# Ordenação topológica

CLRS 22 Elementary Graph Algorithms

CLRS 22.3 e 22.4

# Circuitos em digrafos

Digrafo: grafo dirigido

# Circuitos em digrafos

Digrafo: grafo dirigido

Como detectar se há um circuito (dirigido) em um digrafo?

# Circuitos em digrafos

Digrafo: grafo dirigido

Como detectar se há um circuito (dirigido) em um digrafo?

Consegue fazer usando uma busca em largura?
Consegue fazer usando uma busca em profundidade?

# Circuitos em digrafos

Digrafo: grafo dirigido

Como detectar se há um circuito (dirigido) em um digrafo?

Consegue fazer usando uma busca em largura?
Consegue fazer usando uma busca em profundidade?

Faça uma busca em profundidade
e verifique se existe algum arco de retorno!

# Busca em profundidade - elementos

Vértice branco: ainda não descoberto

Vértice cinzento: descoberto, em processamento

Vértice preto: processado

DFS termina descrevendo via $\pi$ uma floresta DF (ou BP).

$u.\pi$: predecessor ou pai de $u$ na floresta DF

# Busca em profundidade - elementos

Vértice branco: ainda não descoberto

Vértice cinzento: descoberto, em processamento

Vértice preto: processado

DFS termina descrevendo via $\pi$ uma floresta DF (ou BP).

$u.\pi$: predecessor ou pai de $u$ na floresta DF

Faz duas marcas de tempo:

$u.d$: momento da descoberta de $u$
$u.f$: momento da finalização de $u$

$u$ é branco antes de $u.d$,
cinzento entre $u.d$ e $u.f$,
preto depois de $u.f$.

# Busca em profundidade

```
DFS(G)
 1  para cada u ∈ V(G) faça
 2       u.cor ← branco        u.π ← nil
 3  tempo ← 0
 4  para cada u ∈ V(G) faça
 5       se u.cor = branco
 6            então DFS-Visit(u)
```

```
DFS-Visit(G, u)
 1  u.cor ← cinzento    u.d ← tempo    tempo ← tempo + 1
 3  para cada v ∈ u.Estrela faça
 4       se v.cor = branco
 5            então v.π ← u
 6                 DFS-Visit(G, v)
 7  u.cor ← preto
 8  u.f ← tempo          tempo ← tempo + 1
```

# Consumo de tempo

Cada vértice é descoberto uma única vez, pois é branco
e, ao ser descoberto, passa a ser cinzento, e depois preto.

A lista de adjacência de cada vértice descoberto
é percorrida uma única vez.

Logo o consumo de tempo é $O(n+m)$,
onde $n = |V|$ e $m = |E|$, pois
a inicialização custa $\Theta(n)$ e a soma do tamanho
das listas de adjacências percorridas é $O(m)$.

O consumo de tempo de uma DFS é
linear no tamanho do grafo.

# Propriedades da DFS de um grafo

Mesma estrutura que sequência válida de parênteses.

Teorema: *Para vértices $u$ e $v$ com $u.d < v.d$, exatamente uma das duas condições abaixo valem na floresta resultante:*

a) *Os intervalos $[u.d, u.f]$ e $[v.d, v.f]$ são disjuntos, e nem $u$ é descendente de $v$, nem $v$ é descendente de $u$.*

b) *O intervalo $[v.d, v.f]$ está contido no intervalo $[u.d, u.f]$, e $v$ é descendente de $u$.*

Corolário: *O vértice $v$ é um descendente próprio do vértice $u$ na floresta resultante da DFS sse $u.d < v.d < v.f < u.f$.*

# Classificação dos arcos

Quatro tipos de arcos derivados de uma DFS:

a) Arcos da arborecência: arcos da floresta DF.

b) Arcos de retorno: de um vértice para um ascendente na floresta DF.

c) Arcos de avanço: de um vértice para um descendente na floresta DF.

d) Arcos cruzados: todos os outros arcos.

# Classificação dos arcos

Quatro tipos de arcos derivados de uma DFS:

a) Arcos da arborecência: arcos da floresta DF.

b) Arcos de retorno: de um vértice para um ascendente na floresta DF.

c) Arcos de avanço: de um vértice para um descendente na floresta DF.

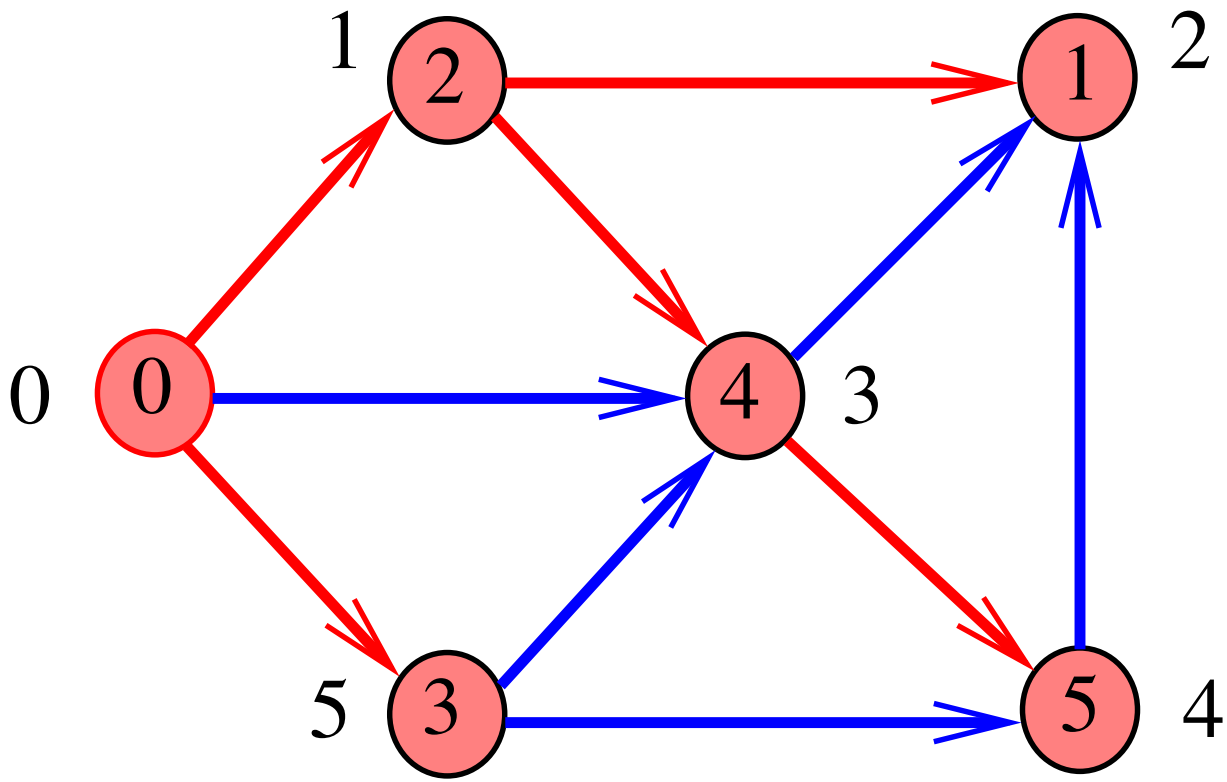
d) Arcos cruzados: todos os outros arcos.

Como identificar o tipo do arco durante a busca?

# Arcos da arborecência

Arcos da arborescência são os arcos *v*-*w* que DFS percorre para visitar *w* pela primeira vez.

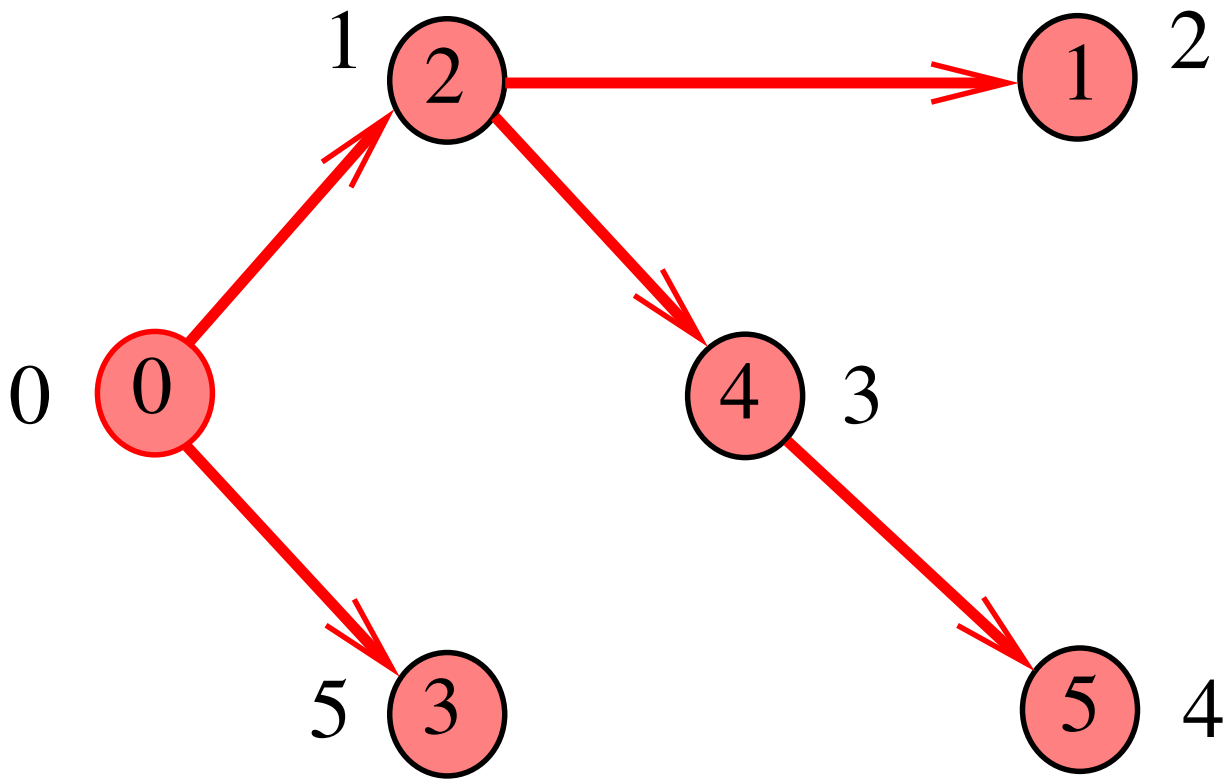
Exemplo: arcos em vermelho são arcos da arborescência



# Arcos da arborecência

Arcos da arborescência são os arcos *v*-*w* que DFS percorre para visitar *w* pela primeira vez.

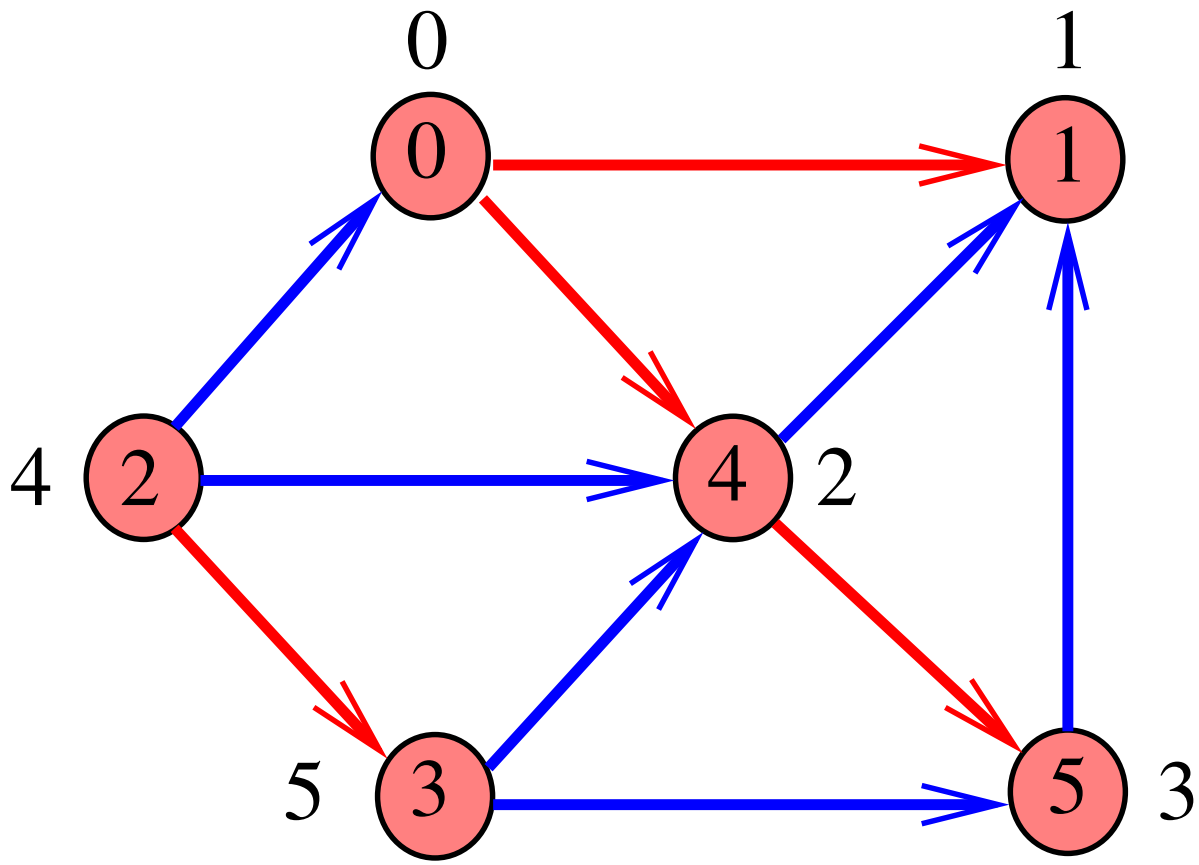
Exemplo: arcos em vermelho são arcos da arborescência



# Floresta DF

Conjunto de arborescências é
a floresta da busca em profundidade (= *floresta DF*).

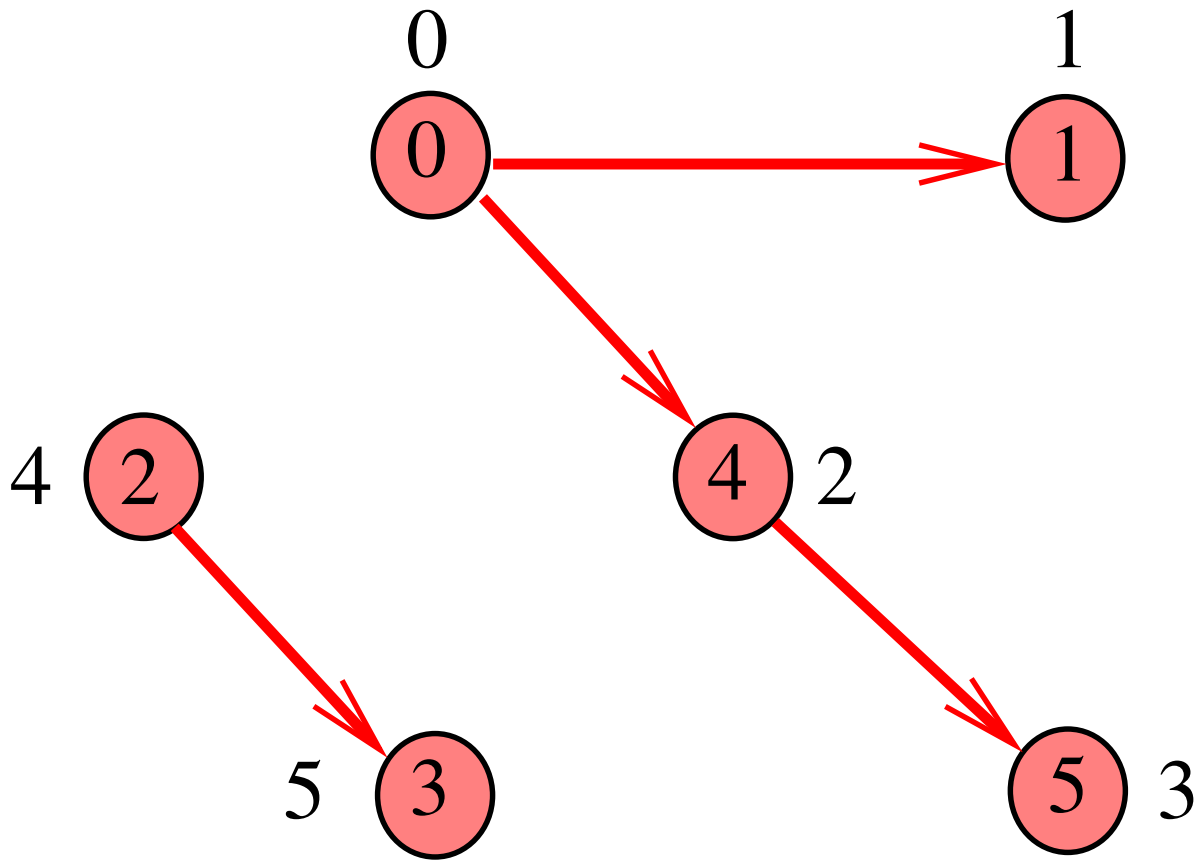
Exemplo: arcos em vermelho formam a floresta DF



# Floresta DF

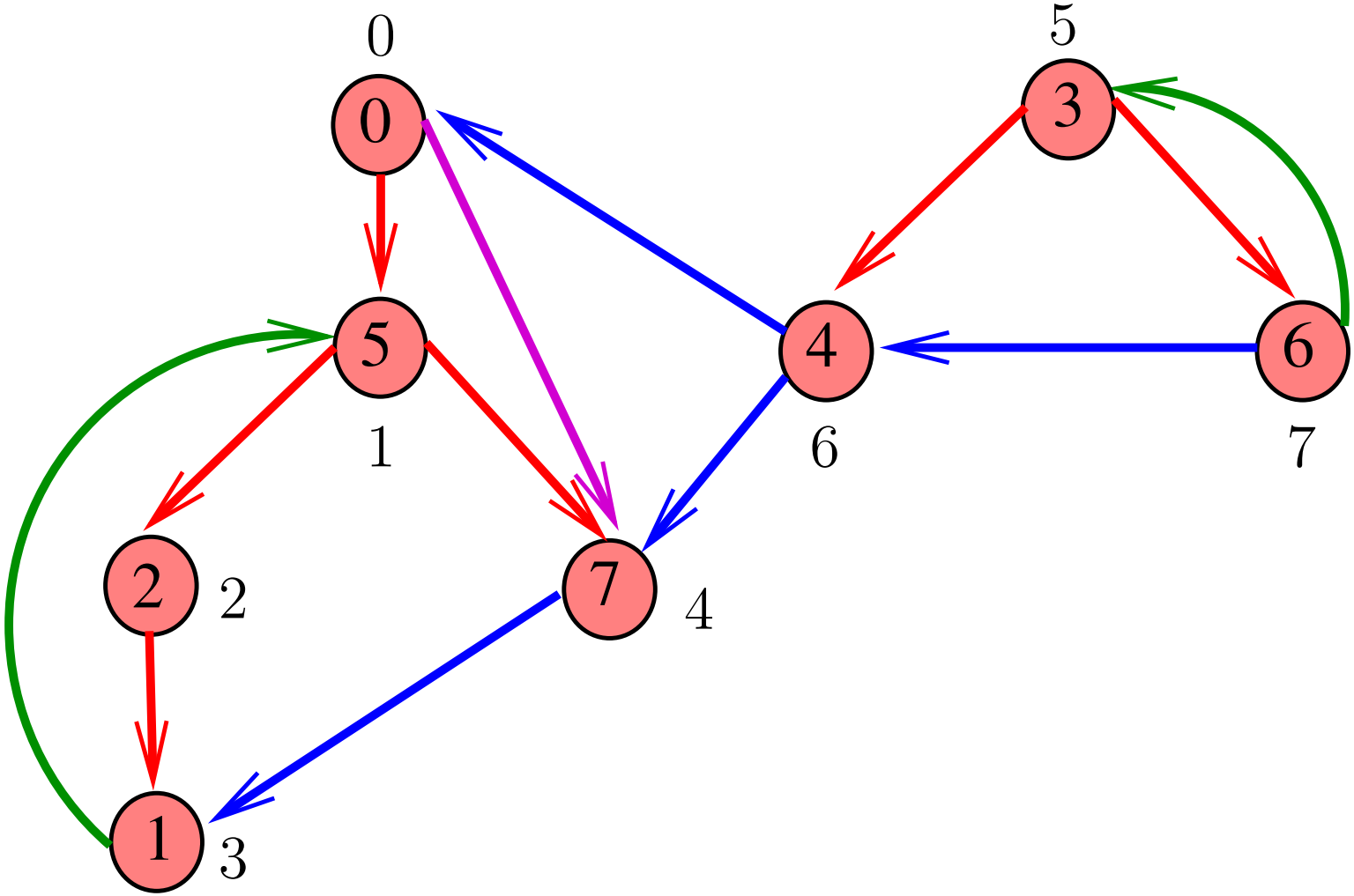
Conjunto de arborescências é
a floresta da busca em profundidade (= *floresta DF*).

