

IA881 – Otimização Linear

Aula: Caminho Mínimo (*shortest path*)

Ricardo C. L. F. Oliveira

Faculdade de Engenharia Elétrica e de Computação
Universidade Estadual de Campinas

1º Semestre 2019

Tópicos

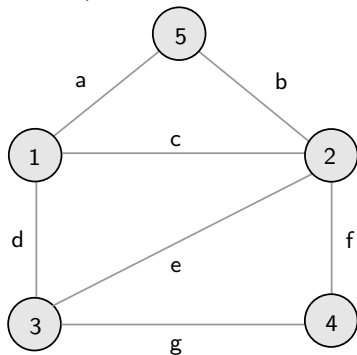
- 1 Conceitos, Definições, Notações
- 2 Caminhos mínimos

Conceitos, Definições, Notações I

Definição 1

Sejam N um conjunto de **vértices** e A um conjunto de **arestas** ligando os vértices $v \in N$. Define-se grafos como sendo $G(N, A)$. $n = |N|$ representa o número (cardinalidade) de vértices e $m = |A|$ o número (cardinalidade) de arestas.

- Observação: vértice = nó; aresta = ramo (não-orientado) ou arco (orientado)



- Nós = $\{1, 2, 3, 4, 5\}$
- Arestas = $\{a, b, c, d, e, f, g\}$

Figura 1: Grafo não orientado.

Conceitos, Definições, Notações II

- Uma alternativa para representar uma aresta é por meio da notação (x, y) com $x, y \in N$. Por exemplo, no grafo da Figura 1, a aresta b poderia ser representada por $(2, 5)$ ou $(5, 2)$. Caso a aresta seja direcionada (arco), convencionou-se que a primeira componente seja o vértice de origem e a segunda o vértice de destino.

Definição 2

Grafo **orientado** ou **direcionado** (não orientado ou não direcionado) – quando as arestas têm (não têm) orientação.

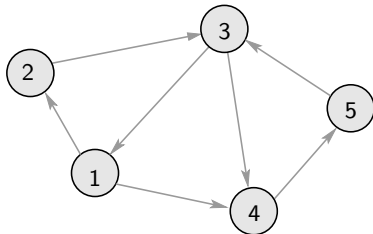


Figura 2: Exemplo de um grafo orientado.

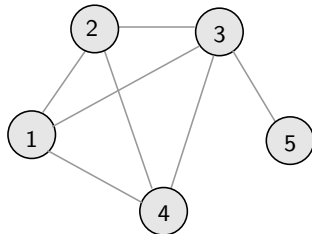


Figura 3: Exemplo de um grafo não orientado.

Conceitos, Definições, Notações III

Definição 3

Um grafo é dito **ponderado** (ou valorado) se suas arestas possuem custos (ou pesos) associados. Usa-se a notação c_{ij} (ou $c(i,j)$) para denotar o custo da aresta entre os vértices i e j .

Definição 4

$G_S(N_S, A_S)$ é um **sub-grafo** de $G(N, A)$ se $N_S \subseteq N$ e $A_S \subseteq A$ tal que se $(i, j) \in A_S \Rightarrow i, j \in N_S$. Um grafo $G_S(N_S, A_S)$ é um **sub-grafo gerador** de $G(N, A)$ se $N_S = N$ e $A_S \subseteq A$.

Conceitos, Definições, Notações IV

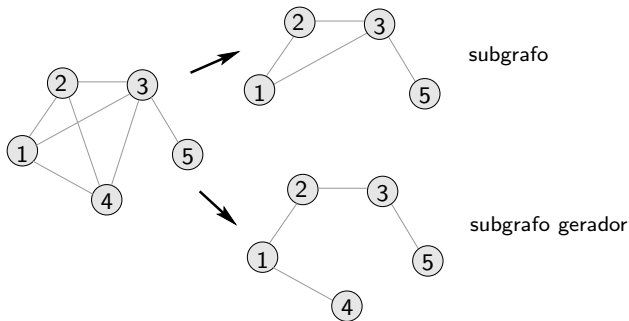


Figura 4: Exemplo de subgrafos (gerador e não gerador).

Conceitos, Definições, Notações V

Definição 5

Grau de um vértice é o número de arestas que incidem nele (no caso orientado, arcos que entram mais que saem).

Definição 6

Cadeia é uma sequência consecutiva de arestas em que todos os nós visitados são distintos. Exemplo: na Figura 2 – $\{(2,3)(5,3)(4,5)(1,4)\}$.

