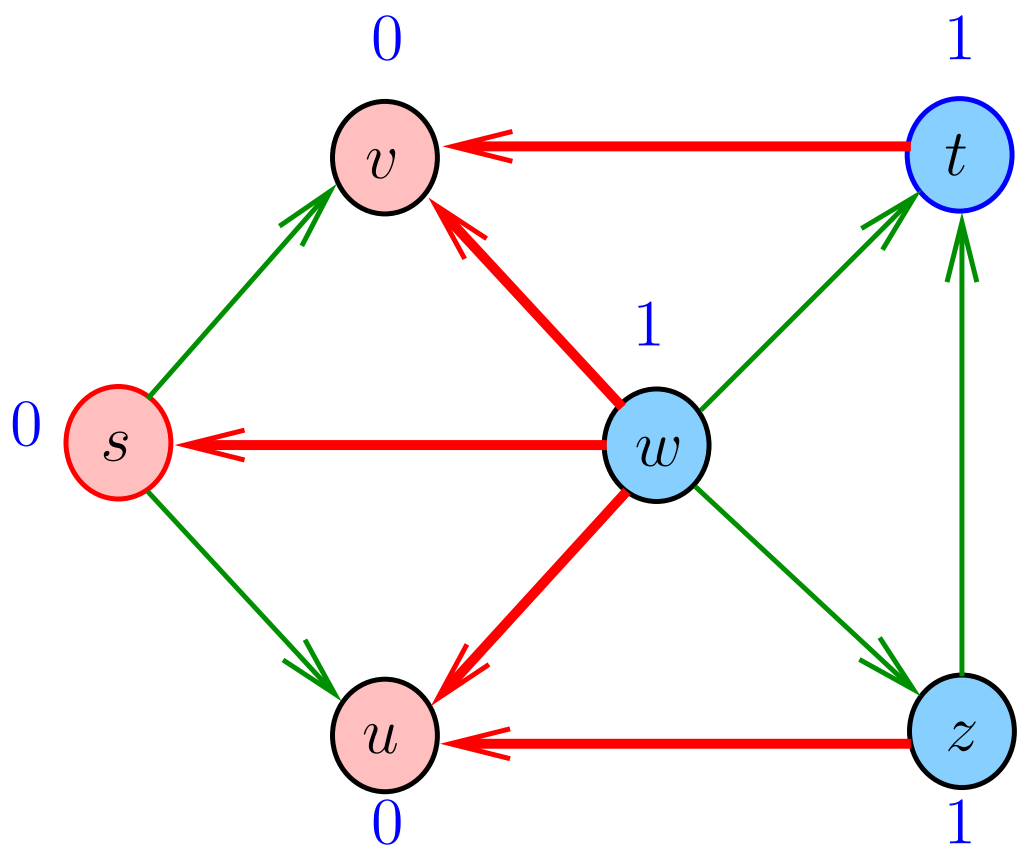
# Melhores momentos

AULA 2

# Propriedade de $0$-Potenciais

Se $y$ é um $0$-potencial e existe um passeio de $s$ a $t$ então

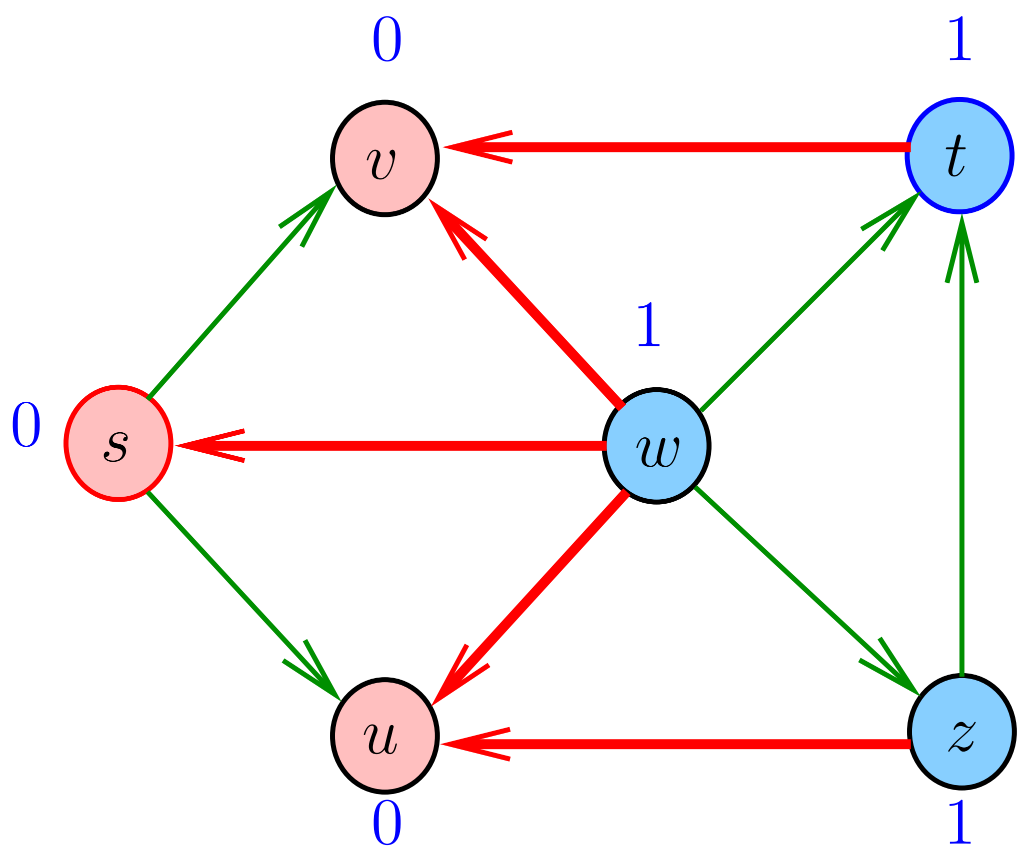
$$y(t) - y(s) \leq 0.$$



# Propriedade de $0$-Potenciais

Para mostrar que não existe um caminho de $s$ a $t$ basta exibir um $0$-potencial tal que

$$y(t) - y(s) > 0.$$



# Busca genérico (2)

Recebe dois nós $s$ e $t$ de um grafo $(N, A)$ e devolve uma caminho de $s$ a $t$ ou um $0$-potencial $y$ tal que $y(t) - y(s) > 0$.