Exemplo: arcos em vermelho formam a floresta DF



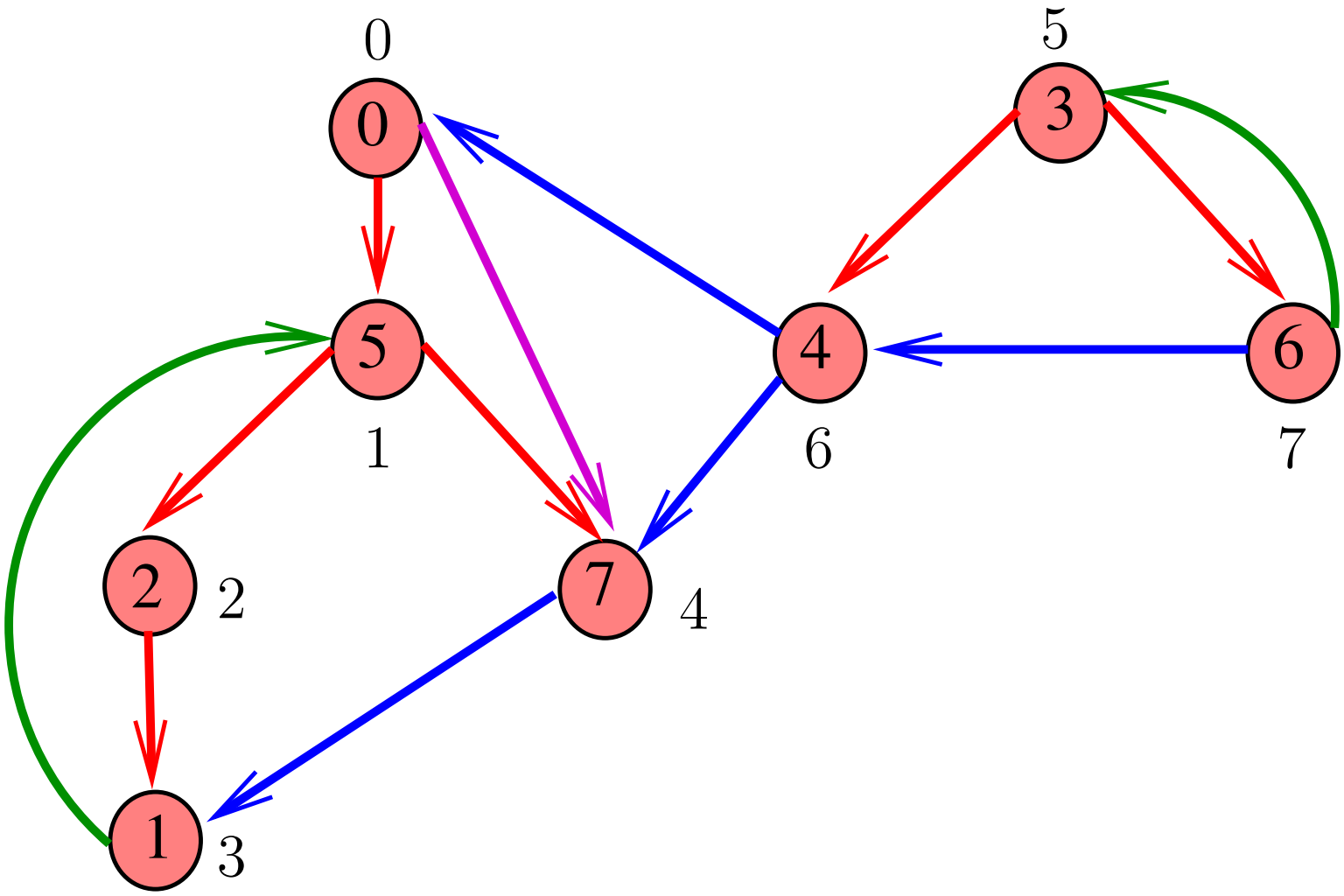
# Arcos de arborescência

*v*-*w* é arco de arborescência se foi usado para visitar *w* pela primeira vez (arcos **vermelhos**).



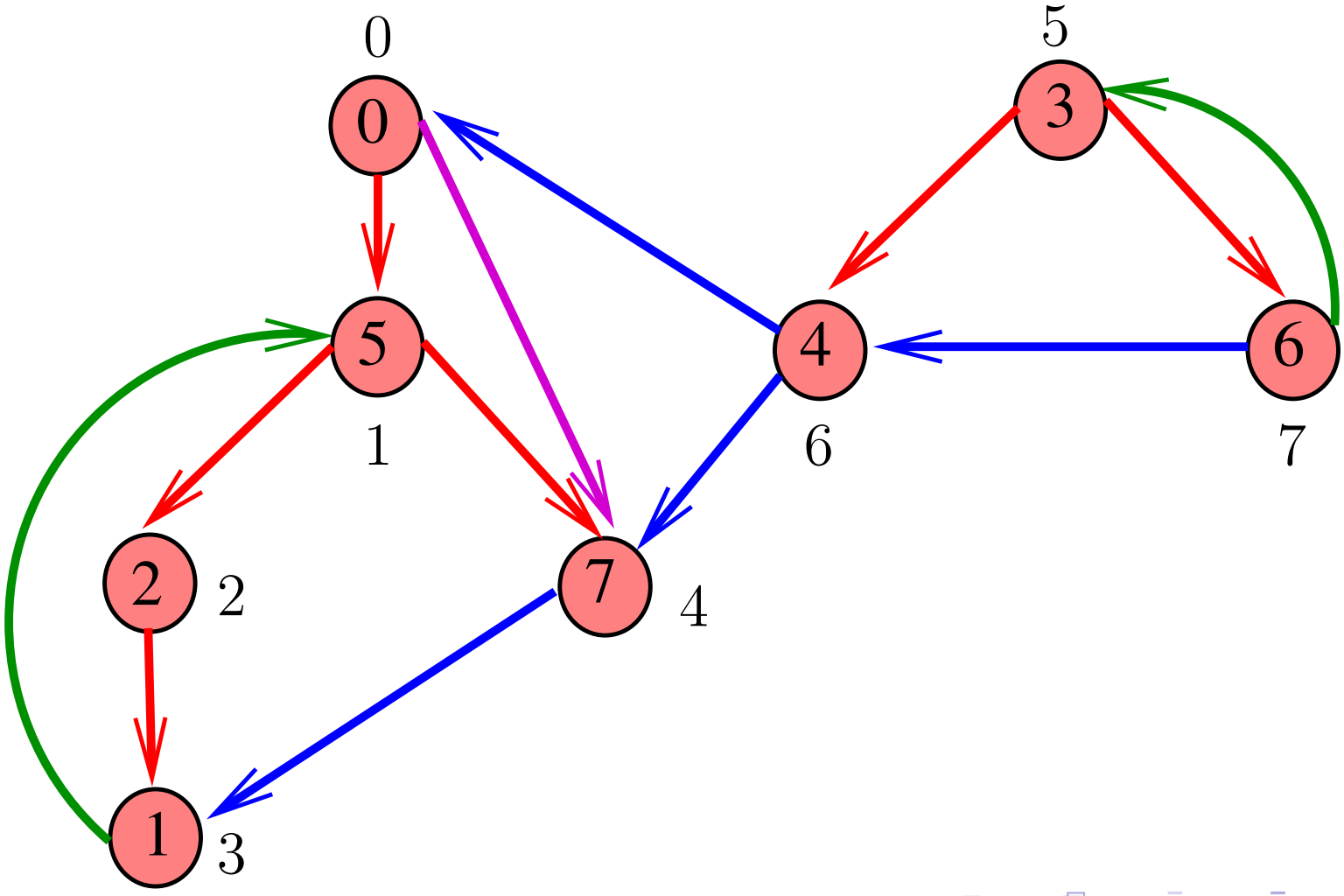
# Arcos de retorno

*v*-*w* é **arco de retorno** se *w* é ancestral de *v* (arcos **verdes**).



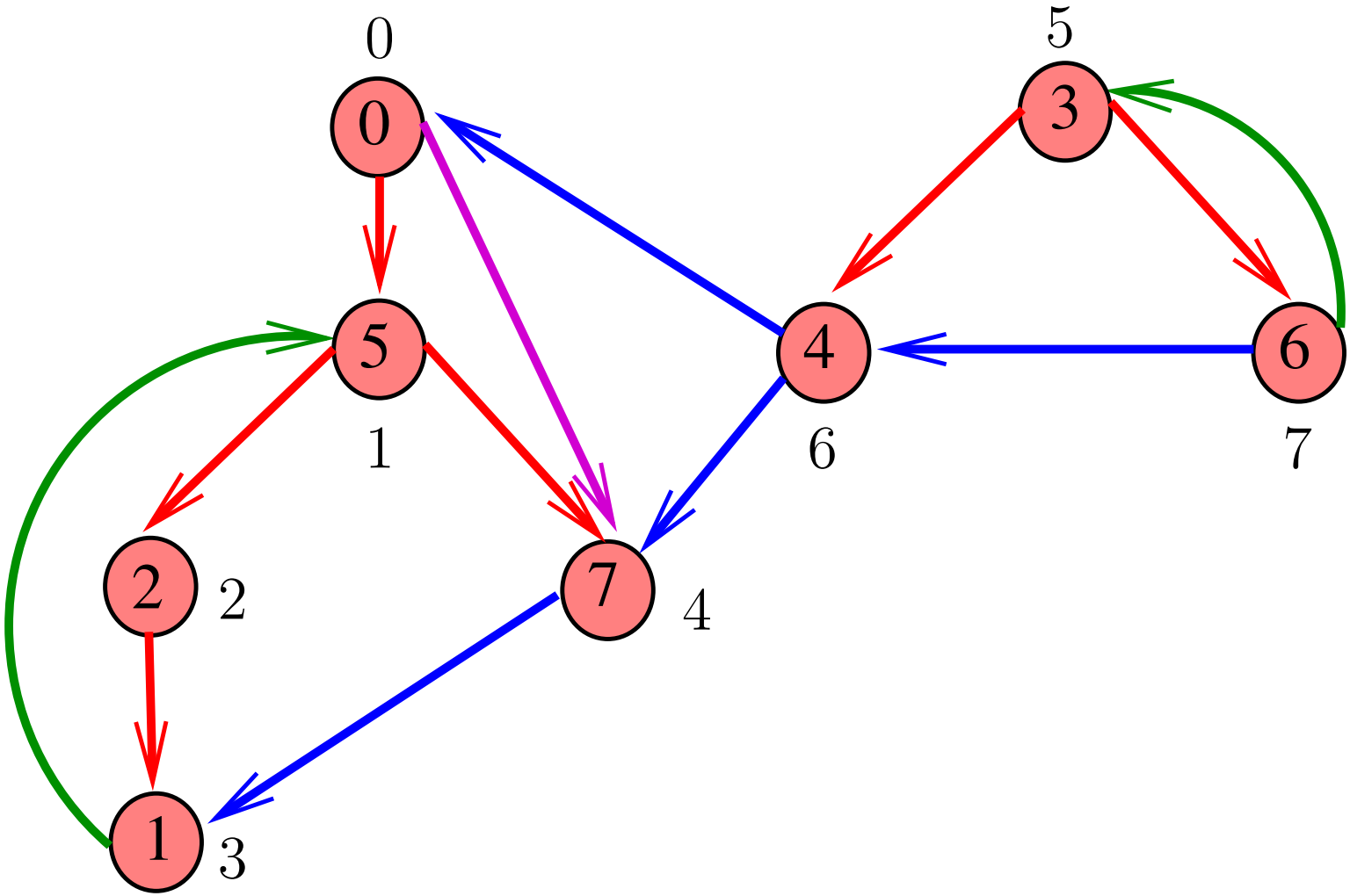
# Arcos de avanço

$v$-$w$ é **de avanço** se $w$ é descendente de $v$,
mas não é filho de $v$ (arco **roxo**).

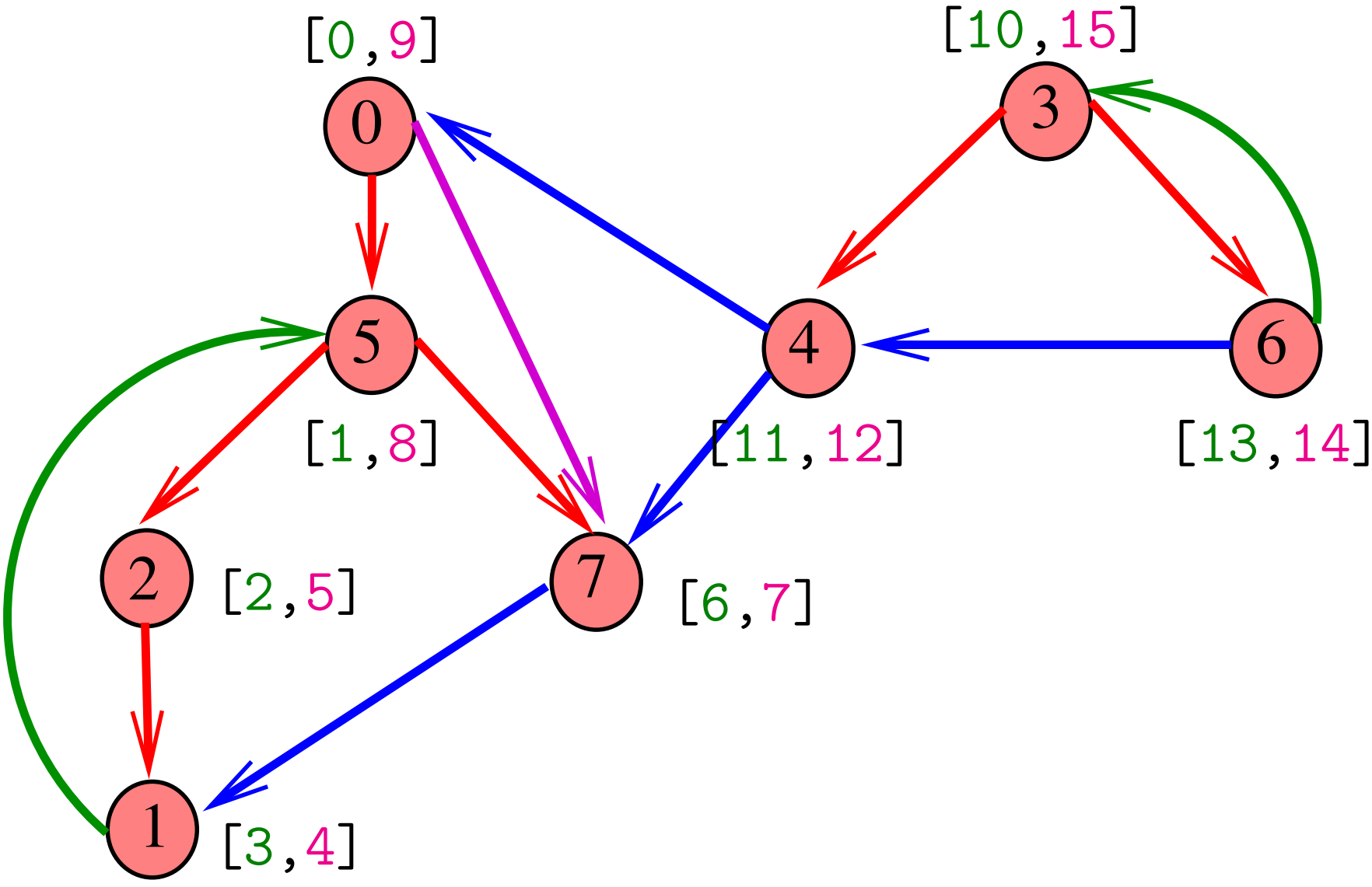


# Arcos cruzados

*v*-*w* é **arco cruzado** se *w* não é ancestral nem descendente de *v* (arcos **azuis**).

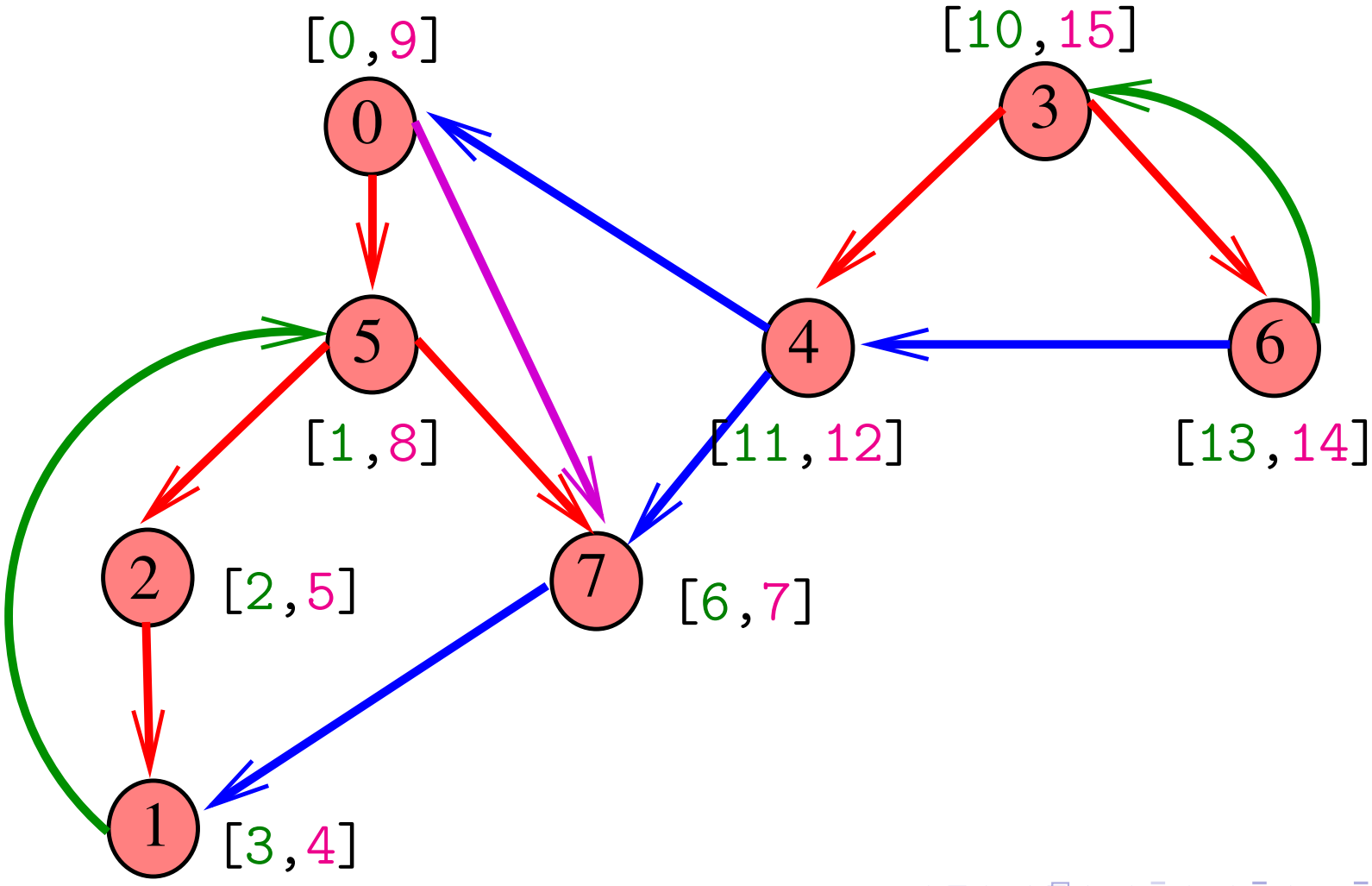


# Classificação dos arcos



# Arcos de arborescência ou de avanço

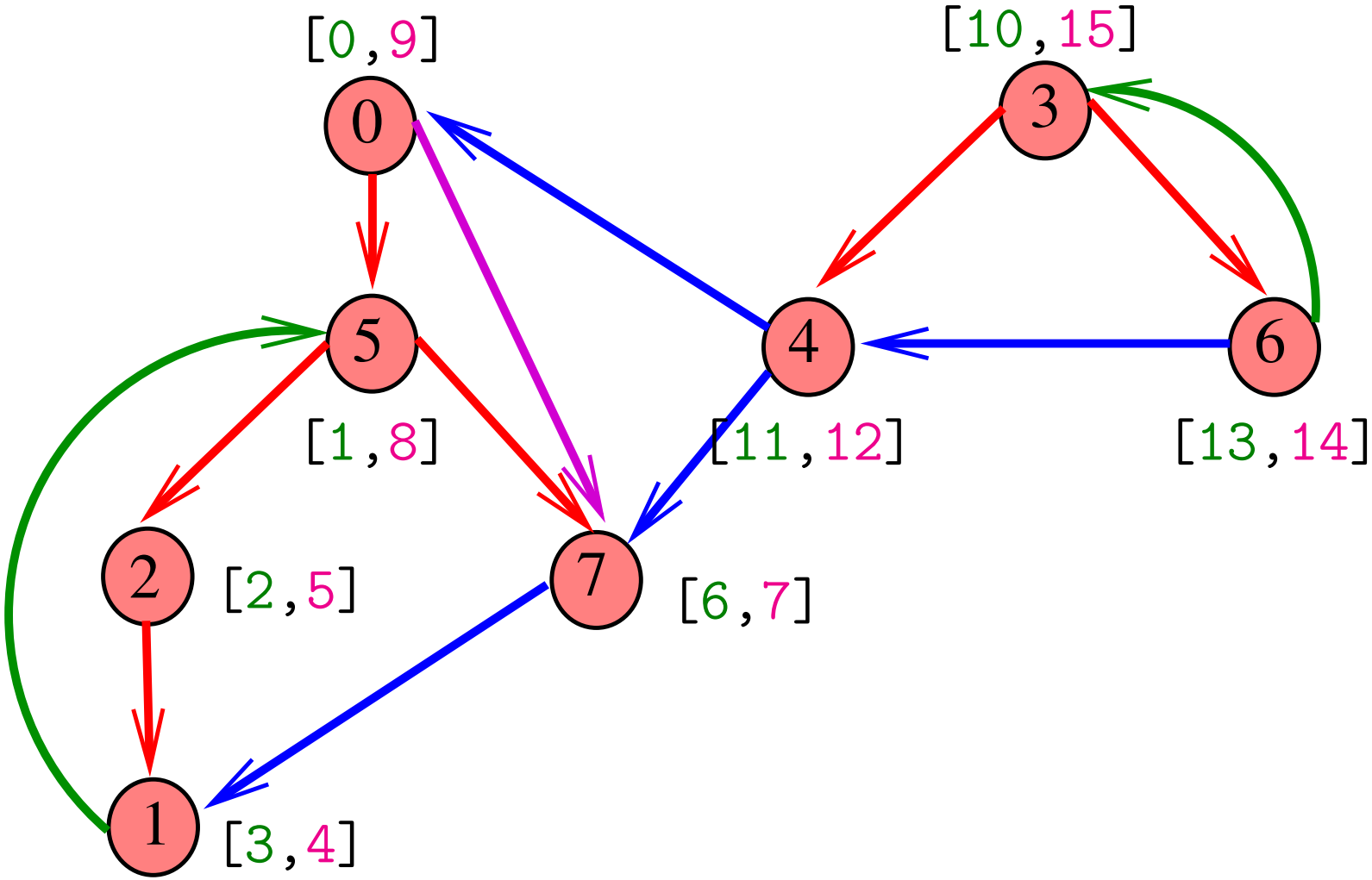
$v$-$w$ é **arco de arborescência** ou **de avanço**
se e somente se $\mathtt{d}[v] < \mathtt{d}[w] < \mathtt{f}[w] < \mathtt{f}[v]$.