Definição 7

Caminho é um caso particular de cadeia na qual os arcos têm os mesmos sentidos. Exemplo Figura 2 – $\{(2,3)(3,1)(1,4)(4,5)\}$

Definição 8

Comprimento de um caminho é a soma dos pesos (ou custos) das arestas do caminho.

Conceitos, Definições, Notações VI

Definição 9

Um grafo é dito ser *conexo* se sempre existe uma cadeia entre qualquer par de vértices.

Definição 10

Ciclo ou *laço* é uma cadeia fechada (termina no nó que iniciou). Exemplo na Figura 2 – $\{(3,1)(1,4)(3,4)\}$

Definição 11

Circuito (ciclo direcionado) é um caminho fechado. Exemplo na Figura 2 – $\{(2,3)(3,1)(1,2)\}$

Definição 12

Uma *árvore* é um grafo conexo que não contém ciclos.

Conceitos, Definições, Notações VII

- Exemplos de árvore obtidas a partir do grafo da Figura 2: (1) removendo-se as arestas $(2,3)$, $(1,4)$ e $(5,3)$; (2) removendo-se as arestas $(1,4)$, $(3,1)$ e $(3,4)$; Existem outras possibilidades.

Motivação

- Qual é o menor caminho entre o Terminal Central de Campinas e a FEEC?

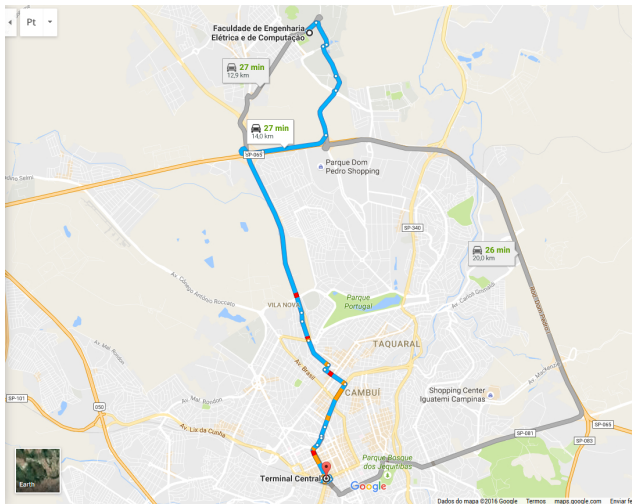


Figura 5: Fonte: Google maps.

Formulação matemática

- Se consideramos as ruas como **arestas** e os cruzamentos das ruas como **vértices**, podemos formular o problema de encontrar o caminho mínimo (em inglês — *shortest path*) usando a teoria de grafos.

Problema

Dado um grafo $G(N,A)$, orientado e com peso nas arestas, encontrar o **caminho mínimo** (de menor comprimento) entre os vértices $s \in N$ (origem) e $d \in N$ (destino).

- Intuitivamente as arestas denotam distâncias, mas outras variáveis poderiam ser consideradas, por exemplo, tempo.
- Inúmeras aplicações: Rotamento de veículos, planejamento de tráfego urbano, navegação robótica, roteamento em telecomunicações, e muitas outras.

Hipóteses e Propriedades I

- Assume-se que o grafo $G(N, A)$ não possui ciclos com comprimento negativo ("ciclo negativo"). Caso contrário não é possível determinar o caminho mínimo pelos algoritmos apresentados.

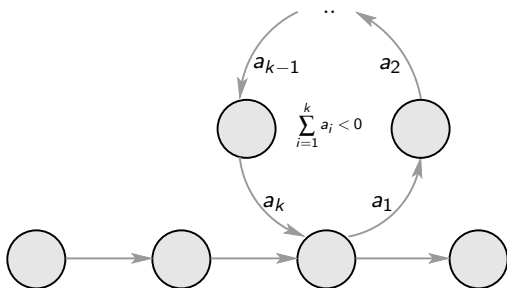


Figura 6: Grafo com ciclo negativo.

Hipóteses e Propriedades II

- O caminho mínimo entre dois vértices pode não ser único. Nesse caso qualquer um deles servirá como solução.
- Também usaremos a seguinte hipótese simplificadora: existe um caminho mínimo entre os vértices s e d .
- Tipos de problemas de caminho mínimo comumente investigados
 - Encontrar o caminho mínimo entre um nó origem (*single-source*) e todos os outros (com ou sem arcos de pesos negativos).
 - Encontrar o caminho mínimo entre todos os nós e um nó destino (*single-sink*).
 - Entre todos os pares de vértices.
- Essas generalizações possuem algoritmos eficientes (complexidade polinomial). Trabalharemos com o *single-source*.

