Algoritmos e Estruturas de Dados II

Ordenação Topológica em Grafos

Prof. Tiago Eugenio de Melo
tmelo@uea.edu.br

www.tiagodemelo.info

Observações

- As palavras com a fonte `Courier` indicam as palavras-reservadas da linguagem de programação.

Referências

- **Projetos de Algoritmos – com implementações em Pascal e C**. Nivio Ziviani. 2ª edição. Thomson, 2005.
- Material baseado nas notas de aula do professor Edirlei Soares de Lima. Acessado em 12/11/2019: http://edirlei.3dgb.com.br
- Material baseado nas notas de aula do professor Marco Antônio Moreira de Carvalho. Acessado em 01/11/2019: http://www.decom.ufop.br/marco/ensino/bcc204/material-das-aulas

Ordenação Topológica

Ordenação Topológica

Ordenação Topológica

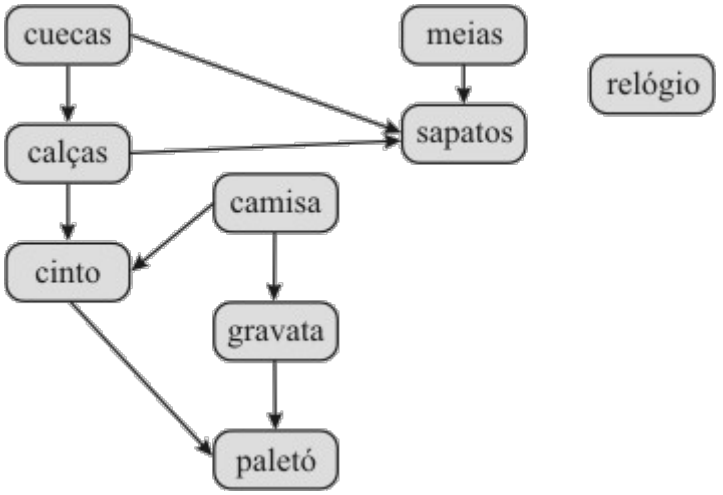
- Uma ordenação topológica de um **grafo acíclico direcionado** (GAD) é uma ordenação linear de seus vértices, na qual cada vértice aparece antes de seus descendentes.

Ordenação Topológica

- Uma ordenação topológica de um **grafo acíclico direcionado** (GAD) é uma ordenação linear de seus vértices, na qual cada vértice aparece antes de seus descendentes.
- Exemplo:

Ordenação Topológica

- Uma ordenação topológica de um **grafo acíclico direcionado** (GAD) é uma ordenação linear de seus vértices, na qual cada vértice aparece antes de seus descendentes.
- Exemplo:



Ordenação Topológica

Ordenação Topológica

- O grafo abaixo pode ter muitas ordenações topológicas.

Ordenação Topológica

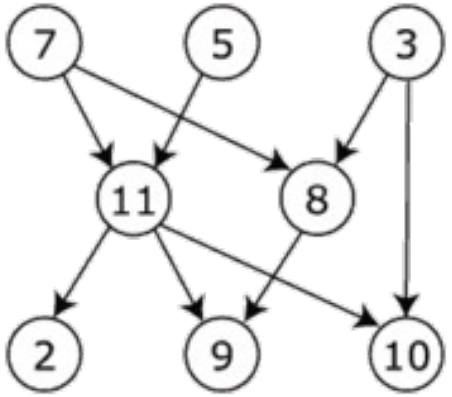
- O grafo abaixo pode ter muitas ordenações topológicas.
- Não necessariamente a única.

Ordenação Topológica

- O grafo abaixo pode ter muitas ordenações topológicas.
- Não necessariamente a única.
- Exemplo:

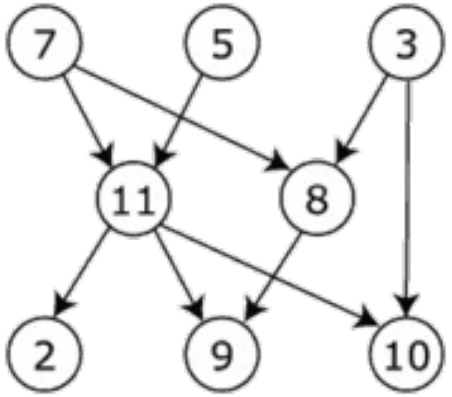
Ordenação Topológica

- O grafo abaixo pode ter muitas ordenações topológicas.
- Não necessariamente a única.
- Exemplo:



Ordenação Topológica

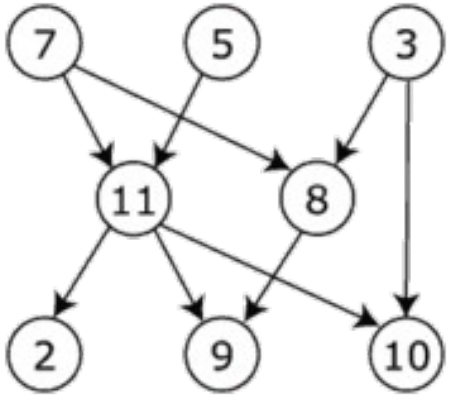
- O grafo abaixo pode ter muitas ordenações topológicas.
- Não necessariamente a única.
- Exemplo:



7, 5, 3, 11, 8, 2, 9, 10

Ordenação Topológica

- O grafo abaixo pode ter muitas ordenações topológicas.
- Não necessariamente a única.
- Exemplo:

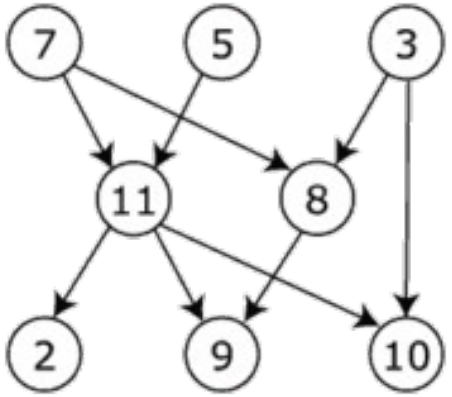


7, 5, 3, 11, 8, 2, 9, 10

3, 5, 7, 8, 11, 2, 9, 10

Ordenação Topológica

- O grafo abaixo pode ter muitas ordenações topológicas.
- Não necessariamente a única.
- Exemplo:



7, 5, 3, 11, 8, 2, 9, 10

3, 5, 7, 8, 11, 2, 9, 10

5, 7, 3, 8, 11, 10, 2, 9

Ordenação Topológica

Ordenação Topológica

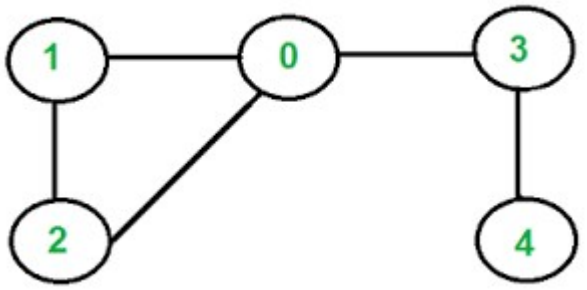
- Caso um grafo possua ciclos ou seja não direcionado, não é possível estabelecer uma relação de precedência entre os vértices e, portanto, é impossível estabelecer uma ordenação topológica.

Ordenação Topológica

- Caso um grafo possua ciclos ou seja não direcionado, não é possível estabelecer uma relação de precedência entre os vértices e, portanto, é impossível estabelecer uma ordenação topológica.
- Exemplo:

Ordenação Topológica

- Caso um grafo possua ciclos ou seja não direcionado, não é possível estabelecer uma relação de precedência entre os vértices e, portanto, é impossível estabelecer uma ordenação topológica.
- Exemplo:



Aplicações

Aplicações

- Roteiro de instalação de pacotes.

Aplicações

- Roteiro de instalação de pacotes.
- O make (makefile).

Aplicações

- Roteiro de instalação de pacotes.
- O make (makefile).
- Confecção de dicionários.

Aplicações

- Roteiro de instalação de pacotes.
- O make (makefile).
- Confecção de dicionários.
- Organização de banco de dados.

Aplicações

- Roteiro de instalação de pacotes.
- O make (makefile).
- Confecção de dicionários.
- Organização de banco de dados.
- Sistemas geográficos.

Aplicações

- Roteiro de instalação de pacotes.
- O make (makefile).
- Confecção de dicionários.
- Organização de banco de dados.
- Sistemas geográficos.
- Alocação de projetos.

Histórico

Histórico

- Algoritmos de ordenação topológica começaram a ser estudados na década de 60, no contexto da técnica PERT/CPM.

