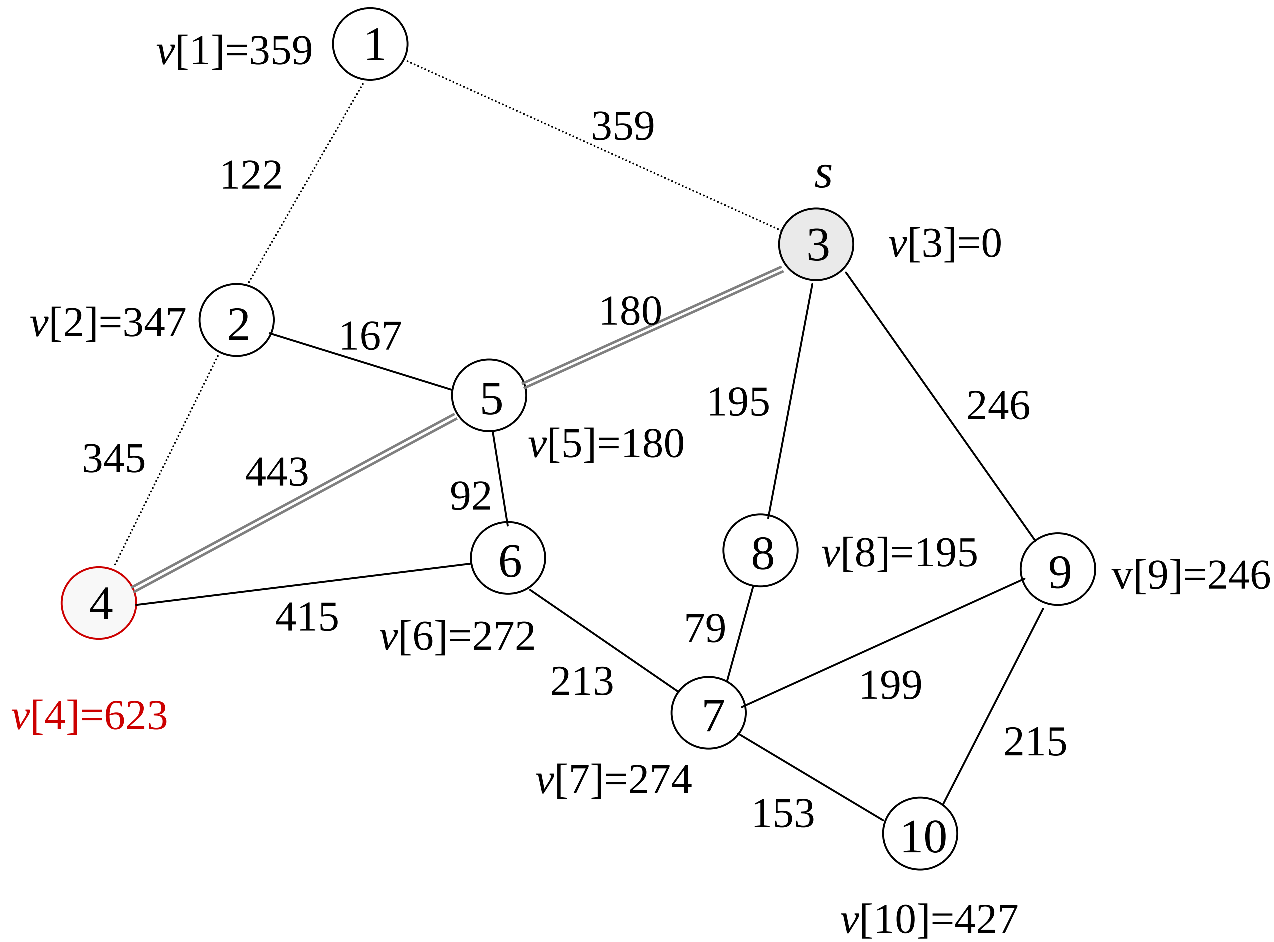
$v[k] = \min \{ v[i] + c_{ik} : (i,k) \text{ existe} \} \; \forall k \neq s$

Se $\nexists$ (i,k) então min { nada } = $+\infty$

Valores $v[k]$ de vértices em um grafo sem ciclos negativos são comprimentos dos caminhos mínimos de um dado vértice s se e somente se eles satisfazem as equações funcionais.