Objetivo e estrutura de dados I

Definição 13

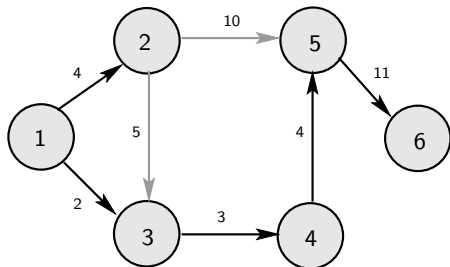
Dado um grafo $G(N,A)$ orientado com pesos nas arestas e um vértice de origem s , uma *árvore de caminhos mínimos* (em inglês shortest-paths tree) é um subgrafo contendo s e todos os vértices alcançáveis a partir de s que forma uma árvore direcionada com raiz em s tal que todo caminho da árvore é um caminho mínimo no grafo.

Objetivo

Encontrar o caminho mínimo entre o vértice de origem s e todos os outros vértices, produzindo como solução uma árvore de caminhos mínimos.

- A árvore de caminhos mínimos pode ser representada por dois vetores indexados pelos vértices v
 - $\text{dist}[v]$: armazena o comprimento do caminho mínimo entre s e v .
 - $\text{prev}[v]$: armazena a última aresta do caminho mínimo entre s e v .

Objetivo e estrutura de dados II



v	prev[v]	dist[v]
1	—	0
2	(1,2)	4
3	(1,3)	2
4	(3,4)	5
5	(4,5)	9
6	(5,6)	20

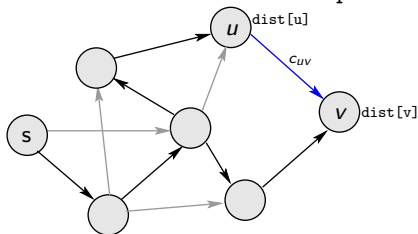
Figura 7: Árvore de caminhos mínimos (arestas pretas).

Conceito de relaxação

Algorithm 1 Relaxação

- 1: Seja um arco (u, v) e seu custo associado c_{uv}
 - 2: **se** $\text{dist}[v] > \text{dist}[u] + c_{uv}$ **então**
 - 3: $\text{dist}[v] \leftarrow \text{dist}[u] + c_{uv}$
 - 4: $\text{prev}[v] \leftarrow (u, v)$
 - 5: **fim se**
-

- Se a aresta (u, v) fornece um caminho para o vértice v via o vértice u , atualiza-se os vetores $\text{dist}[v]$ e $\text{prev}[v]$.



- As arestas em preto representam o estado atual de $\text{prev}[\cdot]$

Condições de otimalidade I

Teorema 1

Seja $G(N, A)$ um grafo orientado e ponderado. O vetor $\text{dist}[u]$ fornece as distâncias dos caminhos mínimos entre $u \in N$ e o vértice de origem *se e somente se* as seguintes condições forem satisfeitas

- $\text{dist}[s]=0$.
- Para cada vértice u , $\text{dist}[u]$ é o comprimento de um caminho de s até u .
- Para cada arco (v, w) com custo c_{vw} , tem-se que $\text{dist}[w] \leq \text{dist}[v] + c_{vw}$.

■ Prova: (Necessidade): Suponha que para um vértice v , associado a uma aresta $u(v, w)$, temos $\text{dist}[w] > \text{dist}[v] + c_{vw}$. Então essa aresta forneceria um caminho da origem s para w com um comprimento menor, o que é uma contradição à otimalidade do caminho.

■ (Suficiência): Suponha que o caminho mínimo entre s e w passe pela seguinte seqüência de vértices: $s = v_0 \rightarrow v_1 \rightarrow v_2 \rightarrow \dots \rightarrow v_k = w$, com custo ótimo dado

Condições de otimalidade II

por $c^* = c_{01} + c_{12} + \dots + c_{k-1k}$. Aplicando as condições de otimalidade em todos os vértices do caminho, tem-se

$$\begin{aligned} dist[v_k] &\leq dist[v_{k-1}] + c_{k-1k} \\ dist[v_{k-1}] &\leq dist[v_{k-2}] + c_{k-2k-1} \\ &\vdots \leq \vdots \\ dist[v_2] &\leq dist[v_1] + c_{12} \\ dist[v_1] &\leq dist[v_0] + c_{01} \end{aligned}$$

Somando as desigualdades e substituindo $dist[v_0] = 0$, tem-se

$$dist[v_k] = dist[w] \leq c_{01} + c_{12} + \dots + c_{k-1k} = c^*$$

Como $dist[w]$ não pode ser menor que o valor ótimo c^* , a restrição é atendida na igualdade.