Histórico

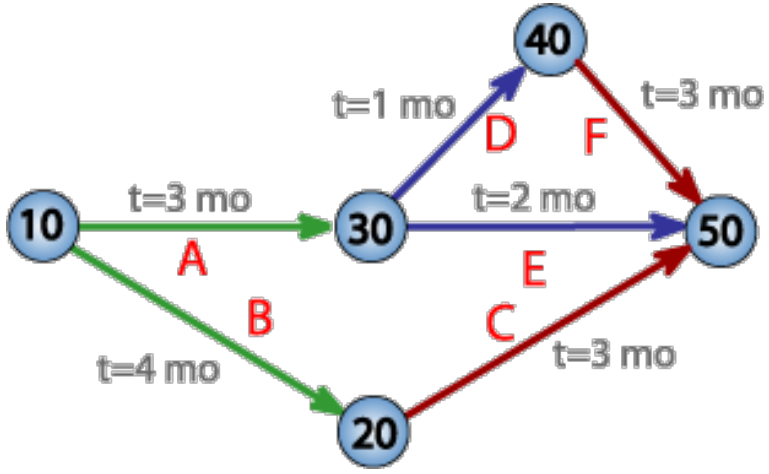
- Algoritmos de ordenação topológica começaram a ser estudados na década de 60, no contexto da técnica PERT/CPM.
- O PERT prevê o cálculo da duração de atividades complexas a partir da duração possível dessas atividades.

Histórico

- Algoritmos de ordenação topológica começaram a ser estudados na década de 60, no contexto da técnica PERT/CPM.
- O PERT prevê o cálculo da duração de atividades complexas a partir da duração possível dessas atividades.
- Exemplo:

Histórico

- Algoritmos de ordenação topológica começaram a ser estudados na década de 60, no contexto da técnica PERT/CPM.
- O PERT prevê o cálculo da duração de atividades complexas a partir da duração possível dessas atividades.
- Exemplo:



Histórico

- O CPM é um método de identificação do caminho crítico dada uma sequência de atividades, isto é, quais atividades de uma sequência não podem sofrer alteração de duração sem que isso reflita na duração final de um projeto.

Exemplo

Exemplo

- Um exemplo simples do encadeamento de atividades é relacionado na tabela abaixo para a fabricação de uma estante.

Exemplo

- Um exemplo simples do encadeamento de atividades é relacionado na tabela abaixo para a fabricação de uma estante.

Atividade	Discrição	Duração	Anterior	Posterior
A	Comprar tábuas	1 dia	–	3
B	Comprar parafusos	1 dia	–	5
C	Cortar as tábuas	2 dias	1	4
D	Pintar as tábuas	1 dia	2	5
E	Montar as tábuas com parafusos	1 dia	4,2	6
F	Transportar a estante	1 dia	5	–

Exemplo (cont.)

Exemplo (cont.)

- Considerando que a coluna duração tenha sido calculada pelo método PERT, é possível, com base na tabela, organizar um grafo do sequenciamento lógico das atividades de fabricação da estante:

Exemplo (cont.)

- Considerando que a coluna duração tenha sido calculada pelo método PERT, é possível, com base na tabela, organizar um grafo do sequenciamento lógico das atividades de fabricação da estante:
  - Os vértices representam eventos instantâneos que caracterizam o início e o término das atividades específicas nos arcos.

Exemplo (cont.)

- Considerando que a coluna duração tenha sido calculada pelo método PERT, é possível, com base na tabela, organizar um grafo do sequenciamento lógico das atividades de fabricação da estante:
  - Os vértices representam eventos instantâneos que caracterizam o início e o término das atividades específicas nos arcos.
  - A sequência das atividades é o principal ponto destacado.

Exemplo (cont.)

- Considerando que a coluna duração tenha sido calculada pelo método PERT, é possível, com base na tabela, organizar um grafo do sequenciamento lógico das atividades de fabricação da estante:
  - Os vértices representam eventos instantâneos que caracterizam o início e o término das atividades específicas nos arcos.
  - A sequência das atividades é o principal ponto destacado.
  - O arco pontilhado representa a dependência lógica da atividade de montagem das tábuas em relação à atividade de compra dos parafusos.

Exemplo (cont.)

- Considerando que a coluna duração tenha sido calculada pelo método PERT, é possível, com base na tabela, organizar um grafo do sequenciamento lógico das atividades de fabricação da estante:
  - Os vértices representam eventos instantâneos que caracterizam o início e o término das atividades específicas nos arcos.
  - A sequência das atividades é o principal ponto destacado.
  - O arco pontilhado representa a dependência lógica da atividade de montagem das tábuas em relação à atividade de compra dos parafusos.
  - Essa atividade é denominada de atividade fantasma e sua duração é igual a zero (instantânea).

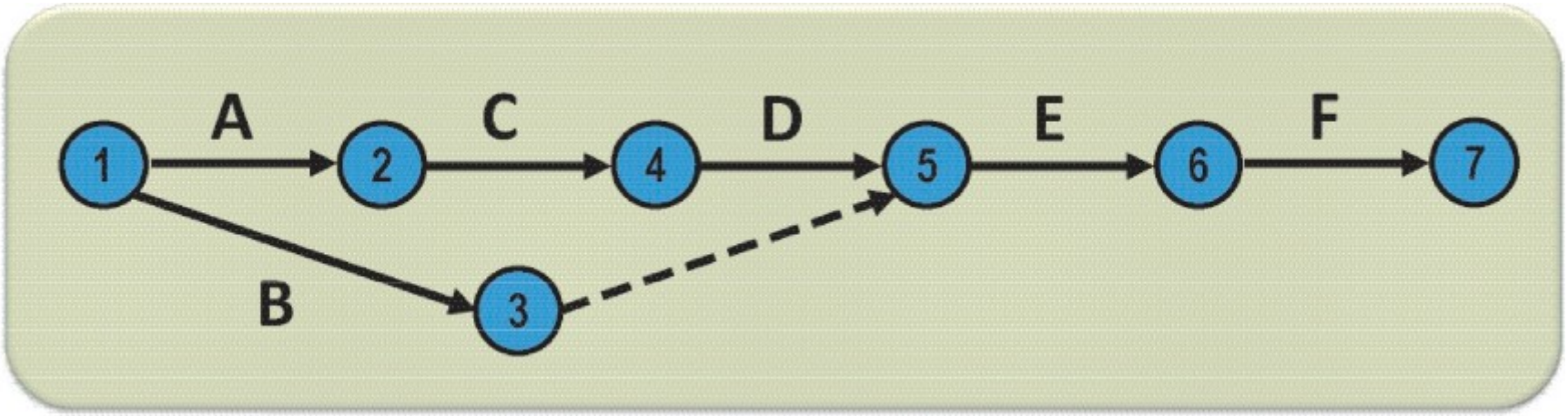
Exemplo (cont.)

Exemplo (cont.)

Atividade	Discrição	Duração	Anterior	Posterior
A	Comprar tábuas	1 dia	–	3
B	Comprar parafusos	1 dia	–	5
C	Cortar as tábuas	2 dias	1	4
D	Pintar as tábuas	1 dia	2	5
E	Montar as tábuas com parafusos	1 dia	4,2	6
F	Transportar a estante	1 dia	5	–

Exemplo (cont.)

Atividade	Discrição	Duração	Anterior	Posterior
A	Comprar tábuas	1 dia	–	3
B	Comprar parafusos	1 dia	–	5
C	Cortar as tábuas	2 dias	1	4
D	Pintar as tábuas	1 dia	2	5
E	Montar as tábuas com parafusos	1 dia	4,2	6
F	Transportar a estante	1 dia	5	–



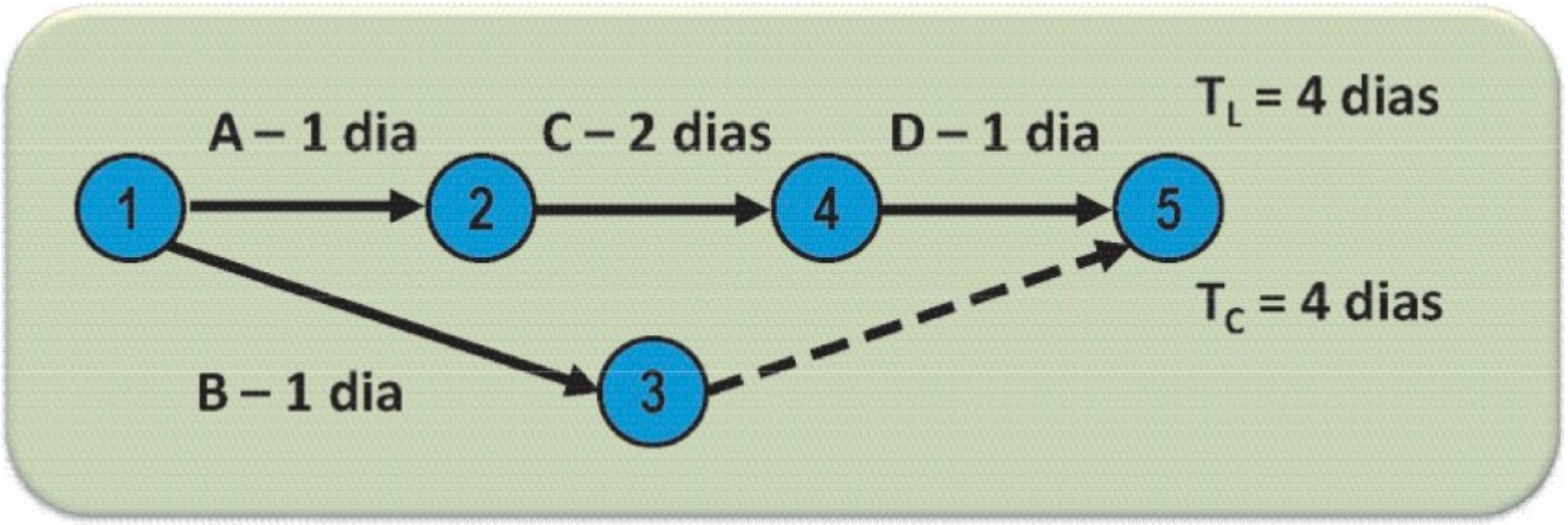
Exemplo (cont.)