$v[k]$: valor do caminho mínimo entre origem s e o vértice k

$v[k] = +\infty$ se não existe caminho $s \rightarrow k$


v[1]=359
1
359
122
s
3
v[3]=0
v[2]=347
2
167
180
5
195
246
345
443
v[5]=180
92
8
v[8]=195
9
v[9]=246
4
6
415
v[6]=272
79
213
199
7
v[4]=623
215
v[7]=274
153
10
v[10]=427

1 - $v[3] = 0$

2 - $v[4] = \mathrm{v}[i] + c_{i4}$ ($i = 5$ minimiza) $\Rightarrow$ $v[4] = v[5] + c_{54}$

$v[5] = v[j] + c_{j5}$ ($j = 3$ minimiza) $\Rightarrow$ $v[5] = v[\mathrm{s}] + \mathrm{c}_{s5}$

$v[4] + v[5] = v[5] + v[s] + c_{s5} + c_{54}$

$v[4] = c_{s5} + c_{54}$ comprimento caminho 3 – 5 – 4

3 - Nenhum outro caminho 3 $\rightarrow$ 4 tem comprimento menor

considerar, por exemplo, caminho 3 – 1 – 2 – 4

valores satisfazem as equações funcionais

$$v[1] \leq v[s] + c_{s1}$$

$$v[2] \leq v[1] + c_{12}$$

$$v[4] \leq v[2] + c_{24}$$

$$v[1] + v[2] + v[4] \leq v[s] + v[1] + v[2] + c_{s1} + c_{12} + c_{24}$$

$$v[4] \leq v[s] + c_{s1} + c_{12} + c_{24}$$

$$v[4] \leq c_{s1} + c_{12} + c_{24}$$

caminho 3 – 1 – 2 – 4 não pode ter comprimento menor que $v[4]$!

- Equações funcionais: $k \rightarrow l, \quad \forall k$, l

$v[k,k] = 0$

$v[k,l] = \min \{c_{kl}\ , v[k,i] + \mathrm{v}[i,l] : \ i \neq k,\ l\ \} \ \forall k \neq l$

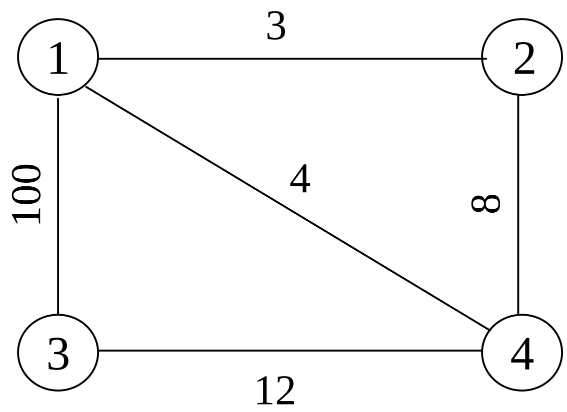
Se $\nexists$ (i,k) então min { nada } = $+\propto$

Valores $v[k,l]$ de vértices em um grafo sem ciclos negativos são comprimentos dos caminhos mínimos entre os vértice k e l se e somente se eles satisfazem as equações funcionais.

$v[k,l]$: valor do caminho mínimo entre vértices k e l

$v[k,l] = +\propto$ se não existe caminho $k \rightarrow l$

Exemplo



Caminhos mínimos

	$j = 1$		$j = 2$		$j = 3$		$j = 4$	
i	v	caminho	v	caminho	v	caminho	v	caminho
1	0	-	3	1-2	16	1-4-3	4	1-4
2	3	2-1	0	-	19	2-1-4-3	7	2-1-4
3	16	3-4-1	19	3-4-1-2	0	-	12	3-4
4	4	1-4	7	4-1-2	12	4-3	0	-

Equações funcionais:

$v[1,4] = \min \{ c_{14}, v[1,2] + v[2,4], v[1,3] + v[3,4] \} = \min \{ 4, 3 + 7, 16 + 12 \} = 4$

$v[2,3] = \min \{ c_{23}, v[2,1] + v[1,3], v[2,4] + v[4,3] \} = \min \{ +\infty, 3 + 16, 7 + 12 \} = 19$

Algoritmo de Bellman-Ford

- Caminho mínimo de um vértice s a todos os outros de um grafo

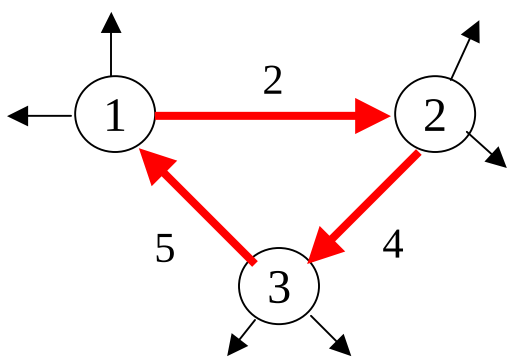
- Equações funcionais: $s \rightarrow k, \quad \forall k \neq s$

$v[s] = 0$

$v[k] = \min \{ v[i] + c_{ik} : (i,k) \text{ existe} \} \; \forall k \neq s$

Se $\nexists$ (i,k) então min { nada } = $+\infty$

- Ciclos introduzem dependências circulares:



$v[1] = \min \{ v[3] + 5, ...\}$

$v[2] = \min \{ v[1] + 2, ...\}$

$v[3] = \min \{ v[2] + 4, ...\}$

Algoritmo de Bellman-Ford

Passo 0 Inicializar: s vértice origem; $v^0[k] = 0$ se $k = s$; senão $v^0[\mathrm{k}] = +\infty$

$t \leftarrow 1$;

Passo 1 Avaliar: calcular $v^t[k] \leftarrow \min \{v^t[i] + c_{ik}: (i,k) \text{ existe}\}\ \forall k \neq s$;

se $v^t[k] < v^{t-1}[\mathrm{k}]$ então $d[k]$ = vértice que precede k no melhor caminho;

Passo 2 Parar: se $v^t[k] = v^{t-1}[k]$ ou t = número de vértices no grafo;

se $v^t[k]$ muda no último t então existe ciclo negativo;

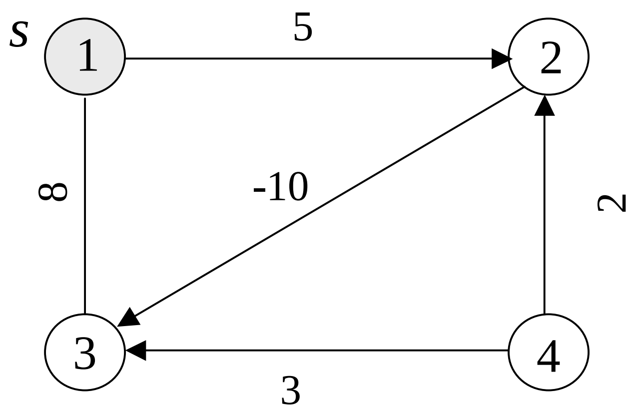
Passo 3 Avançar: se algum $v^t[k]$ mudou e t < número de vértices então

$t \leftarrow t + 1$; ir para o Passo 1;

$v^t[k]$ = valor de $v[k]$ obtido na t-ésima iteração

$d[k]$ = vértice que precede k no melhor caminho $s \rightarrow k$ atual

Exemplo



Inicializar: $v^0[1] = 0$; $v^0[2] = v^0[3] = v^0[4] = +\infty$; $t = 1$

$v^1[1] = 0$

$v^1[2] = \min\{v^0[1] + c_{12}, v^0[4] + c_{42}\} = \min\{0 + 5, \infty + 2\} = 5$; $d[2] = 1$

$v^1[3] = \min\{v^0[1] + \mathrm{c}_{13}, v^0[4] + c_{43}, v^0[2] + c_{23}\} = \min\{8, \infty, \infty\} = 8$; $d[3] = 1$

$v^1[4] = \min\{\ \} = +\infty$

$v^2[2] = \min \{ v^1[1] + c_{12}, v^1[4] + c_{42} \} = \min \{0 + 5, \propto + 2\} = 5;$ $d[2] = 1$

$v^2[3] = \min \{v^1[1] + c_{13}, v^1[4] + c_{43}, v^1[2] + c_{23}\} = \min \{8, \propto, -5\} = -5;$ $d[3] = 2$

$v^2[4] = \min \{ \} = +\propto$

$v^3[2] = \min \{v^2[1] + c_{12}, v^2[4] + c_{42} \} = \min \{0 + 5, \propto + 2 = 5;$ $d[2] = 1$

$v^3[3] = \min \{v^2[1] + c_{13}, v^2[4] + c_{43}, v^2[2] + c_{23}\} = \min \{8, \propto, -5\} = -5;$ $d[3] = 2$