Algoritmo genérico

Algorithm 2 Algoritmo genérico para caminho mínimo.

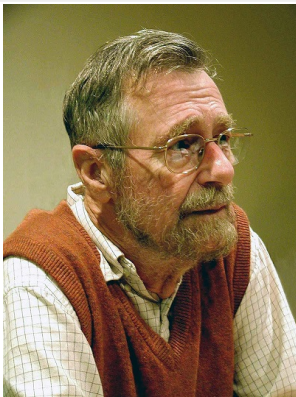
- 1: Inicialize $\text{dist}[s]=0$ e $\text{dist}[v]=\infty$ para os outros vértices
 - 2: **enquanto** as condições de otimalidade não forem satisfeitas **faça**
 - 3: Relaxe alguma aresta
 - 4: **fim enquanto**
-

Proposição

O algoritmo genérico determina a árvore de caminhos mínimos a partir de s .

- Elementos da prova: O vetor $\text{dist}[v]$ sempre armazena o comprimento de um caminho (simples) de s até v ; Uma relaxação pode apenas diminuir o valor de $\text{dist}[v]$; O número de diminuições em $\text{dist}[v]$ é finito (uma para cada possibilidade de caminho entre s e v).
- O algoritmo genérico não especifica a ordem na qual as arestas são relaxadas.

Algoritmo de Dijkstra I



Edsger Wybe Dijkstra.

- Em 1959 Dijkstra (1930–2002) sugeriu um algoritmo de rotulação para caminhos em grafos com **arcos não negativos**, utilizando indução e ajuste, eficiente e de fácil implementação computacional.
- Grafo deve ser conexo.
- Funciona em grafos direcionados e não direcionados.
- Complexidade: $\mathcal{O}(n^2)$
(implementações mais eficientes: $\mathcal{O}(m \log n)$)

Algoritmo de Dijkstra II

- “Assim como Prim está para a árvore geradora mínima, Dijkstra está para árvore de caminhos mínimos”.

Algorithm 3 Algoritmo de Dijkstra.

- 1: Inicialize $\text{dist}[s]=0$ e $\text{dist}[v]=\infty$ para os outros vértices
 - 2: **enquanto** a árvore não estar completa **faça**
 - 3: Insira na árvore o vértice v com menor $\text{dist}[v]$
 - 4: Relaxe os arcos que saem de v
 - 5: **fim enquanto**
-

- Sobre o critério “a árvore não estar completa”, podemos considerar: (1) o número de vértices na árvore não for n ; (2) todos os vértices fora da árvore não terem valor finito em $\text{dist}[v]$.

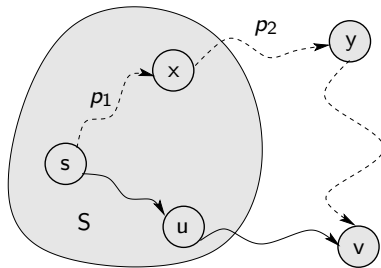
Proposição

O algoritmo de Dijkstra determina a árvore de caminhos mínimos a partir de s para qualquer grafo ponderado com pesos **não negativos** nas arestas.

Algoritmo de Dijkstra III

- Prova: Cada aresta (v, w) do grafo é relaxada apenas uma vez (quando o vértice v está sendo relaxado), deixando $\text{dist}[w] \leq \text{dist}[v] + c_{vw}$. A desigualdade se mantém até o término da execução por dois motivos: (1) $\text{dist}[w]$ não pode aumentar (monotonicamente decrescente); (2) $\text{dist}[v]$ não vai mudar pois, a cada passo da execução, escolhe-se o $\text{dist}[v]$ de menor valor para ser relaxado e os c_{vw} são não negativos. Como conclusão, após o término da execução, as condições de otimalidade são satisfeitas.
- Outra interpretação:

Algoritmo de Dijkstra IV



Exemplos I

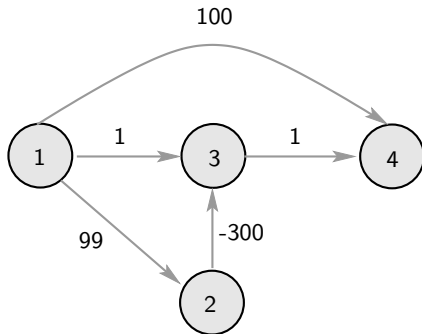


Figura 8: Grafo exemplo para o algoritmo de Dijkstra (vai falhar).