Exemplo (cont.)

- É possível ainda representar a duração de cada atividade representada no grafo e calcular o tempo mais curto (Tc) e o tempo mais longo de conclusão das tarefas (Tl).

Exemplo (cont.)

- É possível ainda representar a duração de cada atividade representada no grafo e calcular o tempo mais curto (Tc) e o tempo mais longo de conclusão das tarefas (Tl).



Exemplo (cont.)

Exemplo (cont.)

- Alternativamente, o grafo resultante pode ser modelado de outra maneira:

Exemplo (cont.)

- Alternativamente, o grafo resultante pode ser modelado de outra maneira:
  - Tarefas a serem desempenhadas em um projeto são representadas por vértices.

Exemplo (cont.)

- Alternativamente, o grafo resultante pode ser modelado de outra maneira:
  - Tarefas a serem desempenhadas em um projeto são representadas por vértices.
  - Existe um arco entre as tarefas ***v*** e ***w***, caso a tarefa ***v*** obrigatoriamente deva ser terminada antes que a tarefa ***w*** comece.

Exemplo (cont.)

- Alternativamente, o grafo resultante pode ser modelado de outra maneira:
  - Tarefas a serem desempenhadas em um projeto são representadas por vértices.
  - Existe um arco entre as tarefas ***v*** e ***w***, caso a tarefa ***v*** obrigatoriamente deva ser terminada antes que a tarefa ***w*** comece.
  - Uma ordenação topológica dos vértices fornece uma maneira de realizar todas as tarefas sem violar as precedências entre tarefas.

Algoritmos

Algoritmos

- Existem diferentes algoritmos para ordenações topológicas em grafos.

Algoritmos

- Existem diferentes algoritmos para ordenações topológicas em grafos.
- Os algoritmos de melhor desempenho possuem complexidade linear.

Algoritmos

- Existem diferentes algoritmos para ordenações topológicas em grafos.
- Os algoritmos de melhor desempenho possuem complexidade linear.
- Algoritmos mais utilizados:

Algoritmos

- Existem diferentes algoritmos para ordenações topológicas em grafos.
- Os algoritmos de melhor desempenho possuem complexidade linear.
- Algoritmos mais utilizados:
  - Algoritmo de Kahn.

Algoritmos

- Existem diferentes algoritmos para ordenações topológicas em grafos.
- Os algoritmos de melhor desempenho possuem complexidade linear.
- Algoritmos mais utilizados:
  - Algoritmo de Kahn.
  - Busca em profundidade (DFS).

Algoritmo de Kahn

Algoritmo de Kahn

- O algoritmo de Kahn data de 1962.

Algoritmo de Kahn

- O algoritmo de Kahn data de 1962.
- Possui como princípio determinar a cada instante os vértices que não possuam arcos de entrada e inserir na solução.

Algoritmo de Kahn

- O algoritmo de Kahn data de 1962.
- Possui como princípio determinar a cada instante os vértices que não possuam arcos de entrada e inserir na solução.
- A cada vértice inserido na solução, todos seus arcos correspondentes são removidos do grafo.

Algoritmo de Kahn

- O algoritmo de Kahn data de 1962.
- Possui como princípio determinar a cada instante os vértices que não possuam arcos de entrada e inserir na solução.
- A cada vértice inserido na solução, todos seus arcos correspondentes são removidos do grafo.
- Também detecta a existência de ciclos no grafo.

Algoritmo de Kahn

Algoritmo de Kahn

- Terminologia:

Algoritmo de Kahn

- Terminologia:
  - L: lista que conterá os elementos da ordenação topológica.

Algoritmo de Kahn

- Terminologia:
  - L: lista que conterá os elementos da ordenação topológica.
  - S: conjunto de vértices que não possuem arcos de entrada.

Algoritmo de Kahn

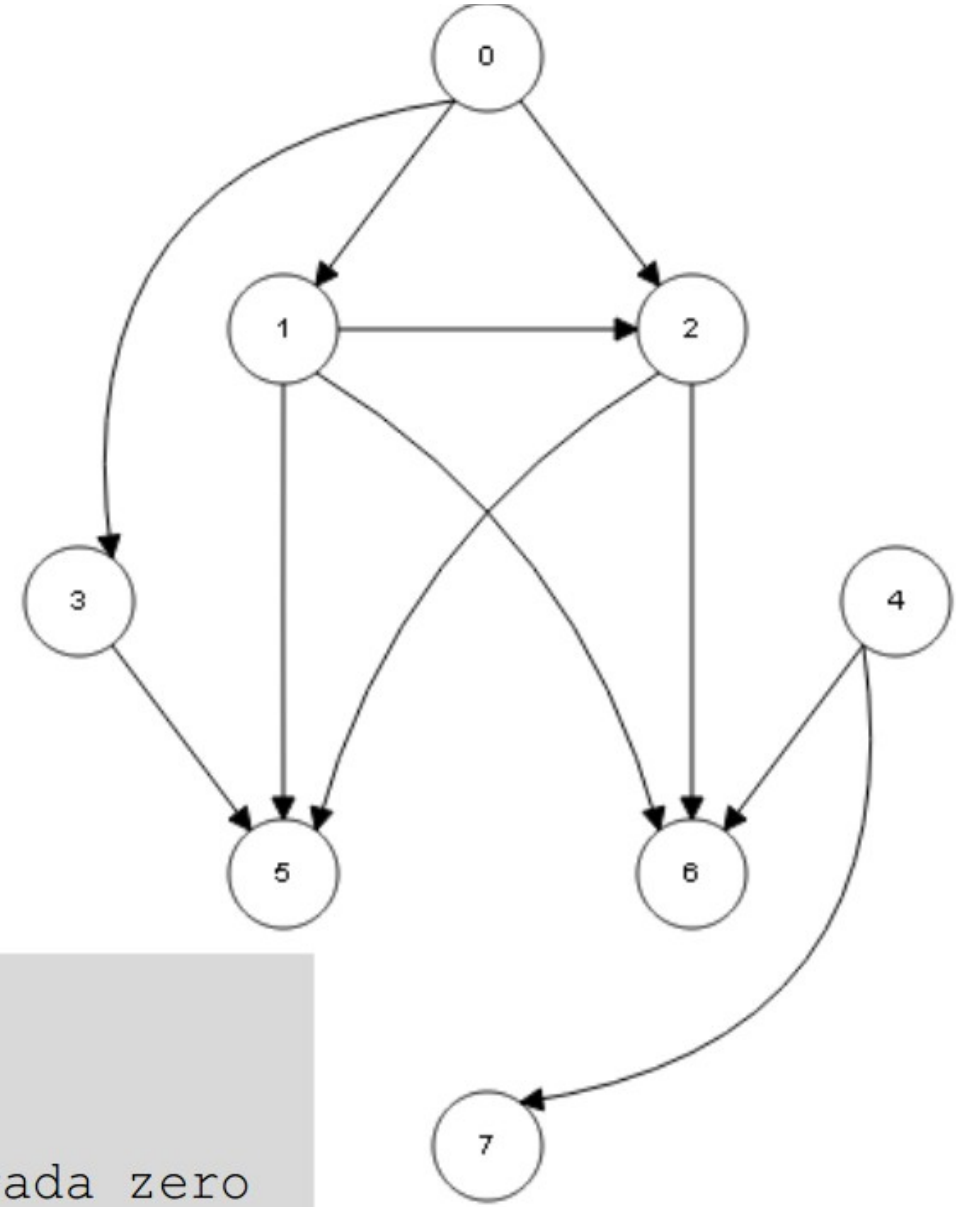
- Terminologia:
  - L: lista que conterá os elementos da ordenação topológica.
  - S: conjunto de vértices que não possuem arcos de entrada.
  - I: lista que conterá os elementos (vértices) com o grau de incidência.