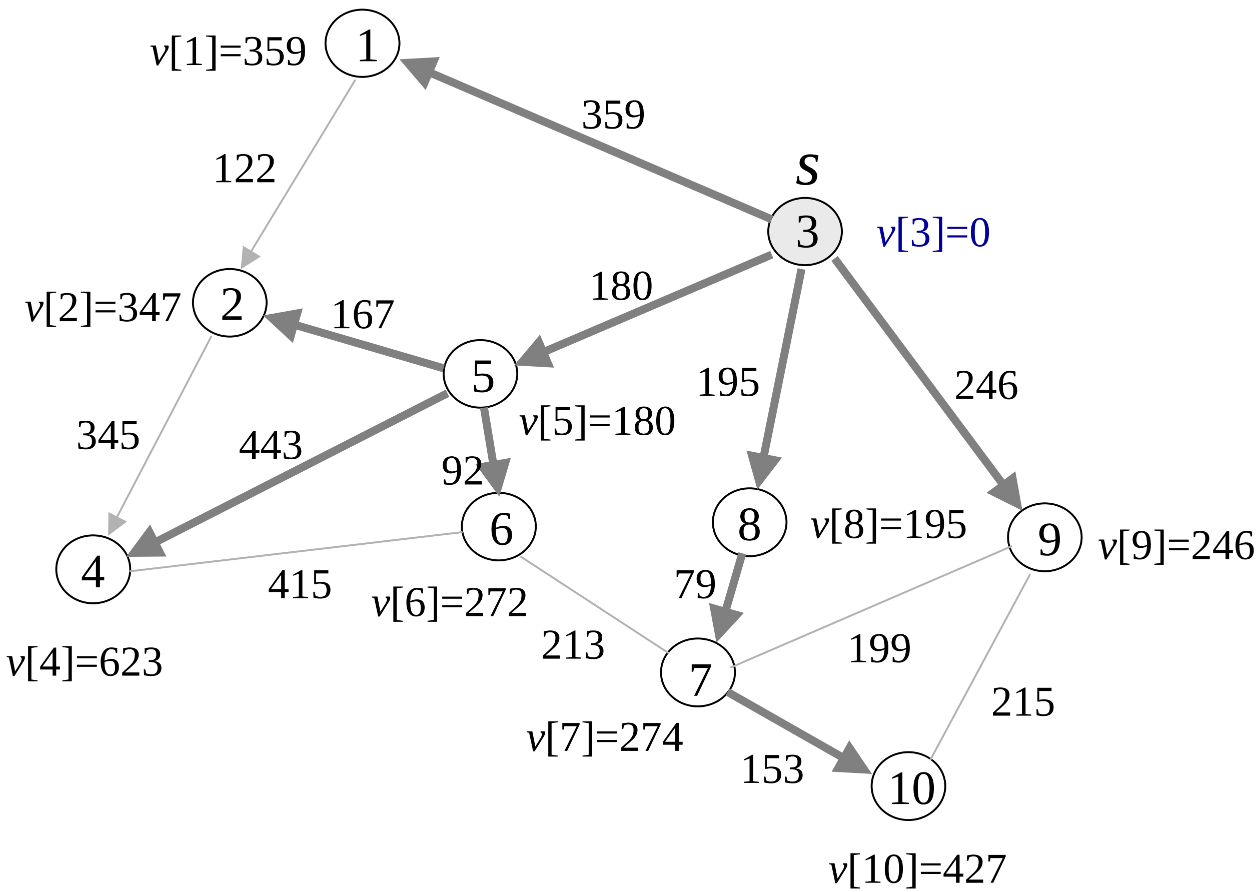
$v^3[4] = \min \{ \} = +\propto$

Exemplo: resumo

t	$v^t[1]$	$v^t[2]$	$v^t[3]$	$v^t[4]$
0	0	+oo	+oo	+oo
1		5	8	
2			-5	
3				

t	$d[1]$	$d[2]$	$d[3]$	$d[4]$
1		1	1	
2			2	
3				

Algoritmo de Bellman-Ford: Justificativa



número arcos caminho < número vértices – 1

Algoritmo de Floyd-Warshall

- Caminho mínimo de cada vértice a todos os outros de um grafo

- Equações funcionais: $k \rightarrow l, \quad \forall k, l$

  $v[k,k] = 0$

  $v[k,l] = \min \{c_{kl}\ ,\ v[k,i] + v[i,l]\!:\ i \neq k,\ l\ \}\ \forall k \neq l$

  Se $\nexists$ (i,k) então min { nada } = $+\infty$

$v^t[k,l]$ = valor do caminho mínimo $k \rightarrow l$ obtido na t-ésima iteração
$d[k,l]$ = vértice que precede l no melhor caminho $k \rightarrow l$ atual

Algoritmo de Floyd-Warshall

Passo 0 Inicializar: numerar vértices consecutivamente a partir de 1;

se existe arco (k,l) então $v^0[\mathrm{k,l}] \leftarrow c_{kl}$; $d[k,l] \leftarrow \mathrm{k}$;

senão $v^0[k,l] = 0$ se $k = l$; $v^0[k,l] = +\propto$ caso contrário;

$t \leftarrow 1$;

Passo 1 Avaliar: $v^t[k,l] \leftarrow \min \{v^{t-1}[k,l], v^{t-1}[k,t] + v^{t-1}[t,l]\}, \ \forall k, l \neq t$;

se $v^t[k,l] < v^{t-1}[k,l]$ então $d[k,l] \leftarrow d[t,l]$;