Exemplos II

■ Adiantaria somar uma constante positiva a todas as arestas (de modo que todas fiquem não negativas)?

Não, o caminho mínimo poderia mudar.

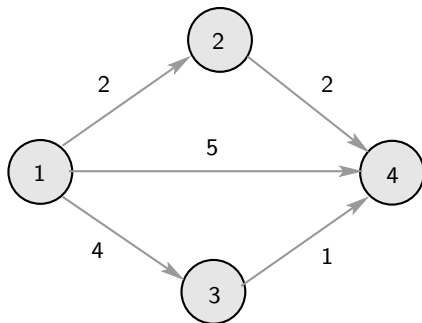


Figura 9: Grafo exemplo para o algoritmo de Dijkstra.

Exemplos III

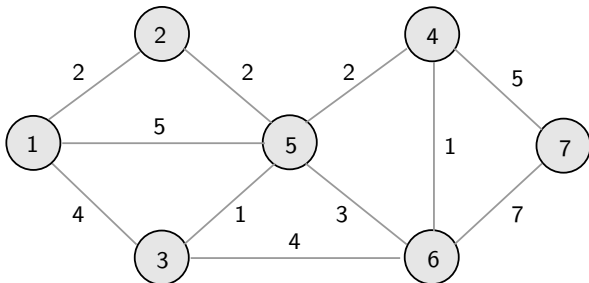


Figura 10: Grafo não direcionado para o algoritmo de Dijkstra.

Algoritmo de Bellman-Ford



Lester Randolph Ford Jr.



Richard Ernest Bellman

- Publicado em 1956 por Ford, em 1958 por Bellman e em 1957 por Edward F. Moore. Também conhecido como algoritmo de Bellman-Ford-Moore.
- Menos eficiente do que Dijkstra, mas trata arestas com pesos **negativos**. É capaz de detectar ciclos negativos.
- Complexidade $\mathcal{O}(nm)$

Exemplo motivador

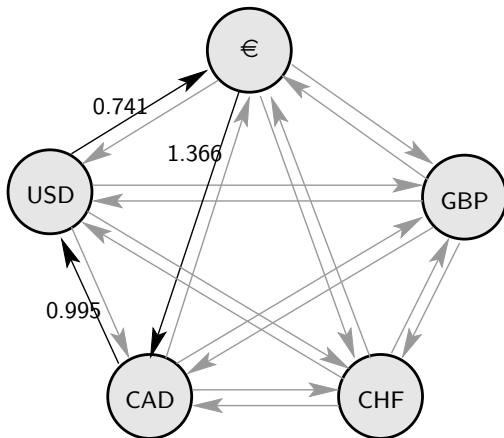
- Dada as moedas e as taxas de câmbio, qual é o melhor caminho para converter mil dólares americanos em dólares canadenses? Opção 1: $1000 \text{ USD} \rightarrow 1005 \text{ CAD}$. Opção dois: $1000 \text{ USD} \rightarrow 741 \text{ €} \rightarrow 1012.21 \text{ CAN}$ (mais vantajosa).

Moeda	USD	€	GBP	CHF	CAD
USD	1	0.741	0.657	1.061	1.005
€	1.349	1	0.888	1.433	1.366
GBP	1.521	1.126	1	1.614	1.538
CHF	0.942	0.698	0.619	1	0.953
CAD	0.995	0.732	0.650	1.049	1

- Problema interessante: Arbitragem financeira:
 $1000 \text{ USD} \rightarrow 741 \text{ €} \rightarrow 1012.21 \text{ CAN} \rightarrow 1007.14 \text{ USD}$. **Lucro de 7.14 dólares!**

Modelagem via grafos

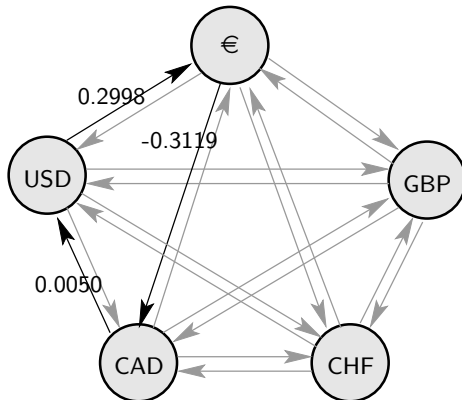
- Vértices: moedas; Arestas: transação de câmbio (peso igual à taxa de câmbio)



- Como modelar como um problema de detecção de ciclos negativos?