# Arcos de retorno

$v$-$w$ é **arco de retorno** se e somente se

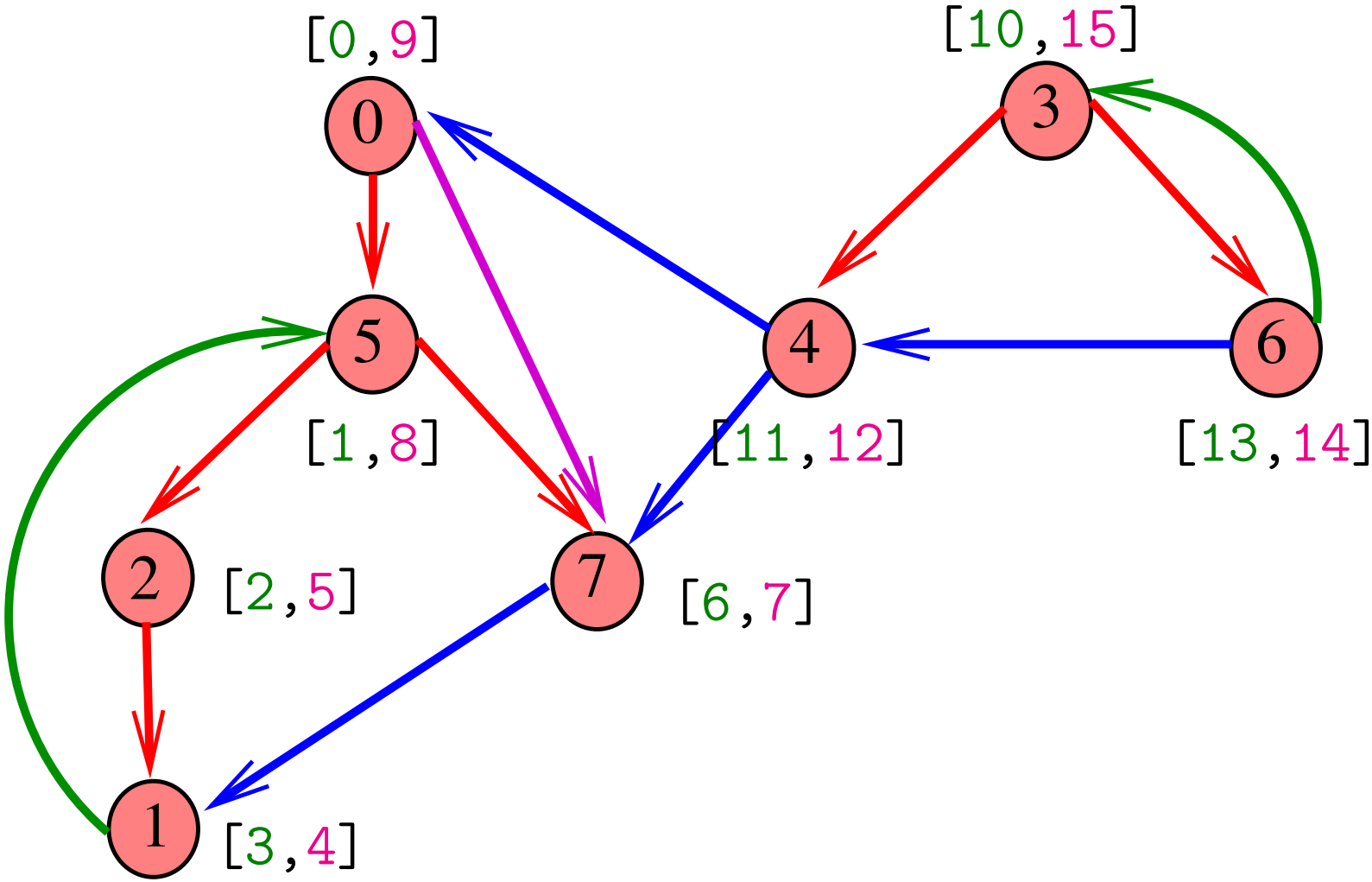
$$\mathtt{d}[w] < \mathtt{d}[v] < \mathtt{f}[v] < \mathtt{f}[w].$$



# Arcos cruzados

$v$-$w$ é arco **cruzado** se e somente se

$$\mathtt{d}[w] < \mathtt{f}[w] < \mathtt{d}[v] < \mathtt{f}[v].$$



# Conclusões

$v$-$w$ é:

- **arco de arborescência** se e somente se $\mathtt{d}[v] < \mathtt{d}[w] < \mathtt{f}[w] < \mathtt{f}[v]$ e $\pi[w] = v$;
- **arco de avanço** se e somente se $\mathtt{d}[v] < \mathtt{d}[w] < \mathtt{f}[w] < \mathtt{f}[v]$ e $\pi[w] \neq v$;
- **arco de retorno** se e somente se $\mathtt{d}[w] < \mathtt{d}[v] < \mathtt{f}[v] < \mathtt{f}[w]$;
- **arco cruzado** se e somente se $\mathtt{d}[w] < \mathtt{f}[w] < \mathtt{d}[v] < \mathtt{f}[v]$.

# Conclusões

$v$-$w$ é:

- **arco de arborescência** se e somente se $\mathtt{d}[v] < \mathtt{d}[w] < \mathtt{f}[w] < \mathtt{f}[v]$ e $\pi[w] = v$;
- **arco de avanço** se e somente se $\mathtt{d}[v] < \mathtt{d}[w] < \mathtt{f}[w] < \mathtt{f}[v]$ e $\pi[w] \neq v$;
- **arco de retorno** se e somente se $\mathtt{d}[w] < \mathtt{d}[v] < \mathtt{f}[v] < \mathtt{f}[w]$;
- **arco cruzado** se e somente se $\mathtt{d}[w] < \mathtt{f}[w] < \mathtt{d}[v] < \mathtt{f}[v]$.

Com essa informação, sabem alterar a DFS
para detectar se há circuito (dirigido) num digrafo?

# Como é um digrafo sem circuitos (dirigidos)?

# Como é um digrafo sem circuitos (dirigidos)?

Seja $G = (V, A)$ um digrafo sem circuitos dirigidos (DAG).

# Como é um digrafo sem circuitos (dirigidos)?

Seja $G = (V, A)$ um digrafo sem circuitos dirigidos (DAG).

Podemos ordenar os vértices de $G$ de modo que
todos os arcos vão da esquerda para a direita.

# Como é um digrafo sem circuitos (dirigidos)?

Seja $G = (V, A)$ um digrafo sem circuitos dirigidos (DAG).

Podemos ordenar os vértices de $G$ de modo que
todos os arcos vão da esquerda para a direita.

Uma ordem assim é chamada de ordenação topológica de $G$.

# Como é um digrafo sem circuitos (dirigidos)?

Seja $G = (V, A)$ um digrafo sem circuitos dirigidos (DAG).

Podemos ordenar os vértices de $G$ de modo que
todos os arcos vão da esquerda para a direita.

Uma ordem assim é chamada de ordenação topológica de $G$.

Como obter uma ordenação topológica de um DAG?

# Algoritmo de eliminação de fontes

Fonte: vértice que não é ponta final de nenhum arco.

Repetidamente, encontre uma fonte no grafo e a remova.

# Algoritmo de eliminação de fontes

Fonte: vértice que não é ponta final de nenhum arco.

Repetidamente, encontre uma fonte no grafo e a remova.

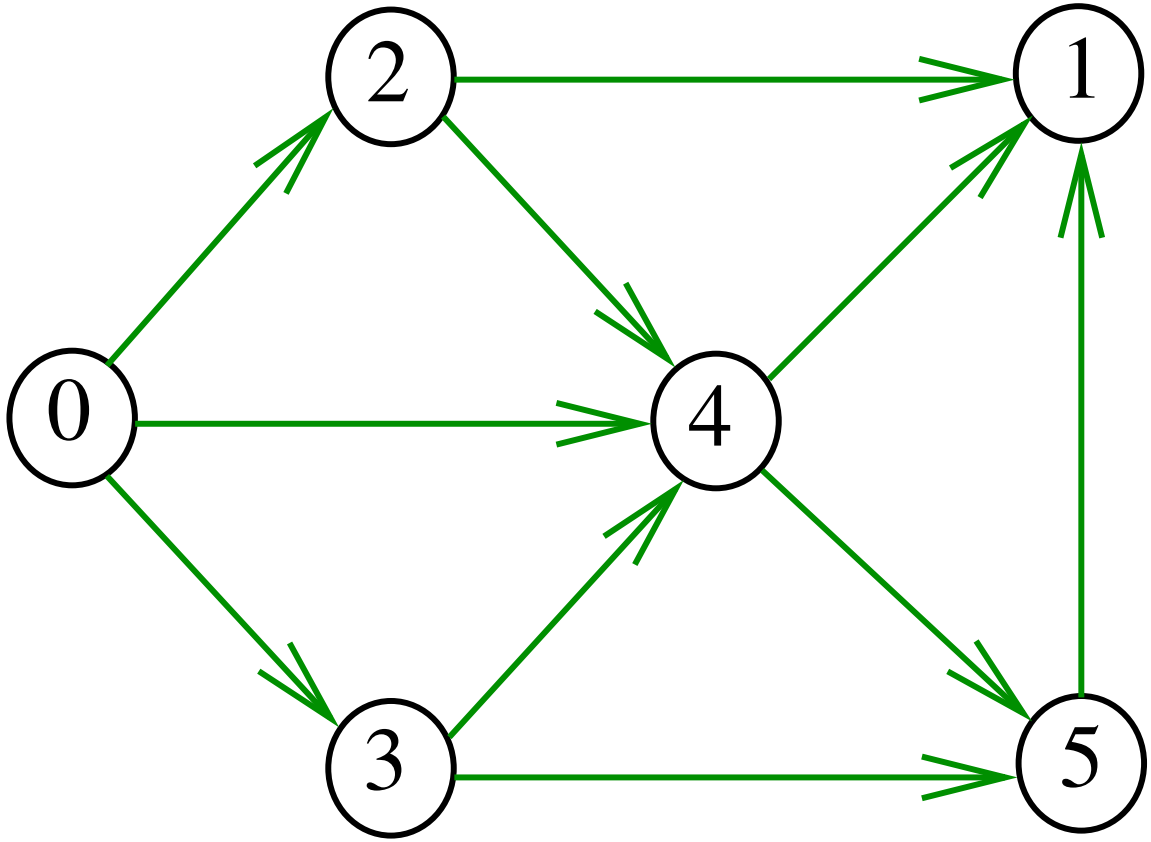
Armazena em ts$[1..i]$ uma permutação de um subconjunto do conjunto de vértices de $G$ e devolve o valor de $i$.

Se $i = n$ então ts$[0..i-1]$ é uma ordenação topológica de $G$.

Caso contrário, $G$ **não** é um DAG.

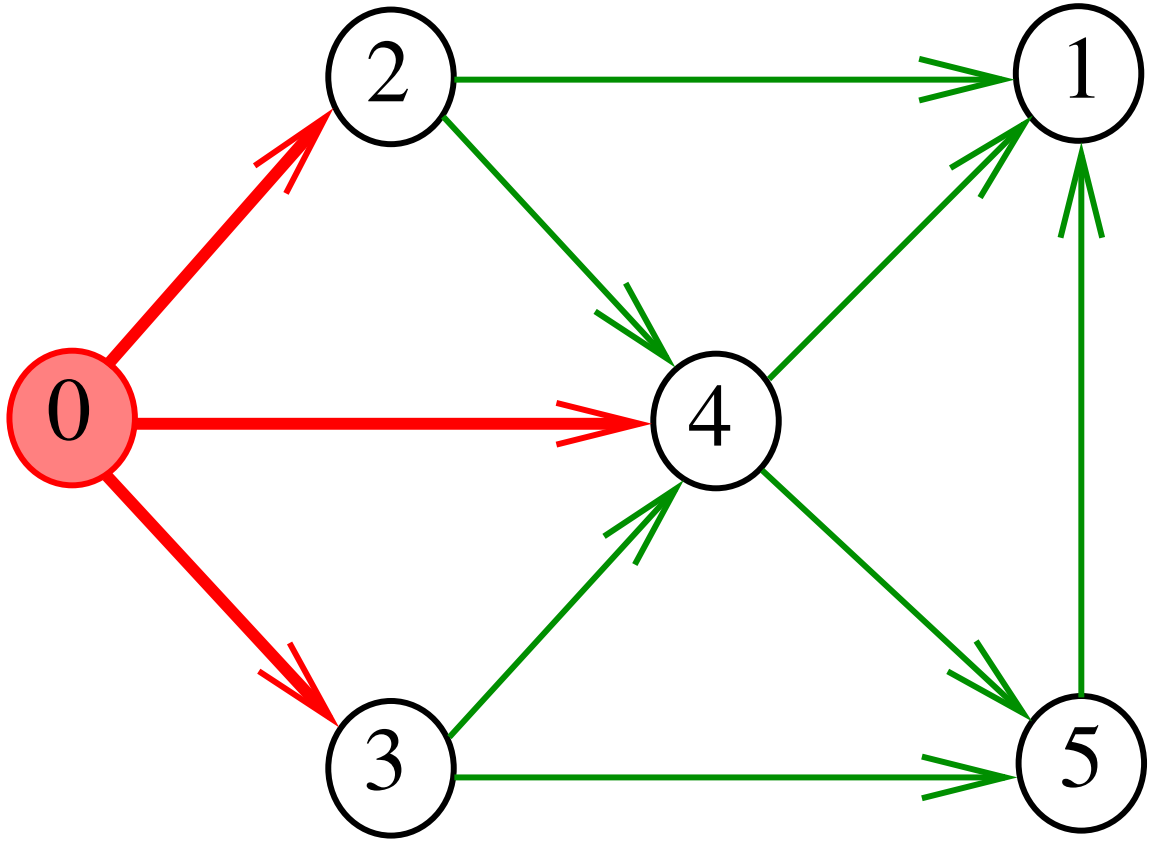
# Exemplo

| i | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| ts[i] | | | | | | |



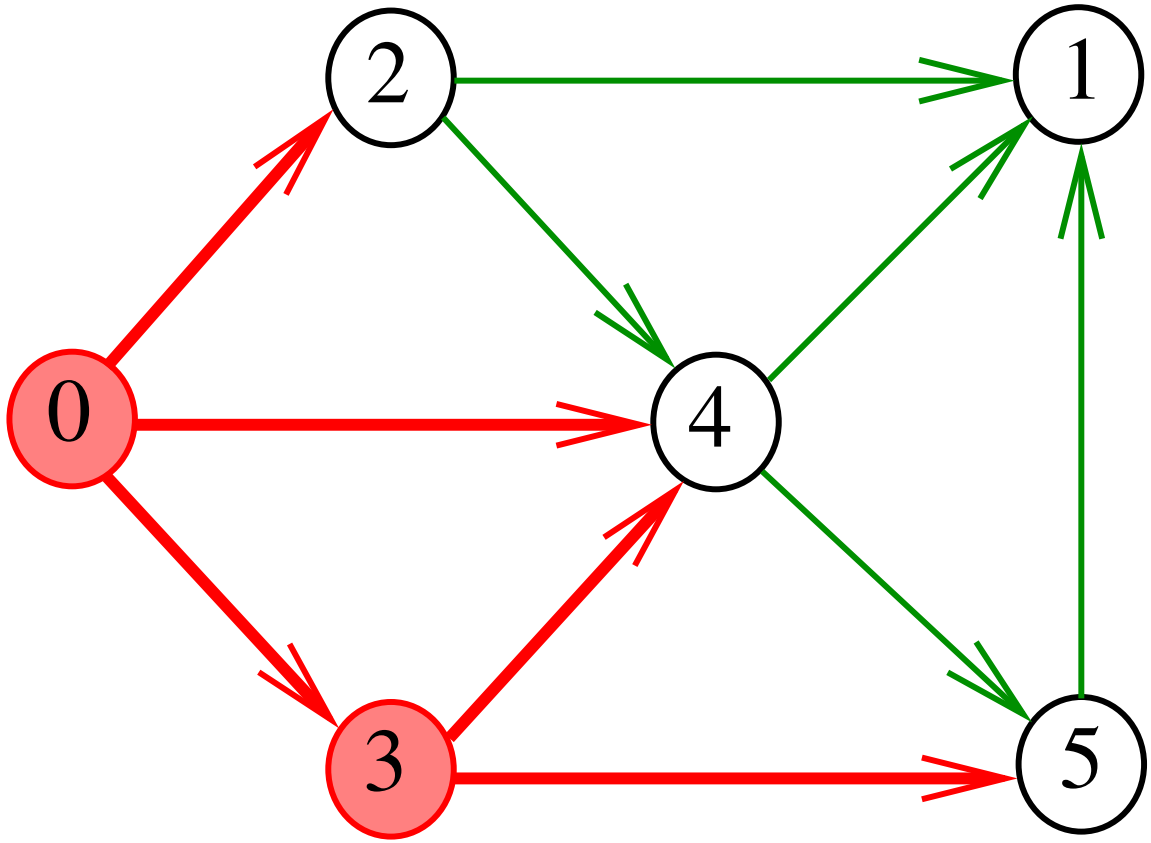
# Exemplo

| i | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| ts[i] | 0 | | | | | |



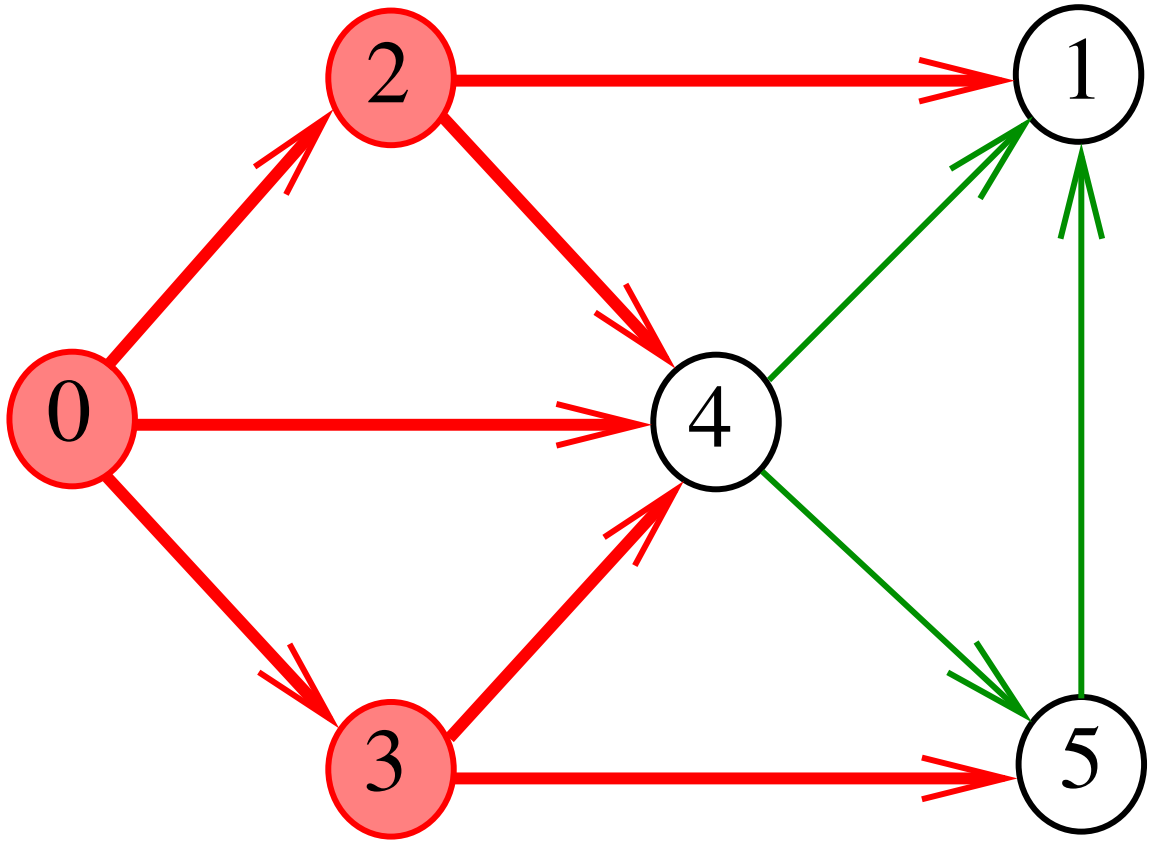
# Exemplo

| i | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| ts[i] | 0 | 3 | | | | |



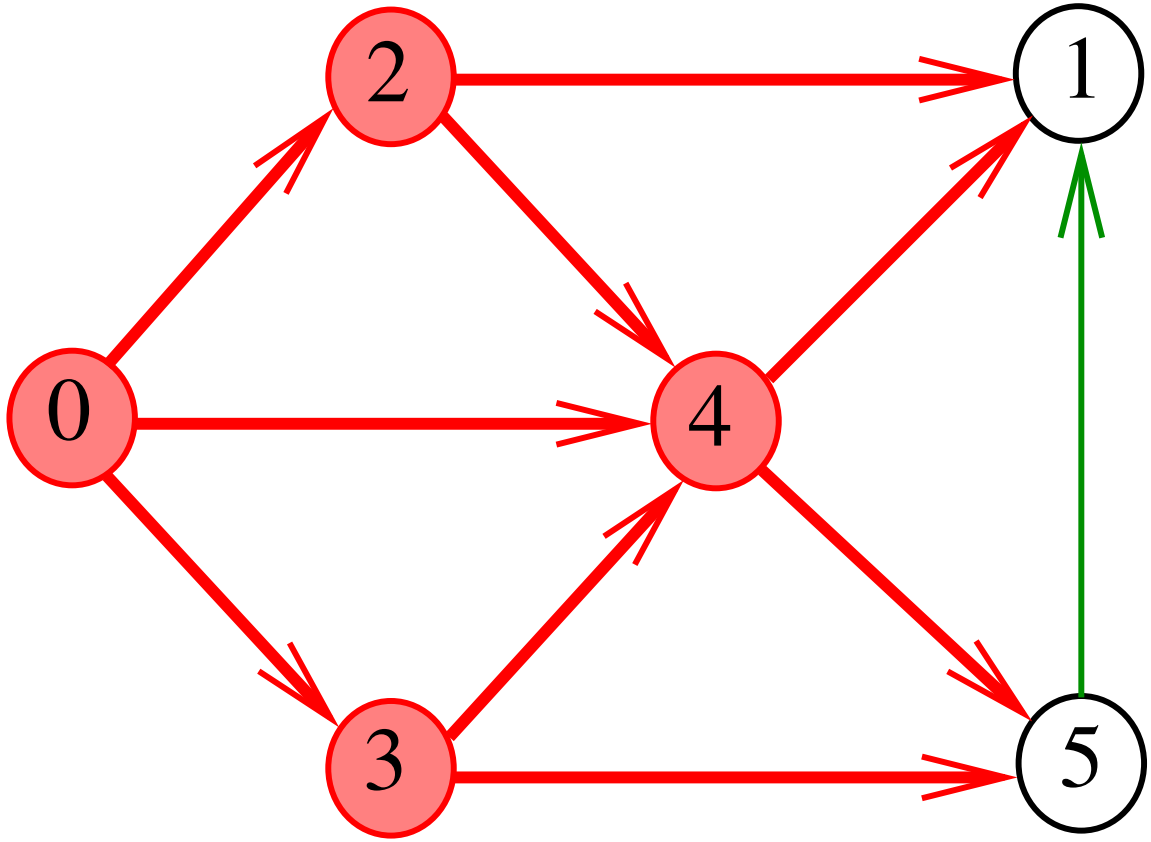
# Exemplo

| i | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| ts[i] | 0 | 3 | 2 | | | |