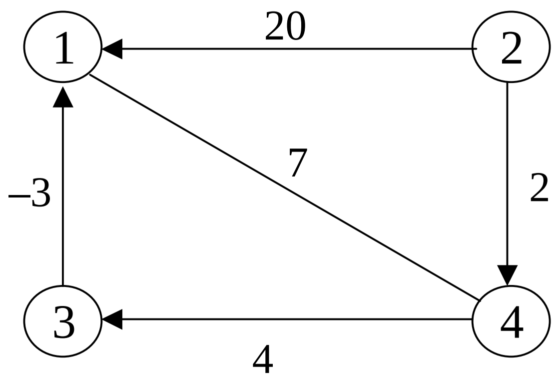
Passo 2 Parar: se $v^t[k,k] < 0$ para algum k, ou t = número de vértices;

se $v^t[k,k] < 0$ para algum k, então existe ciclo negativo;

Passo 3 Avançar: se t < número de vértices e $v^t[k,k] > 0, \forall k$, então

$t \leftarrow \mathrm{t} + 1$; ir para o Passo 1;

Exemplo



		$v_0[k,l]$				$d[k,l]$		
k	$l=1$	$l=2$	$l=3$	$l=4$	$l=1$	$l=2$	$l=3$	$l=4$
1	0	oo	oo	7	-	-	-	1
2	20	0	oo	2	2	-	-	2
3	-3	oo	0	oo	3	-	-	-
4	7	oo	4	0	4	-	4	-

	$v^1[k,l] = v^2[k,l]$					$d[k,l]$		
k	$l=1$	$l=2$	$l=3$	$l=4$	$l=1$	$l=2$	$l=3$	$l=4$
1	0	oo	oo	7	-	-	-	1
2	20	0	oo	2	2	-	-	2
3	-3	oo	0	4	3	-	-	1
4	7	oo	4	0	4	-	4	-

		$v3[k,l]$				$d[k,l]$		
k	$l=1$	$l=2$	$l=3$	$l=4$	$l=1$	$l=2$	$l=3$	$l=4$
1	0	oo	oo	7	-	-	-	1
2	20	0	oo	2	2	-	-	2
3	-3	oo	0	4	3	-	-	1
4	1	oo	4	0	3	-	4	-

		$v4[k,l]$				$d[k,l]$		
k	$l=1$	$l=2$	$l=3$	$l=4$	$l=1$	$l=2$	$l=3$	$l=4$
1	0	oo	11	7	-	-	4	1
2	3	0	6	2	3	-	4	2
3	-3	oo	0	4	3	-	-	1
4	1	oo	4	0	3	-	4	-

Algoritmo de Dijkstra

- Caminho mínimo do vértice *s* a todos os outros com $c_{ij} \geq 0$ $\forall i, j$

Passo 0 Inicializar: *s* vértice origem; $v[i] = 0$ se $i = s$; senão $v[i] = +\infty$

$p \leftarrow s$ e marcar *s* como vértice permanente;

Passo 1 Avaliar: marcar *p* como vértice permanente; $\forall$arco (p,i)

calcular $v[\mathrm{i}] \leftarrow \min\,\{v[\mathrm{i}], v[p] + c_{pi}\}$, *i* temporário;

se o valor de $v[i]$ modifica então $d[i] \leftarrow p$;

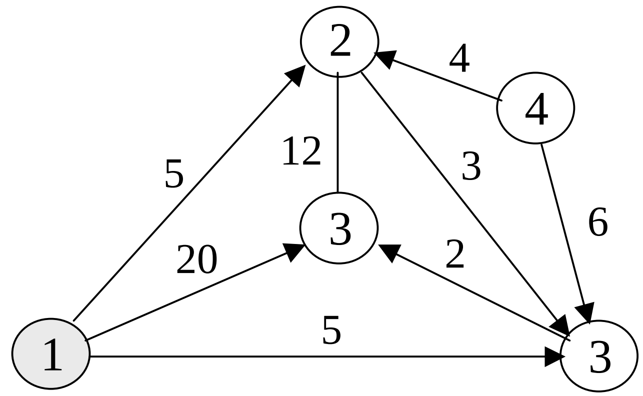
Passo 2 Parar: se não restar nenhum vértice temporário,

$v[i]$ são os valores ótimos;

Passo 3 Permanente: escolher o próximo vértice permanente *p* tal que

$v[p] = \min\,\{v[i] : i \text{ temporário}\}$; ir para o Passo 1;

Exemplo

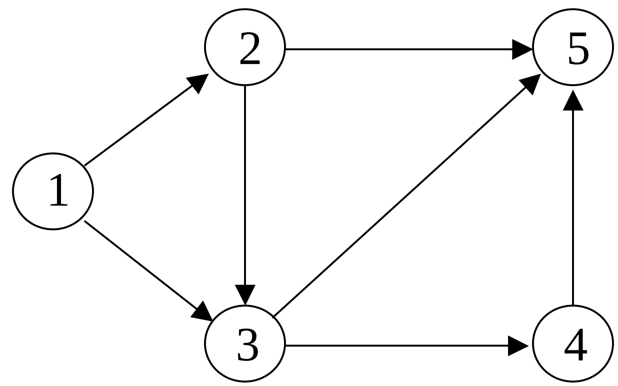


p	*v* [1]	*v* [2]	*v* [3]	*v* [4]	*v* [5]
inicio	0	oo	oo	oo	oo
1	*p*	5	20		5
2		*p*	17		
5			7		*p*
3			*p*		
4				*p*	
fim	0	5	7	oo	5