Algoritmo de Kahn

```
OrdenacaoTopologicaKahn(G)
  L ← ∅
  for each uv ∈ A[G]
    I[v] ← I[v] + 1
  S ← vertices com grau de entrada zero (S = pilha)
  while S ≠ ∅
    v ← unstack(S)
    stack(v, L)
    for each u ∈ Adj[v]
      I[u] ← I[u] - 1
      if I[u] = 0
        stack(u, S)
  return L
```

Algoritmo de Kahn

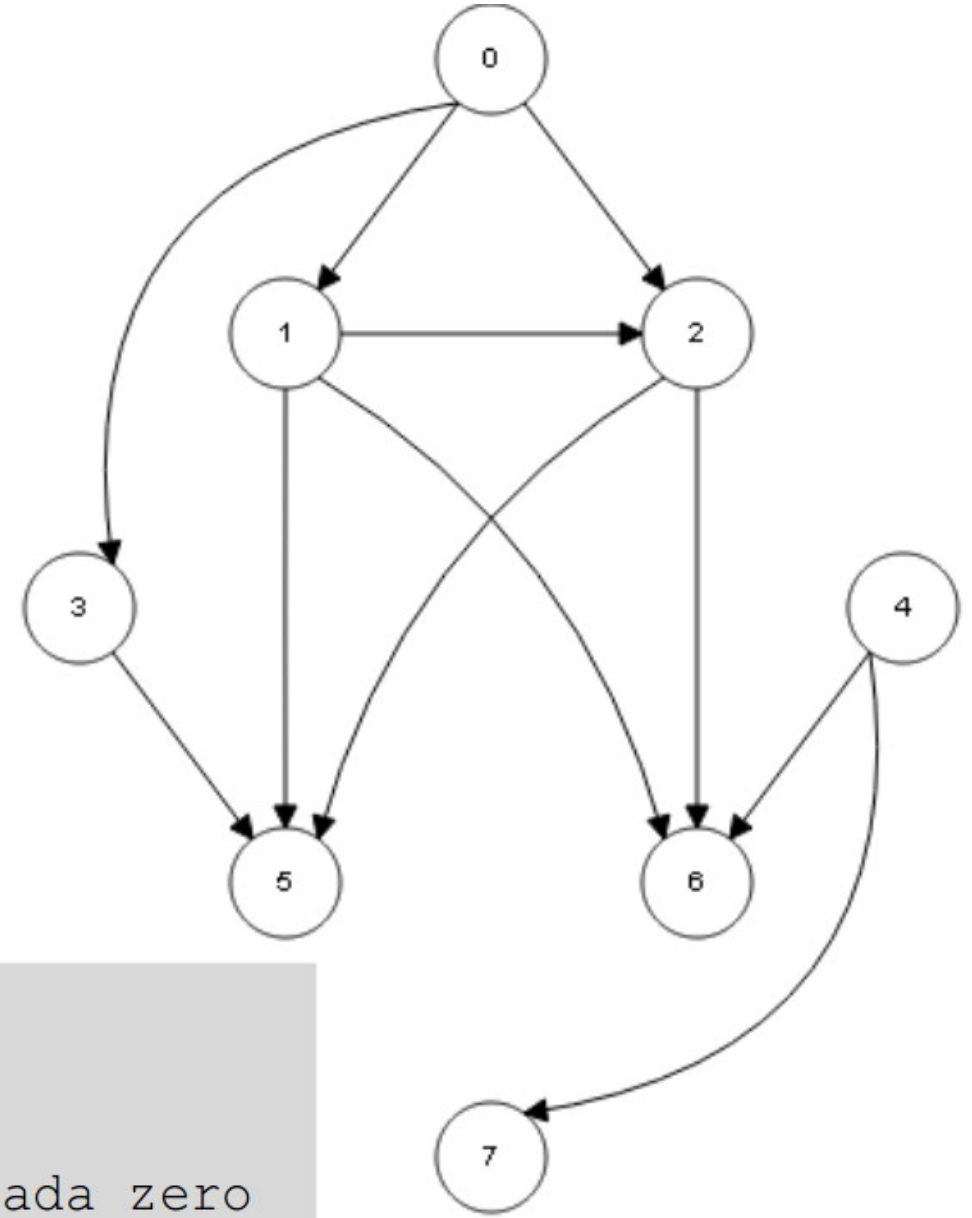
	I	S	L
0	0	/	/
1	0		
2	0		
3	0		
4	0		
5	0		
6	0		
7	0		




```
L ← ∅
for each uv ∈ A[G]
  I[v] ← I[v] + 1
S ← vertices com grau de entrada zero
```

Algoritmo de Kahn

	I	S	L
0	0	/	/
1	1		
2	2		
3	1		
4	0		
5	3		
6	3		
7	1		



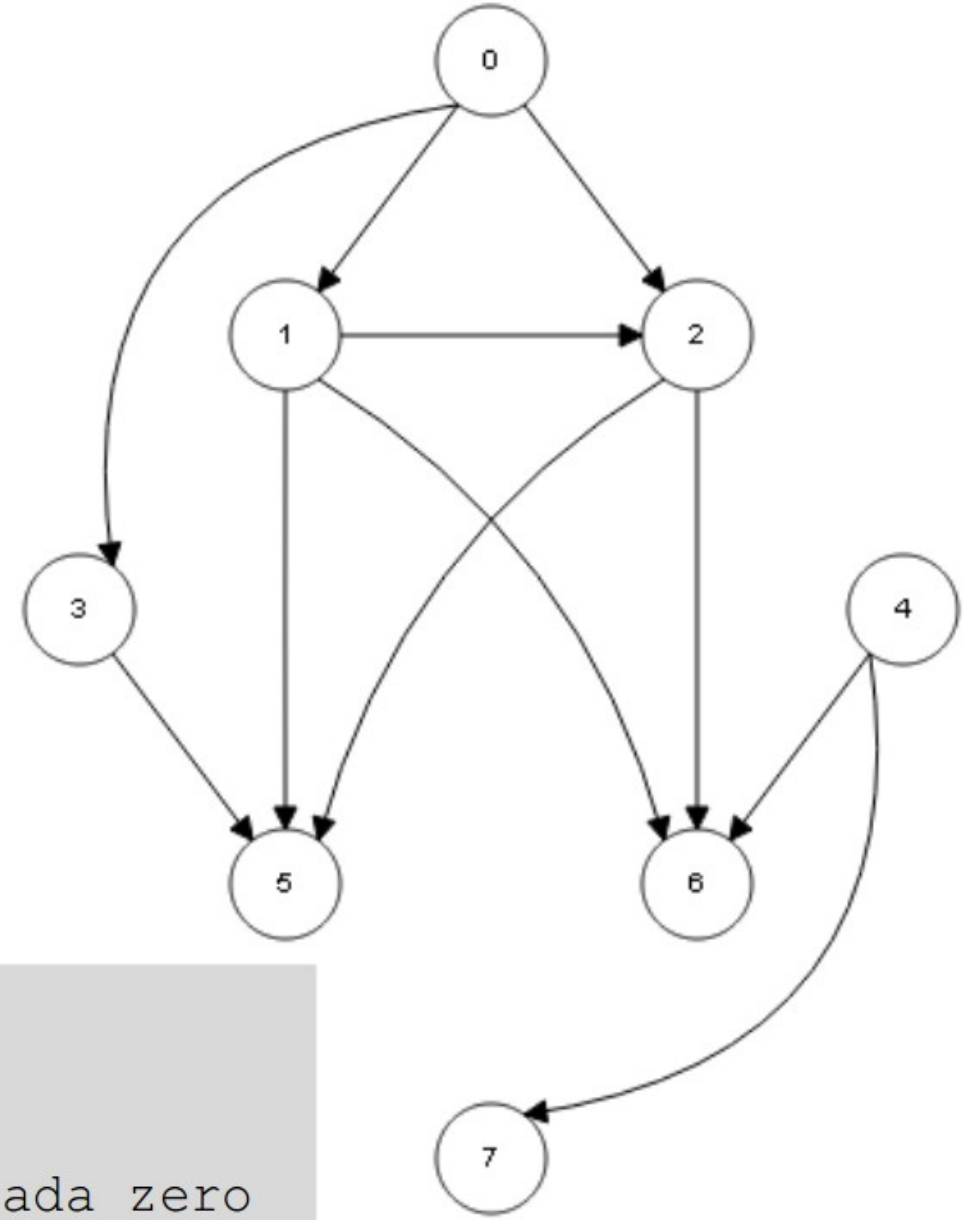

```
L ← ∅
for each uv ∈ A[G]
  I[v] ← I[v] + 1
S ← vertices com grau de entrada zero
```