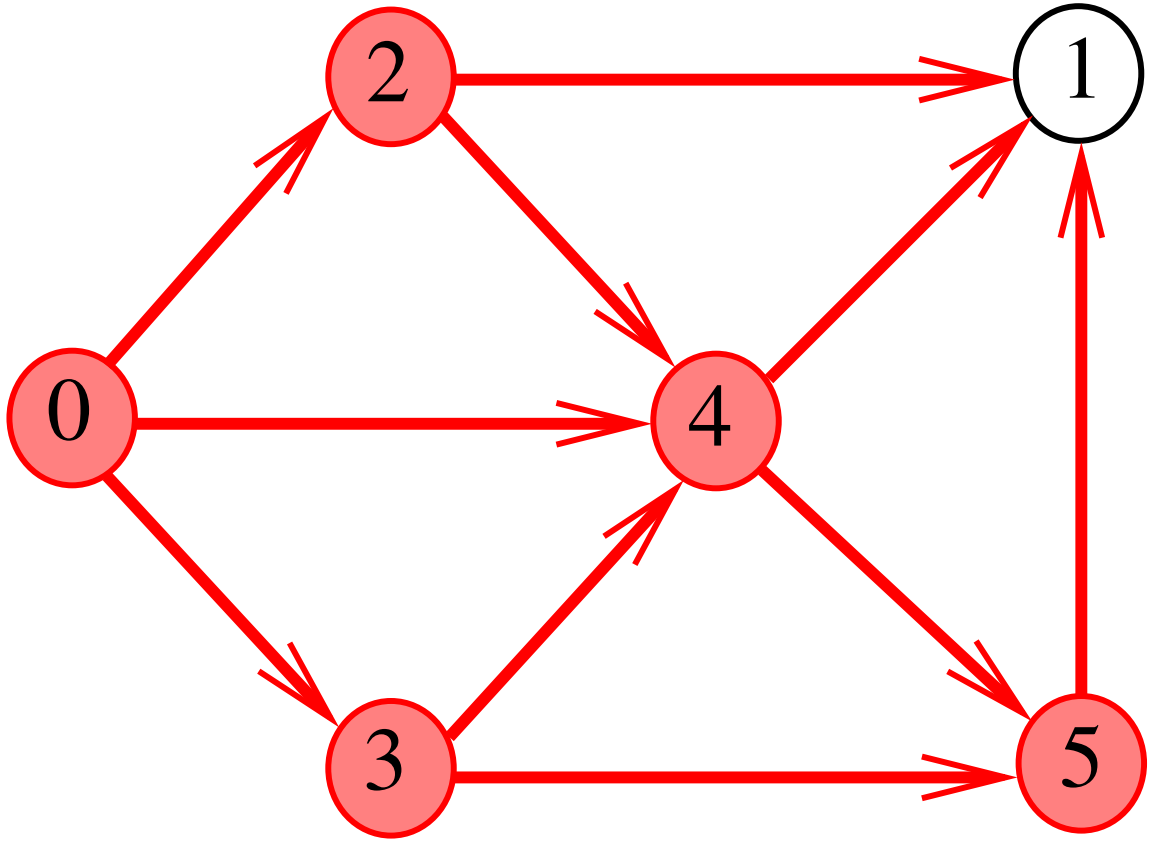
# Exemplo

| i | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| ts[i] | 0 | 3 | 2 | 4 | | |



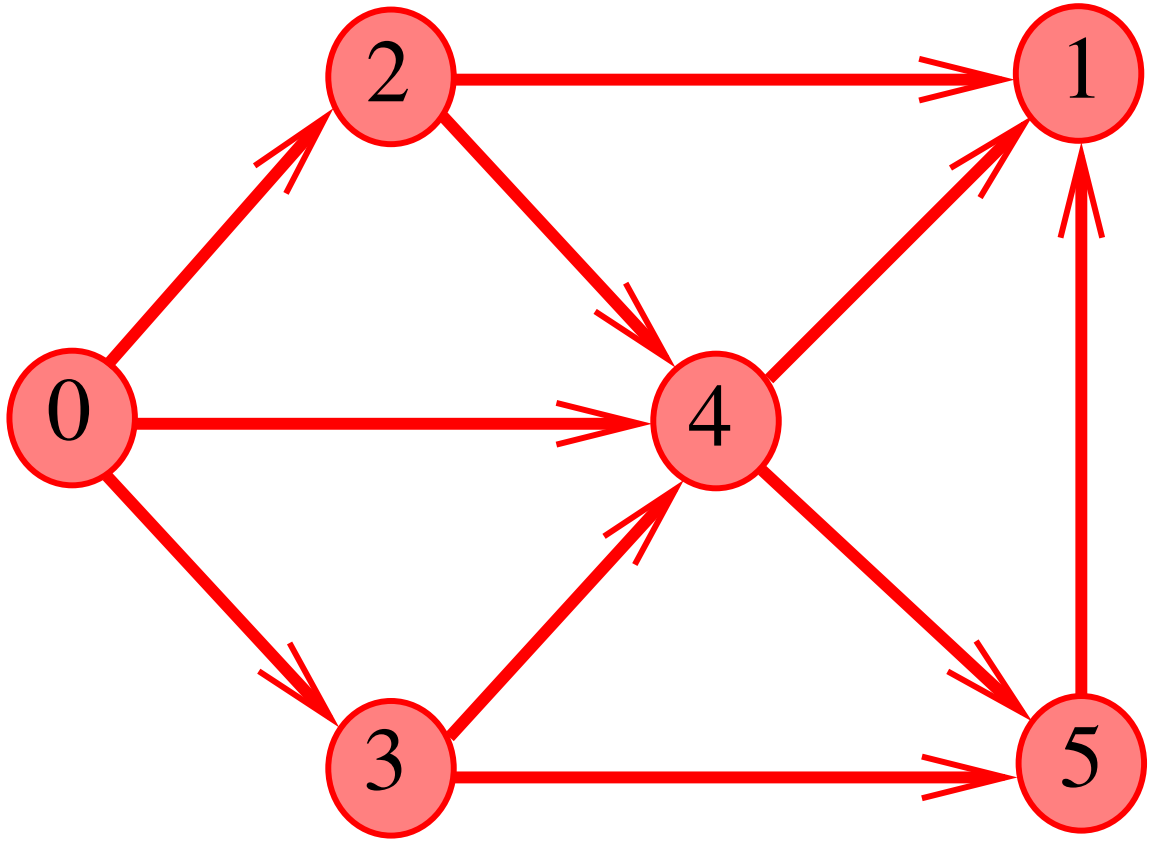
# Exemplo

| i | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| ts[i] | 0 | 3 | 2 | 4 | 5 | |



# Exemplo

| i | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| ts[i] | 0 | 3 | 2 | 4 | 5 | 1 |



# Consumo de tempo

O consumo de tempo desse algoritmo para vetor de listas de adjacência é $O(n+m)$.

O consumo de tempo desse algoritmo para matriz de adjacências é $O(n^2)$.

# Consumo de tempo

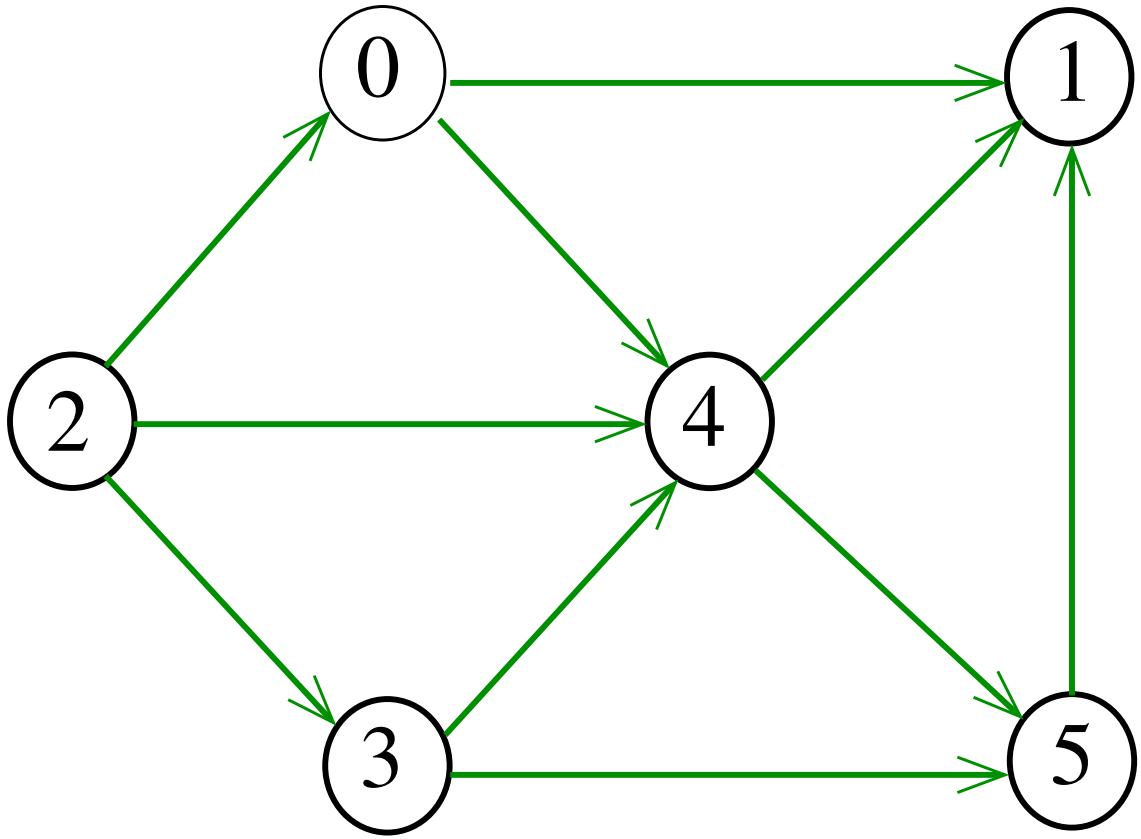
O consumo de tempo desse algoritmo para vetor de listas de adjacência é $O(n+m)$.

O consumo de tempo desse algoritmo para matriz de adjacências é $O(n^2)$.

Próxima missão:
Modificar a DFS para, dado um digrafo $G$, encontrar um circuito dirigido em $G$ ou produzir uma ordenação topológica de $G$.