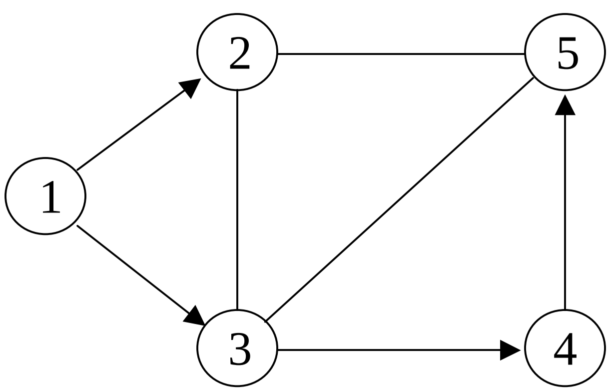
p	*d* [1]	*d* [2]	*d* [3]	*d* [4]	*d* [5]
1		1	1		1
2			2		
5			5		
3					
4					
fim	-	1	5	-	1

Grafos acíclicos

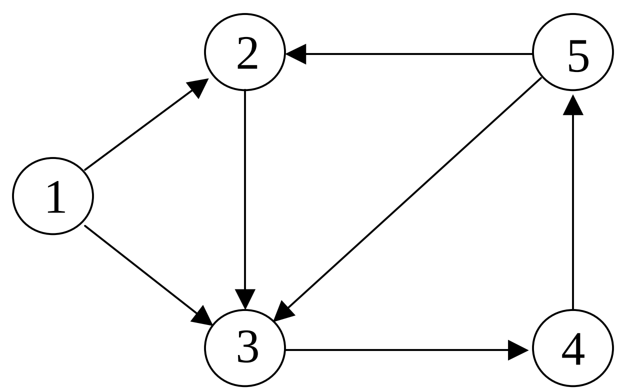
- grafo que é dirigido e sem ciclos
- vértices podem ser numerados tal que todo arco (i, j) tem $i < j$



acíclico



grafo não dirigido



grafo com ciclo

Caminhos mínimos em grafos acíclicos

Passo 0 Inicializar: numerar vértices tal que arcos (i, j) tem $i < j$;

s vértice origem; $v[s] \leftarrow 0$;

Passo 1 Parar: se todos $v[k]$ já foram determinados;

caso contrário, seja p um vértice não processado com a menor numeração;

Passo 3 Processar: se $\nexists$ arcos dirigidos para o vértice p; $v[p] \leftarrow +\propto$;

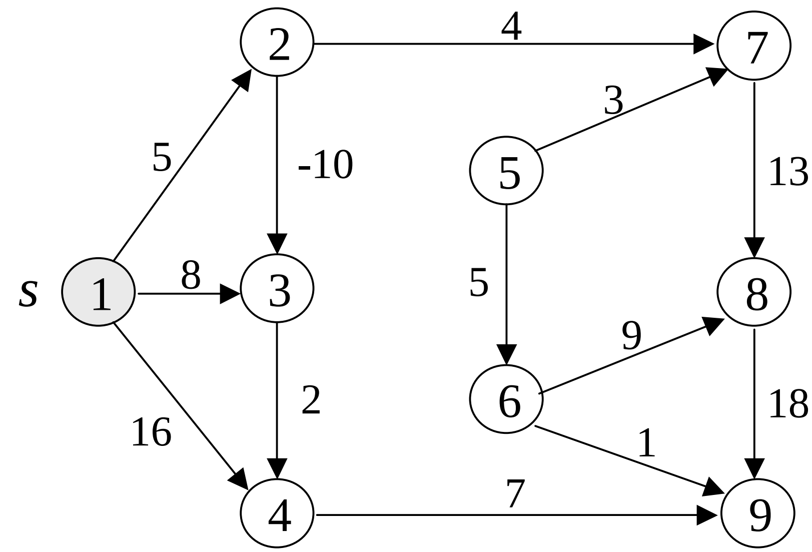
caso contrário, determinar

$v[p] \leftarrow \min \{ v[i] + c_{ip} : (i,p) \text{ existe} \}$

$d[p] \leftarrow$ número do vértice que atinge o mínimo;

ir para o Passo 1;