Algoritmo de Kahn

	I
0	0
1	1
2	2
3	1
4	0
5	3
6	3
7	1

S
0
4

L
/

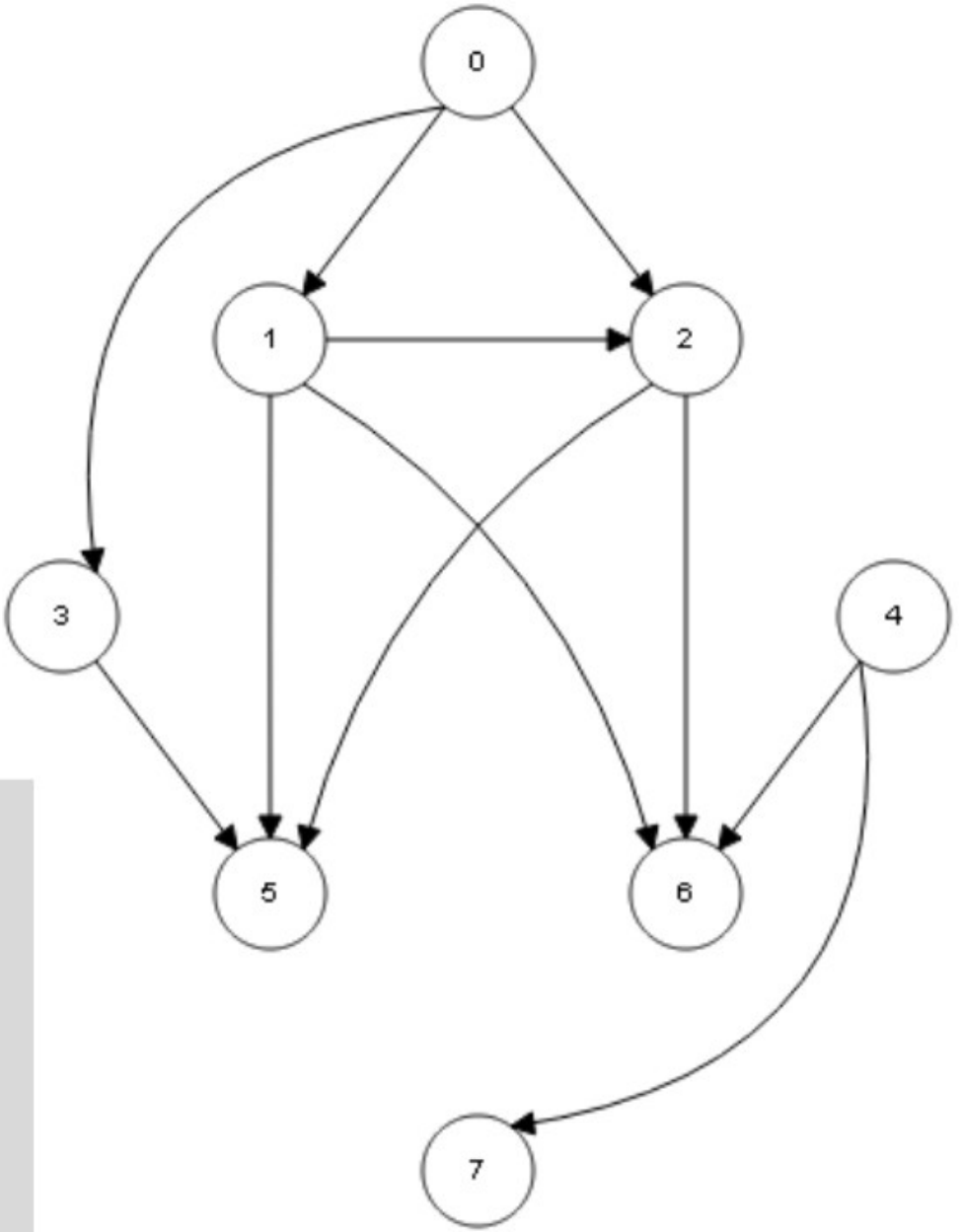



```
L ← ∅
for each uv ∈ A[G]
  I[v] ← I[v] + 1
S ← vertices com grau de entrada zero
```

Algoritmo de Kahn

	I	S	L
0	0	0	4
1	1		
2	2		
3	1		
4	0		
5	3		
6	3		
7	1		

```
while S ≠ ∅
  v ← unstack(S)
  stack(v, L)
  for each u ∈ Adj[v]
    I[u] ← I[u] - 1
    if I[u] = 0
      stack(u, S)
```