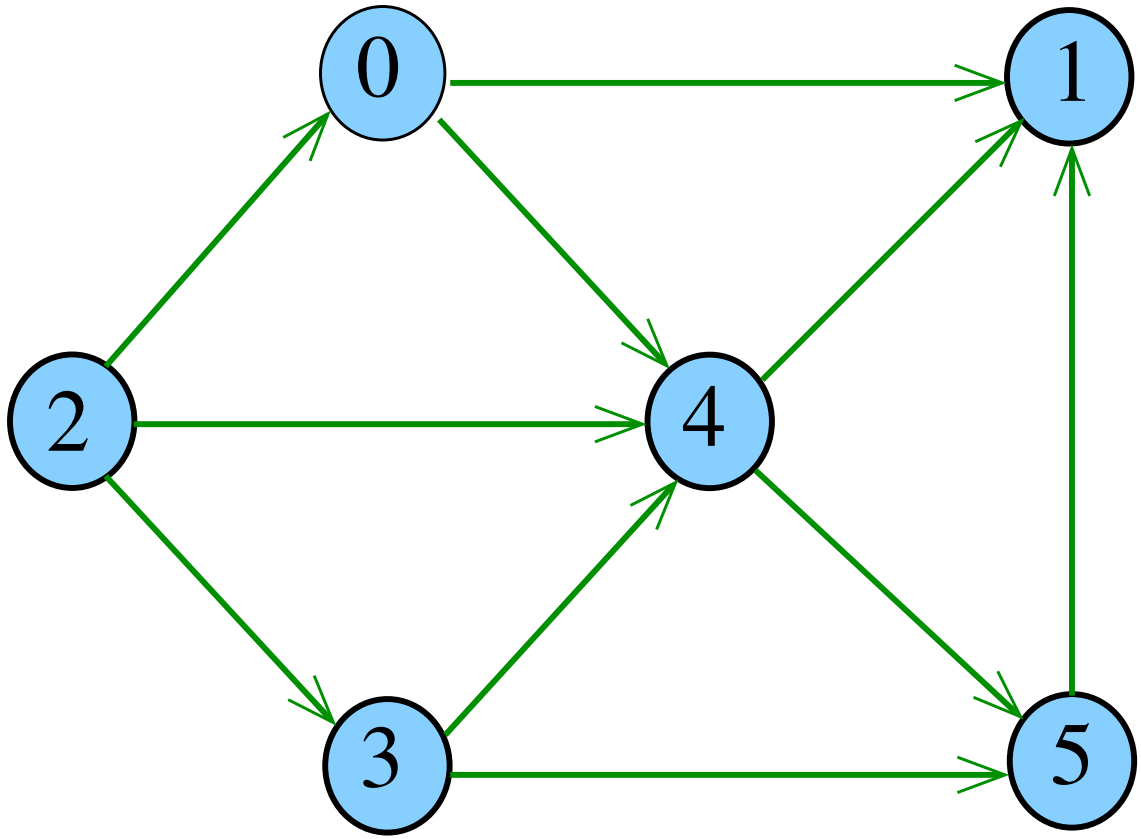
# Algoritmo DFS

| i | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| ts[i] | | | | | | |



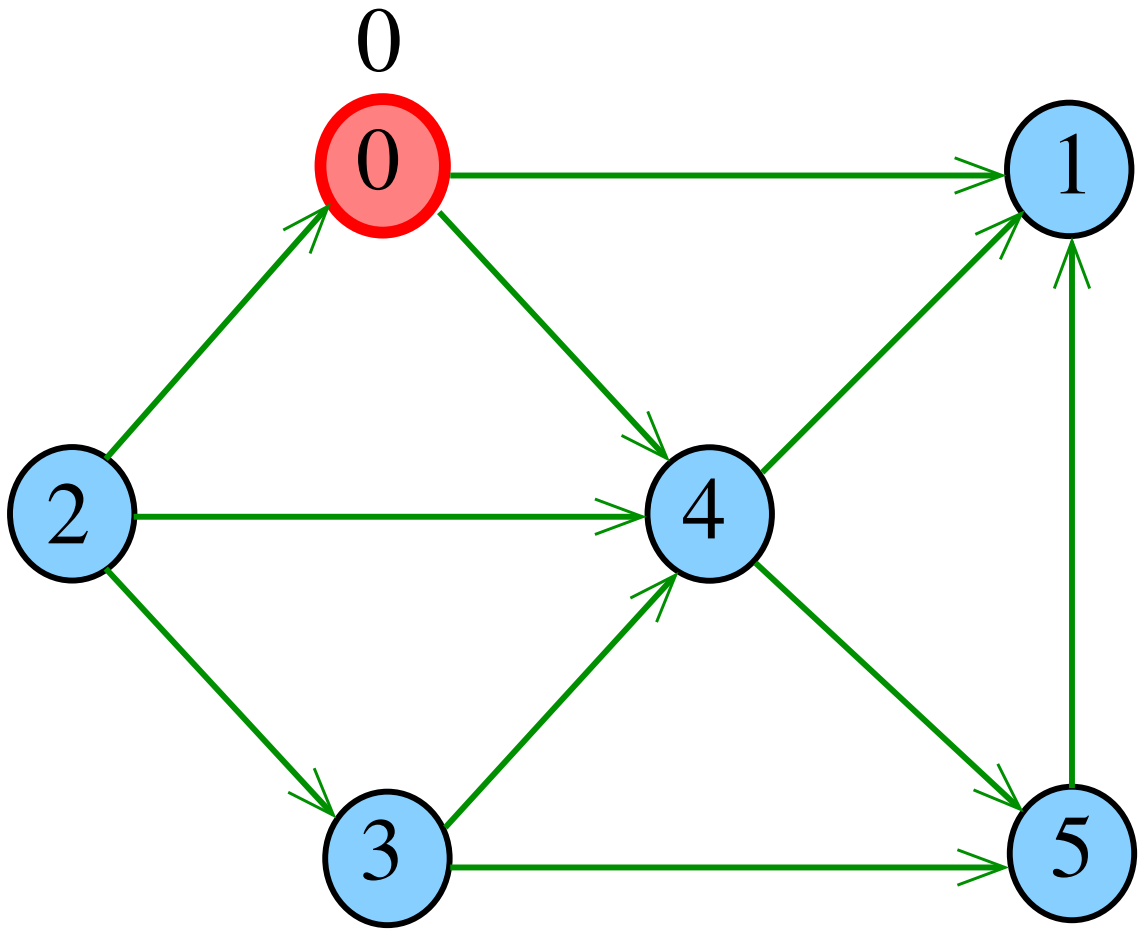
# Algoritmo DFS

| i | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| ts[i] | | | | | | |



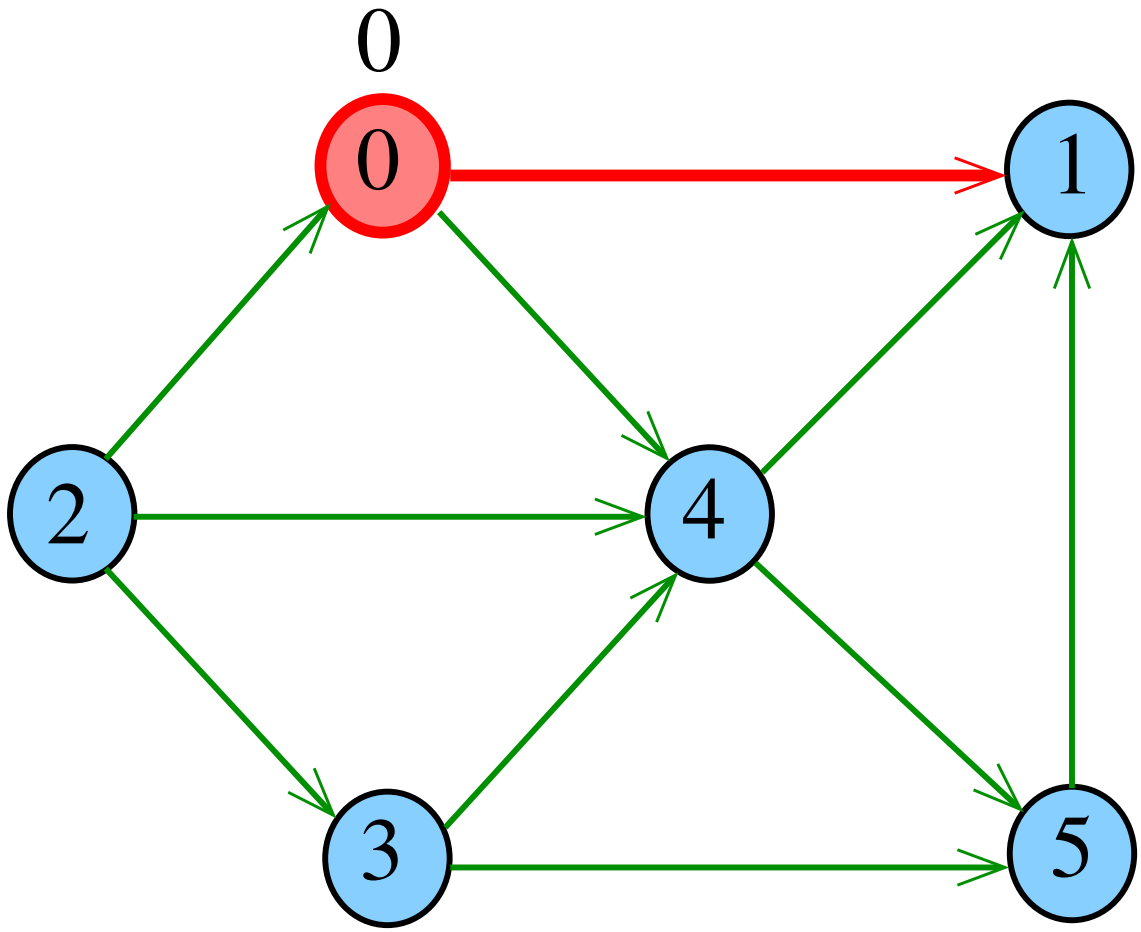
# Algoritmo DFS

| i | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| ts[i] | | | | | | |



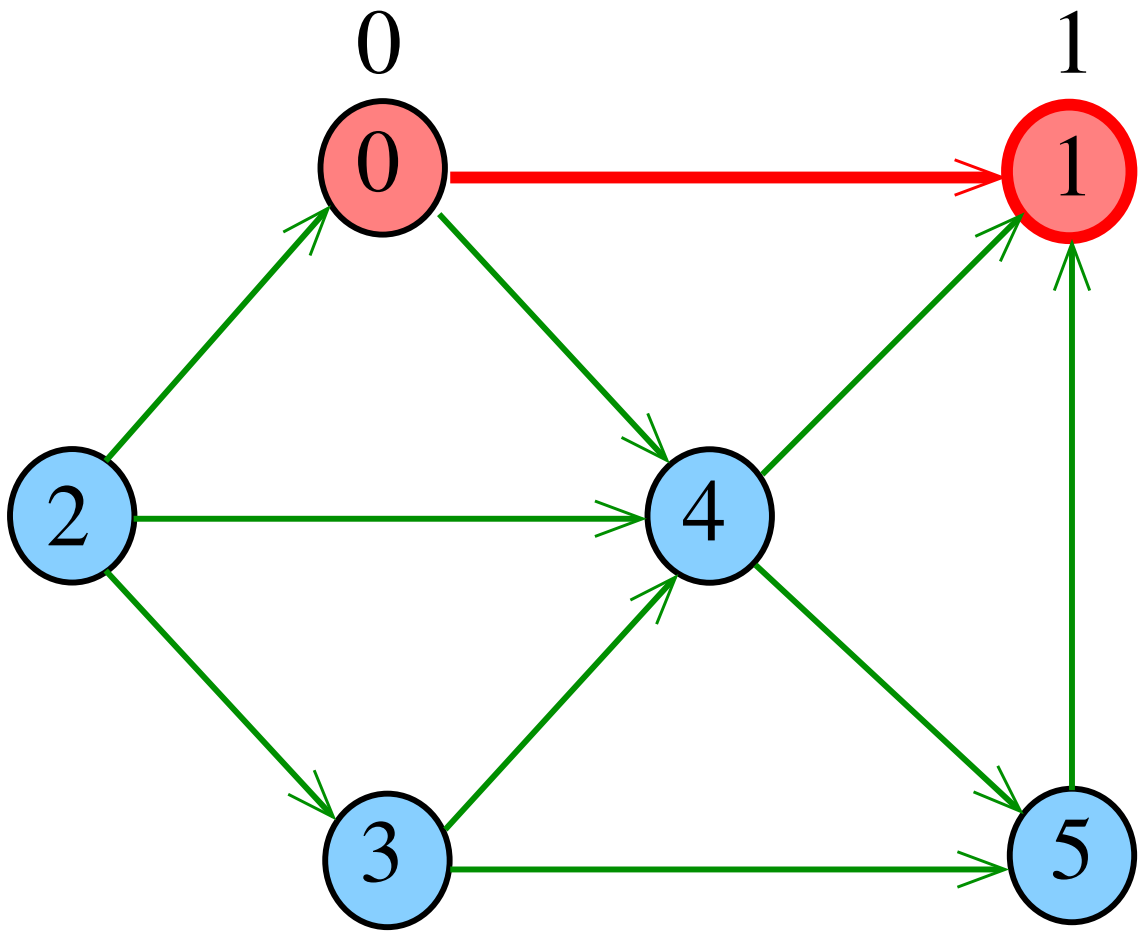
# Algoritmo DFS

| i | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| ts[i] | | | | | | |



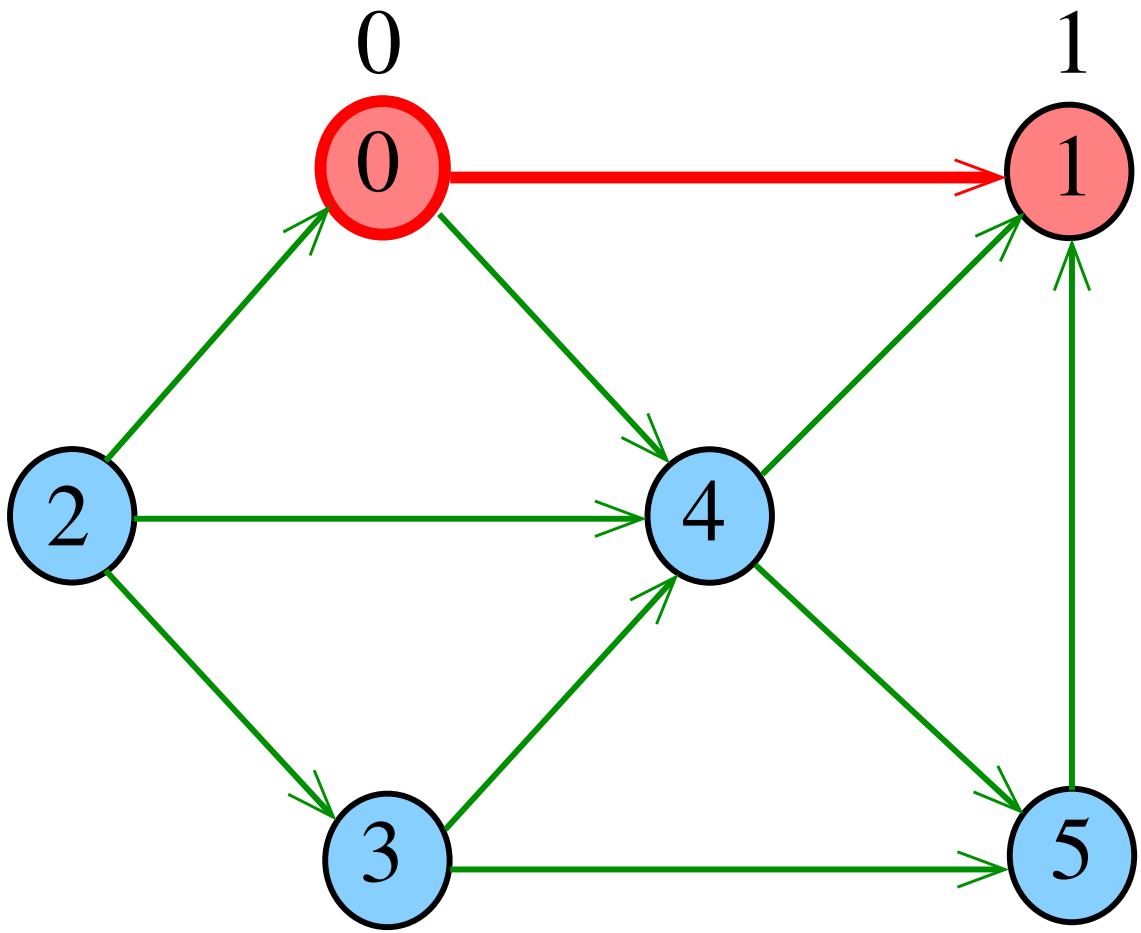
# Algoritmo DFS

| i | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| ts[i] | | | | | | |



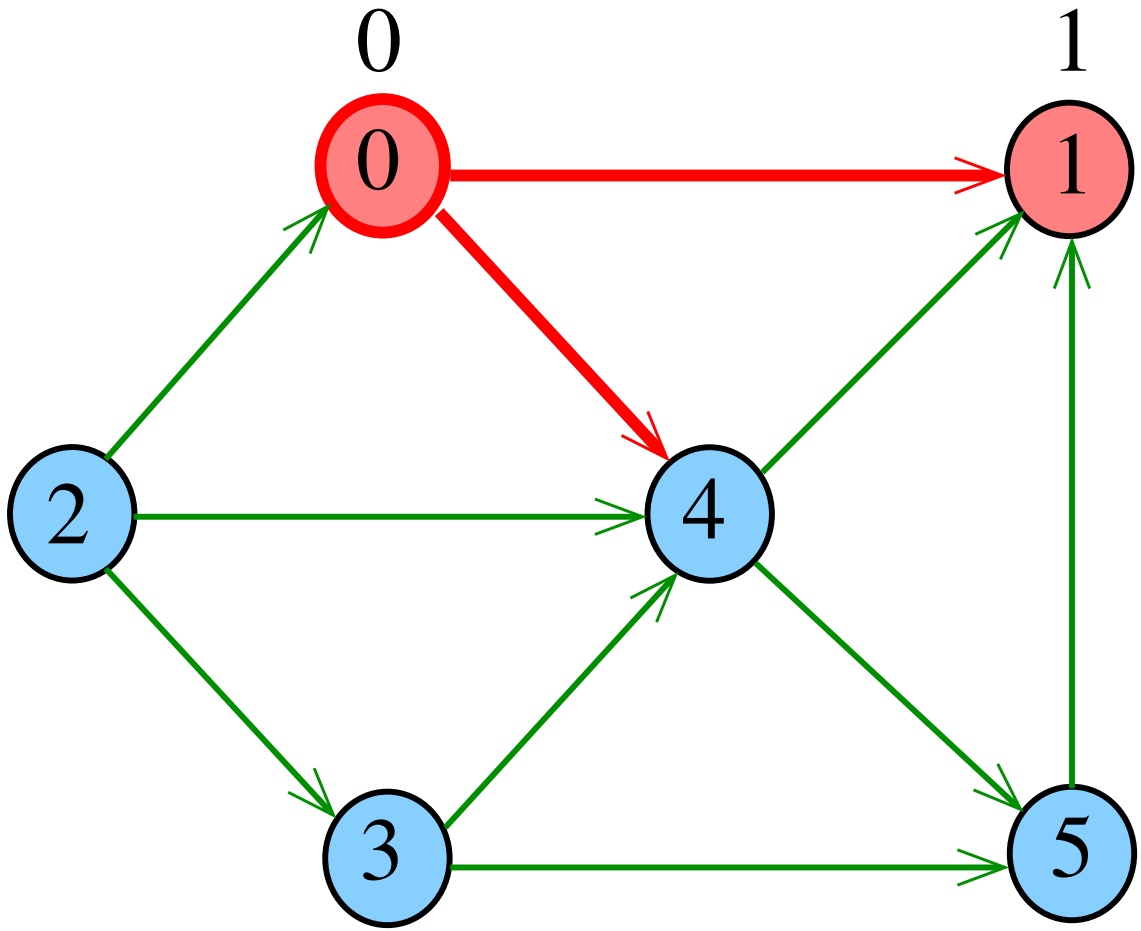
# Algoritmo DFS

| i | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| ts[i] | | | | | | 1 |



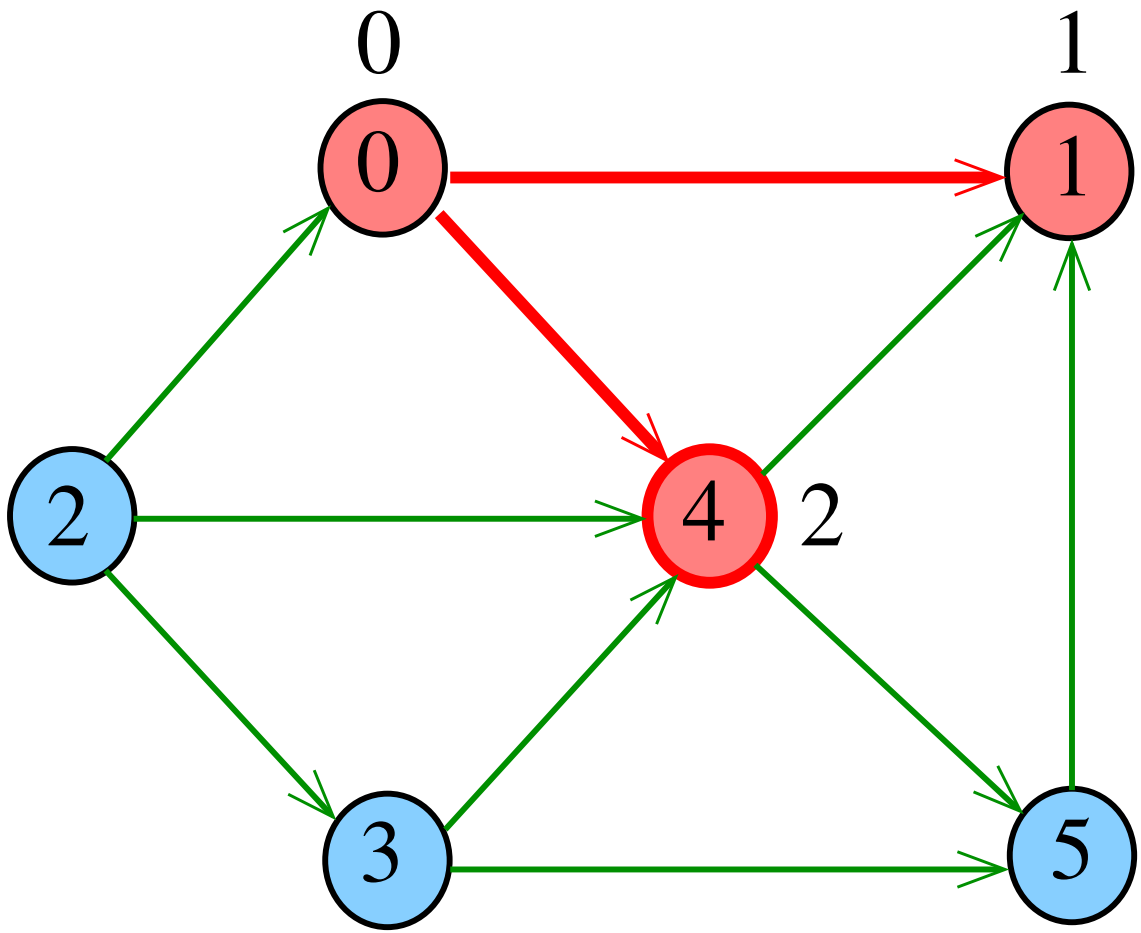
# Algoritmo DFS

| i | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| ts[i] | | | | | | 1 |



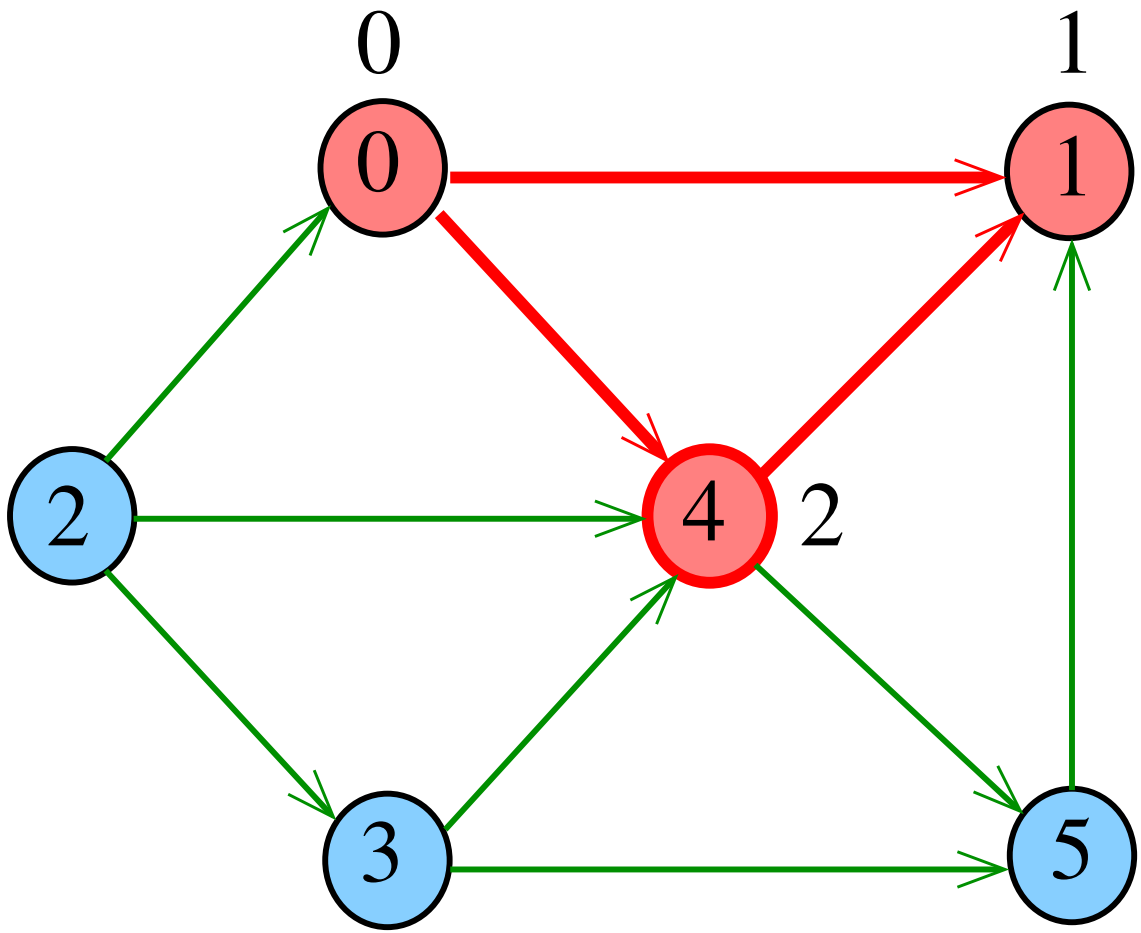
# Algoritmo DFS

| i | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| ts[i] | | | | | | 1 |



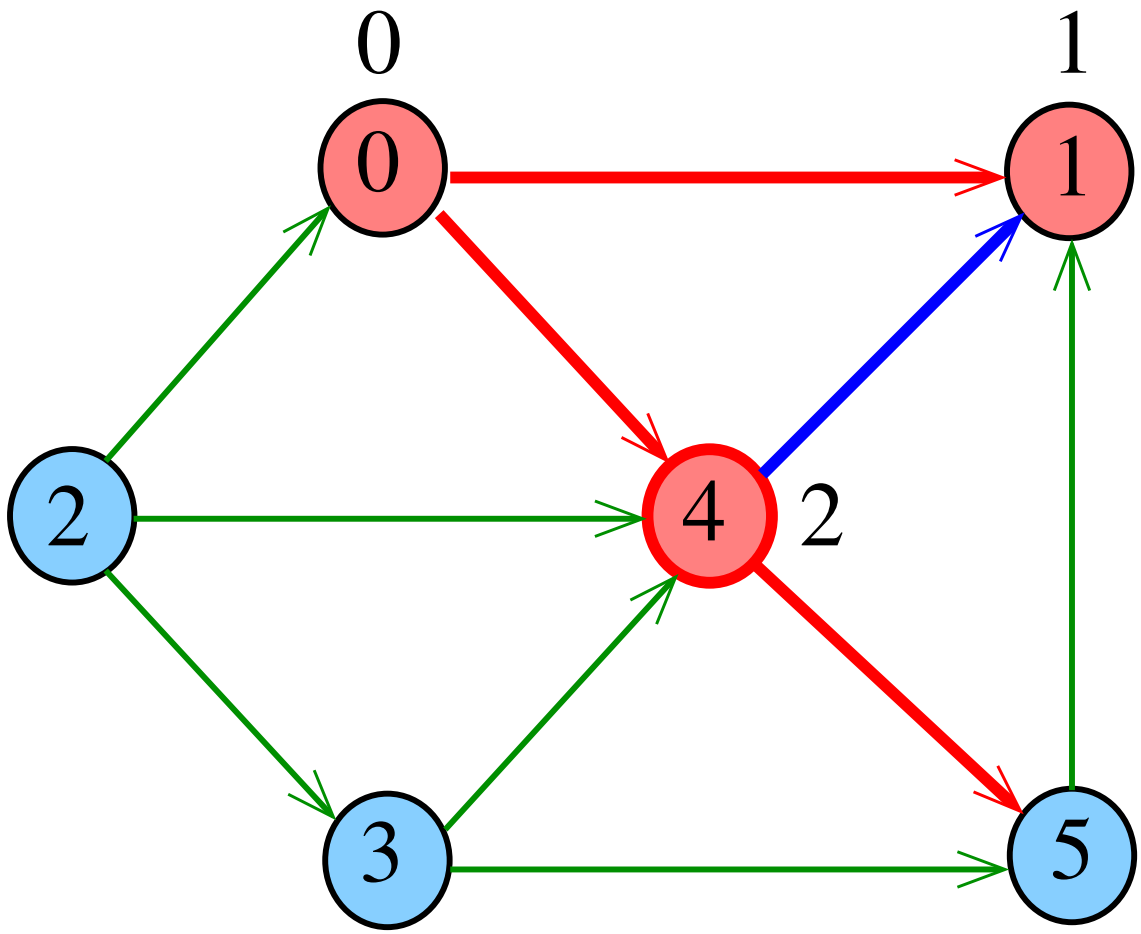
# Algoritmo DFS

| i | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| ts[i] | | | | | | 1 |



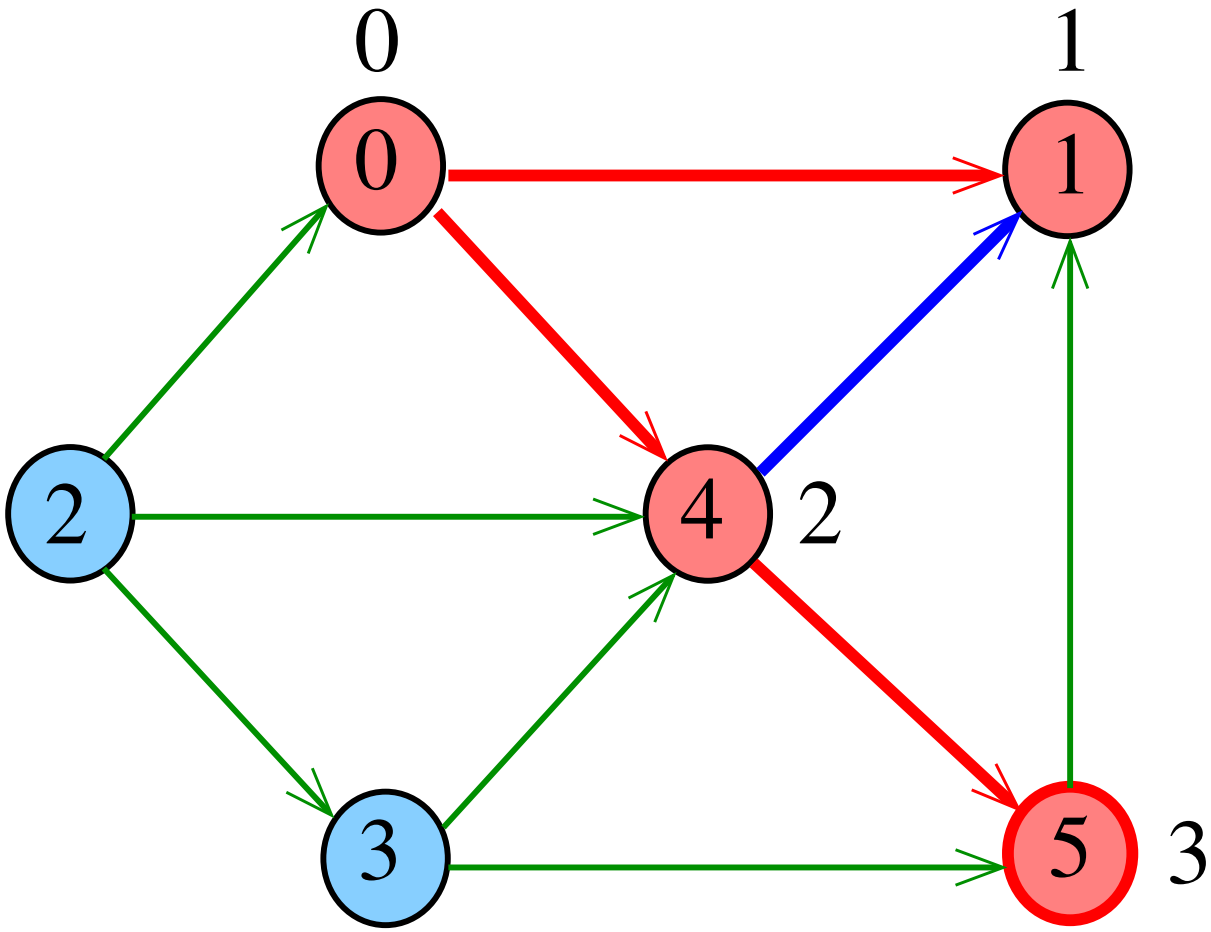
# Algoritmo DFS

| i | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| ts[i] | | | | | | 1 |