Exemplo



p	$v\,[p\,]$	$d\,[p\,]$
1	0	-
2	5	1
3	–5	2
4	–3	3
5	∝	-
6	∝	-
7	9	2
8	22	7
9	4	4

s $\quad v[1] = 0$

$p = 2$ $\quad v[2] = \min\,\{v[1] + c_{12}\} = 5$

$d[2] = 1$

$p = 3$ $\quad v[3] = \min\,\{v[1] + c_{13}, v[2] + c_{23}\}$

$= \min\,\{0 + 8, 5 – 10\} = –5$

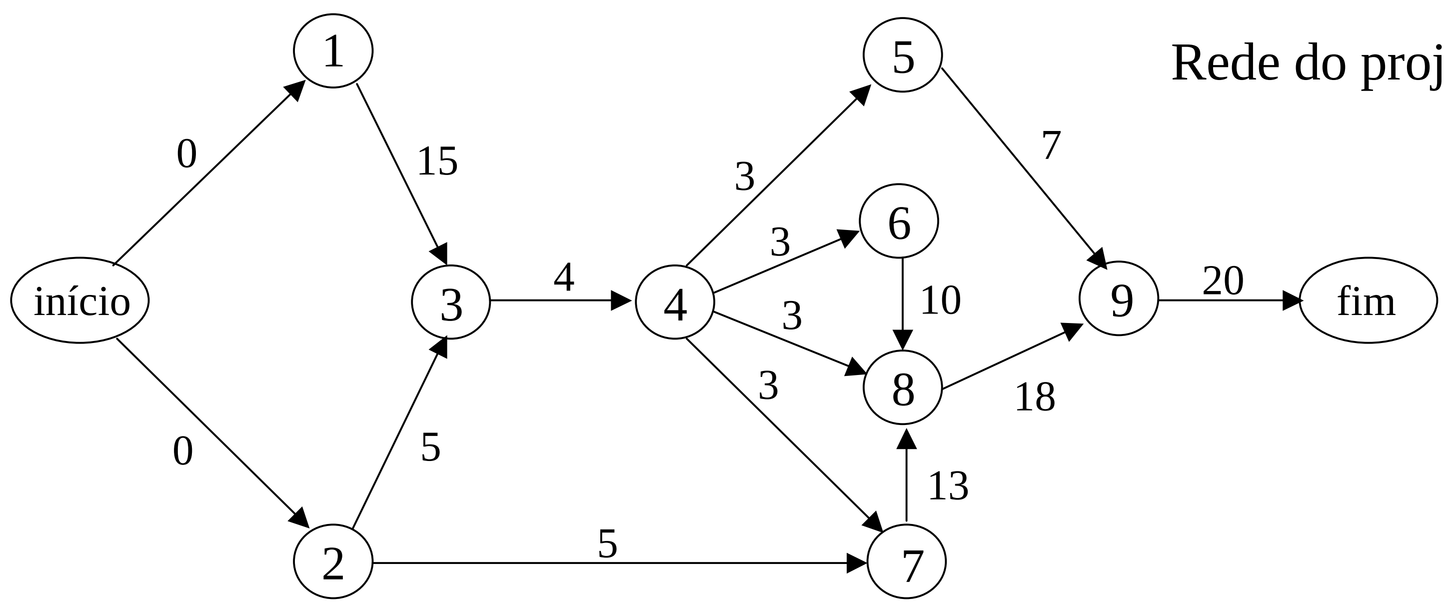
$d[3] = 2$

Aplicação planejamento de projetos: CPM

- Projeto: conjunto de atividades
- Duração de uma atividade k: a_k
- Atividade j precede k se j tem que terminar antes do início de k
- Problema: determinar a data mais cedo para cada atividade, respeitando-se as restrições de precedência entre elas

- Método do caminho crítico: CPM (*Critical Path Method*)
- Projeto pode ser modelado através de um grafo acíclico
- Algoritmos de caminhos ótimos $\rightarrow$ caminhos críticos

k	Atividade	Duração (dias)	Predecessores
1	Fundação	15	-
2	Hidráulica básica	5	-
3	Concretagem vigas	4	1, 2
4	Elementos estruturais	3	3
5	Telhado	7	4
6	Instação elétrica	10	4
7	Condicionamento térmico	13	2, 4
8	Paredes	18	4, 6, 7
9	Acabamento	20	5, 8



Rede do projeto

- Data mais cedo início atividade *k*:
  - – é o comprimento do maior caminho do vértice *início* até o vértice *k* da rede do projeto correspondente

- Tempo mínimo para completar o projeto:
  - – é o comprimento do maior caminho entre os vértices *início* e *fim* da rede do projeto.