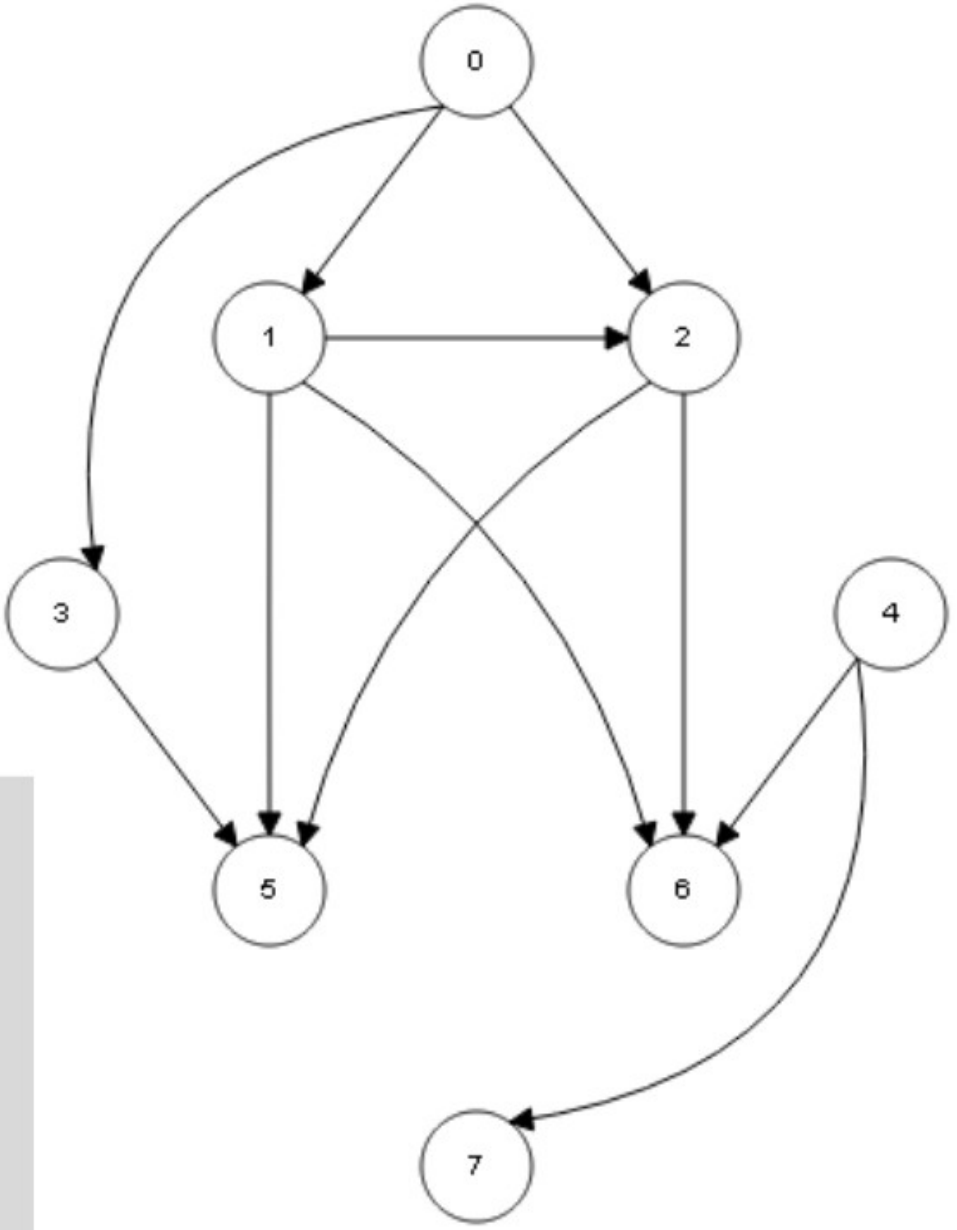


Algoritmo de Kahn

	I
0	0
1	1
2	2
3	1
4	0
5	3
6	3
7	1

S
0

L
4

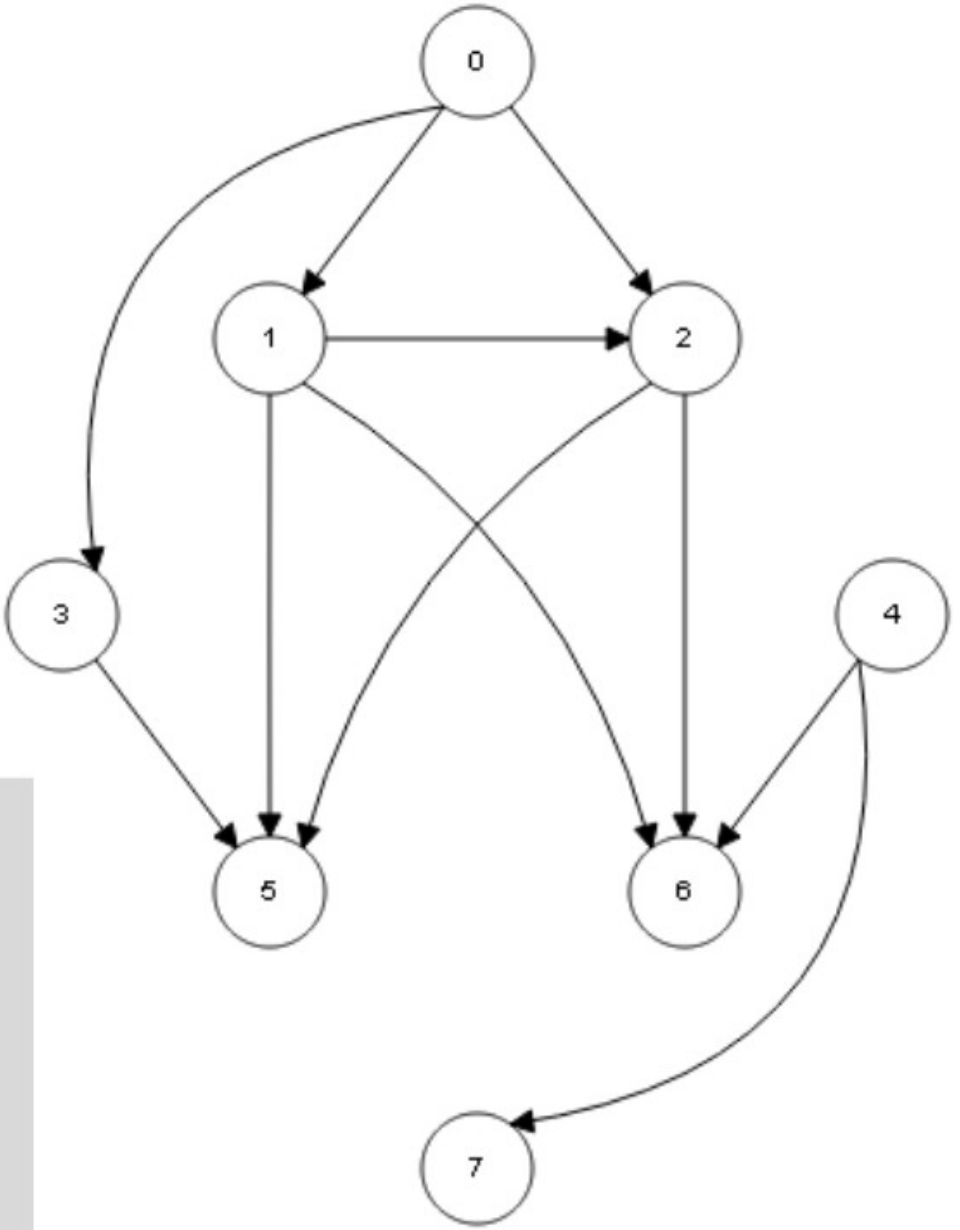



```
while S ≠ ∅
  v ← unstack(S)
  stack(v, L)
  for each u ∈ Adj[v]
    I[u] ← I[u] - 1
    if I[u] = 0
      stack(u, S)
```

Algoritmo de Kahn

	I	S	L
0	0	0	4
1	1		
2	2		
3	1		
4	0		
5	3		
6	2		
7	0		

```
while S ≠ ∅
  v ← unstack(S)
  stack(v, L)
  for each u ∈ Adj[v]
    I[u] ← I[u] - 1
    if I[u] = 0
      stack(u, S)
```

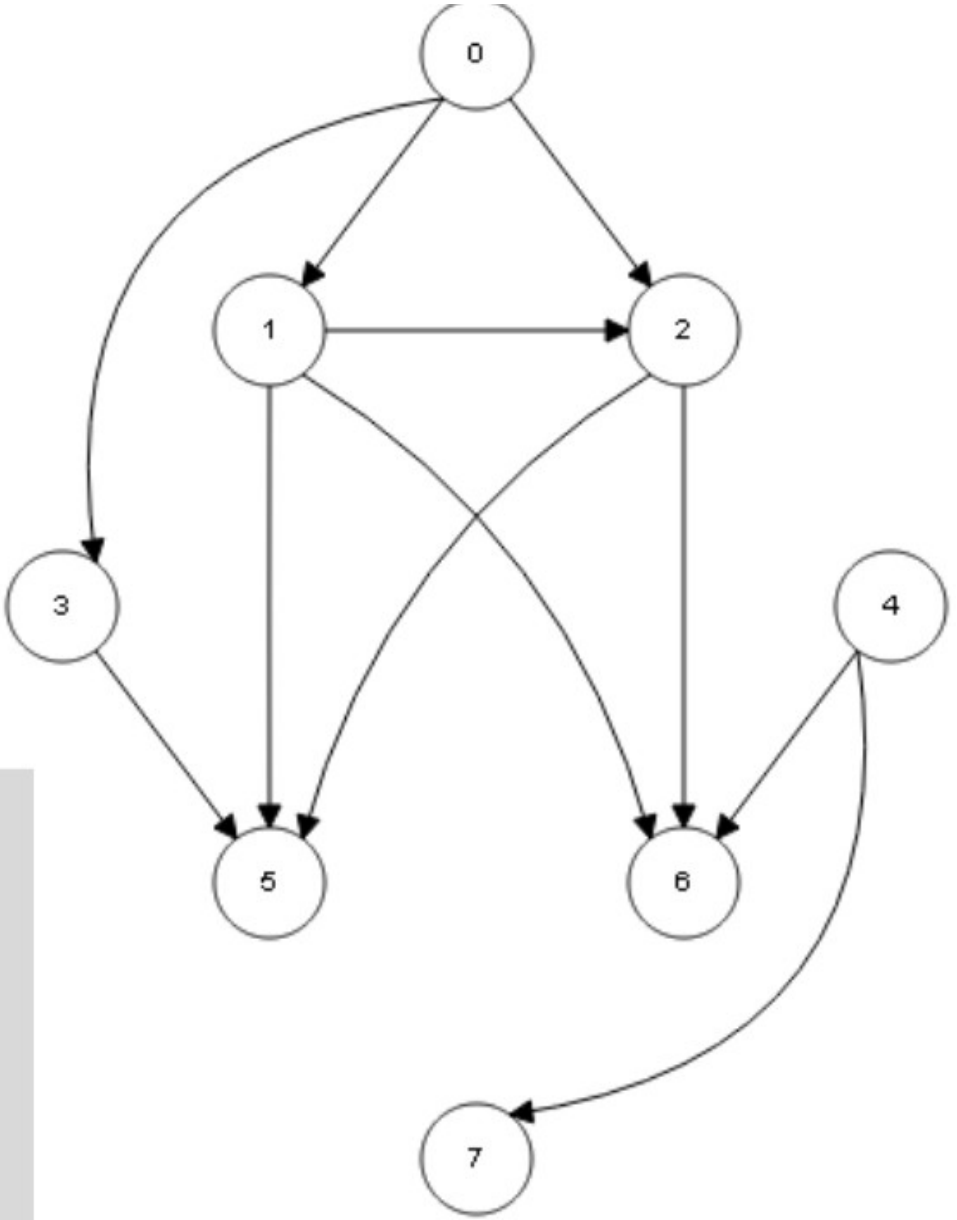


Algoritmo de Kahn

	I
0	0
1	1
2	2
3	1
4	0
5	3
6	2
7	0

S
0
7

L
4



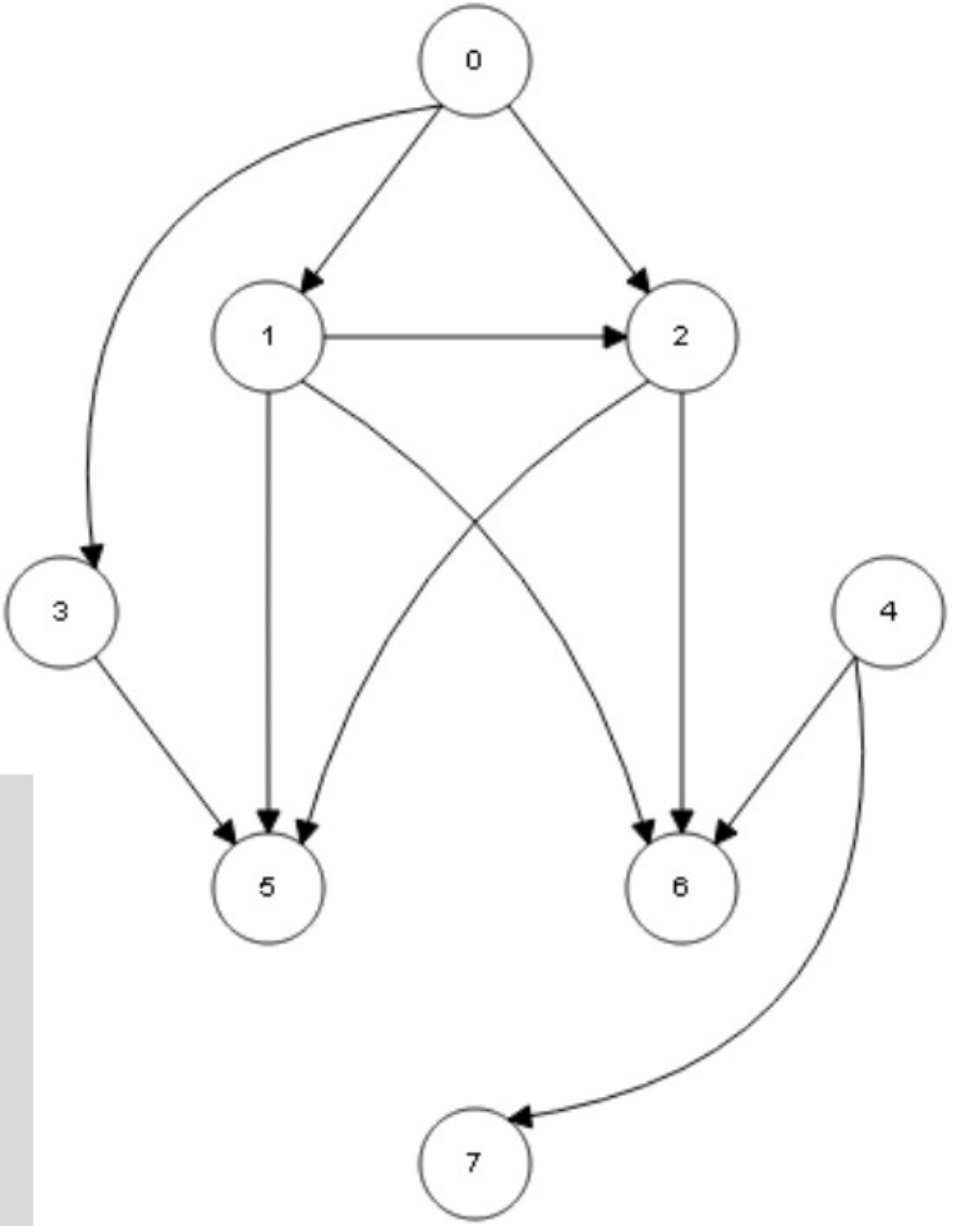

```
while S ≠ Ø
  v ← unstack(S)
  stack(v, L)
  for each u ∈ Adj[v]
    I[u] ← I[u] - 1
    if I[u] = 0
      stack(u, S)
```

