MAC0325+MAC5781

Otimização Combinatória

Otimização Combinatória

"Highways, telephone lines, electric power systems, computer chips, water delivery systems, and rail lines: these physical networks, and many others, are familiar to all of us. In each of these problem settings, we often wish to send some good(s) (vehicles, messages, electricity, or water) from one point to another, typicaly as efficiently as possible — that is, along a shortest route or via some minimum cost flow pattern."

Ahuja, Magnanti, Orlin, and Reddy

Applications of network optimization

Administração

- Página da disciplina: aulas, cadastro, fórum, . . .

  `http://latin.ime.usp.br/moodle/`

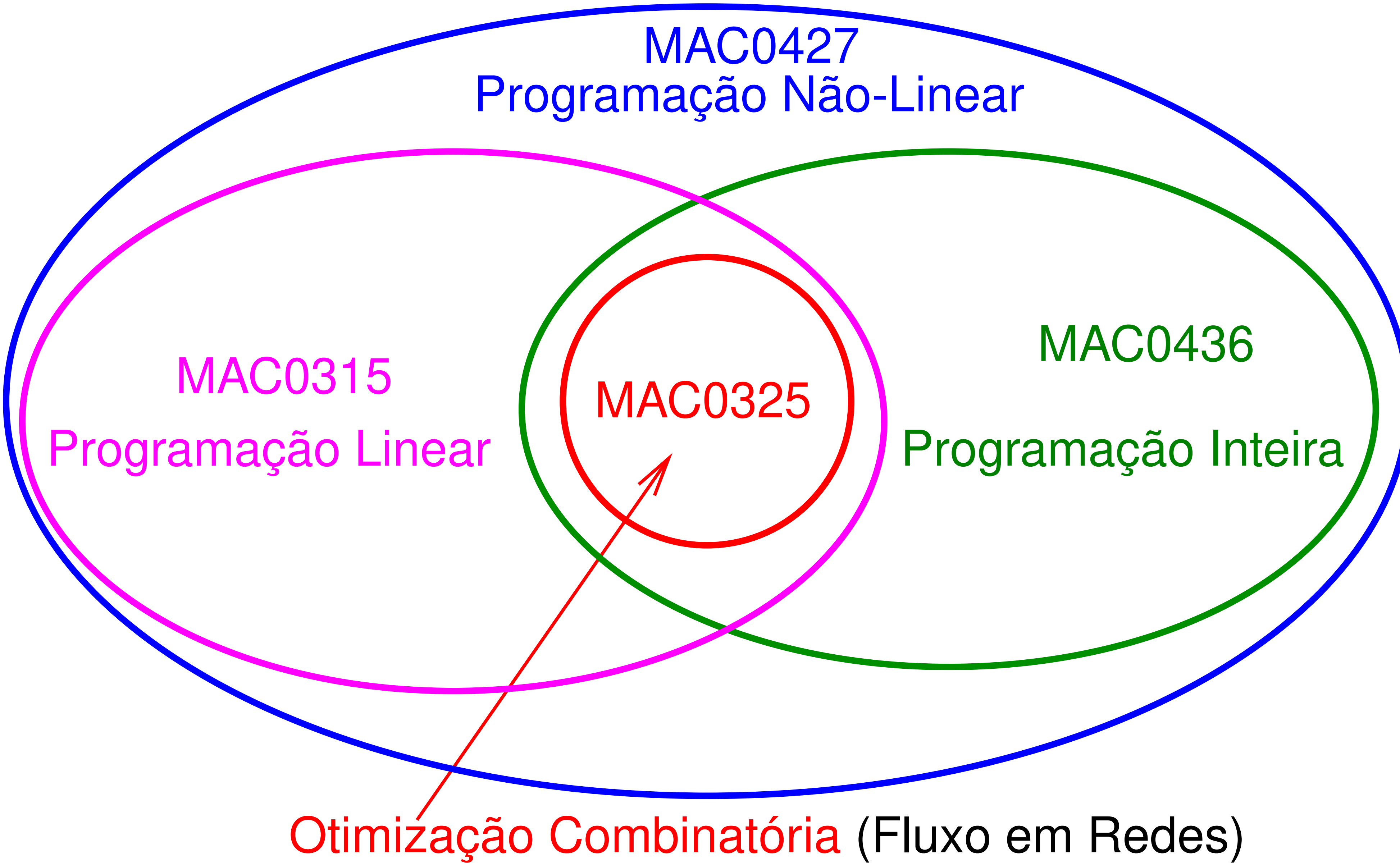
- Livros:
  - PF = Paulo Feofiloff, *Fluxo em Redes*
    `www.ime.usp.br/~pf/`
  - AMO = Ahuja-Magnanti-Orlin, *Network Flows*
  - CCPS = Cook-Cunningham-Pulleyblank-Schrijver, *Combinatorial Optimization*
  - S = Schrijver, *Combinatorial Optimization: Polyhedra and Efficiency*
  - CLRS = Cormen-Leiserson-Rivest-Stein, *Introductions to Algorithms*

Seminários

- Seminário de Teoria de Computação e Combinatória
  Sextas-feiras às 14 horas, na sala 252-A

- Seminários de Algoritmos e Combinatória
  Teças-feiras às 13 horas, na sala ??

Localização



Otimização Combinatória

Otimização Combinatória combina técnicas de

- combinatória
- programação linear
- teoria de algoritmos

para resolver problemas de otimização sobre

- estruturas discretas.

MAC0325+MAC5781

Nesta disciplina vamos estudar uma coleção de diferentes algoritmos para problemas de

Fluxo em Redes

que servirão de paradigma para muitas técnicas e idéias importantes e muito gerais, aplicáveis a muitos outros problemas no futuro (hmmmm).

O foco estará em

algoritmos combinatórios

com pitadas de

combinatória poliédrica

Pré-requisitos

MAC5781 não tem pré-requisitos oficiais.

O pré-requisito oficial de MAC0325 é MAC0122.

No entanto, é recomendável que já tenham cursado

- Análise de Algoritmos (MAC5711 ou MAC0338)
- Estruturas de Dados (MAC5710 ou MAC0323)
- Teoria dos Grafos (MAC5770 ou MAC0328)

Também são úteis conhecimentos de

- Programação Linear (MAC5790 ou MAC0315).

AULA 1

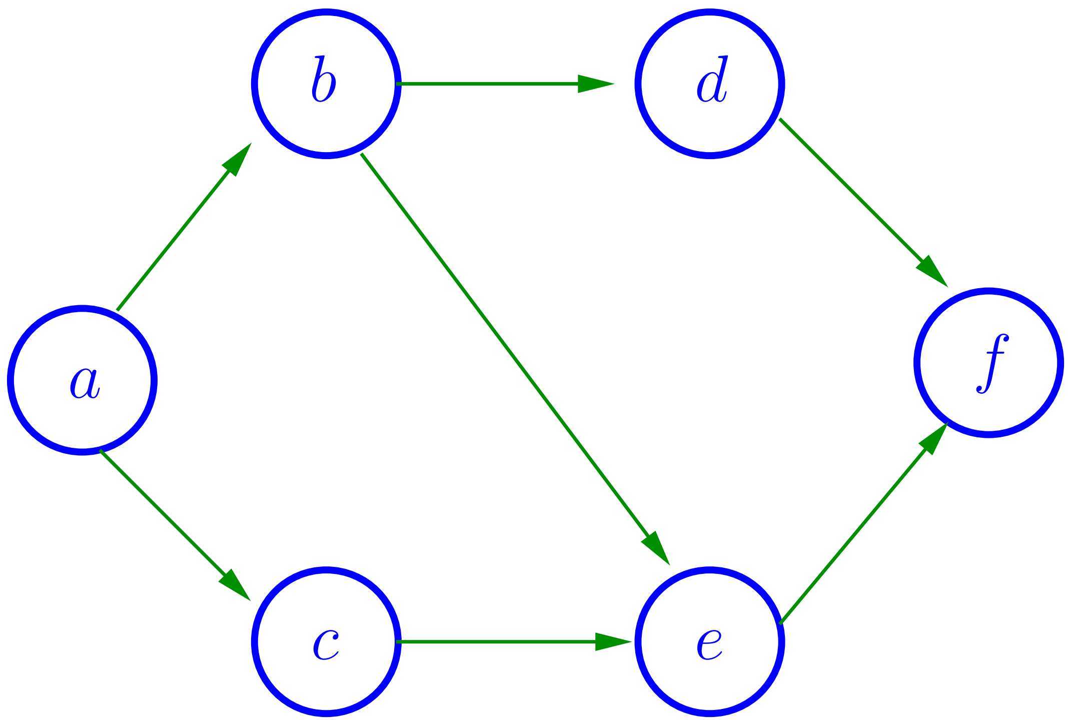
Redes

PF 2.1, 2.2, 2.3, 2.4, 2.5, 2.7

Grafos

Um grafo (= grafo orientado) é um objeto da forma (N, A), onde N é um conjunto finito e A é um conjunto de pares ordenados de elementos de N.