k	Atividade	Inicio Mais Cedo	Caminho Crítico
1	Fundação	0	início - 1
2	Hidráulica básica	0	início - 2
3	Concretagem vigas	15	início - 1 - 3
4	Elementos estruturais	19	início - 1 - 3 - 4
5	Telhado	22	início - 1 - 3 - 4 - 5
6	Instação elétrica	22	início - 1 - 3 - 4 - 6
7	Condicionamento térmico	22	início - 1 - 3 - 4 - 7
8	Paredes	35	início - 1 - 3 - 4 - 7 - 8
9	Acabamento	53	início - 1 - 3 - 4 - 7 - 8 - 9

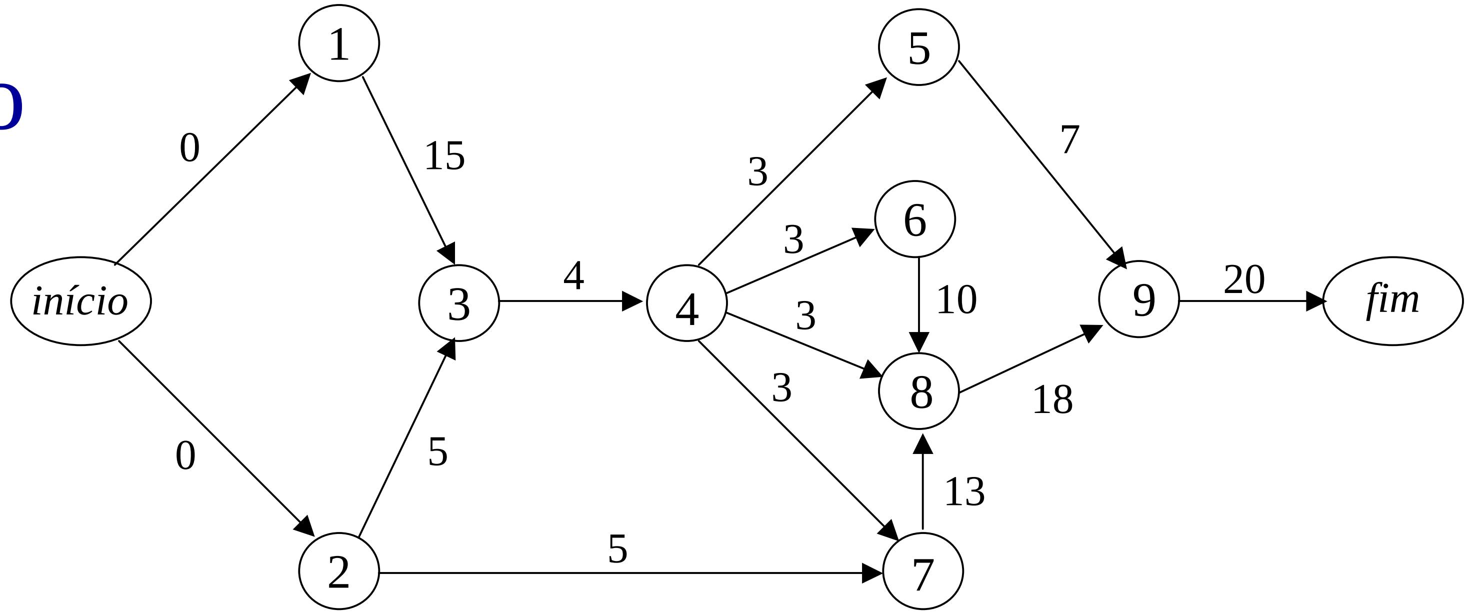
Algoritmo de sequenciamento: CPM

Passo 0 Inicializa: numerar vértices atividade com cada arco precedência (i, j) com $i < j$; $v[início] \leftarrow 0$;

Passo 1 Pára: se o valor de $v[final]$ já foi determinado;
caso contrário, seja p um vértice não processado com a menor numeração;

Passo 3 Processa: determinar
$v[p] \leftarrow \max \{ v[i] + a_i : i$ precede p ;
$d[p] \leftarrow$ número do vértice que atinge o máximo;
ir para o Passo 1;

Exemplo



p	$v[p]$	$d[p]$
início		-
1	0	*início*
2	0	*início*
3	15	1
4	19	3
5	22	4
6	22	4
7	22	4
8	35	7
9	53	8
fim	73	9

Observação

Este material refere-se às notas de aula do curso CT 820 Teoria de Sistemas e Otimização Fuzzy: Introdução e Aplicações da Faculdade de Engenharia Elétrica e de Computação da Unicamp e do Centro Federal de Educação Tecnológica do Estado de Minas Gerais. Não substitui o livro texto, as referências recomendadas e nem as aulas expositivas. Este material não pode ser reproduzido sem autorização prévia dos autores. Quando autorizado, seu uso é exclusivo para atividades de ensino e pesquisa em instituições sem fins lucrativos.