Algoritmo de Kahn

	I
0	0
1	1
2	2
3	1
4	0
5	3
6	2
7	0

S
0

L
4
7



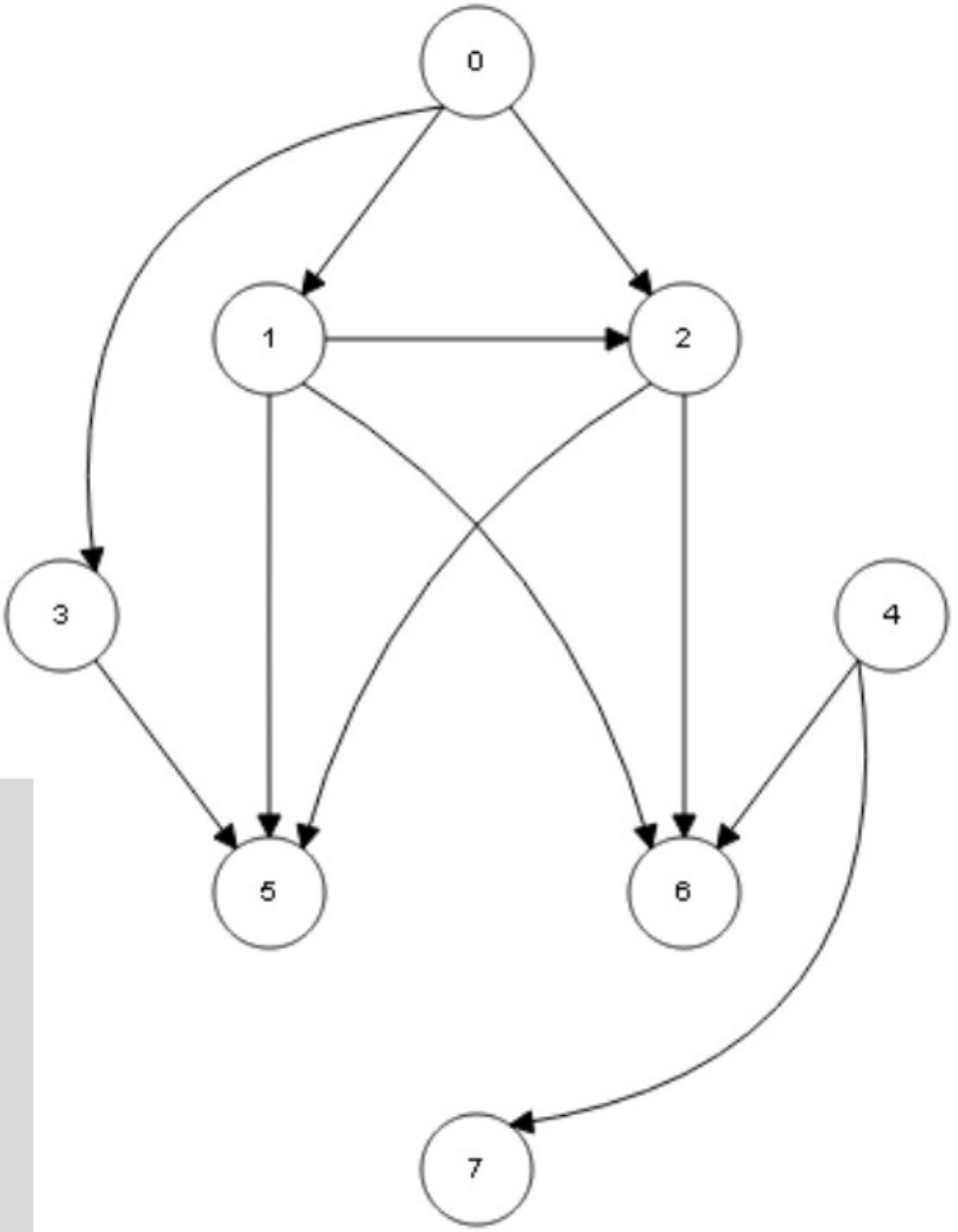

```
while S ≠ ∅
  v ← unstack(S)
  stack(v, L)
  for each u ∈ Adj[v]
    I[u] ← I[u] - 1
    if I[u] = 0
      stack(u, S)
```

Algoritmo de Kahn

	I
0	0
1	1
2	2
3	1
4	0
5	3
6	2
7	0

S
/

L
4
7
0



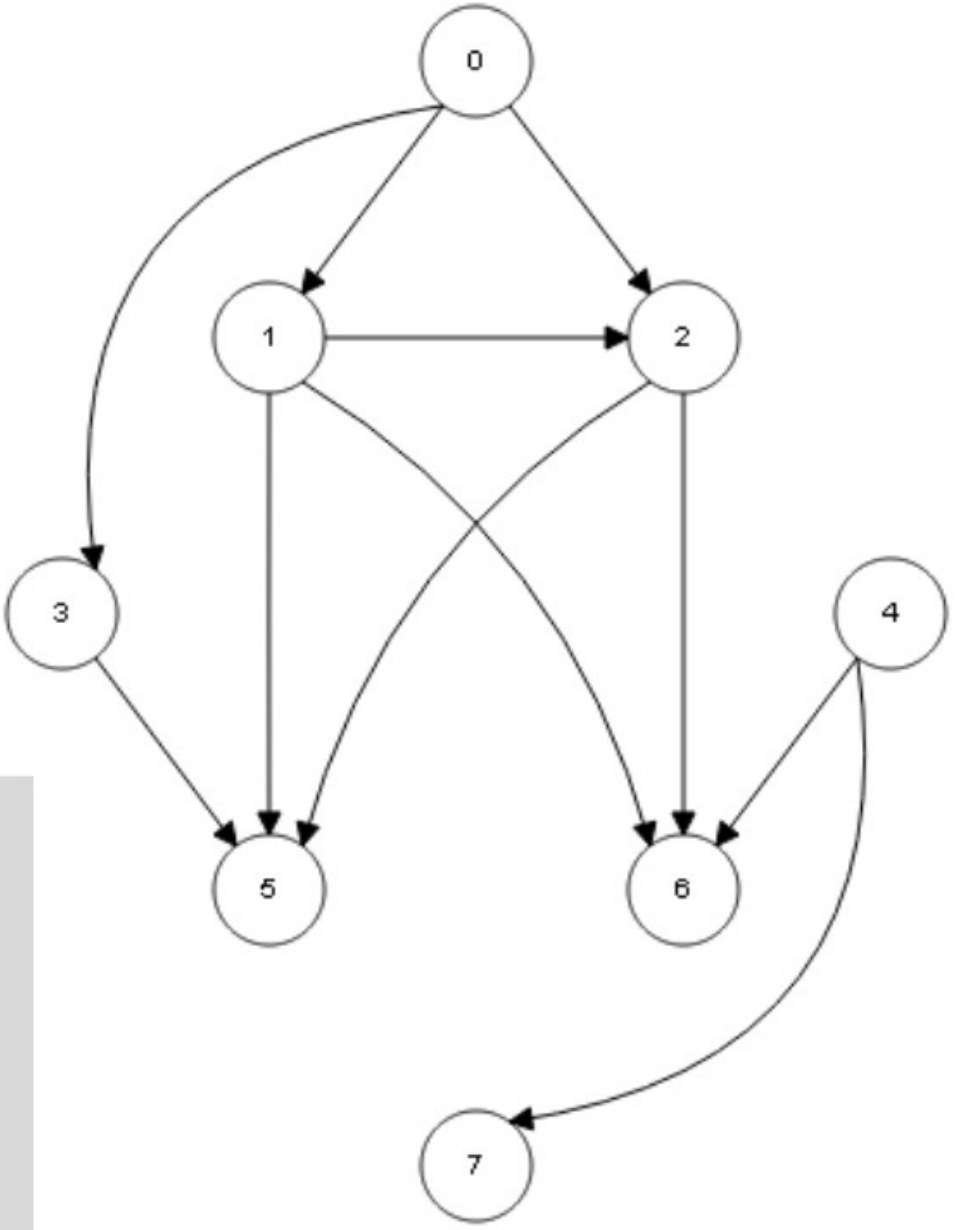

```
while S ≠ ∅
  v ← unstack(S)
  stack(v, L)
  for each u ∈ Adj[v]
    I[u] ← I[u] - 1
    if I[u] = 0
      stack(u, S)
```