BUSCA-GENÉRICO $(N, A, s, t)$

**para cada** $i$ em $N$ **faça**<br>
    $y(i) \leftarrow 1$<br>
    $\pi(i) \leftarrow \mathrm{NIL}$<br>
$y(s) \leftarrow 0$

**enquanto** $y(j) > y(i)$ **para** algum $ij \in A$ **faça**<br>
    $y(j) \leftarrow y(i)$<br>
    $\pi(j) \leftarrow i$

**se** $y(t) = 0$<br>
    **então devolva** o $st$-caminho no grafo $(N, A_\pi)$<br>
    **senão devolva** $y$

# Correção

Início da última iteração:

- $y$ é um $0$-potencial
- se $y(t) = 0$ então (por (i3)) $\pi(t) \neq \text{NIL}$, logo (por (i4)) existe caminho de $s$ a $t$
- se $y(t) = 1$ então (por (i2)) $y(t) - y(s) > 0$

Conclusão: o algoritmo faz o que promete.

# Conclusão

Para quaisquer nós $s$ e $t$ de um grafo $(N, A)$, vale uma e apenas umas das seguintes afirmações:

- existe um caminho de $s$ a $t$
- existe um $0$-potencial $y$ tal que $y(t) - y(s) > 0$.

# Implementação do algoritmo genérico

BUSCA $(N, A, s, t)$

**para cada** $i$ em $N$ **faça**
  $A'(i) \leftarrow A(i) \quad y(i) \leftarrow 1 \quad \pi(i) \leftarrow \mathrm{NIL}$
$y(s) \leftarrow 0 \quad L \leftarrow \{s\}$

**enquanto** $L \neq \emptyset$ **faça**
  escolha um nó $i$ em $L$
  **se** $A'(i) \neq \emptyset$ **então**
    retire um arco $ij$ de $A'(i)$
    **se** $y(j) = 1$ **então**
      $y(j) \leftarrow 0 \quad \pi(j) \leftarrow i \quad L \leftarrow L \cup \{j\}$
  **senão** $L \leftarrow L - \{i\}$

**se** $y(t) = 0$
  **então devolva** o $st$-caminho no grafo $(N, A_\pi)$
  **senão devolva** $y$

# Invariantes

Na linha 3, antes da verificação da condição "$L \neq \emptyset$" valem, além de (i0)–(i3) as seguintes invariantes:

(i4) para cada arco $pq$, se $y(p) = 0$ e $y(q) = 1$ então $p \in L$;

(i5) $y(p) = 0$ para cada $p$ em $L$;

(i6) para cada nó $p$ e cada arco $pq$ em $A(p) - A'(p)$, se $y(p) = 0$ então $y(q) = 0$.

# Correção

No início da última iteração:

- Por (i4), $y(q) - y(p) \leq 0$ para todo $pq$; portanto, $y$ é um $0$-potencial.
- Se $y(t) = 1$, então (por (i1)) $y(t) - y(s) = 1 > 0$.
- Senão (por (i3)), há caminho de $s$ a $t$.

Conclusão: o algoritmo faz o que promete.