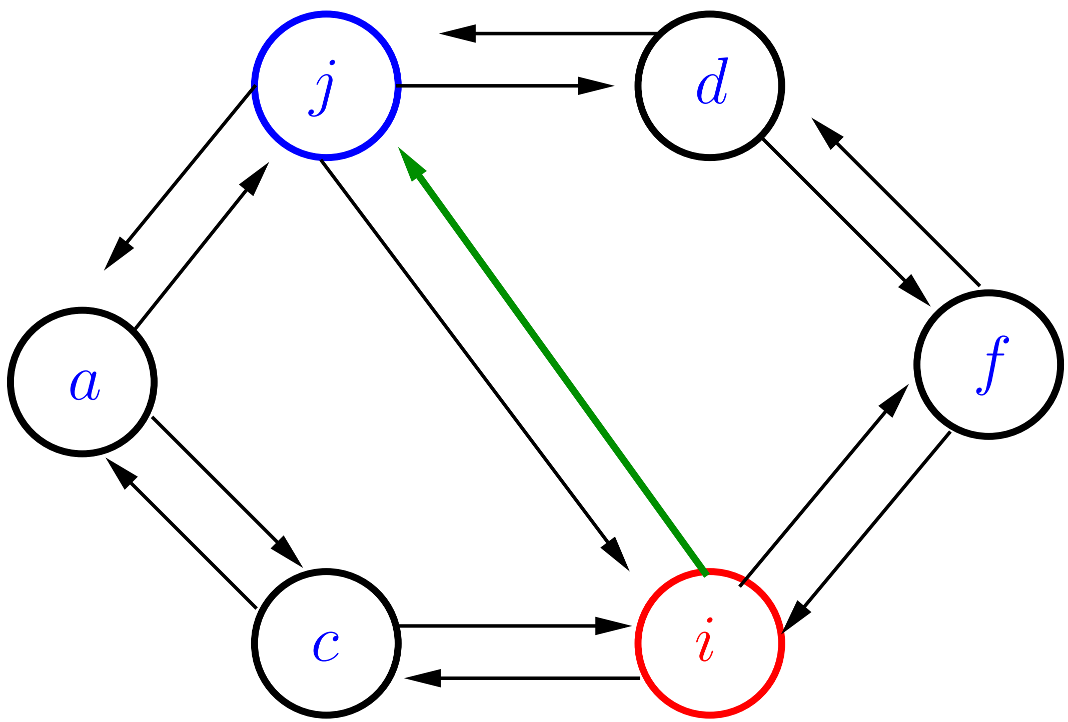
Exemplo:



Nós e arcos

Os elementos de N são chamados nós e os elementos de A são chamados arcos.

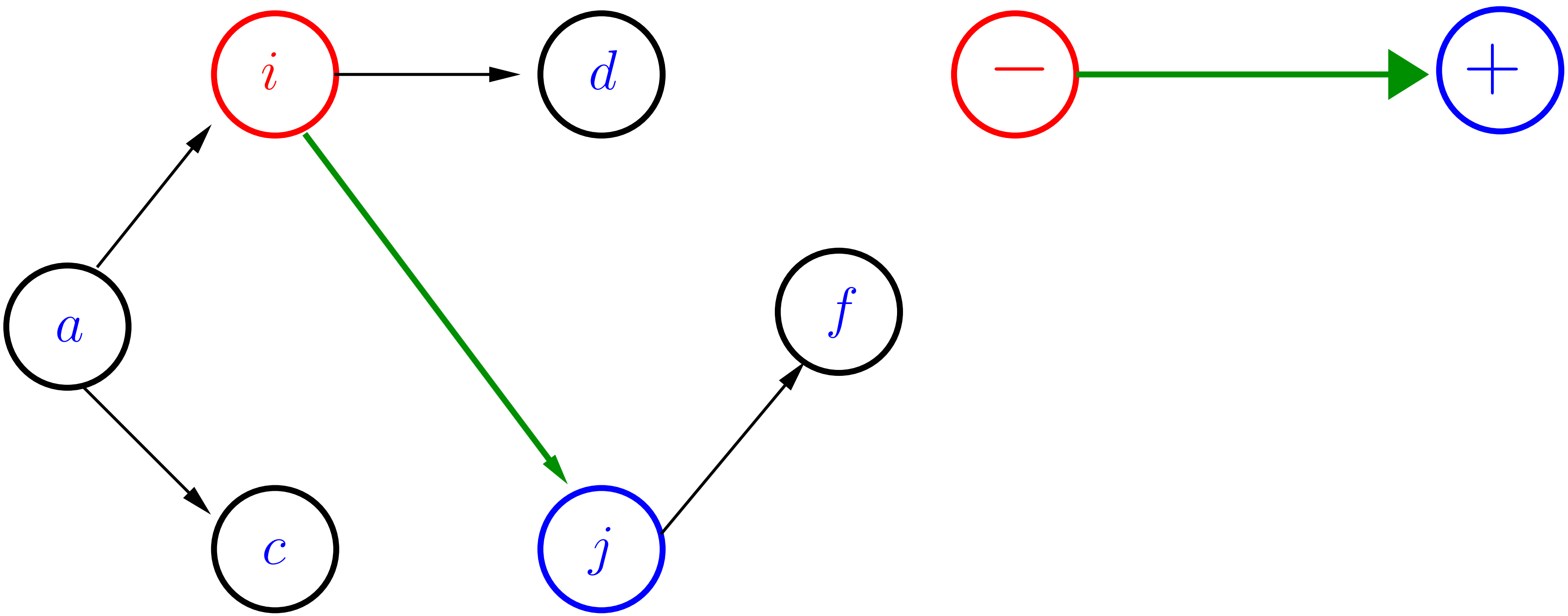
Exemplo: i e j são nós e (i, j) é um arco.



Ponta inicial e final

Para cada arco (i, j), o nó i é a ponta negativa ou ponta inicial e j é a ponta positiva ou ponta final.
Um arco (i, j) também poderá ser denotado por ij.

Exemplo: i é ponta inicial e j é ponta final de (i, j).



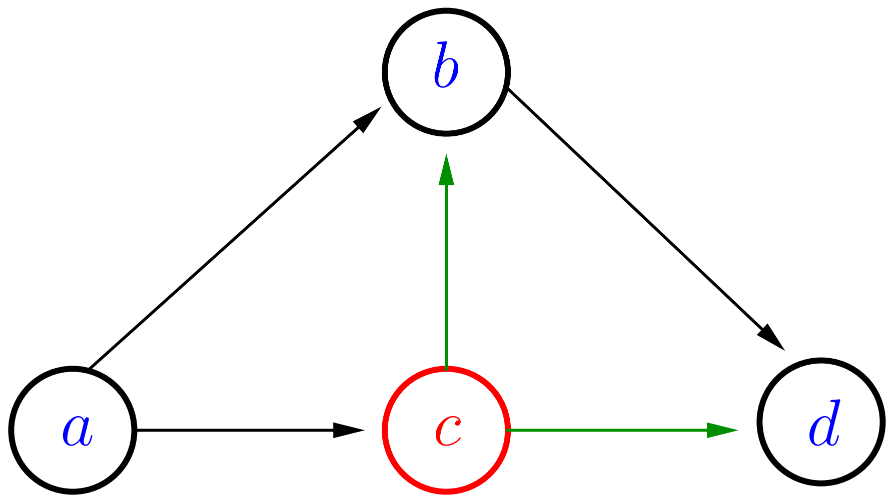
n, m e $A(i)$

$n := |N| \qquad m := |A|$

$A(i) :=$ conjunto dos arcos que saem de i

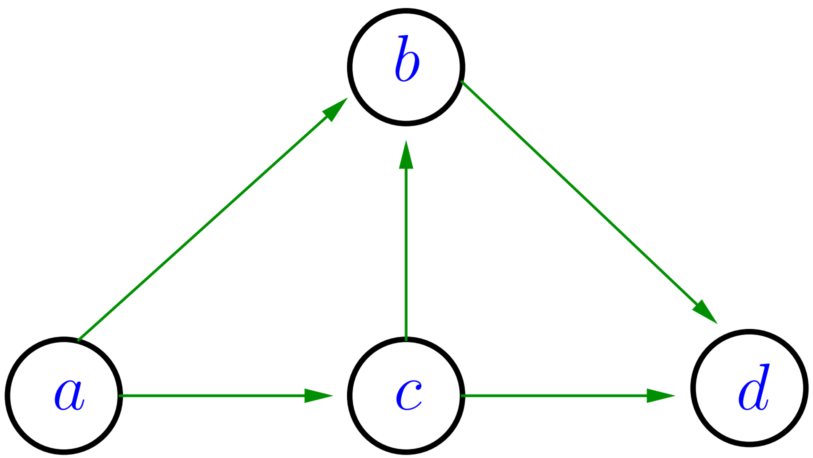
É evidente que $m \leq n(n-1) < n^2$.

Exemplo: $n = 4$, $m = 5$ e $A(c) = \{(c,b),(c,d)\}$.



Grafos no computador

Uma matriz de adjacência de um grafo (N, A) é uma matriz com valores em $\{0, 1\}$, e indexada por $N \times N$, onde cada entrada (i, j) da matriz tem valor 1 se $ij \in A$, e 0 caso contrário.

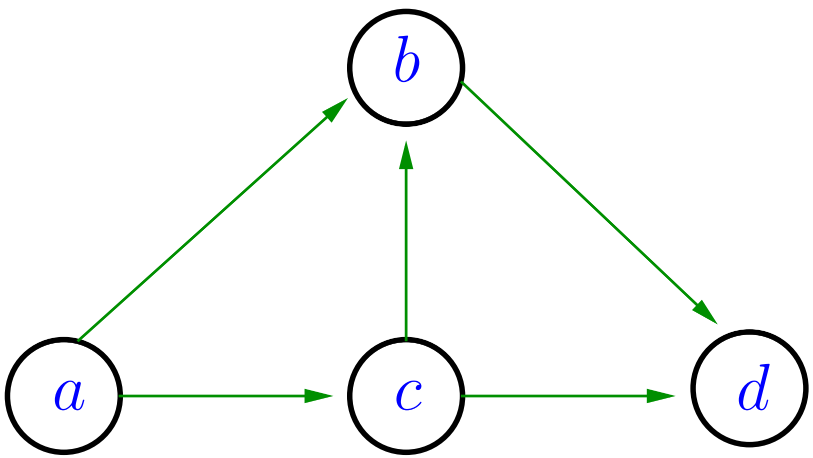


	a	b	c	d
a	0	1	1	0
b	0	0	0	1
c	0	1	0	1
d	0	0	0	0

Consumo de espaço: $\Theta(n^2)$ fácil de implementar

Grafos no computador

Uma matriz de incidências de um grafo (N, A) é uma matriz indexada por $N \times A$ e com valores em $\{-1, 0, +1\}$, onde cada entrada (k, ij) é -1 se $k = i$, $+1$ se $k = j$, e 0 em caso contrário.