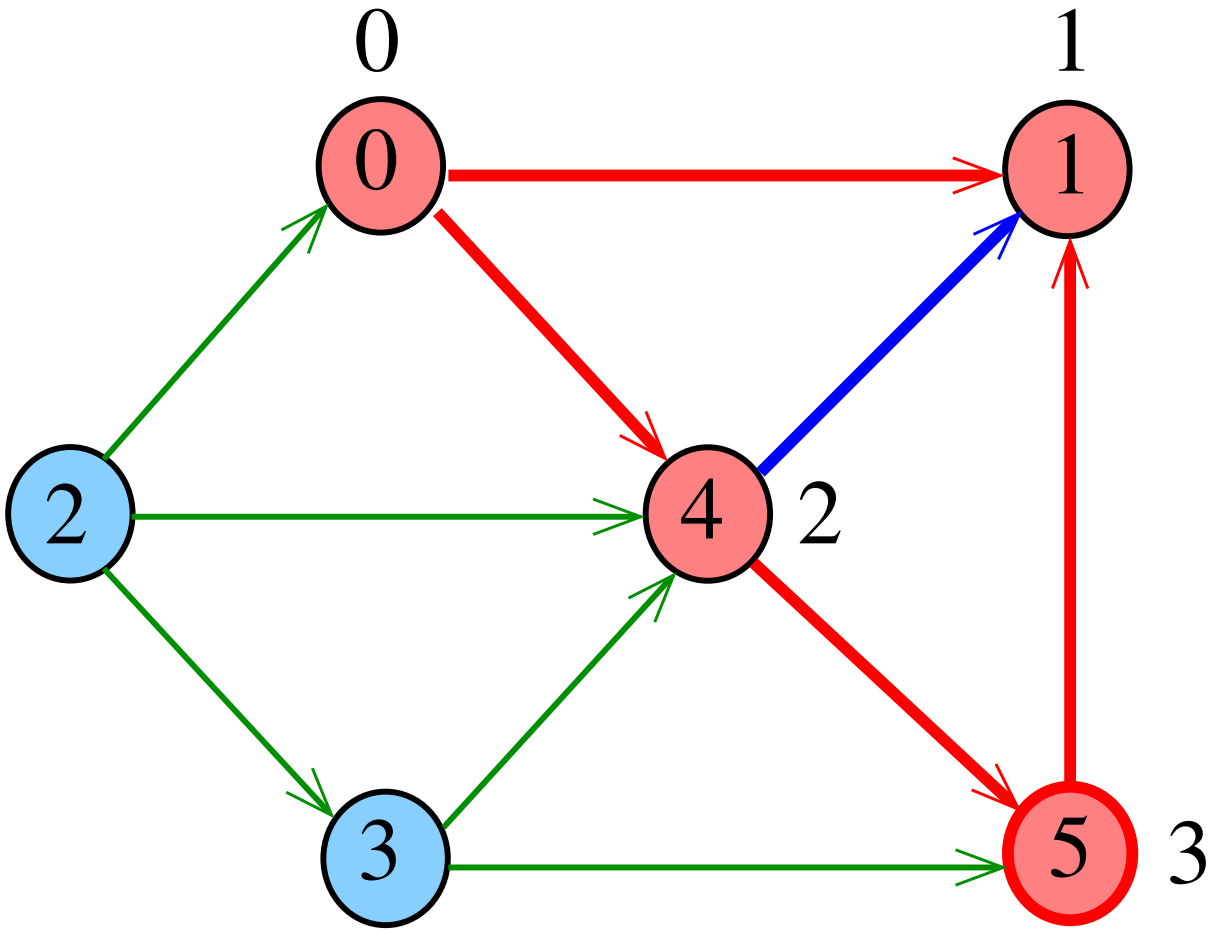
# Algoritmo DFS

| i | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| ts[i] | | | | | | 1 |



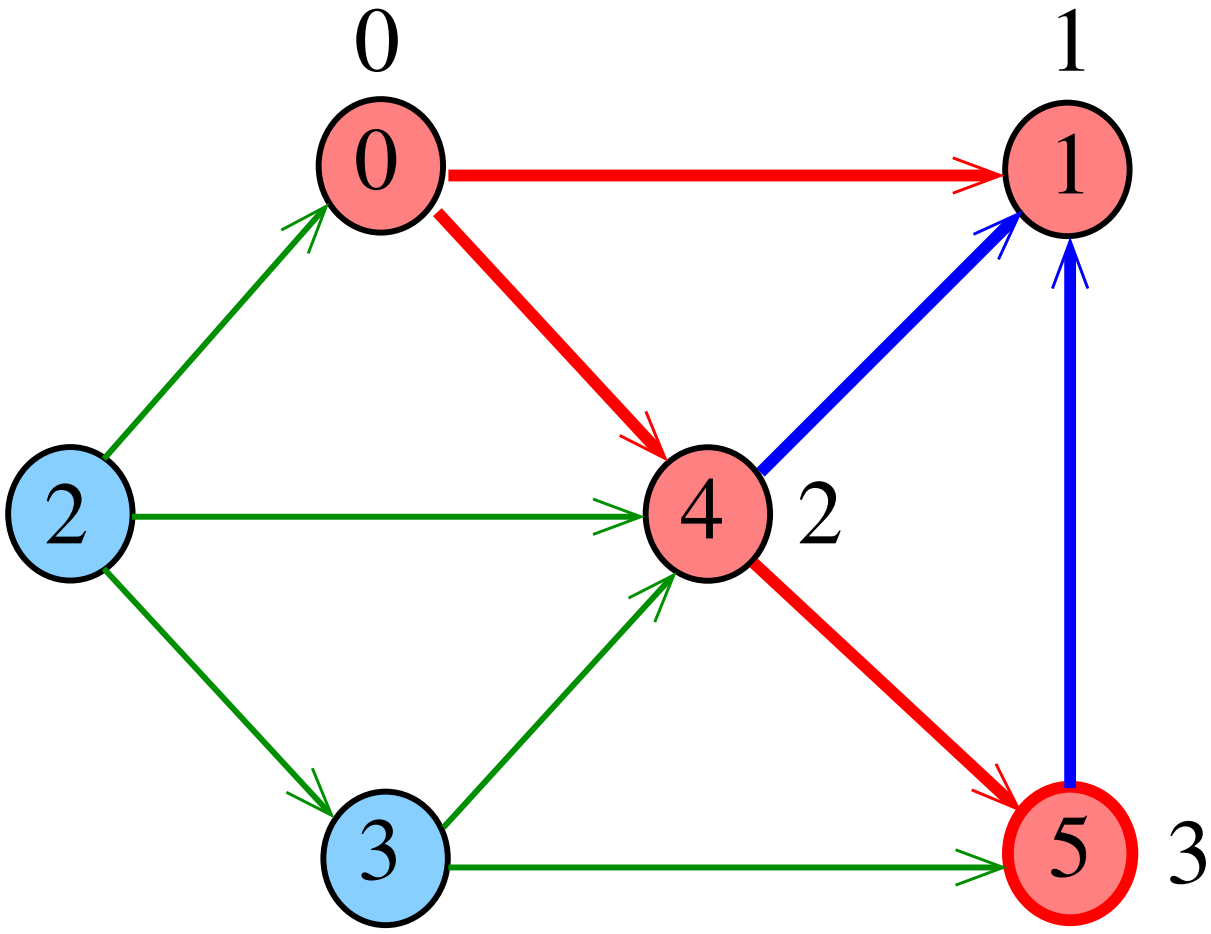
# Algoritmo DFS

| i | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| ts[i] | | | | | | 1 |



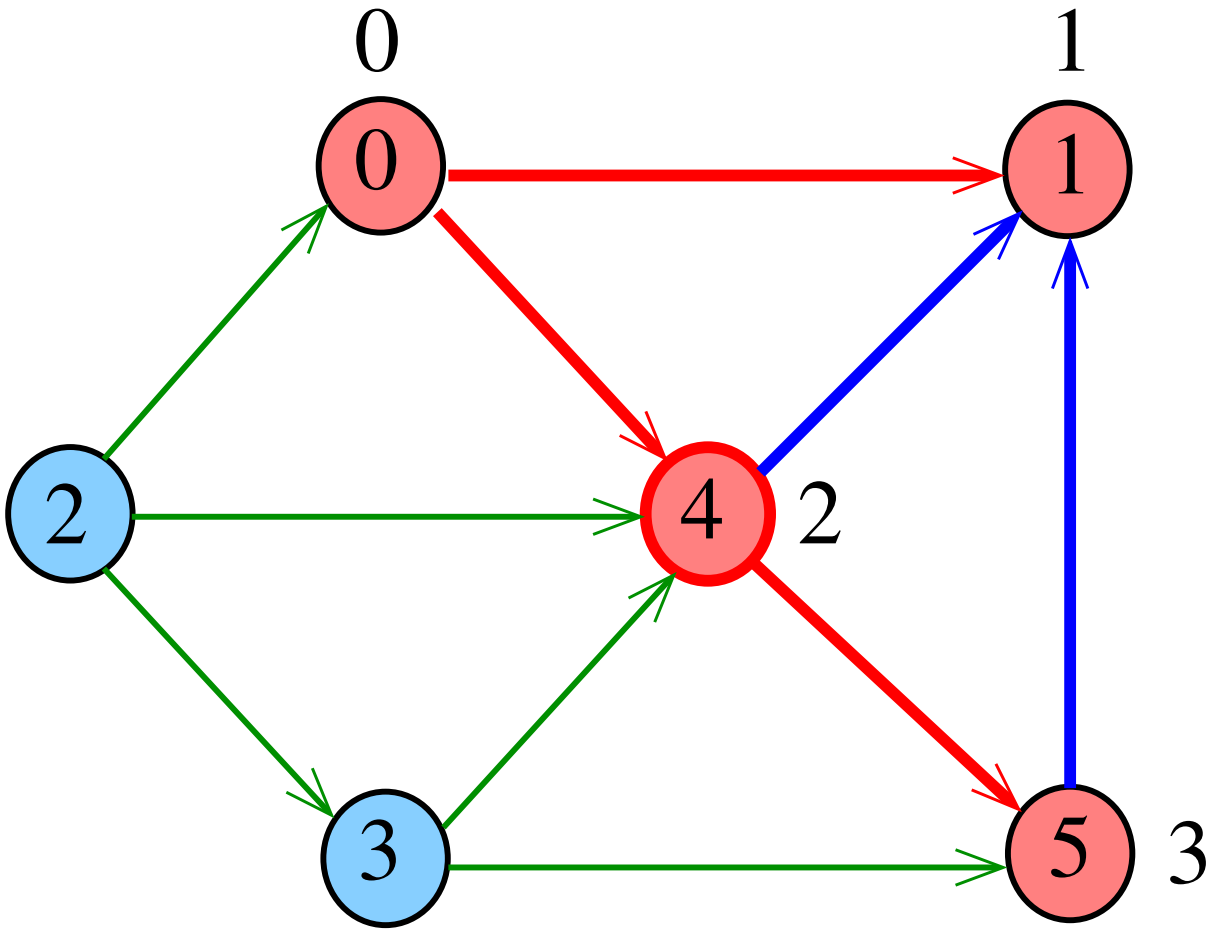
# Algoritmo DFS

| i | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| ts[i] | | | | | | 1 |



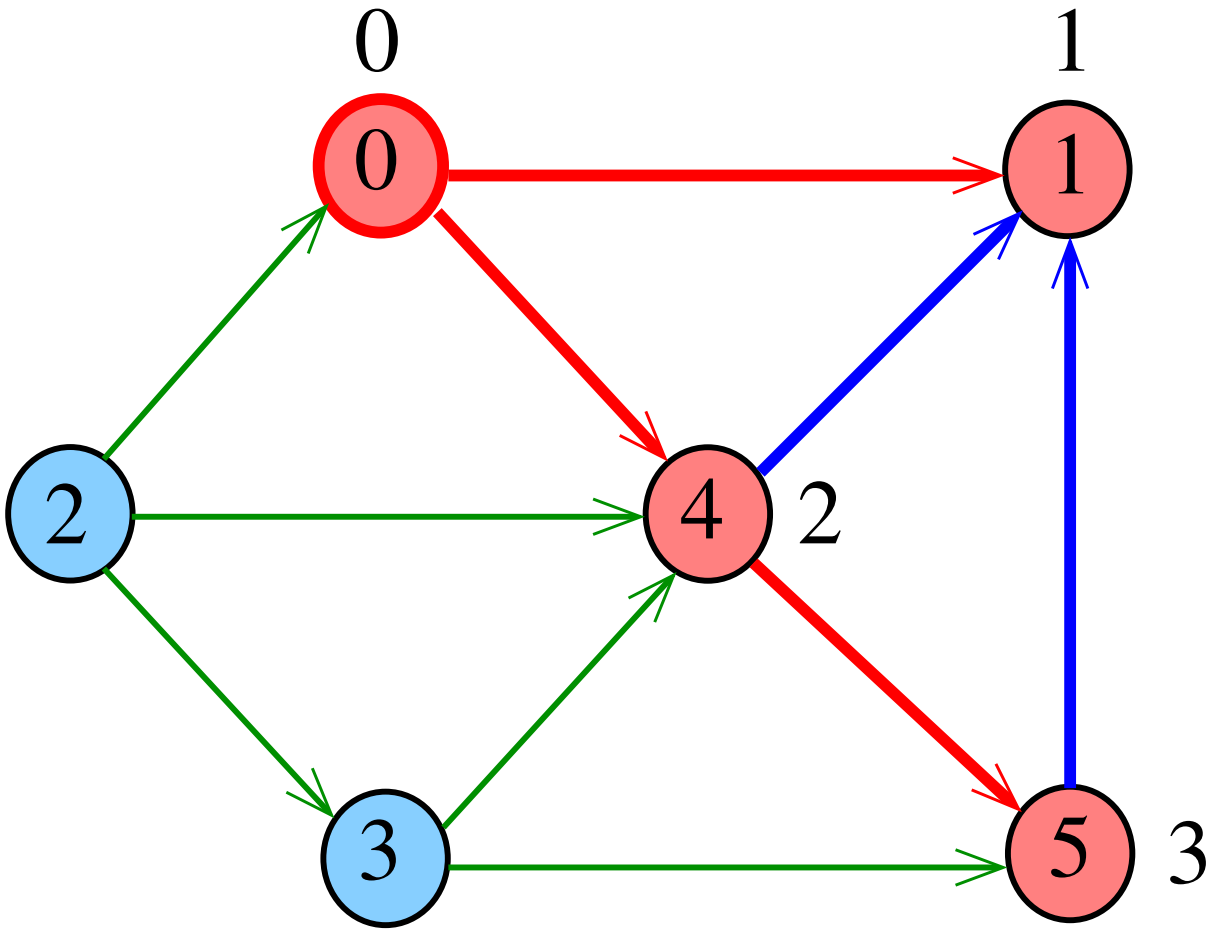
# Algoritmo DFS

| i | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| ts[i] | | | | | 5 | 1 |



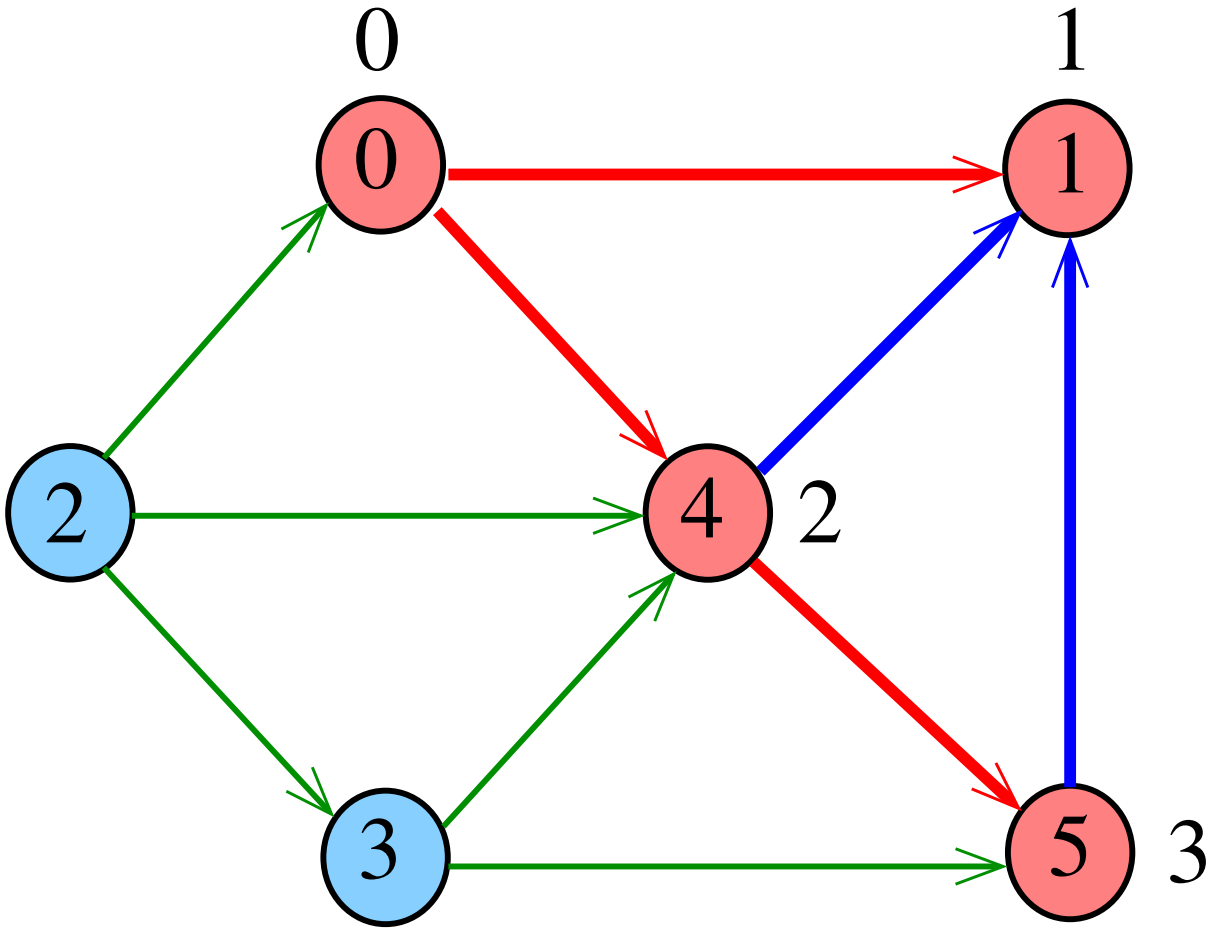
# Algoritmo DFS

| i | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| ts[i] | | | | | 4 | 5 | 1 |



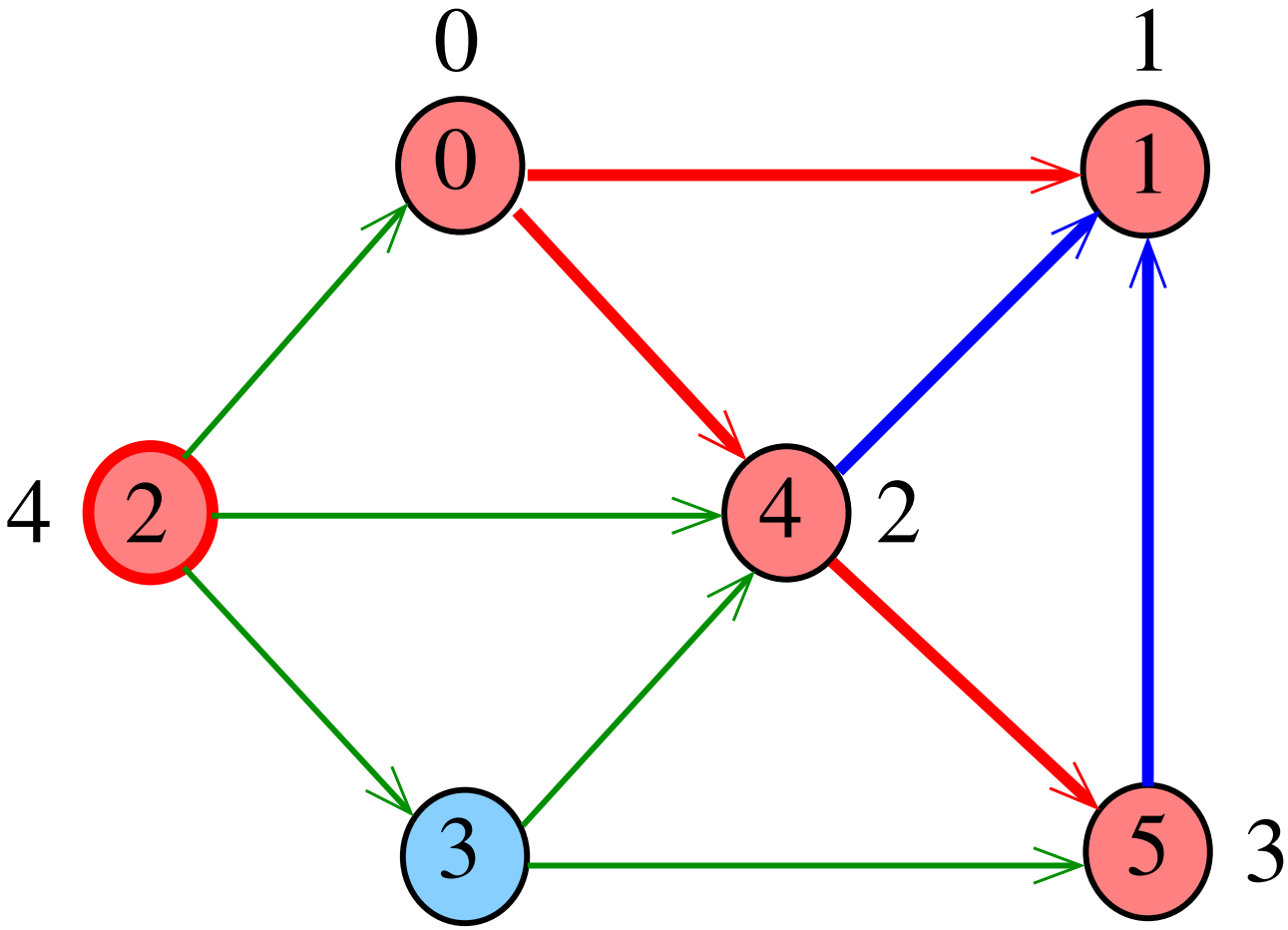
# Algoritmo DFS

| i | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| ts[i] | | | 0 | 4 | 5 | 1 |