# AULA 3

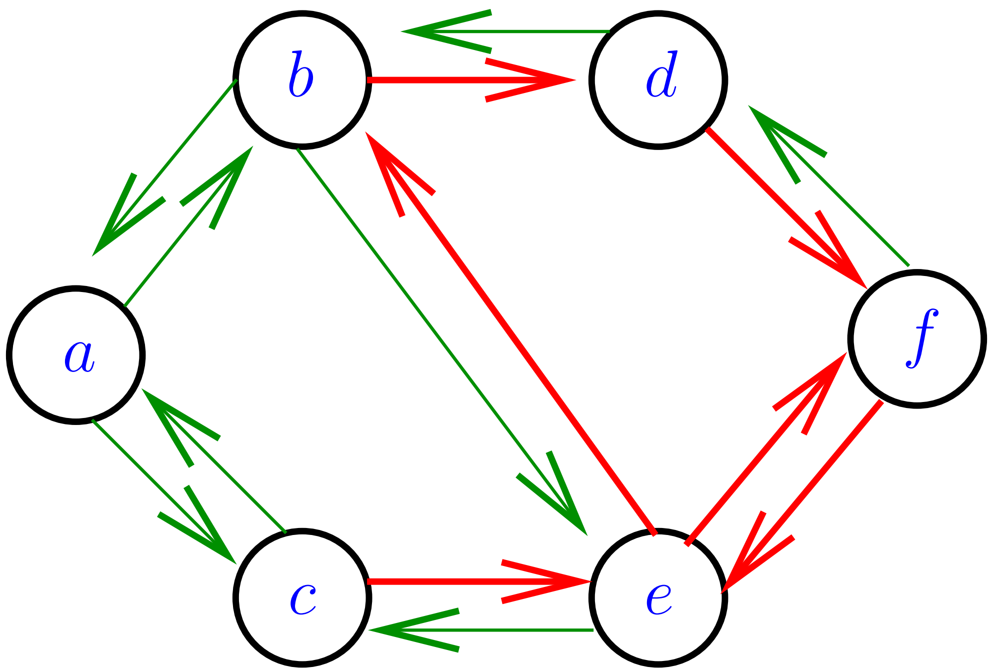
# Ciclos e ordem topológica

PF 4.1, 4.2, 4.3

# 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 $f$.

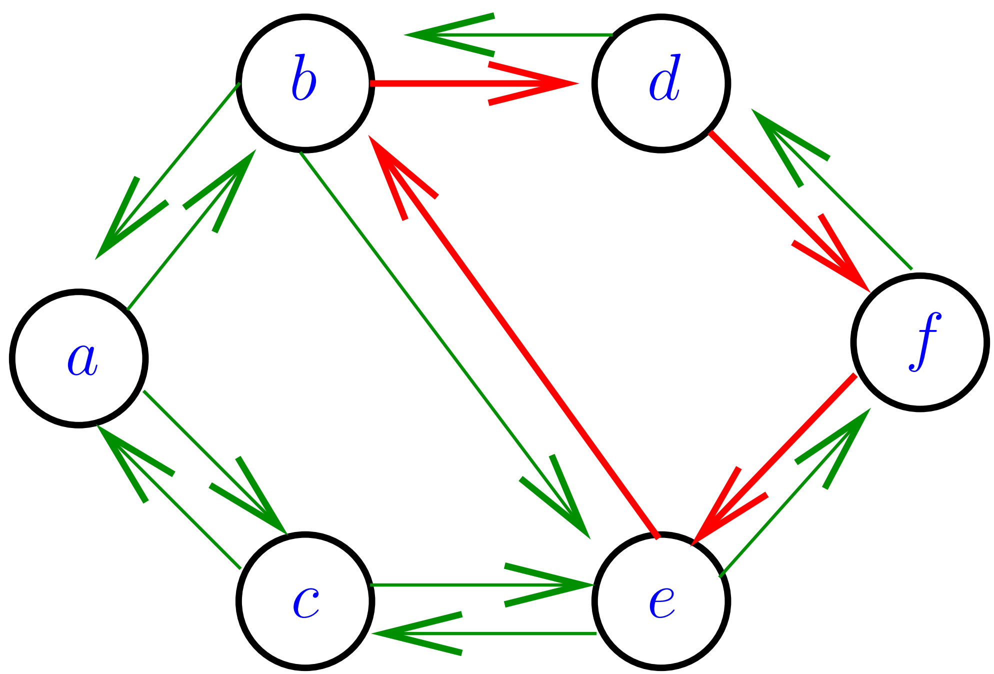


# Ciclo

Um ciclo é um passeio $\langle v_0, v_1, v_2, \ldots, v_p \rangle$ em que $v_1, \ldots, v_p$ são distintos doia a dois e $v_0 = v_p$.

Um grafo é acíclico se não tem ciclos

Exemplo: $\langle f, e, b, d, f \rangle$ é um ciclo.



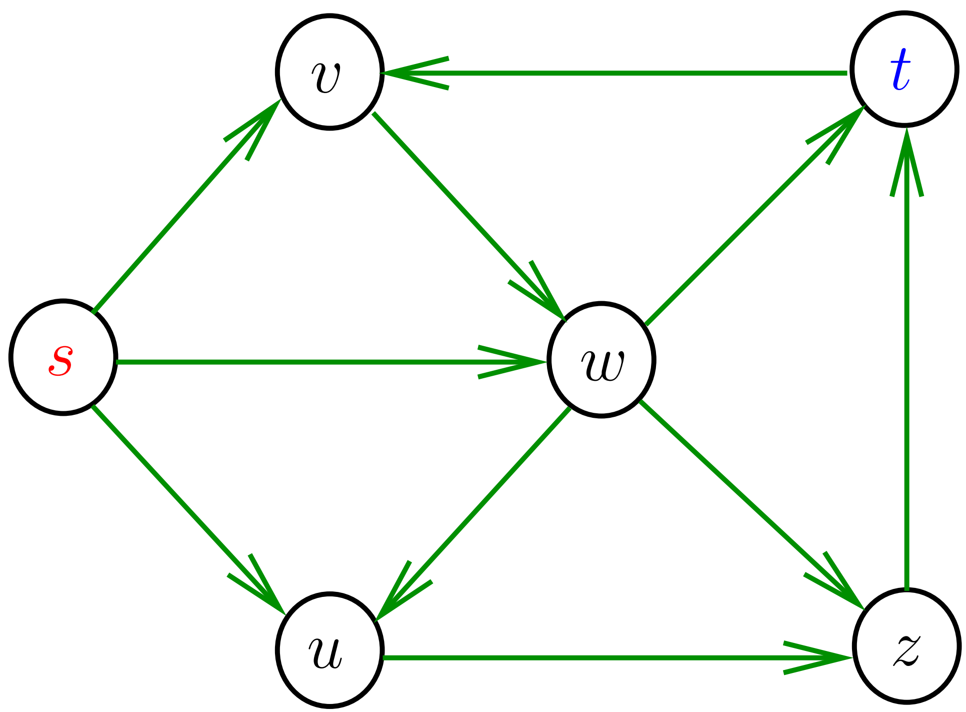
# Problema

Problema do ciclo: Encontrar um ciclo de um grafo dado.

# Problema

Problema do ciclo: Encontrar um ciclo de um grafo dado.