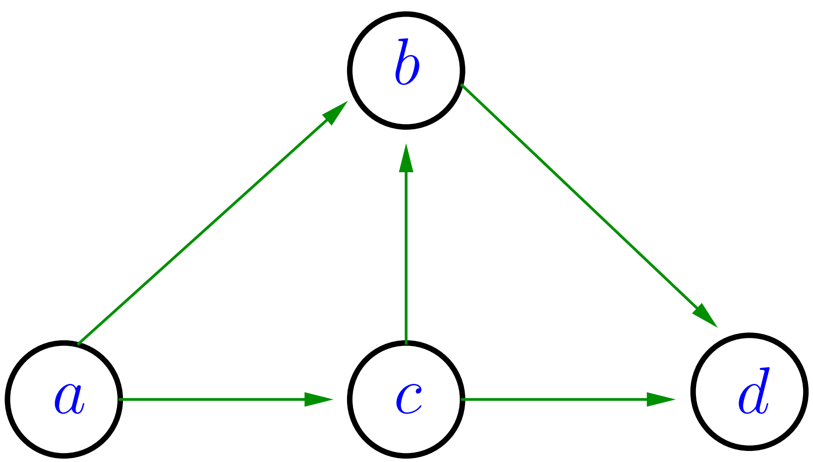


	ab	ac	cb	cd	bd
a	-1	-1	0	0	0
b	$+1$	0	$+1$	0	-1
c	0	$+1$	-1	-1	0
d	0	0	0	$+1$	$+1$

Consumo de espaço: $\Theta(nm)$
Interessante do ponto de vista de programação linear.

Grafos no computador

Na representação de um grafo (N, A) através de listas de adjacências tem-se, para cada vértice i, uma lista dos arcos deixando i. Desta forma, para cada nó i, o conjunto $A(i)$ é representado por uma lista.



$A(a)$: ab, ac

$A(b)$: bd

$A(c)$: cb, cd

$A(d)$:

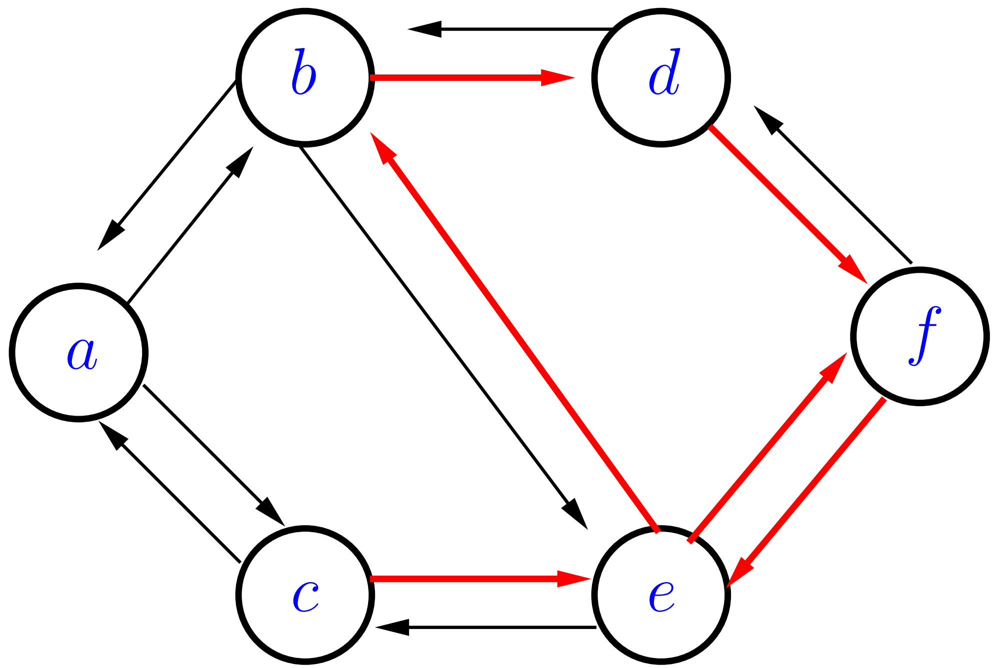
Consumo de espaço: $\Theta(n + m)$ (linear)

Manipulação eficiente

Passeios

Um passeio num grafo (N, A) é qualquer seqüência da forma $\langle v_0, v_1, v_2, \ldots, v_p \rangle$ onde (v_{k-1}, v_k) é um arco para $k = 1, \ldots, p$.

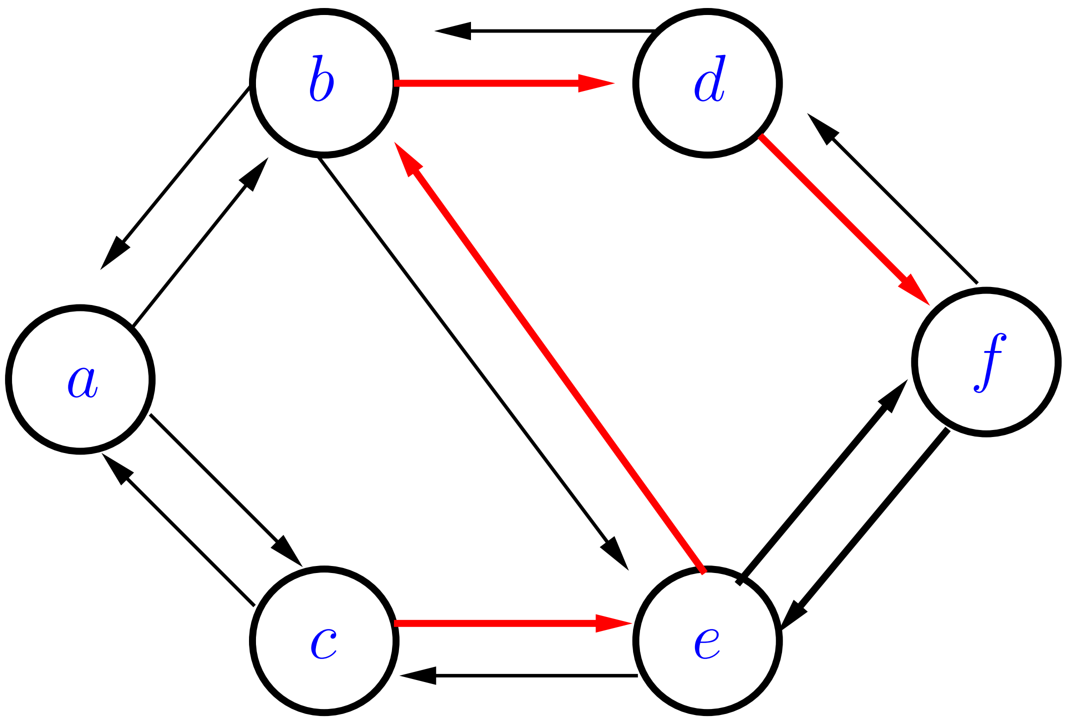
Exemplo: $\langle c, e, b, d, f, e, f \rangle$ é um passeio com origem em c é termino em f.



Caminhos

Um caminho é um passeio sem vértices repetidos.

Exemplo: $\langle c, e, b, d, f \rangle$ é um caminho de c a f.



Caminhos no computador

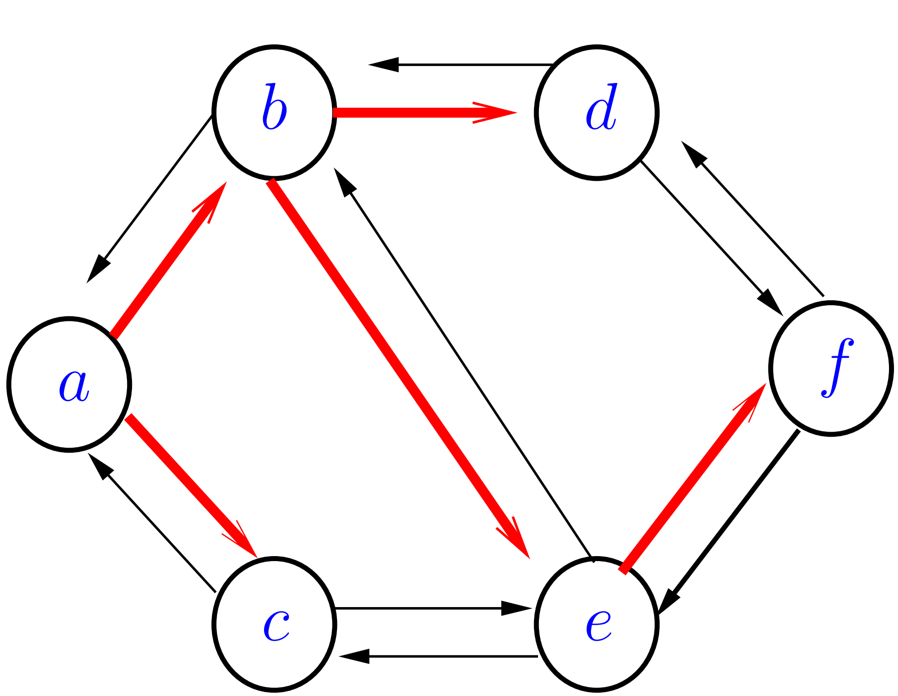
Como representar caminhos no computador?

Caminhos no computador

Uma maneira compacta de representar caminhos de um nó aos nós de um grafo é através de função-predecessor.