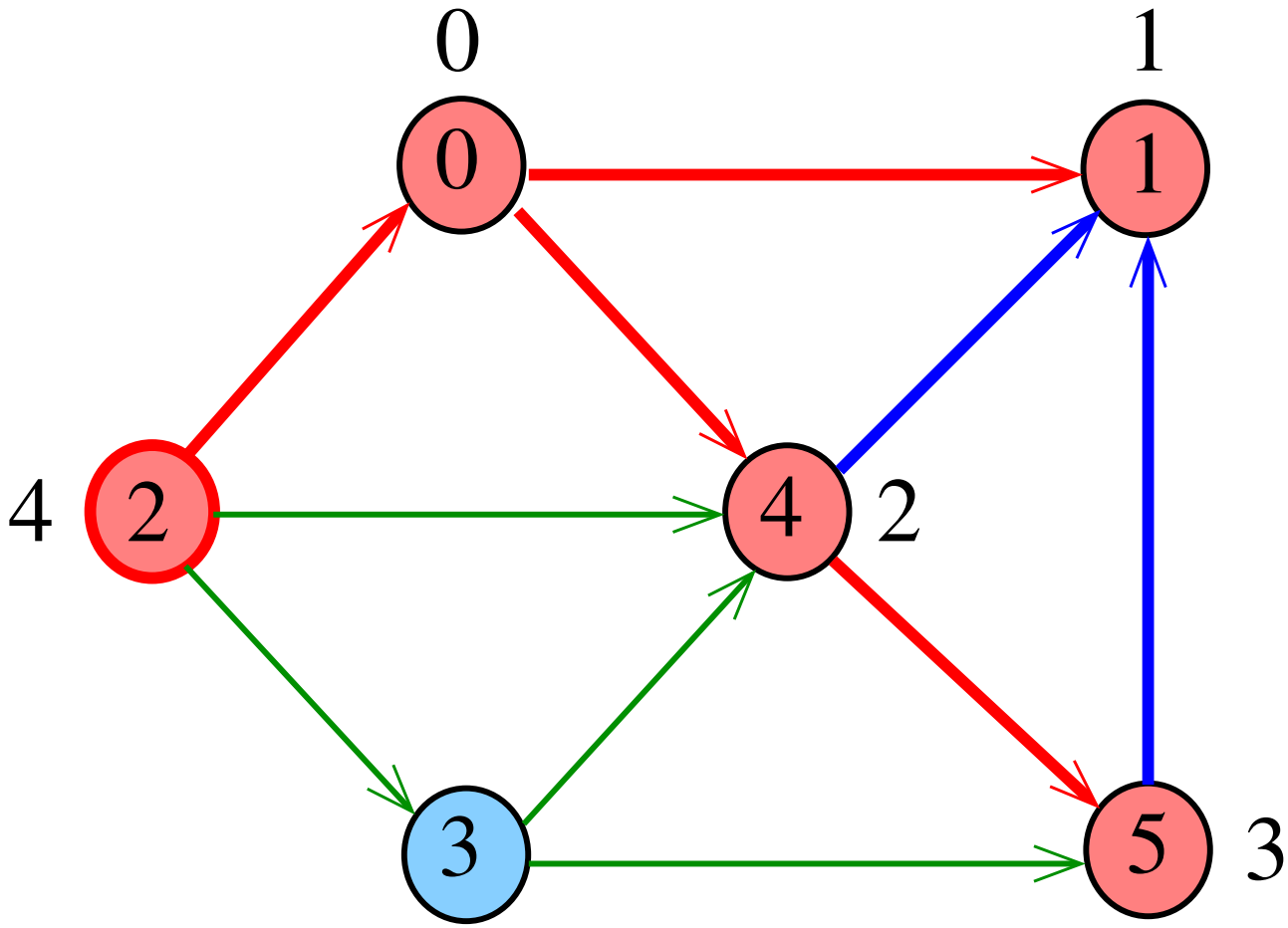
# Algoritmo DFS

| i | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| ts[i] | | | 0 | 4 | 5 | 1 |



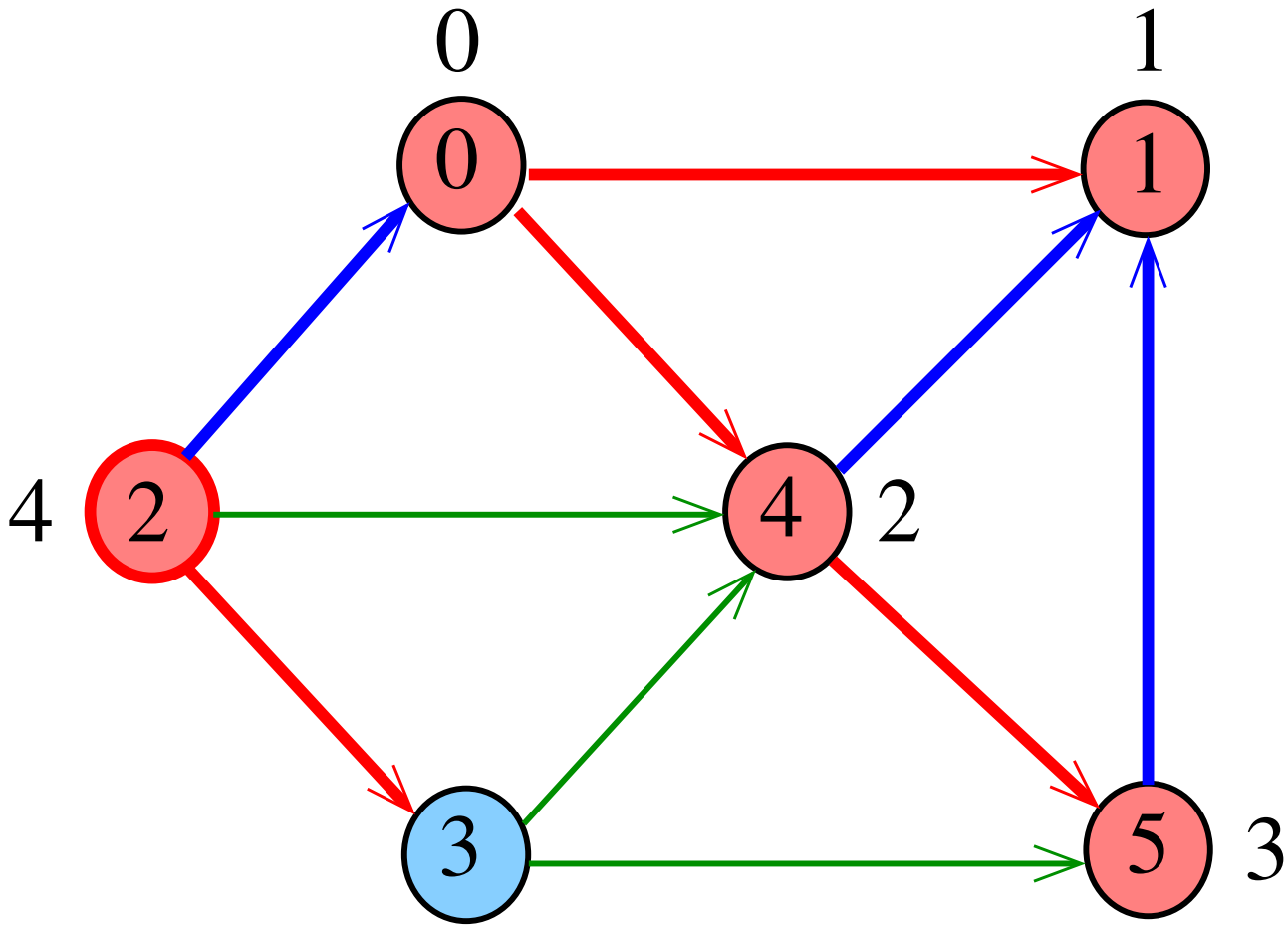
# Algoritmo DFS

| i | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| ts[i] | | | 0 | 4 | 5 | 1 |



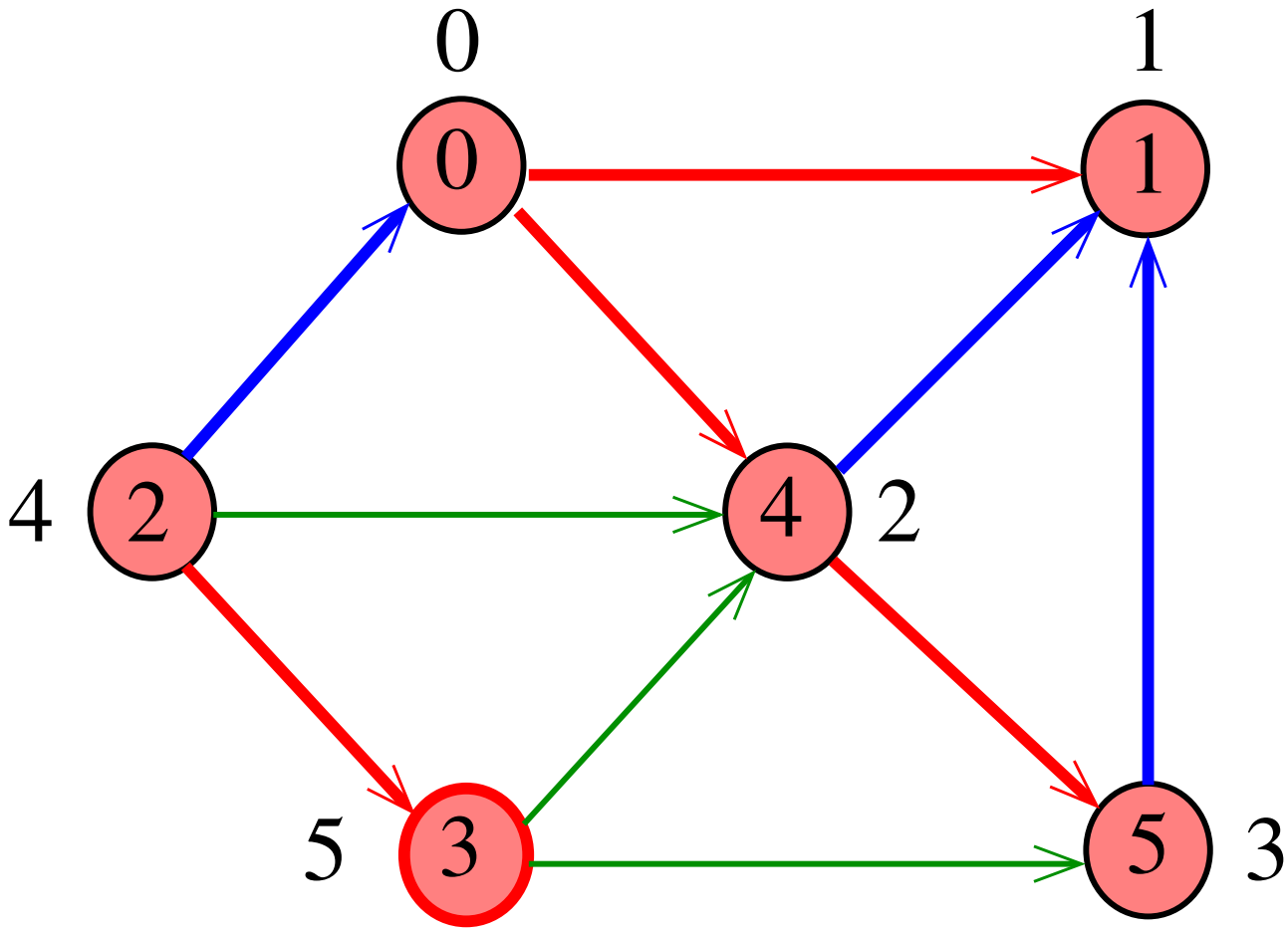
# Algoritmo DFS

| i | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| ts[i] | | | 0 | 4 | 5 | 1 |



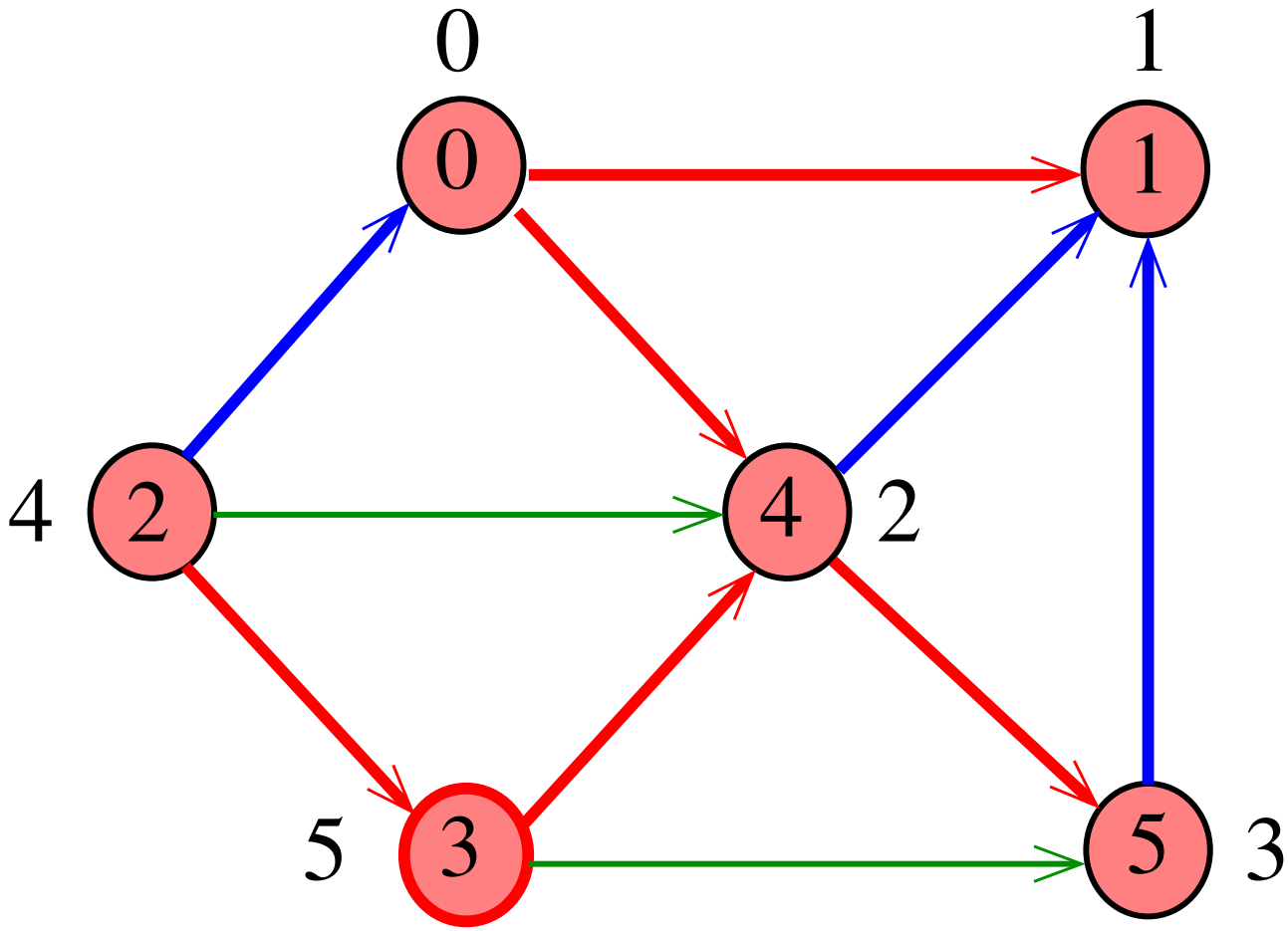
# Algoritmo DFS

| i | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| ts[i] | | | 0 | 4 | 5 | 1 |



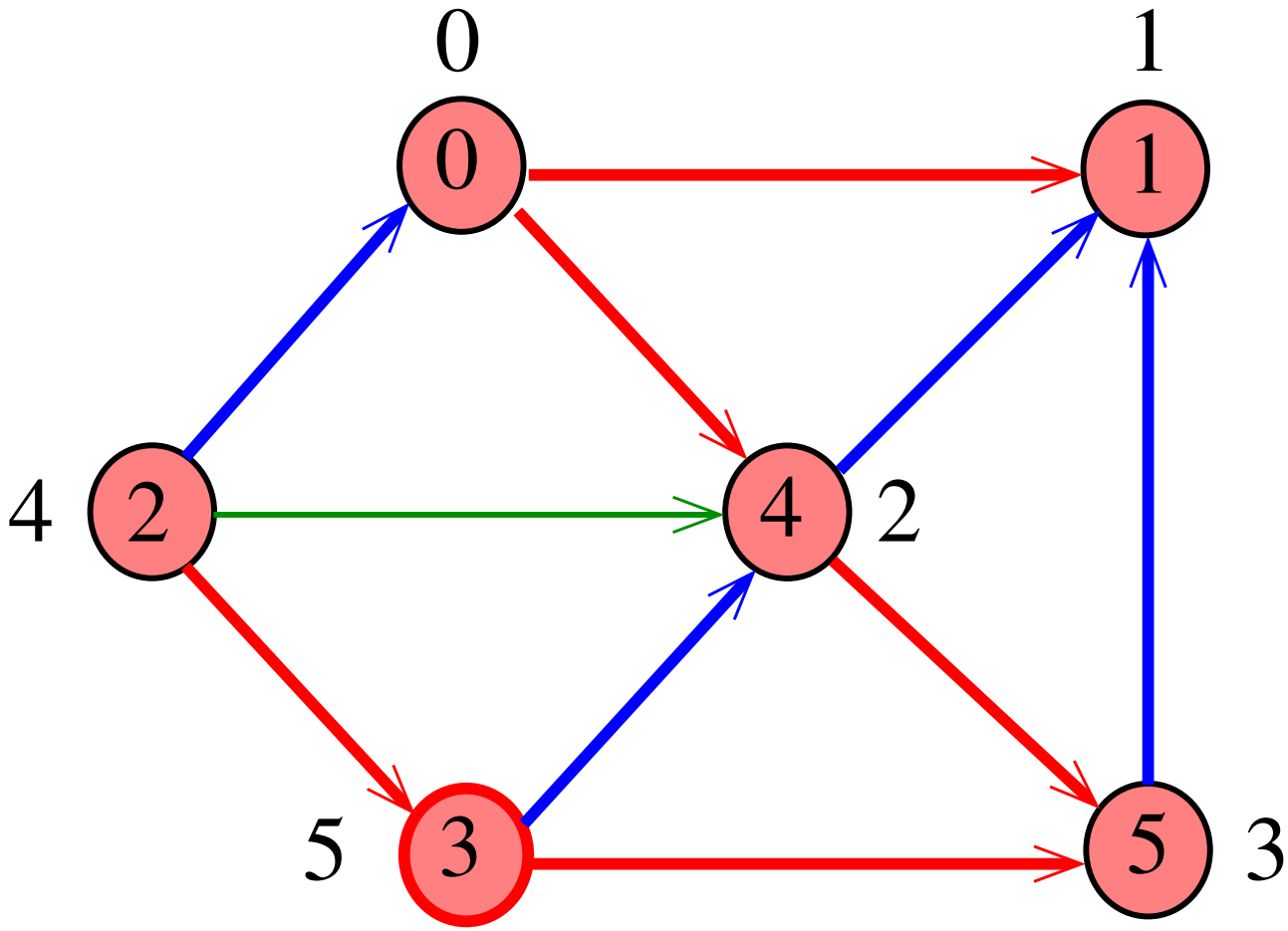
# Algoritmo DFS

| i | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| ts[i] | | | 0 | 4 | 5 | 1 |