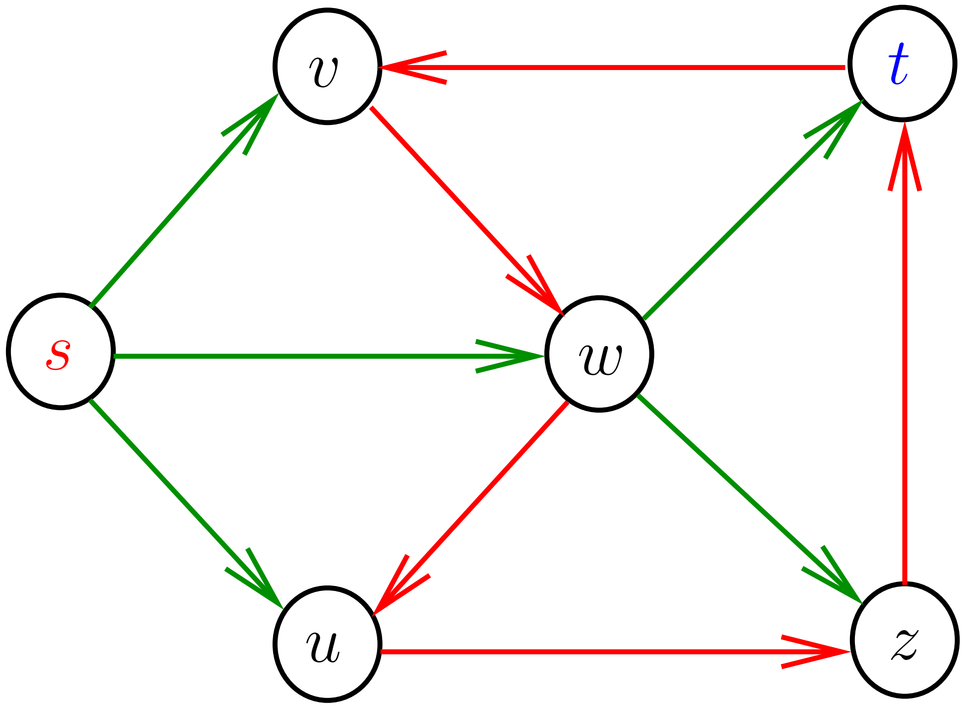
Entra:



# Problema

Problema do ciclo: Encontrar um ciclo de um grafo dado.

Sai:



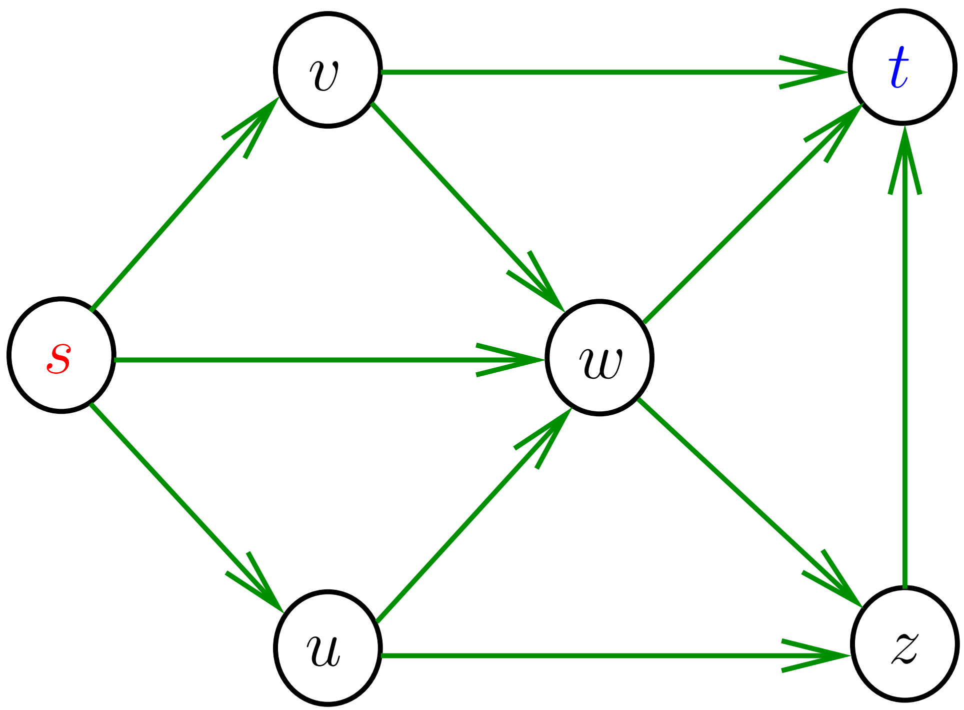
# Condição de inexistência

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

# Condição de inexistência

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

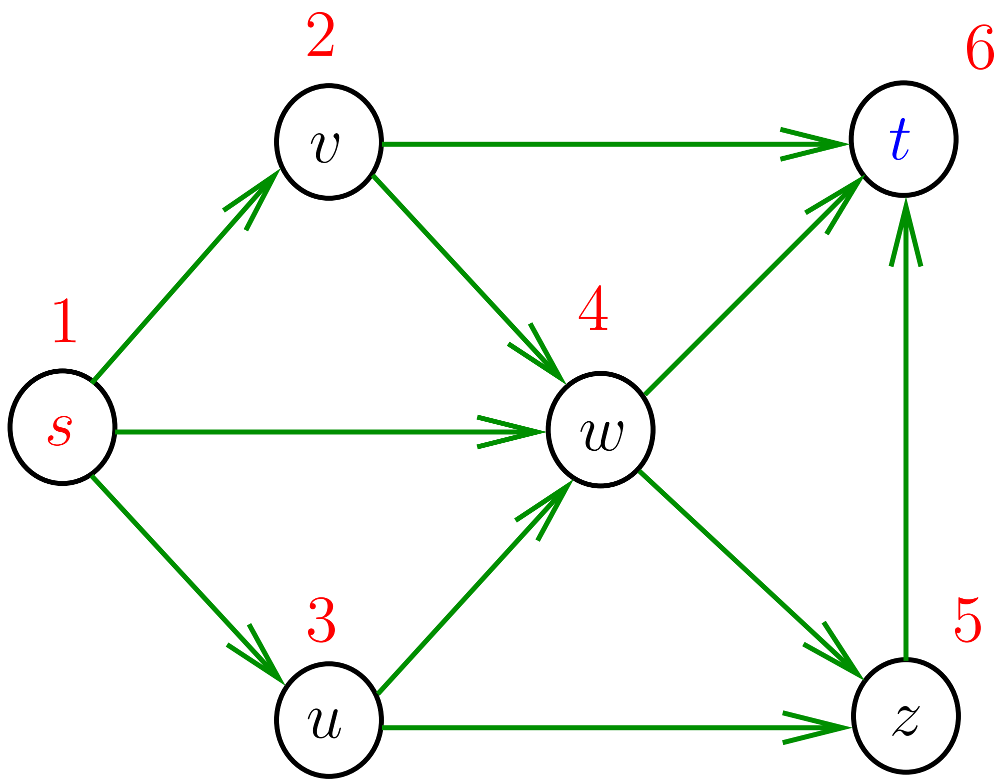
Entra:



# Condição de inexistência

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

Sai: uma "certa ordem" dós nós

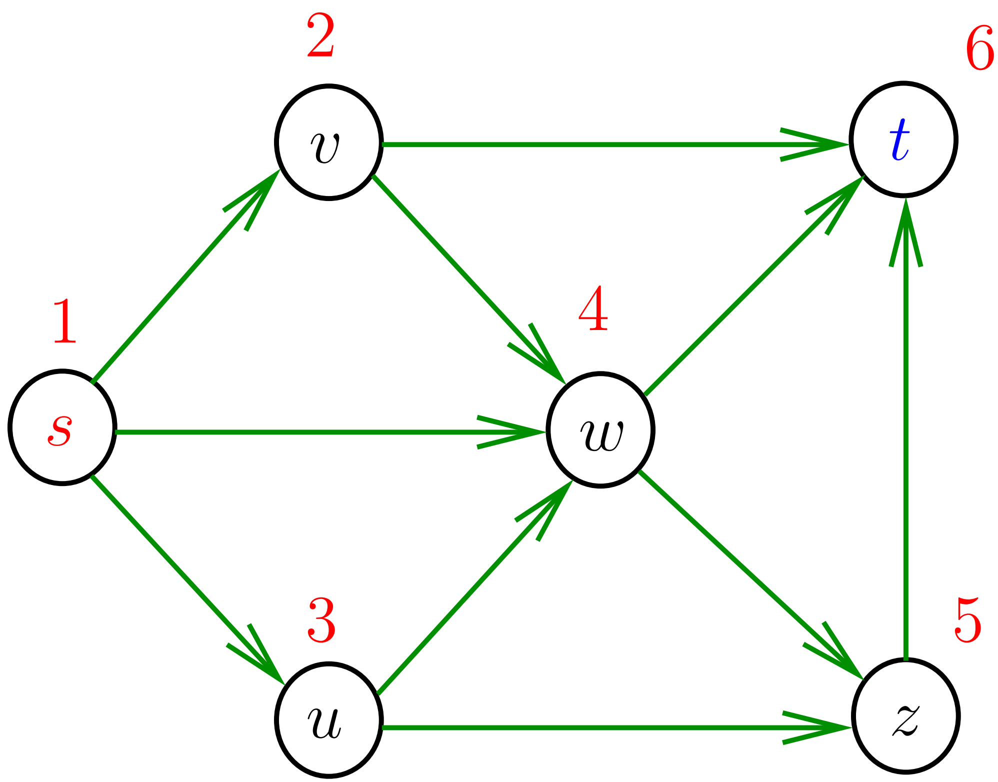


# Ordem topológica

Uma enumeração $\langle v_1, v_2, \ldots, v_n \rangle$ dos nós do grafo tal que

$$(v_p, v_q) \in A \Rightarrow p < q$$

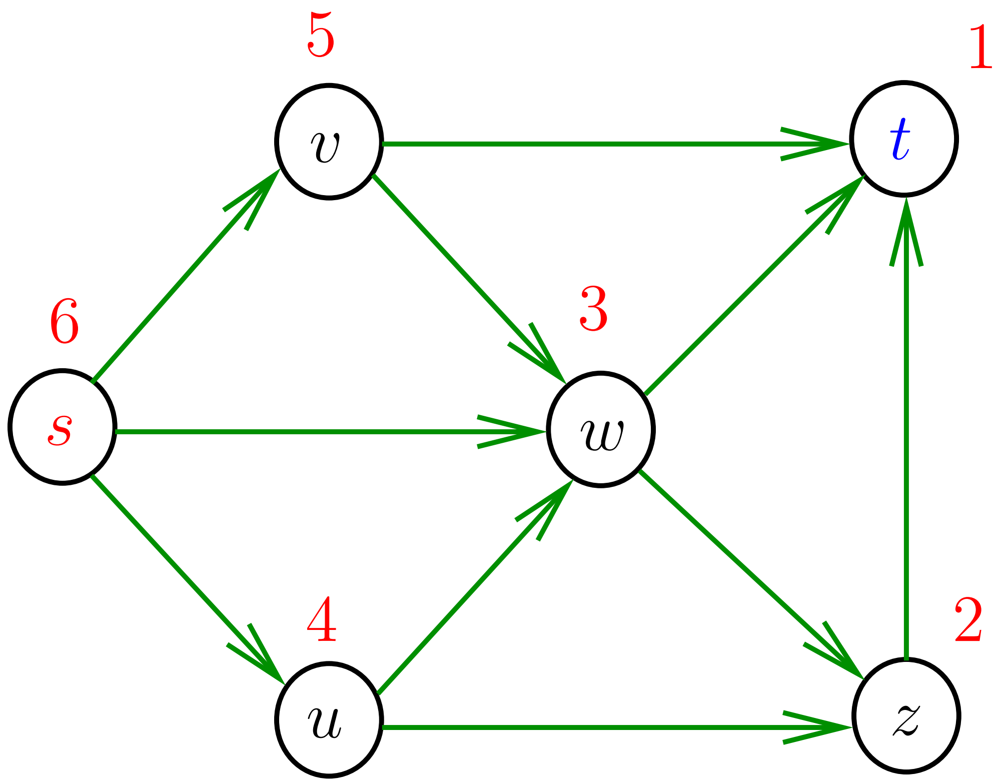
é conhecida como ordem topológica ($=$ topological order).