Uma função-predecessor é uma função “parcial” π de N em N tal que, para cada j em N,

$$\pi(j) = \text{NIL} \quad \text{ou} \quad (\pi(j), j) \in A$$

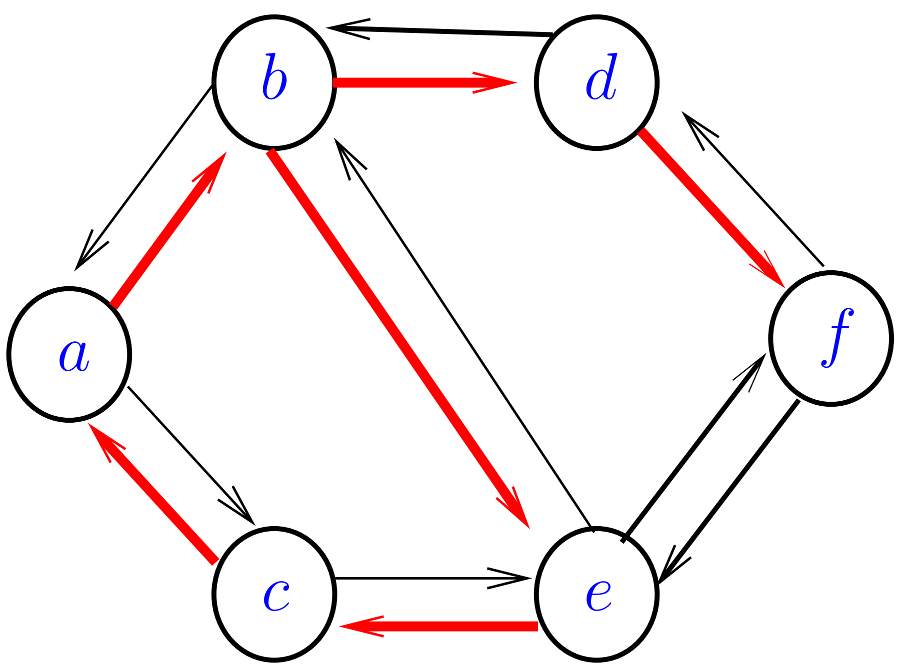


nó	π
a	NIL
b	a
c	a
d	b
e	b
f	e

Grafo de predecessores

Suponha que π é uma função-predecessor.
Denotaremos por A_π os arcos da forma $(\pi(j), j)$.
Diremos que (N, A_π) é o grafo de predecessores.

Exemplo:



nó	π
a	c
b	a
c	e
d	b
e	b
f	d

Brincadeirinha

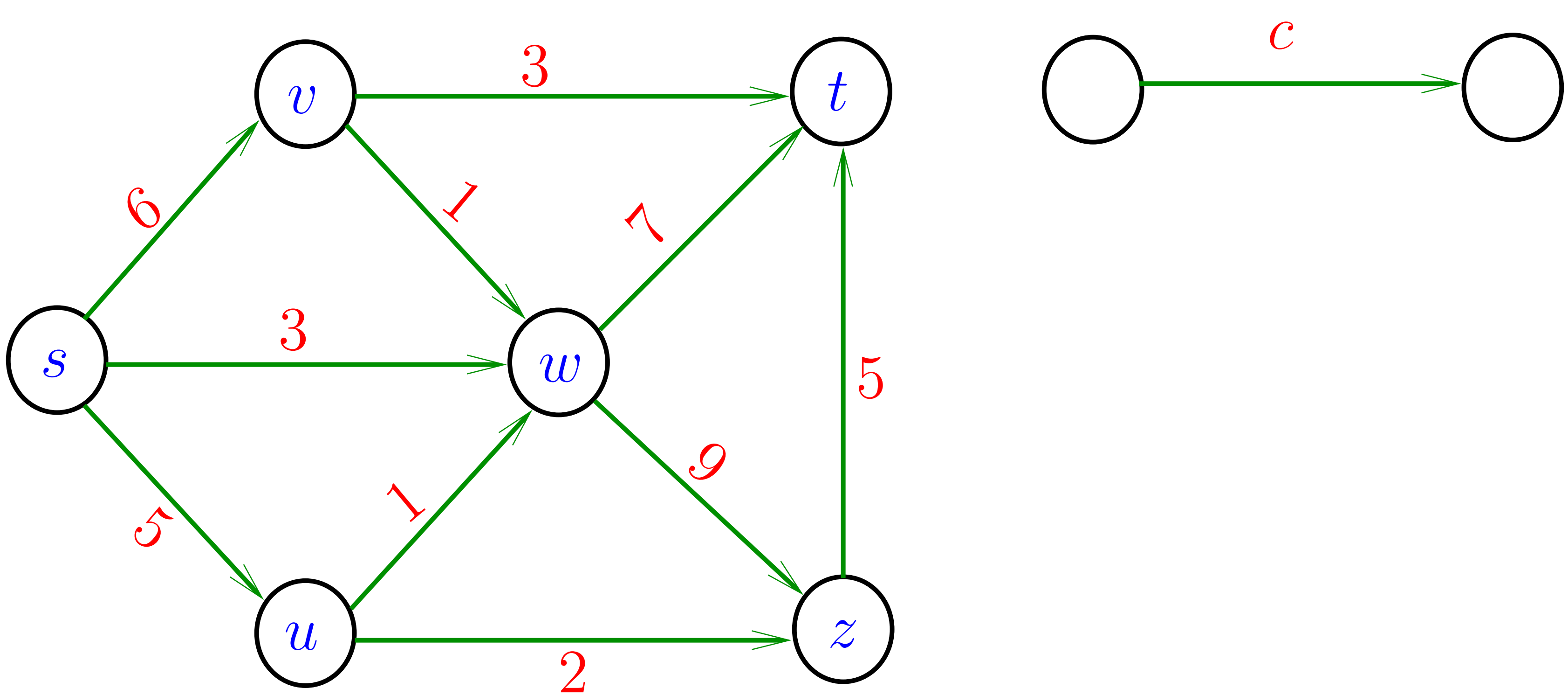
Algoritmo que calcula o caminho ou “quase-caminho” determinado por π a partir de um nó j.

$P \leftarrow \langle j \rangle$
$J \leftarrow \{j\}$
enquanto $\pi(j) \neq \text{NIL}$ **e** $\pi(j) \notin P$ **faça**
    $j \leftarrow \pi(j)$
    acrescente j ao início de P
    $J \leftarrow J \cup \{j\}$
se $\pi(j) = \text{NIL}$
    então P é um caminho
    senão acrescente $\pi(j)$ ao início de P
        P é um quase-caminho

Redes

Uma rede (= *network*) é um grafo (N, A) com uma ou mais funções que atribuem valores aos arcos e/ou arestas.

Exemplo: uma rede com uma “função-custo” c de A em $\mathbb{Z}$.



Algoritmos de Busca

PF 3.1, 3.2, 2.6

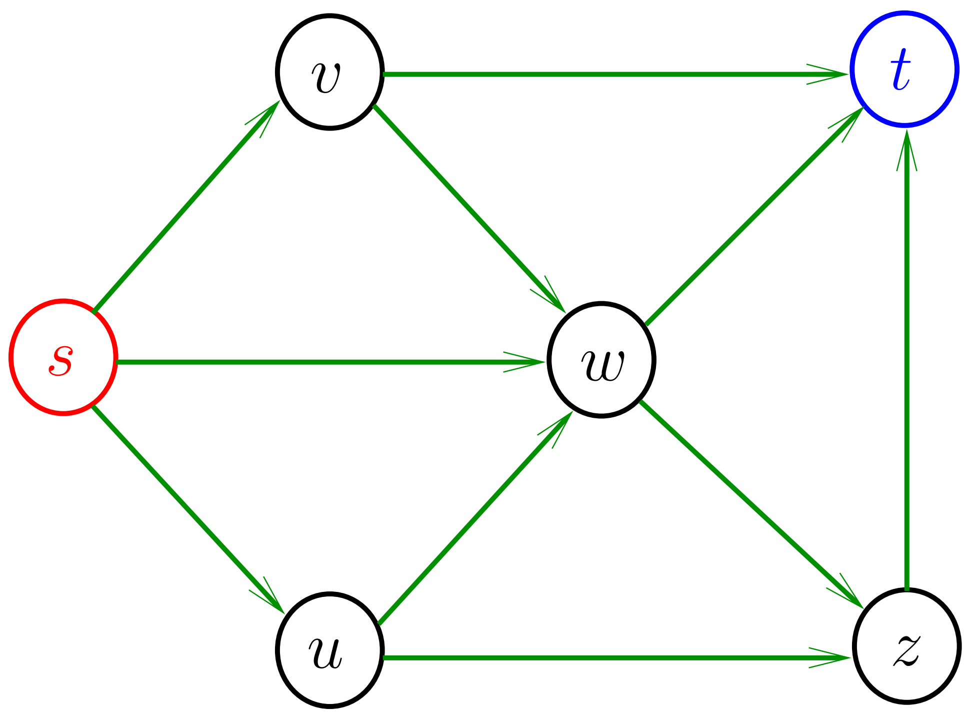
Busca

Problema de busca: Dados nós s e t de um grafo, encontrar um caminho de s a t

Busca

Problema de busca: Dados nós s e t de um grafo, encontrar um caminho de s a t

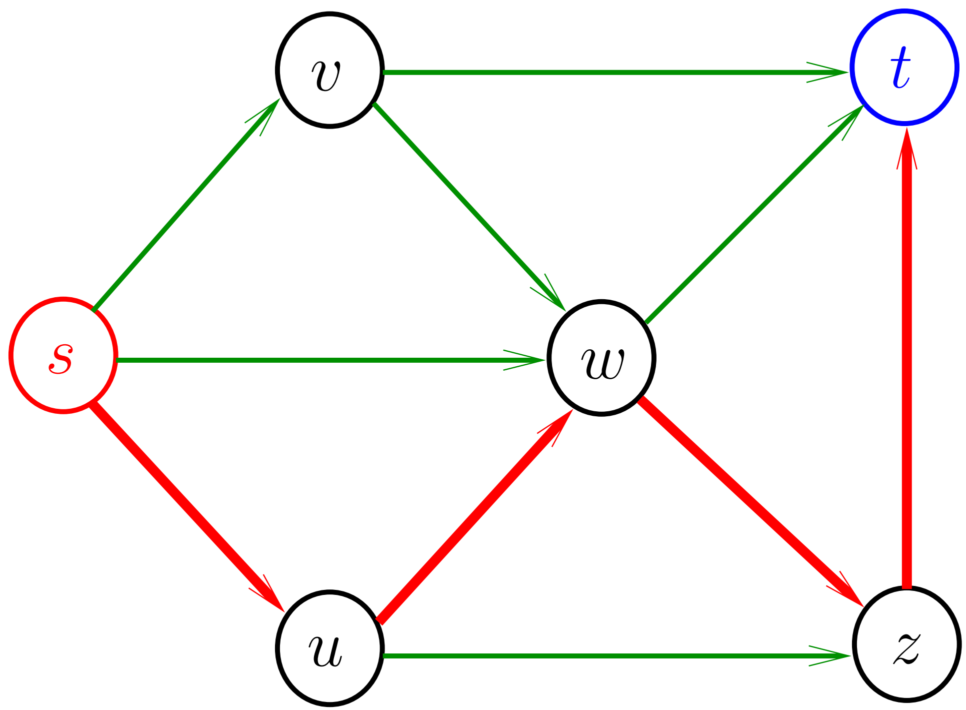
Entra:



Busca

Problema de busca: Dados nós s e t de um grafo, encontrar um caminho de s a t

Sai:



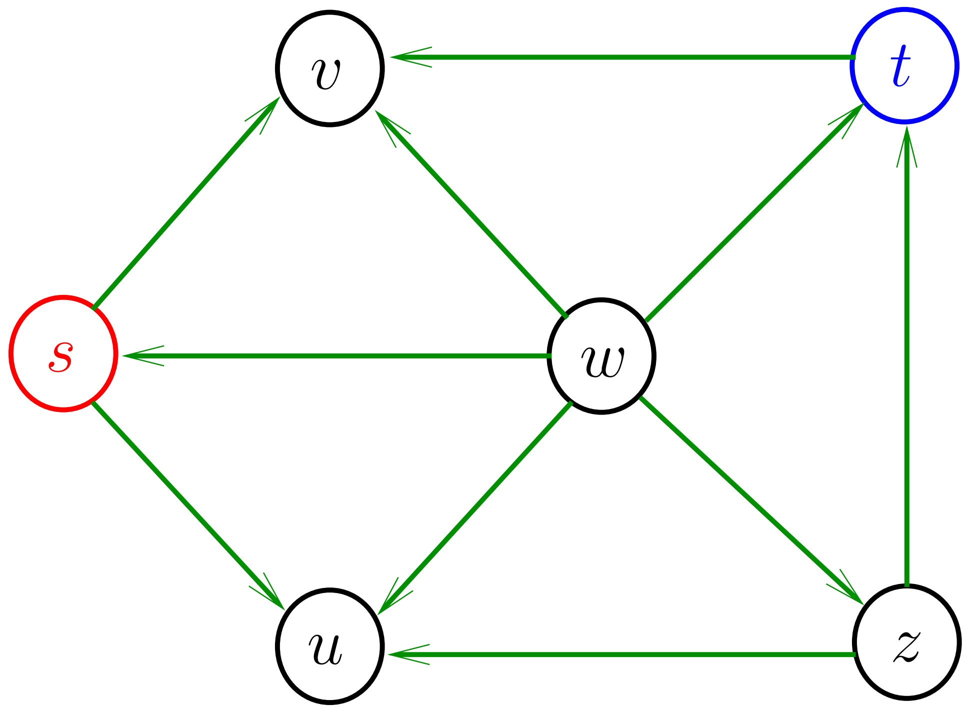
Condição de inexistência

Como é possível demonstrar que o problema **não** tem solução?

Condição de inexistência

Como é possível demonstrar que o problema **não** tem solução?

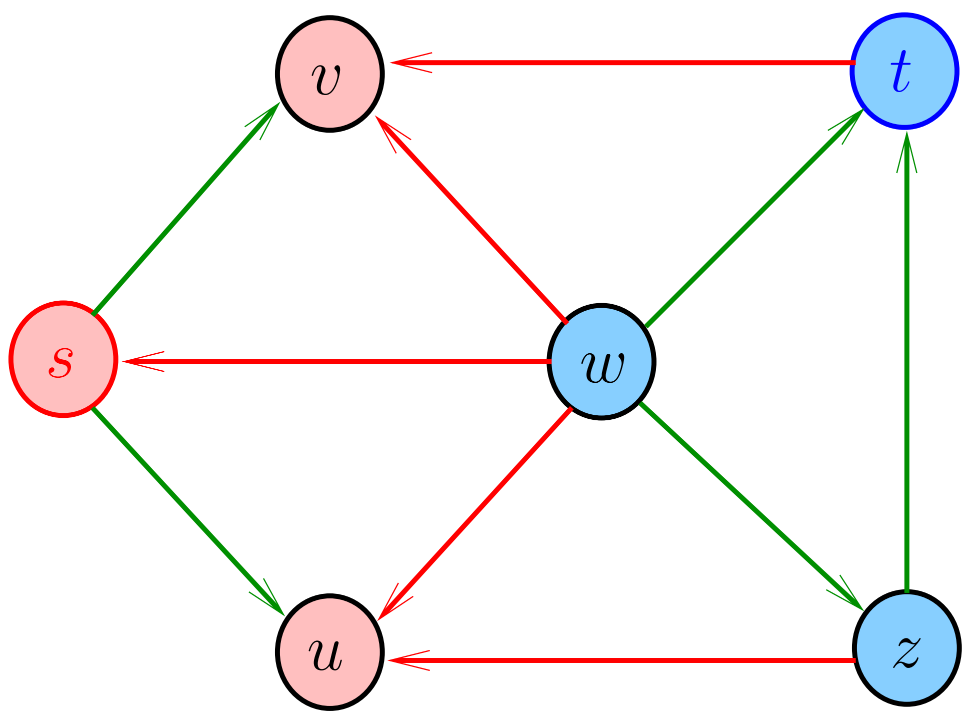
Entra:



Condição de inexistência

Como é possível demonstrar que o problema **não** tem solução?

Sai: um "certo corte" separando s de t

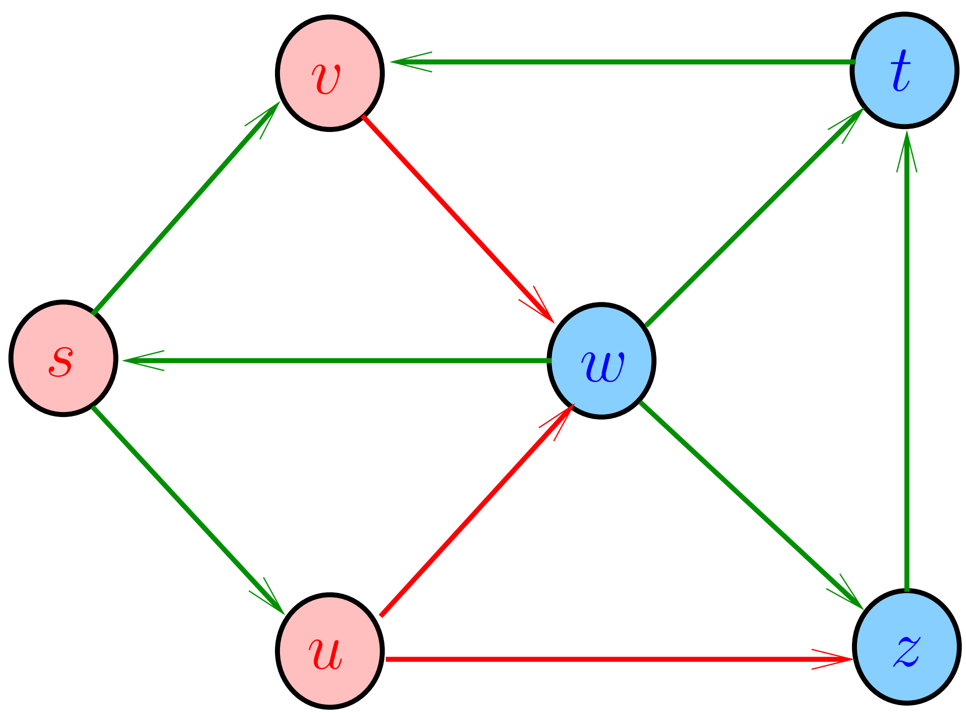


Cortes

Para quaisquer partes S e T de N:

$$\nabla(S,T) := \{st \in A : s \in S \text{ e } t \in T\}.$$

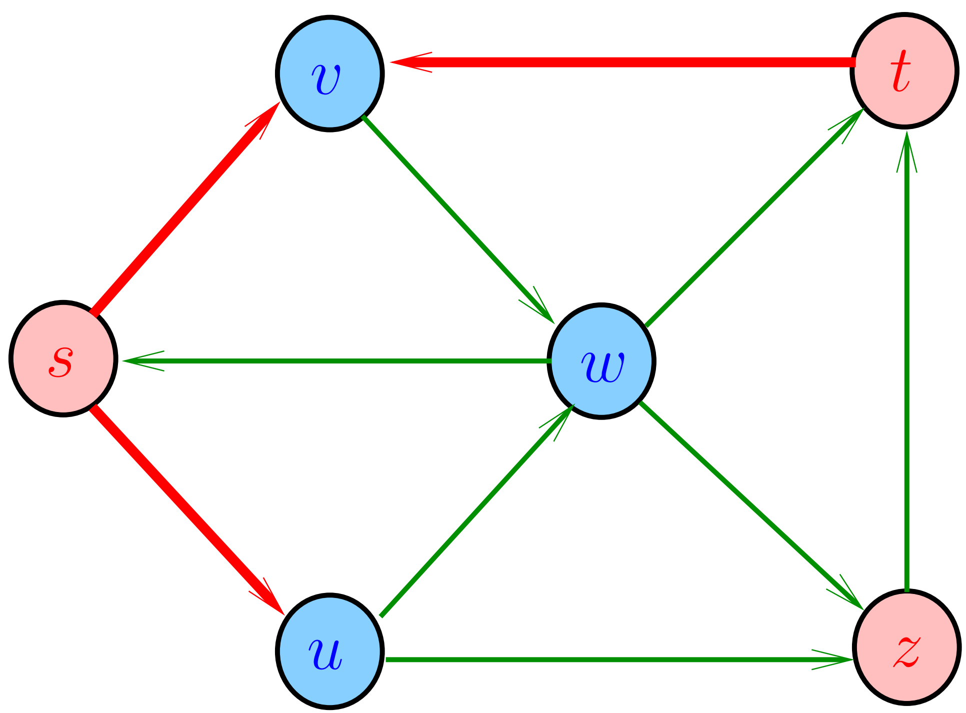
Exemplo: $\nabla(\{s,v,u\},\{w,t,z\}) = \nabla(\{v,u\},\{w,t,z\})$ são os arcos em vermelho.



Cortes

Para qualquer parte T de N, o corte determinado por T é $\nabla(N - T, T)$ (= arcos que entram em T).

Exemplo: Os arcos em vermelho estão no corte determinado por $\{v, w, u\}$.