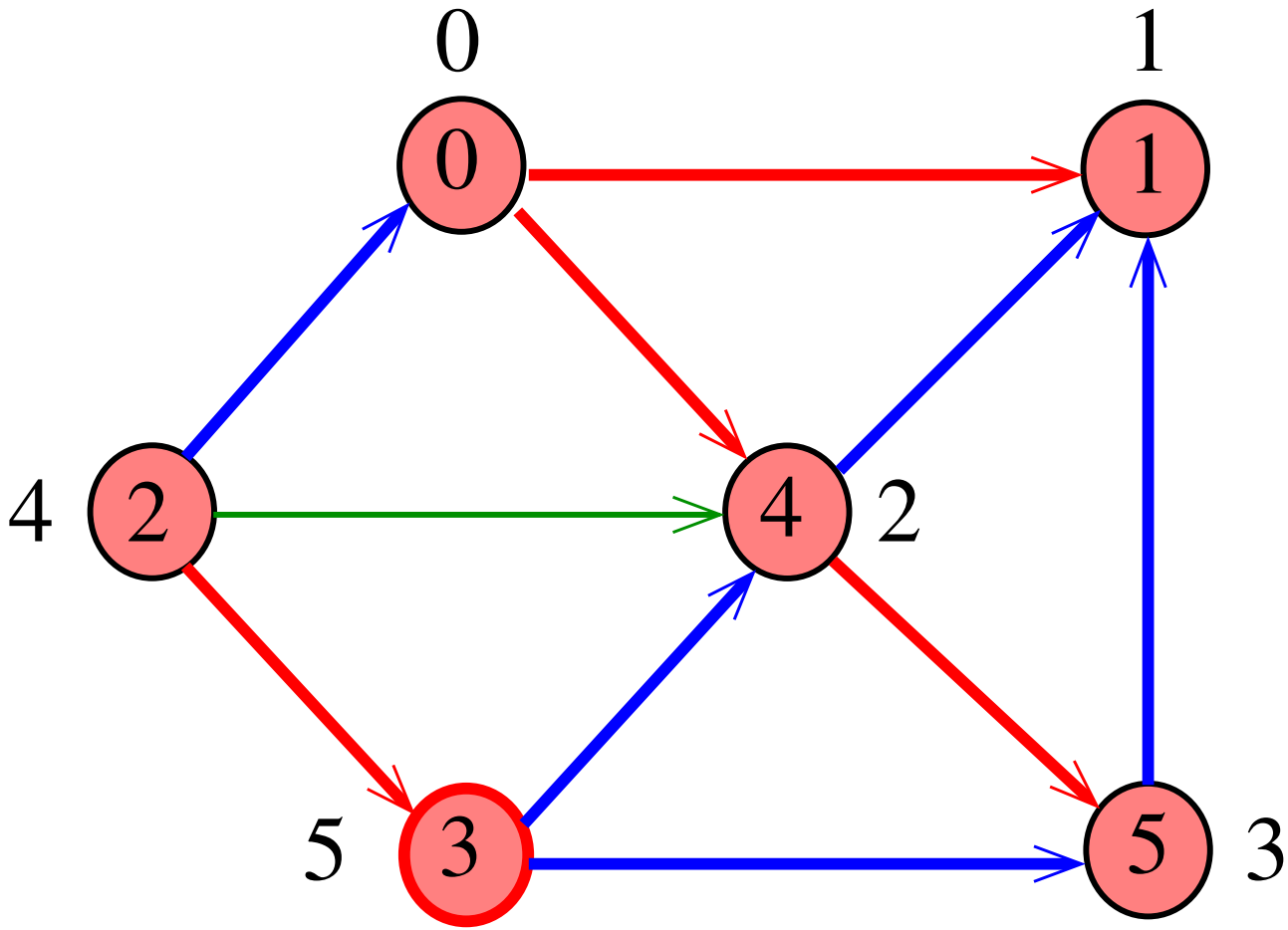
# Algoritmo DFS

| i | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| ts[i] | | | 0 | 4 | 5 | 1 |



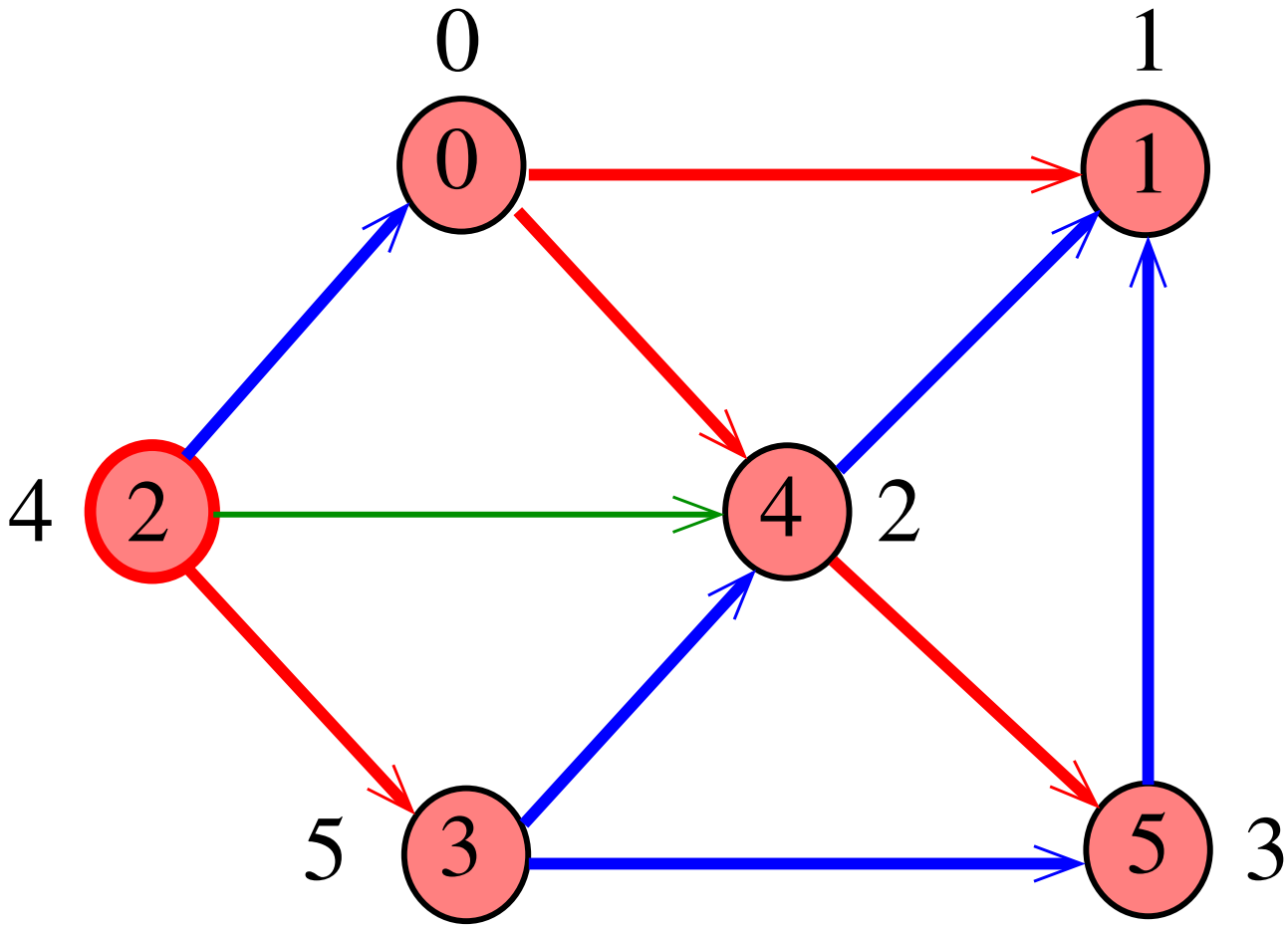
# Algoritmo DFS

| i | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| ts[i] | | | 0 | 4 | 5 | 1 |



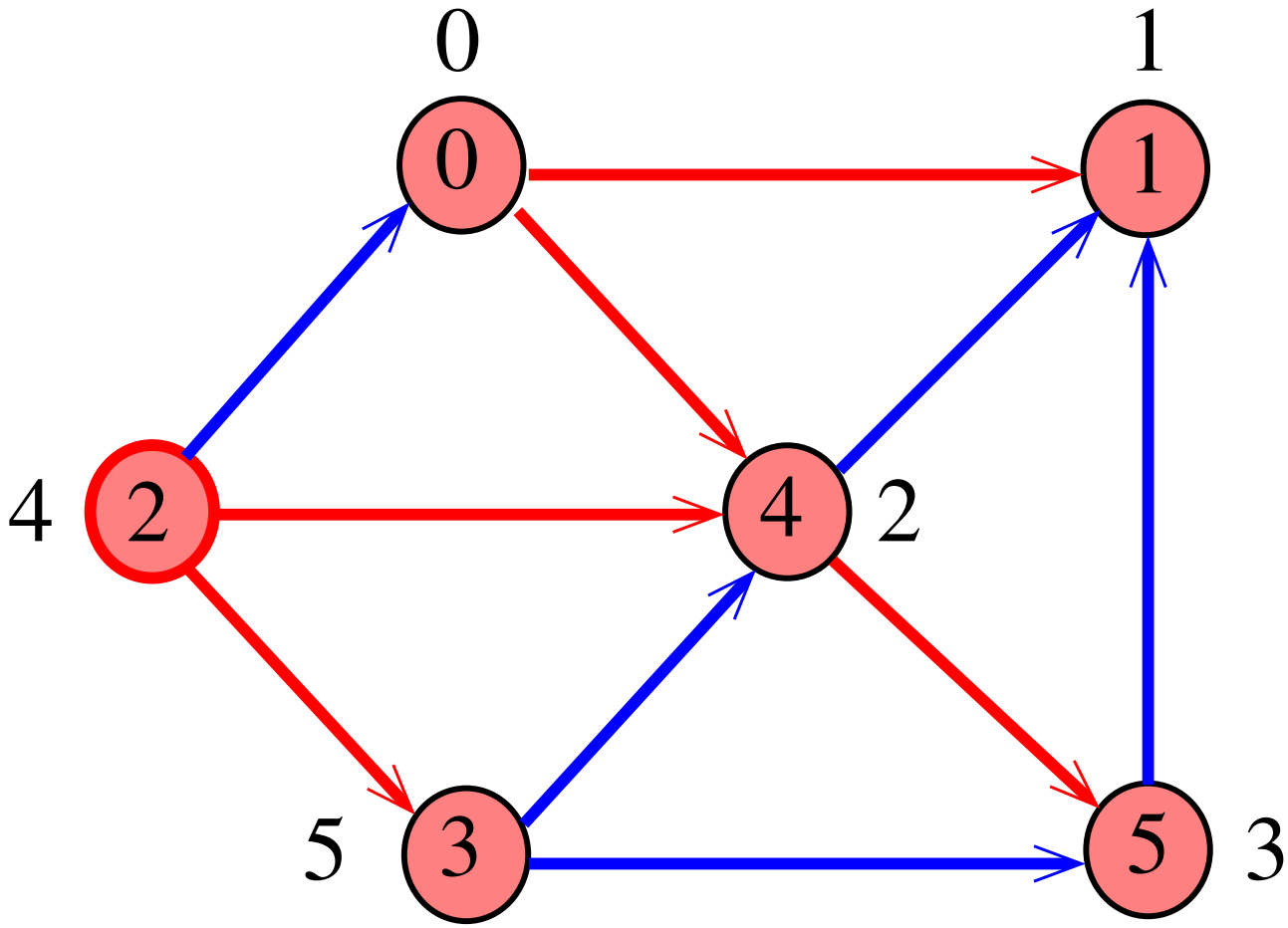
# Algoritmo DFS

| i | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| ts[i] | | 3 | 0 | 4 | 5 | 1 |



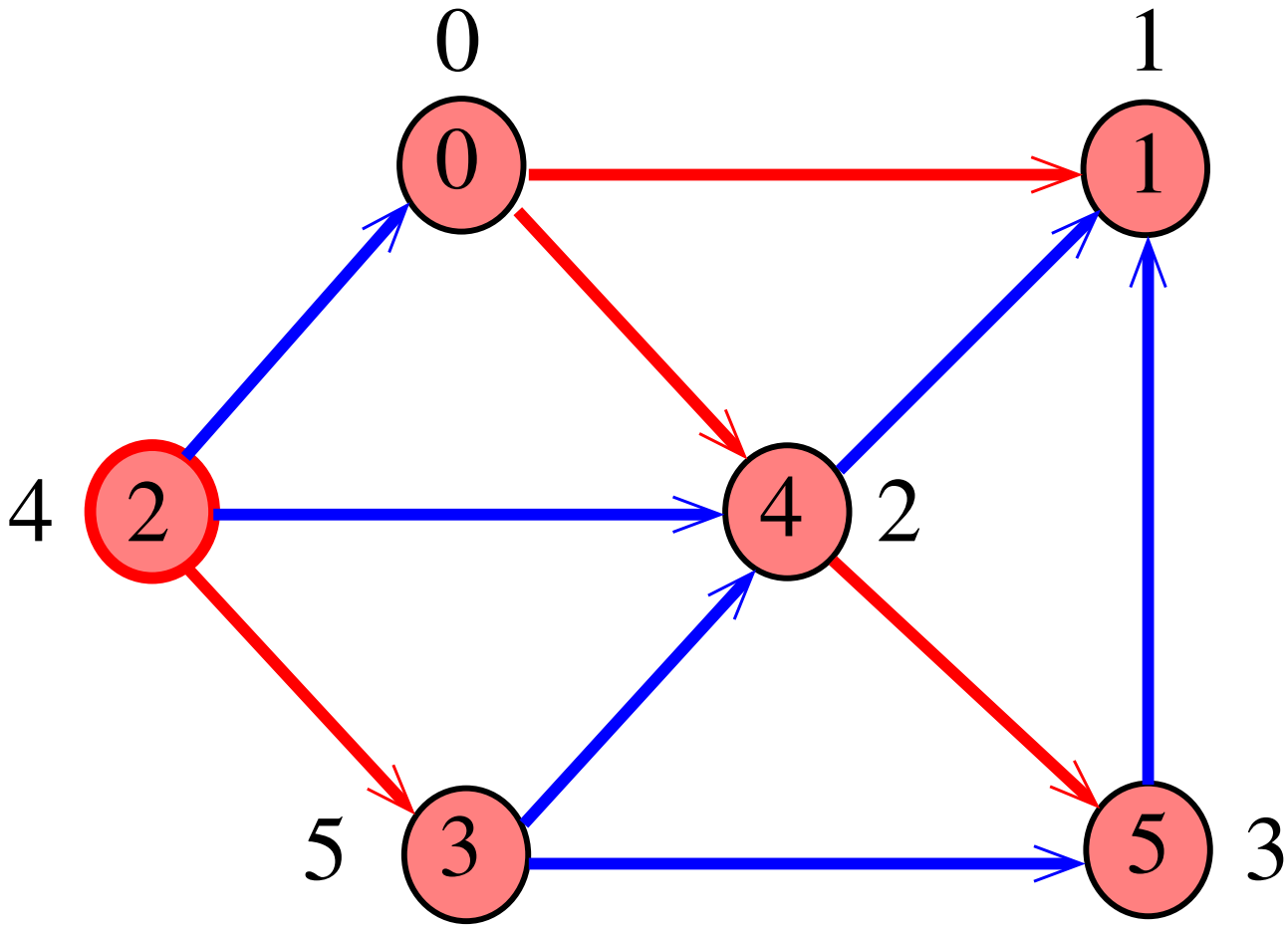
# Algoritmo DFS

| i | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| ts[i] | | 3 | 0 | 4 | 5 | 1 |



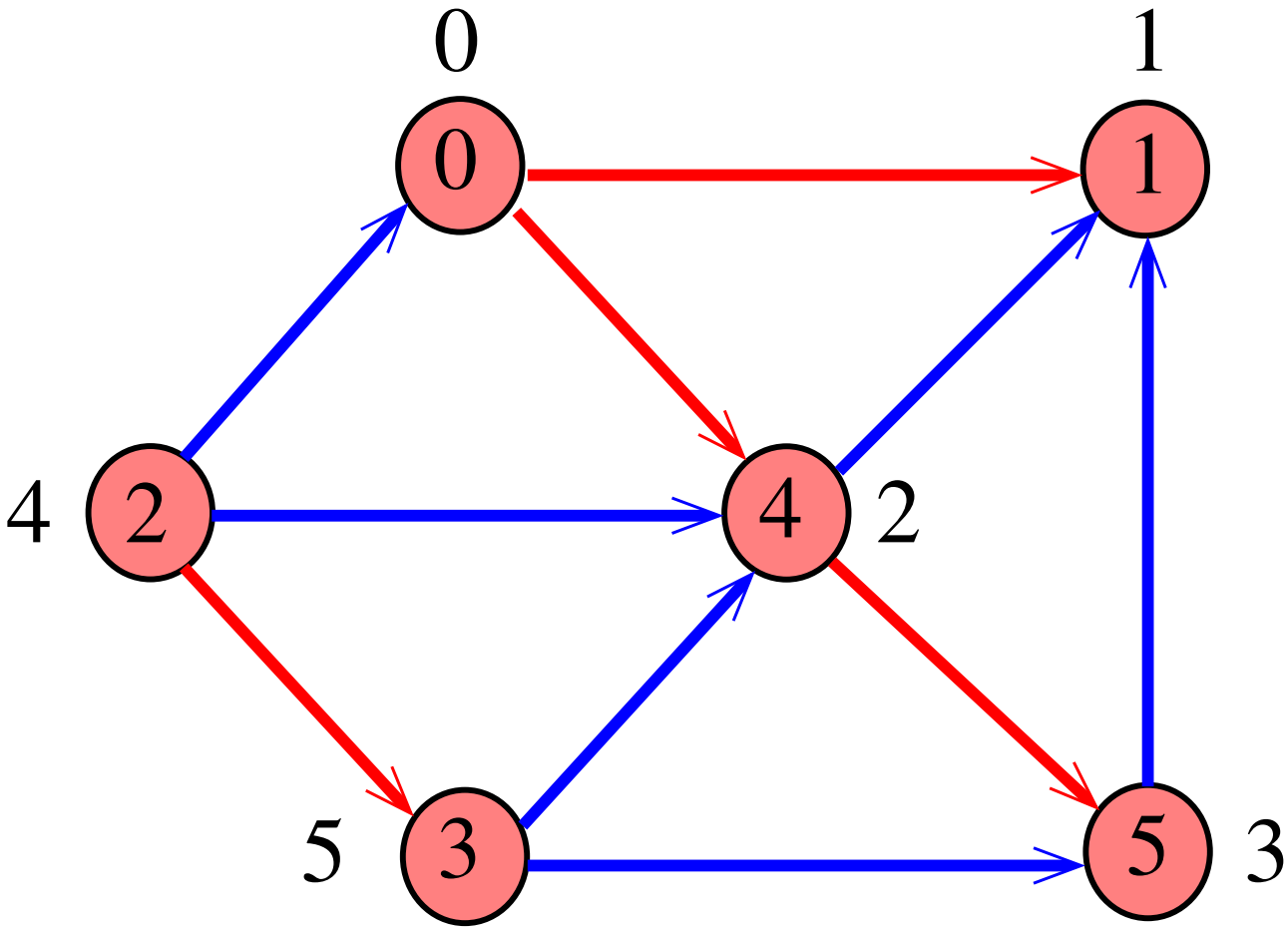
# Algoritmo DFS

| i | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| ts[i] | | 3 | 0 | 4 | 5 | 1 |



# Algoritmo DFS

| i | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| ts[i] | 2 | 3 | 0 | 4 | 5 | 1 |



# Ordenação topológica

Digrafo $G = (V, A)$

Ordenação topológica: Ordenação total de $V$
em que $u$ vem antes de $v$ sempre que $(u, v) \in A$.
Geralmente é expressa como uma numeração de $V$.

# Ordenação topológica

Digrafo $G = (V, A)$

Ordenação topológica: Ordenação total de $V$
em que $u$ vem antes de $v$ sempre que $(u, v) \in A$.
Geralmente é expressa como uma numeração de $V$.

Digrafo $G$ é acíclico se não tem circuito dirigido.

Problema: Dado digrafo $G$ acíclico,
encontrar uma ordenação topológica de $G$.

# Ordenação topológica

Digrafo $G = (V, A)$

Ordenação topológica: Ordenação total de $V$
em que $u$ vem antes de $v$ sempre que $(u, v) \in A$.
Geralmente é expressa como uma numeração de $V$.

Digrafo $G$ é acíclico se não tem circuito dirigido.

Problema: Dado digrafo $G$ acíclico,
encontrar uma ordenação topológica de $G$.

Algoritmo:

Execute uma DFS no grafo calculando $u.\texttt{f}$ para cada $u$.

# Ordenação topológica

Digrafo $G = (V, A)$

Ordenação topológica: Ordenação total de $V$
em que $u$ vem antes de $v$ sempre que $(u, v) \in A$.
Geralmente é expressa como uma numeração de $V$.

Digrafo $G$ é acíclico se não tem circuito dirigido.

Problema: Dado digrafo $G$ acíclico,
encontrar uma ordenação topológica de $G$.

Algoritmo:

Execute uma DFS no grafo calculando $u$.`f` para cada $u$.

Ao terminar um vértice, empilhe-o.

# Ordenação topológica

Digrafo $G = (V, A)$

Ordenação topológica: Ordenação total de $V$
em que $u$ vem antes de $v$ sempre que $(u, v) \in A$.
Geralmente é expressa como uma numeração de $V$.

Digrafo $G$ é acíclico se não tem circuito dirigido.

Problema: Dado digrafo $G$ acíclico,
encontrar uma ordenação topológica de $G$.

Algoritmo:

Execute uma DFS no grafo calculando $u$.`f` para cada $u$.

Ao terminar um vértice, empilhe-o.

Devolva a pilha invertida.

# Ordenação topológica

Problema: Dado digrafo $G$ acíclico,
encontrar uma ordenação topológica de $G$.

Algoritmo:

Execute uma DFS no grafo calculando $u$.`f` para cada $u$.

Ao terminar um vértice, empilhe-o.

Devolva a pilha invertida.

# Ordenação topológica

Problema: Dado digrafo $G$ acíclico,
encontrar uma ordenação topológica de $G$.

Algoritmo:

Execute uma DFS no grafo calculando $u$.`f` para cada $u$.

Ao terminar um vértice, empilhe-o.

Devolva a pilha invertida.

Consumo de tempo: linear no tamanho do grafo

# Ordenação topológica

Problema: Dado digrafo $G$ acíclico,
encontrar uma ordenação topológica de $G$.

Algoritmo:

Execute uma DFS no grafo calculando $u$.`f` para cada $u$.

Ao terminar um vértice, empilhe-o.

Devolva a pilha invertida.

Consumo de tempo: linear no tamanho do grafo

Lema: Um digrafo é acíclico sse
a DFS não detecta nenhuma aresta de retorno.

# Ordenação topológica

Problema: Dado digrafo $G$ acíclico,
encontrar uma ordenação topológica de $G$.

Algoritmo:

Execute uma DFS no grafo calculando $u$.`f` para cada $u$.

Ao terminar um vértice, empilhe-o.

Devolva a pilha invertida.

Consumo de tempo: linear no tamanho do grafo

Lema: Um digrafo é acíclico sse
a DFS não detecta nenhuma aresta de retorno.

Teorema: O algoritmo acima calcula
uma ordenação topológica do grafo.