# Condição de inexistências, ainda ...

Um $-1$-potencial é qualquer função $y$ de $N$ em $\mathbb{Z}$ tal que

$$y(j) - y(i) \leq -1 \quad \text{para todo arco } ij.$$

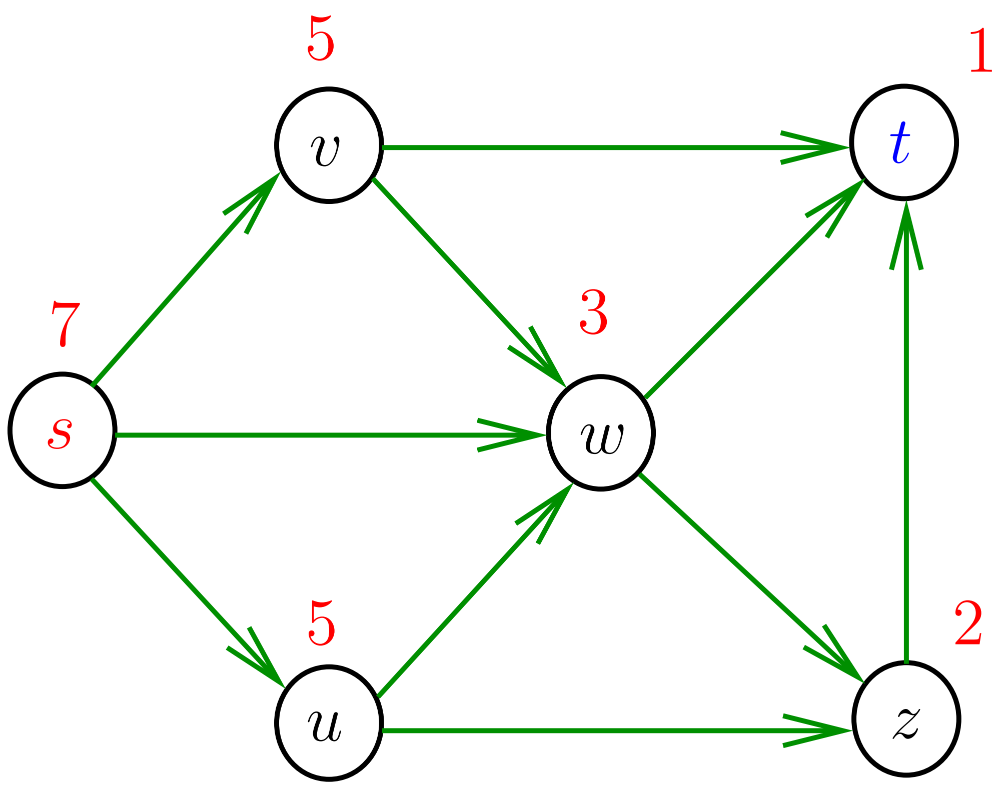


# $-1$-Potenciais e ordem topológica

Se $y$ é um $-1$-potencial então qualquer enumeração $\langle v_1, v_2, \ldots, v_n \rangle$ dos nós tal que

$$y(v_1) \geq y(v_2) \geq \cdots \geq y(v_n)$$

é uma ordem topológica.