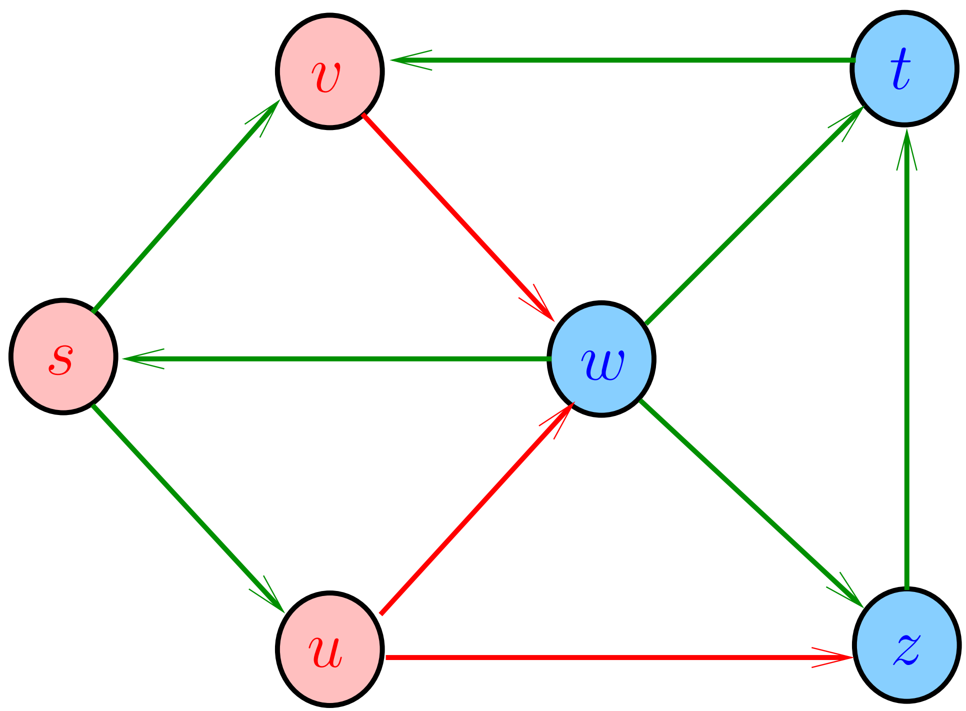


(s, t)-Cortes

Para s em $N - T$ e t em T dizemos que:

- T separa s de t
- $\nabla(N - T, T)$ separa s de t
- $\nabla(N - T, T)$ é um (s, t)-corte.

Exemplo: $\{w, t, z\}$ separa s de t

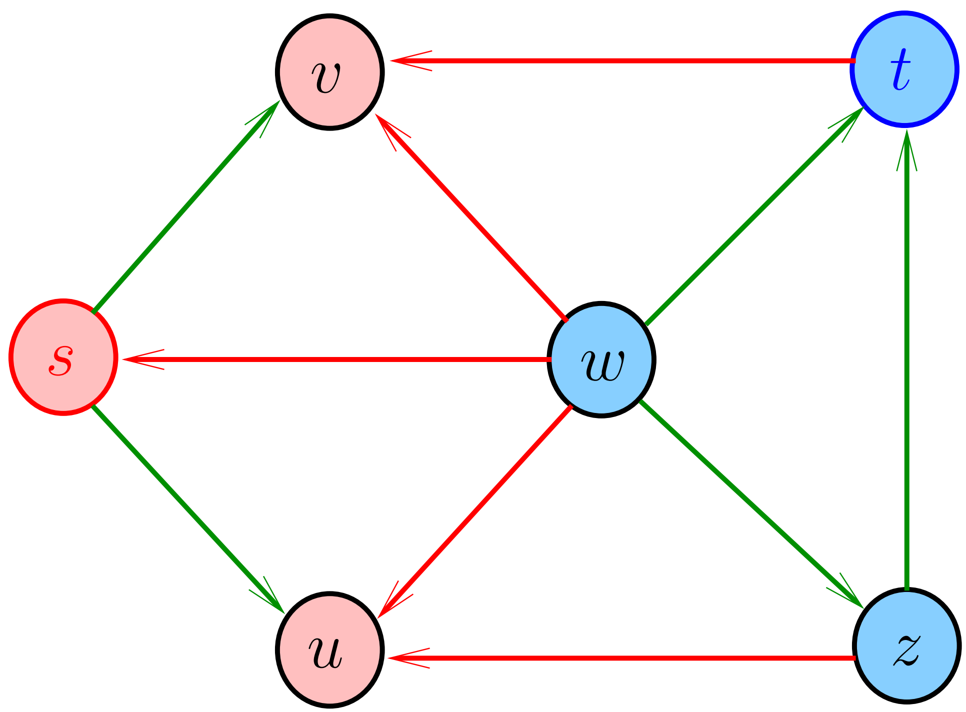


Condição de inexistência

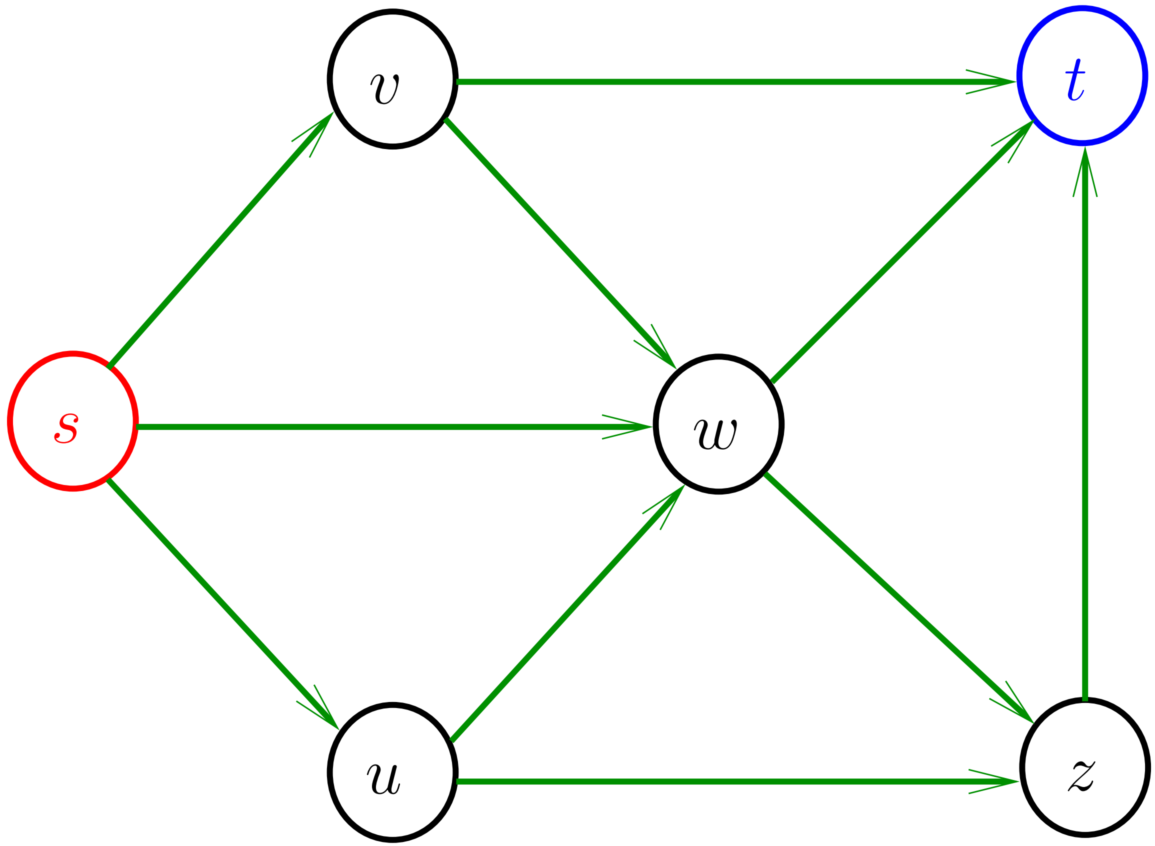
Para demonstrar que uma dada instância não tem solução basta exibir um (s,t)-corte tal que