Modelagem via grafos

- Estratgia: Tomar o logaritmo do pesos das arestas e trocar o sinal. Assim a multiplicação de pesos transforma-se em adição e valores maiores que um tornam-se menores que zero.



- Problema de arbitragem financeira: encontrar ciclos direcionados (circuitos)

Algorithm 4 Algoritmo de Ford-Moore-Bellman.

- 1: Inicialize $\text{dist}[s]=0$ e $\text{dist}[v]=\infty$ para os outros vértices
 - 2: **para** $i = 1$ até n **faça**
 - 3: Relaxe todos os arcos
 - 4: **fim para**
-

Proposição

O algoritmo Ford-Moore-Bellman determina a árvore de caminhos mínimos a partir de s para grafos livres de ciclos negativos.

- Ideia da prova: Após a i -ésima iteração, os caminhos mínimos de comprimento i (ou menores) já estão determinados.
- Observação: Se o valor de $\text{dist}[v]$ não mudar durante a iteração i , então não é necessário relaxar nenhuma aresta partindo de v na iteração $i + 1$.

Exemplo I

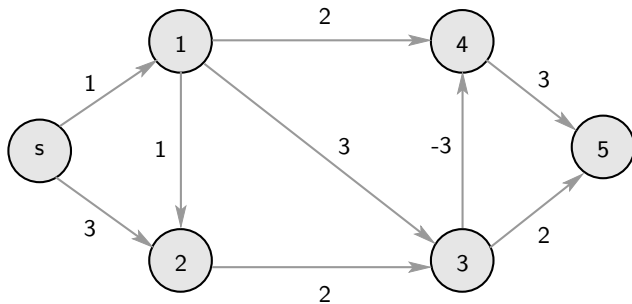


Figura 11: Grafo exemplo para aplicar Bellman-Ford.

Exemplo II

■ Ordem na qual as arestas são relaxadas: $(1,4)$, $(s,1)$, $(4,5)$, $(3,4)$, $(3,5)$, $(2,3)$, $(1,3)$, $(1,2)$, $(s,2)$.

■ Resultado (d =dist, p =prev):





it	0		1		2		3		4	
nó	d	p	d	p	d	p	d	p	d	p
s	0	-	0	-	0	-	0	-	0	-
1	∞	?	1	s	1	s	1	s	1	s
2	∞	?	2	1	2	1	2	1	2	1
3	∞	?	4	1	4	1	4	1	4	1
4	∞	?	∞	?	1	3	1	3	1	3
5	∞	?	∞	?	6	4	4	4	4	4

■ Observação: a solução não é única (pode-se trocar o arco $(1,3)$ pelo $(2,3)$ e obter uma outra árvore de caminhos mínimos com os mesmos custos).

Detecção de ciclos negativos

- Se existir um ciclo negativo no grafo, o algoritmo de Bellman-Ford-Moore entrará em um loop infinito, atualizando sequencialmente $dist[v]$ para todos os vértices v pertencentes ao ciclo negativo.
- Proposta de detecção de ciclo negativo: se $dist[v]$ para algum vértice v é atualizado na última iteração (n), então existe um ciclo negativo. O vetor $prev[v]$ pode ser utilizado para encontrá-lo.

Referências I

-  R. K. Ahuja, T. L. Magnanti, and J. B. Orlin.
Network flows: theory, algorithms, and applications.
Prentice-Hall, Upper Saddle River, NJ, 1993.
-  P. Feofiloff.
Algoritmos para grafos em C via sedgewick.
http://www.ime.usp.br/~pf/algoritmos_para_grafos/index.html.
Acessado: Setembro de 2016.
-  M. C. Goldberg and H. P. L. Luna.
Otimização combinatória e programação linear – Modelos e Algoritmos.
Elsevier, Rio de Janeiro, RJ, 2 edition, 2005.
-  R. Sedgewick and K. Wayne.
Algorithms.
Pearson Education, Boston, MA, 4 edition, 2011.