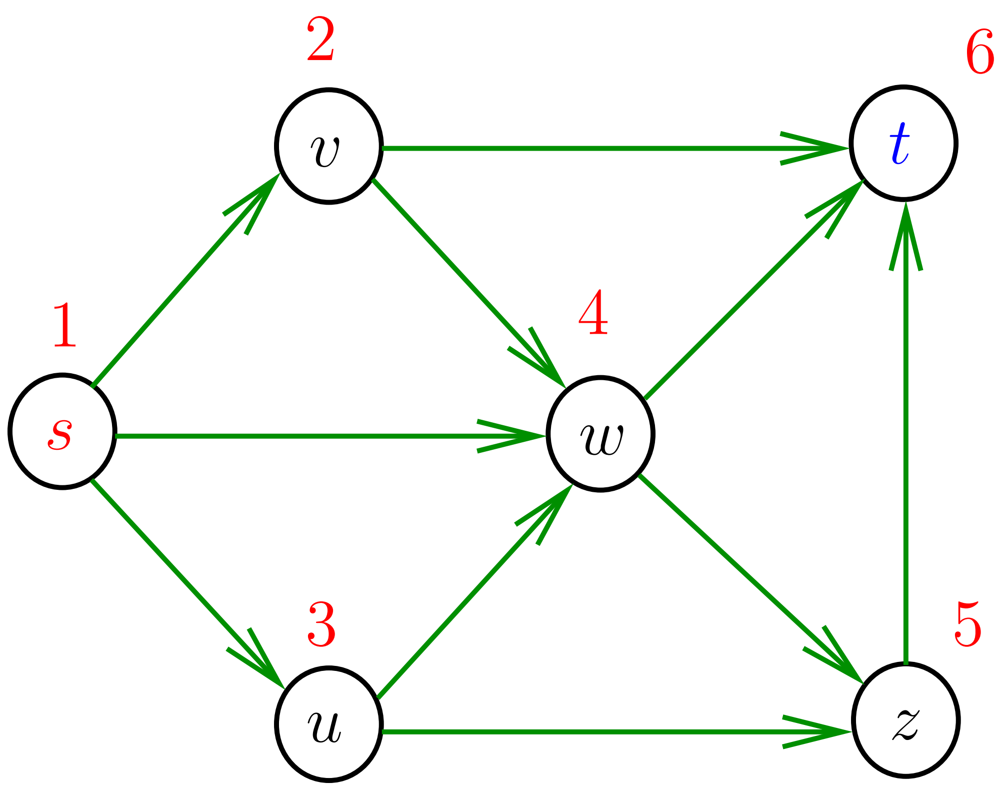


# Ordem topológica e $-1$-potenciais

Se $\langle v_1, v_2, \ldots, v_n \rangle$ é uma ordem topológica então

$$y(v_p) := n - p + 1$$

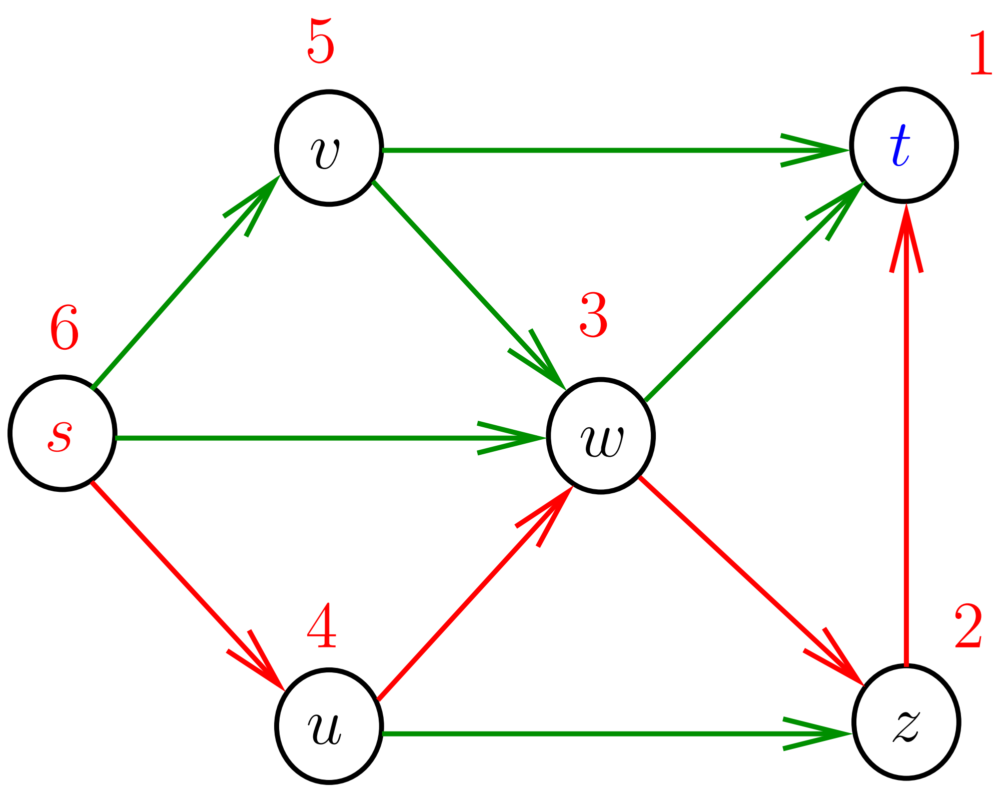
para $p = 1, \ldots, n$ é um $-1$-potencial.



# Propriedade de $-1$-Potenciais

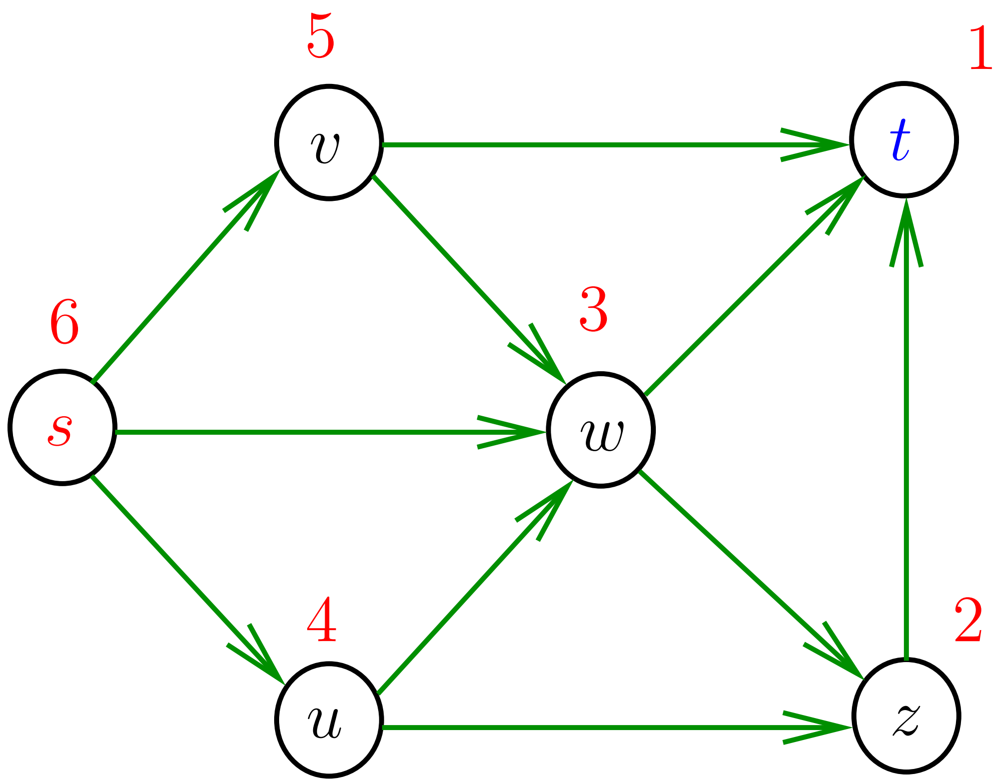
Se $y$ é um $-1$-potencial e $P$ é passeio de $s$ a $t$ então

$$y(t) - y(s) \leq -|P|.$$



# Propriedade de $-1$-Potenciais

Para mostrar que não existe um ciclo basta exibir um $-1$-potencial.