$$\nabla(N-T,T)=\emptyset,$$

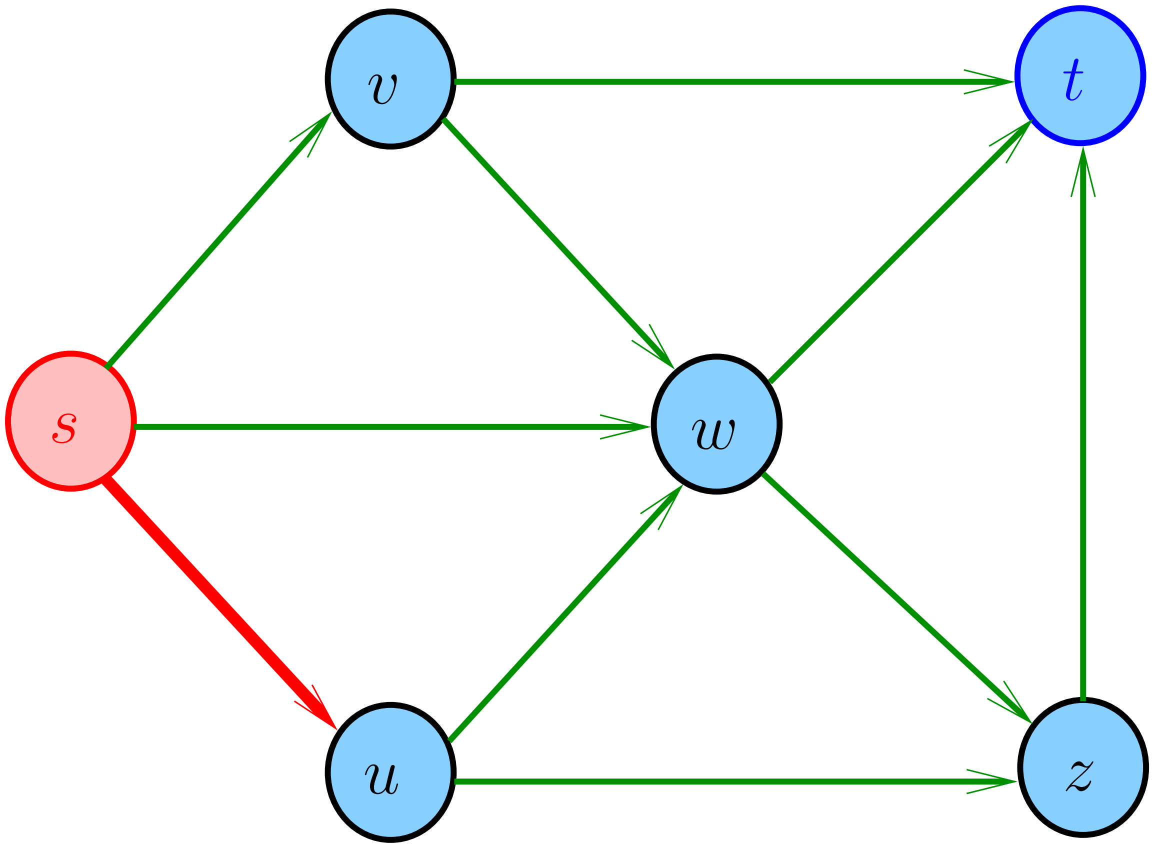
Exemplo: um (s,t)-corte vazio



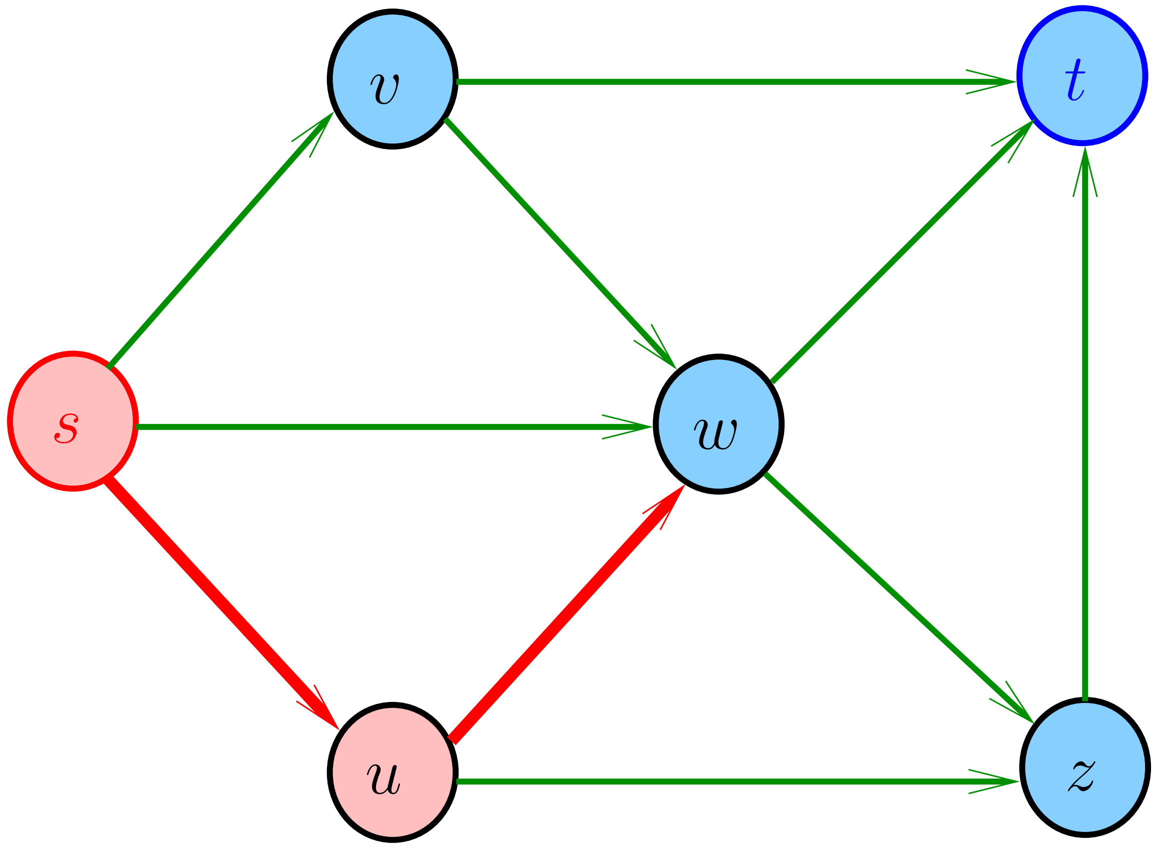
Simulação



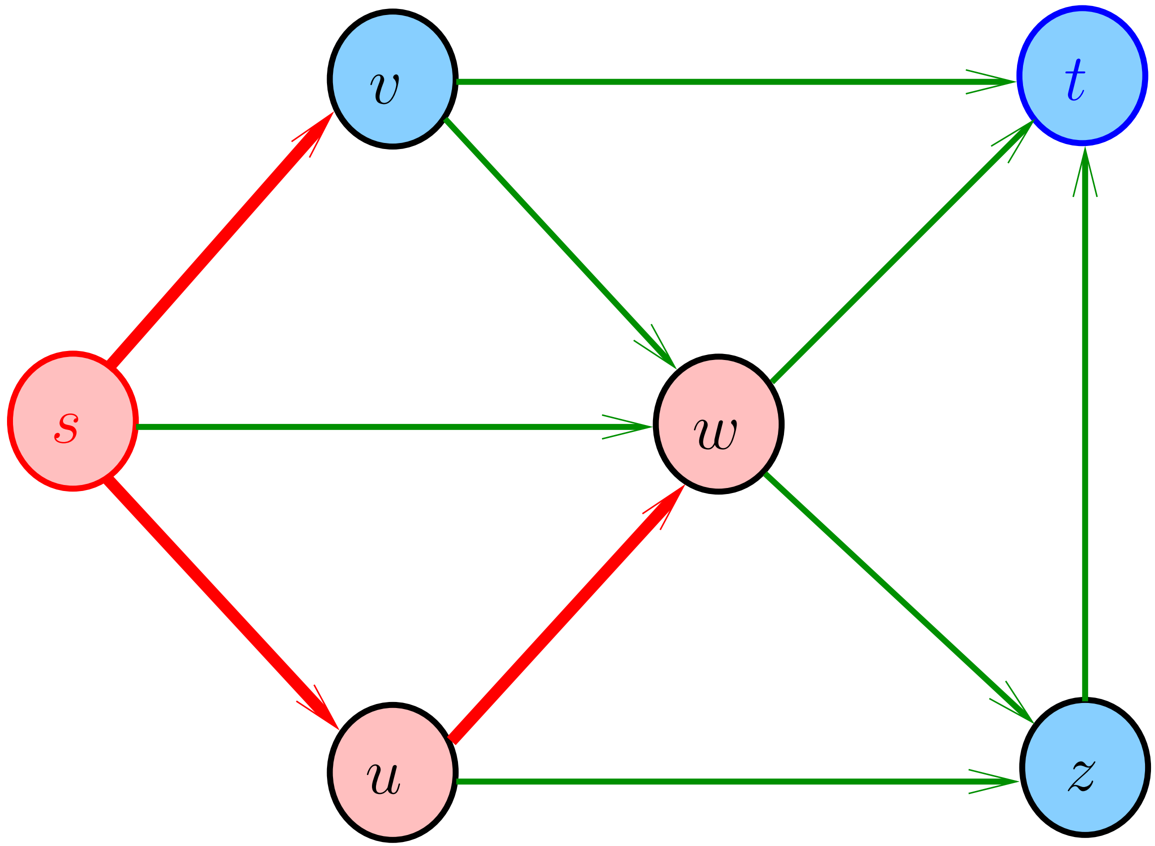
Simulação



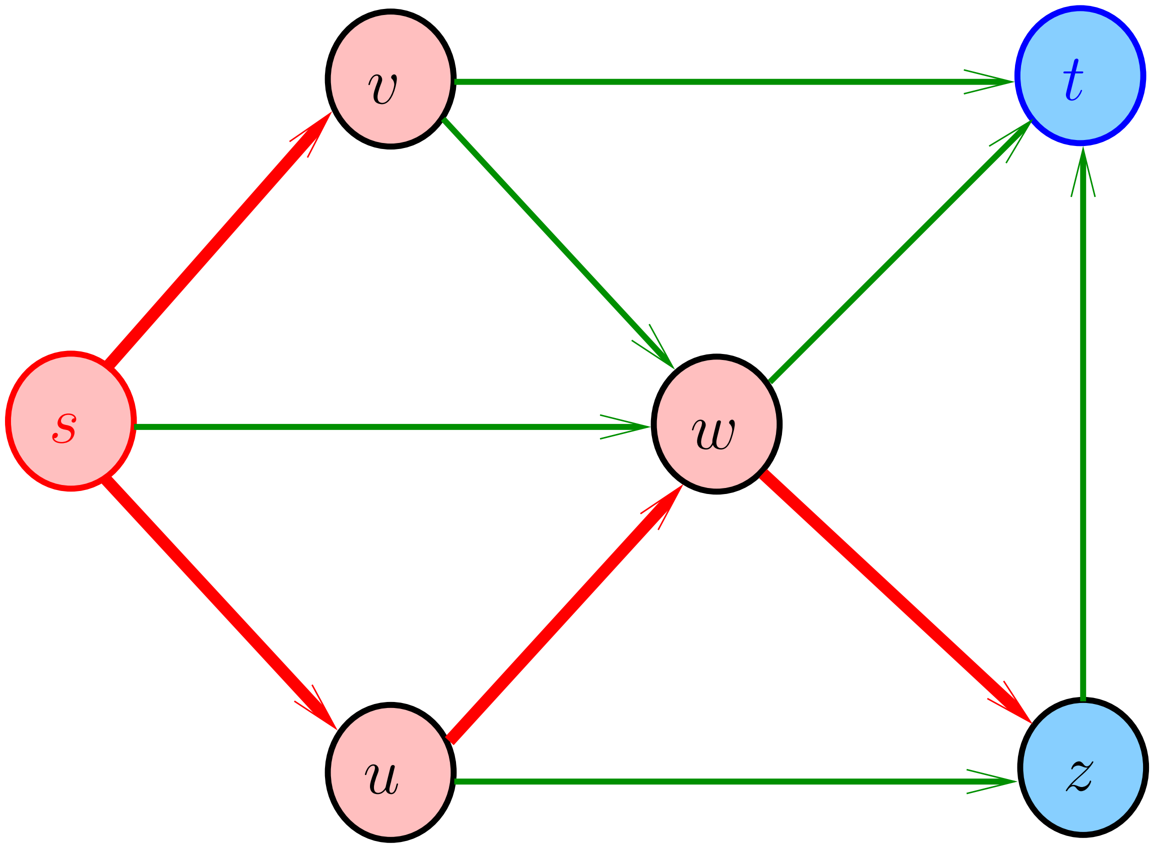
Simulação



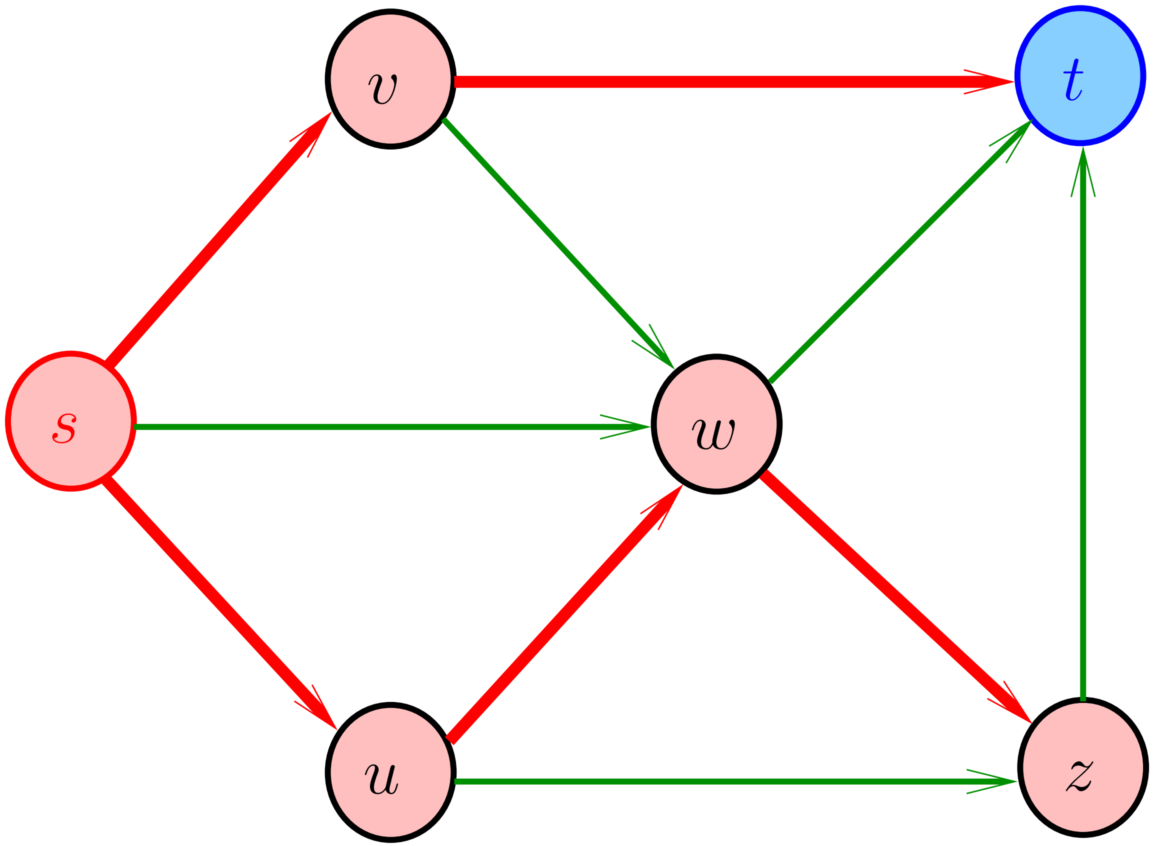
Simulação



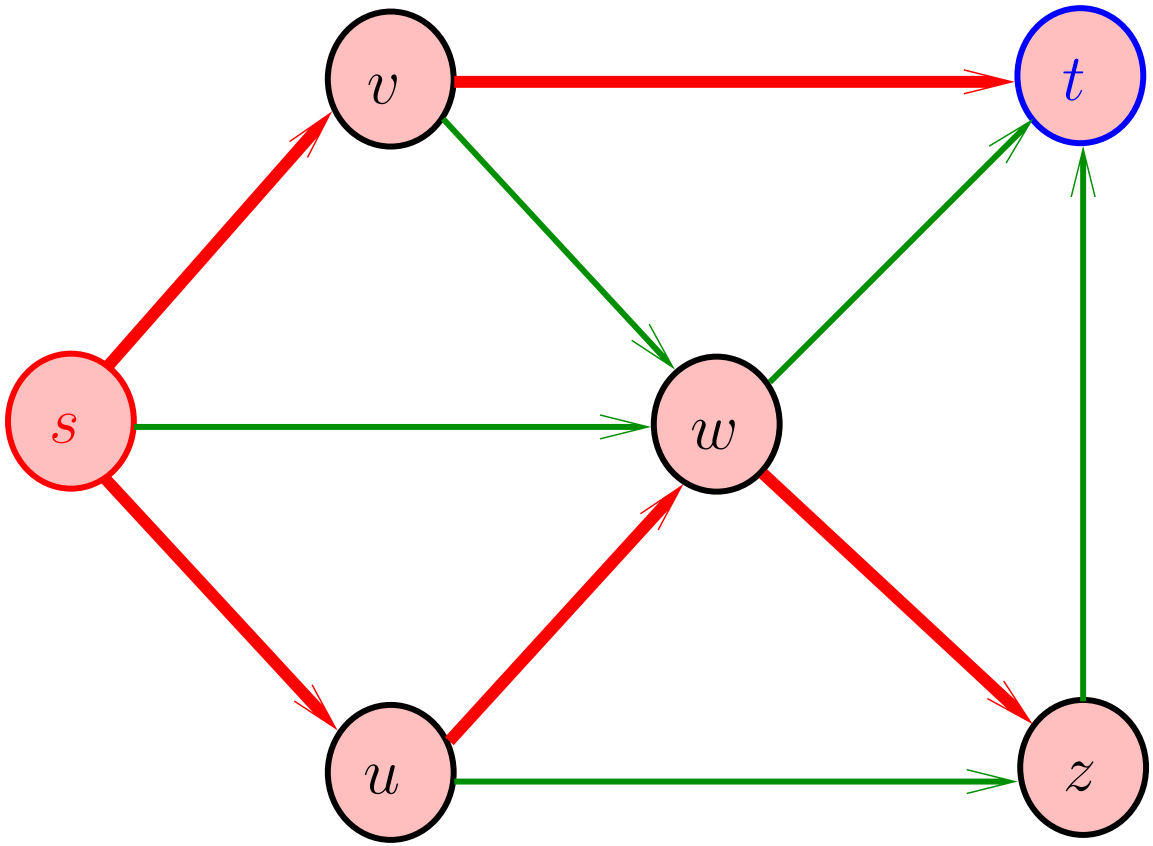
Simulação



Simulação



Simulação



Busca genérico

Recebe dois nós s e t de um grafo (N, A) e devolve uma caminho de s a t ou um (s,t)-corte vazio.

BUSCA-GENÉRICO (N, A, s, t)

para cada i em N **faça**
  $T \leftarrow T \cup \{i\}$
  $\pi(i) \leftarrow \text{NIL}$
$S \leftarrow \{s\} \quad T \leftarrow T - \{s\}$

enquanto existe $ij \in A$ com $i \in S$ e $j \in T$ **faça**
  $S \leftarrow S \cup \{j\} \quad T \leftarrow T - \{j\}$
  $\pi(j) \leftarrow i$

se $t \in S$
  então devolva o st-caminho no grafo (N, A_π)
  senão devolva S e T $\quad \triangleright\ \nabla(S,T) = \emptyset$

Busca genérico

BUSCA-GENÉRICO (N, A, s, t). Recebe um grafo (N, A), e dois nós s e t e devolve st-caminho ou $T \subset N$ que separa s de t e $\nabla(N - T, T) = \emptyset$.

Cada iteração começa com uma partição (S, T) de N e uma função-predecessor π. No início da primeira iteração $S = \{s\}$, $T = N - \{s\}$ e $\pi(i) = \text{NIL}$ para todo nó i. Cada iteração consiste no seguinte.

Busca genérico

BUSCA-GENÉRICO (N, A, s, t). Recebe um grafo (N, A), e dois nós s e t e devolve st-caminho ou $T \subset N$ que separa s de t e $\nabla(N - T, T) = \emptyset$.

Cada iteração começa com uma partição (S, T) de N e uma função-predecessor π. No início da primeira iteração $S = \{s\}$, $T = N - \{s\}$ e $\pi(i) = \text{NIL}$ para todo nó i. Cada iteração consiste no seguinte.

Caso 1: não existe $ij \in A$ com $i \in S$ e $j \in T$

Caso 1A: $t \in S$

Devolva o st-caminho no grafo (N, A_π) e pare.

Caso 1B: $t \notin S$

Devolva T e pare.

Busca genérico

BUSCA-GENÉRICO (N, A, s, t). Recebe um grafo (N, A), e dois nós s e t e devolve st-caminho ou $T \subset N$ que separa s de t e $\nabla(N - T, T) = \emptyset$.

Cada iteração começa com uma partição (S, T) de N e uma função-predecessor π. No início da primeira iteração $S = \{s\}$, $T = N - \{s\}$ e $\pi(i) = \text{NIL}$ para todo nó i. Cada iteração consiste no seguinte.

Caso 2: existe $ij \in A$ com $i \in S$ e $j \in T$

$S' \leftarrow S \cup \{j\}$

$T' \leftarrow T - \{j\}$

Seja π' tal que $\pi'(k) := \begin{cases} \pi(k) & \text{se } k \neq j, \\ i & \text{se } k = j. \end{cases}$

Comece nova iteração com S', T' e π' nos papéis de S, T e π.