# DAG genérico

Recebe um grafo $(N, A)$ e devolve uma ciclo ou um $-1$-potencial.

DAG-GENÉRICO $(N, A)$

**para cada** $i$ em $N$ **faça**
  $y(i) \leftarrow n + 1$  $\triangleright$ $n + 1$ faz o papel de $\infty$
  $\pi(i) \leftarrow \text{NIL}$

**enquanto** $y(j) > y(i) - 1$ **para** algum $ij \in A$ **faça**
  $y(j) \leftarrow y(i) - 1$
  $\pi(j) \leftarrow i$
  **se** $y(j) \leq 0$
    **então devolva** $j$ e **pare**

**devolva** $y$

# Invariantes

Na linha 4, antes da verificação da condição "$y(j) > y(i) - 1 \ldots$" valem as seguintes invariantes:

(i1) para cada arco $pq$ no grafo de predecessores tem-se que $y(q) - y(p) \geq -1$;

(i2) para qualquer **passeio** $P$ no grafo de predecessores,

$$y(t) \geq n + 1 - |P|,$$

onde $t$ é o término de $P$.

(i2') para qualquer nó $t$ vale que

no grafo de predecessores existe uma **passeio** com término em $t$ com $\geq n + 1 - y(t)$ nós.

# Correção

- Devido a condição da linha 4 é evidente que na linha 9 o algoritmo devolve um $-1$-potencial.

- Se o algoritmo pára após a execução da linha 8 temos que, pelo invariante (i2'), existe no grafo de predecessores um passeio com pelo menos

$$n + 1 - y(j) \geq n + 1 - 0 = n + 1 \text{ nós.}$$

  Logo, o grafo de predecessores possui um ciclo. Este ciclo pode ser encontrado a partir de $j$.

Conclusão: o algoritmo faz o que promete.

# Conclusão

Um grafo é acíclico se e somente se admite um $-1$-potencial (ou, se e somente se admite uma ordem topológica).

Para qualquer grafo $(N, A)$, vale uma e apenas umas das seguintes afirmações:

- $(N, A)$ possui um ciclo;
- $(N, A)$ admite um $-1$-potencial (ordem topológica).

# Consumo de tempo

O número de execuções do bloco de linhas 4–8 é

$$\leq n^2.$$

| linha | consumo de todas as execuções da linha |
|---|---|
| 1-3 | $O(n)$ |
| 4 | $n^2 O(m) = O(n^2 m)$ |
| 5-7 | $n^2 O(1) = O(n^2)$ |
| 8 | $O(1)$ |
| 9 | $O(n)$ |
| total | $O(1) + O(n^2) + 2\,O(n) + O(n^2 m)$ |
| | $= O(n^2 m)$ |

# Conclusão

O consumo de tempo do algoritmo
DAG-GENÉRICO é $O(n^2m)$.

# Implementação do algoritmo genérico

Recebe um grafo $(N, A)$ e devolve uma ciclo ou um $-1$-potencial.

DAG $(N, A)$

**para cada** $i$ em $N$ **faça**
    $A'(i) \leftarrow A(i)$
    $y(i) \leftarrow n + 1$
    $\pi(i) \leftarrow \text{NIL}$
*rótulo* $\leftarrow 1$

**enquanto** $y(s) = n + 1$ para algum $s \in N$ **faça**
    DAG$-$DFS$(N, A', s,$ *rótulo*$)$

    **devolva** $y$

O potencial $y$ devolvido é uma numeração DFS pós-ordem.

# DAG-DFS

DAG-DFS $(N, A', s,$ *rótulo*$)$

$L \leftarrow \langle s \rangle$

**enquanto** $L \neq \langle\rangle$ **faça** $\triangleright$ $L$ funciona como uma pilha

  seja $i$ o primeiro elemento de $L$

  **se** $A'(i) \neq \emptyset$ **então**

    retire um arco $ij$ de $A'(i)$

    **se** $y(j) = n + 1$ **então**

      $\pi(j) \leftarrow i$

      **se** $j$ está em $L$

        **então devolva** $j$ e **pare**

        **senão** acrescente $j$ ao início de $L$

    **senão** $y(i) \leftarrow$ *rótulo*

      *rótulo* $\leftarrow$ *rótulo* $+ 1$

      retire $i$ de $L$

# Consumo de tempo

O número de execuções do bloco de linhas 8–19 é $\leq m$.

| linha | consumo de todas as execuções da linha |
|---|---|
| 1-4 | $\mathrm{O}(n)$ |
| 5 | $\mathrm{O}(1)$ |
| 6-7 | $\mathrm{O}(n)$ |
| 8-9 | $\mathrm{O}(n+m)$ |
| 10-16 | $\mathrm{O}(m)$ |
| 17-18 | $\mathrm{O}(n)$ |
| 19 | $\mathrm{O}(n)$ |
| total | $\mathrm{O}(1) + 4\,\mathrm{O}(n) + \mathrm{O}(n+m)$ $= \mathrm{O}(n+m)$ |

# Conclusão

$\boxed{\text{O consumo de tempo do algoritmo DAG é } \mathrm{O}(m+n).}$