Algoritmo de Kahn

	I
0	0
1	1
2	2
3	1
4	0
5	3
6	2
7	0

S
/

L
4
7
0



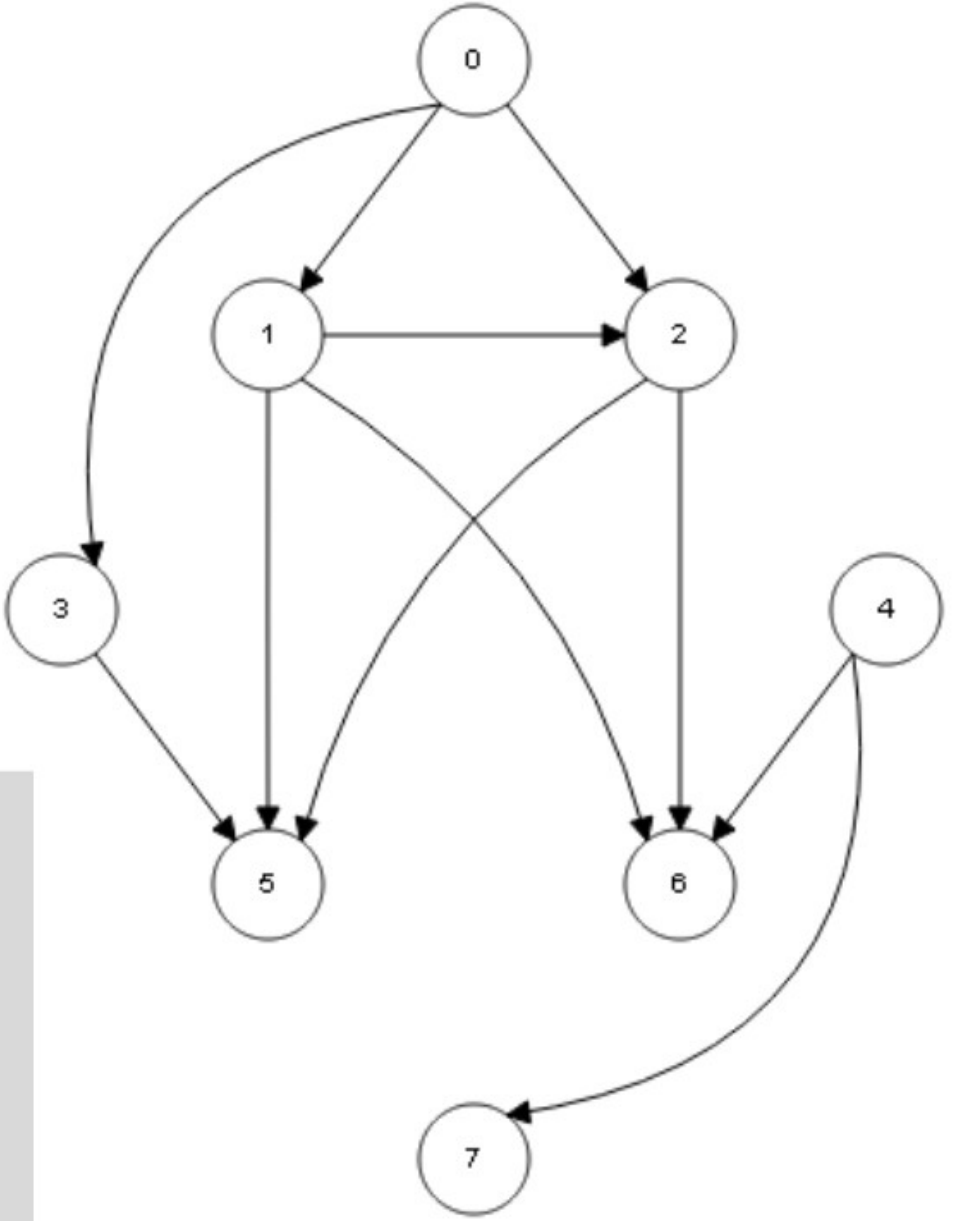

```
while S ≠ ∅
  v ← unstack(S)
  stack(v, L)
  for each u ∈ Adj[v]
    I[u] ← I[u] - 1
    if I[u] = 0
      stack(u, S)
```

Algoritmo de Kahn

	I
0	0
1	0
2	1
3	0
4	0
5	3
6	2
7	0

S
/

L
4
7
0



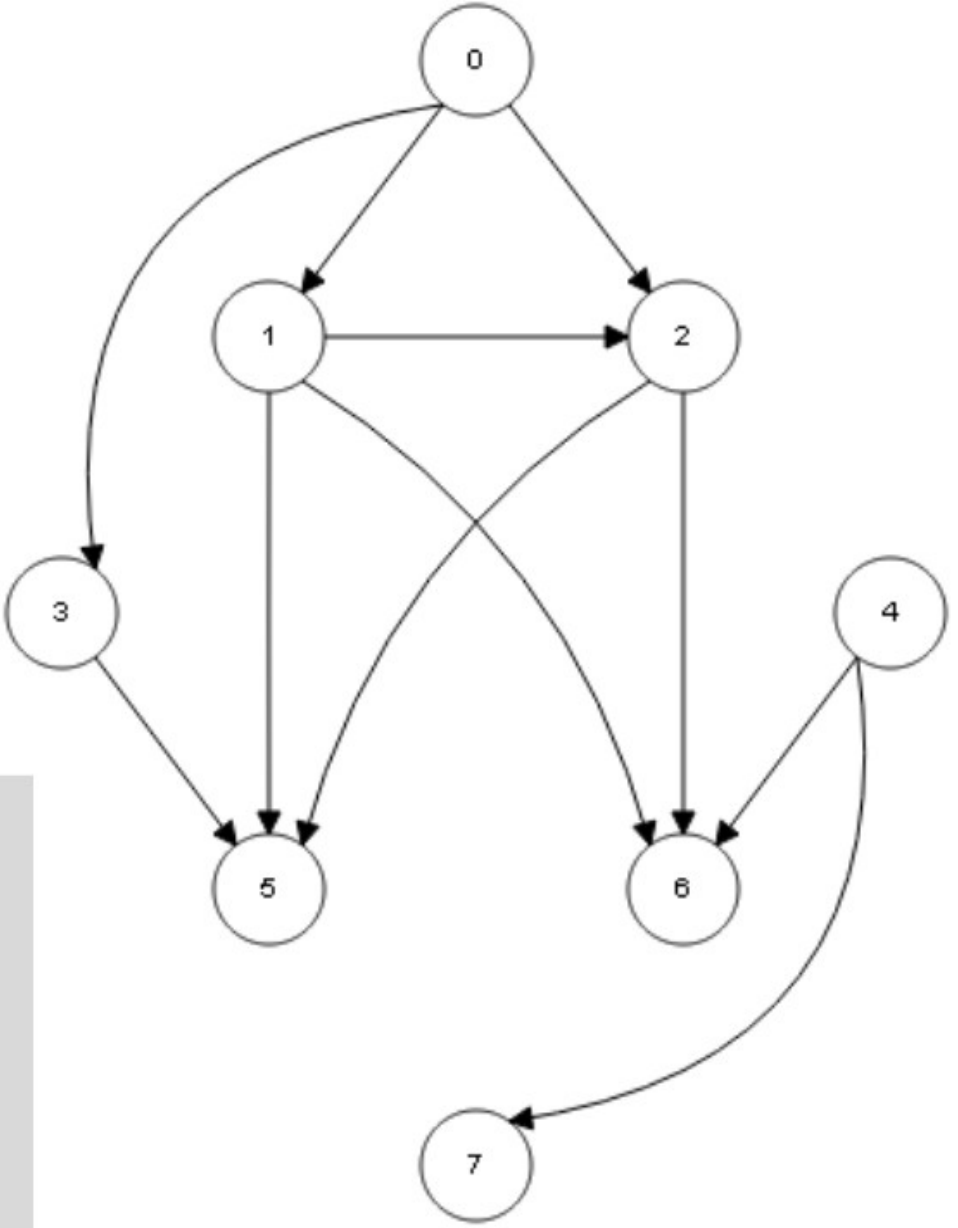

```
while S ≠ ∅
  v ← unstack(S)
  stack(v, L)
  for each u ∈ Adj[v]
    I[u] ← I[u] - 1
    if I[u] = 0
      stack(u, S)
```

Algoritmo de Kahn

	I
0	0
1	0
2	1
3	0
4	0
5	3
6	2
7	0

S
1
3

L
4
7
0



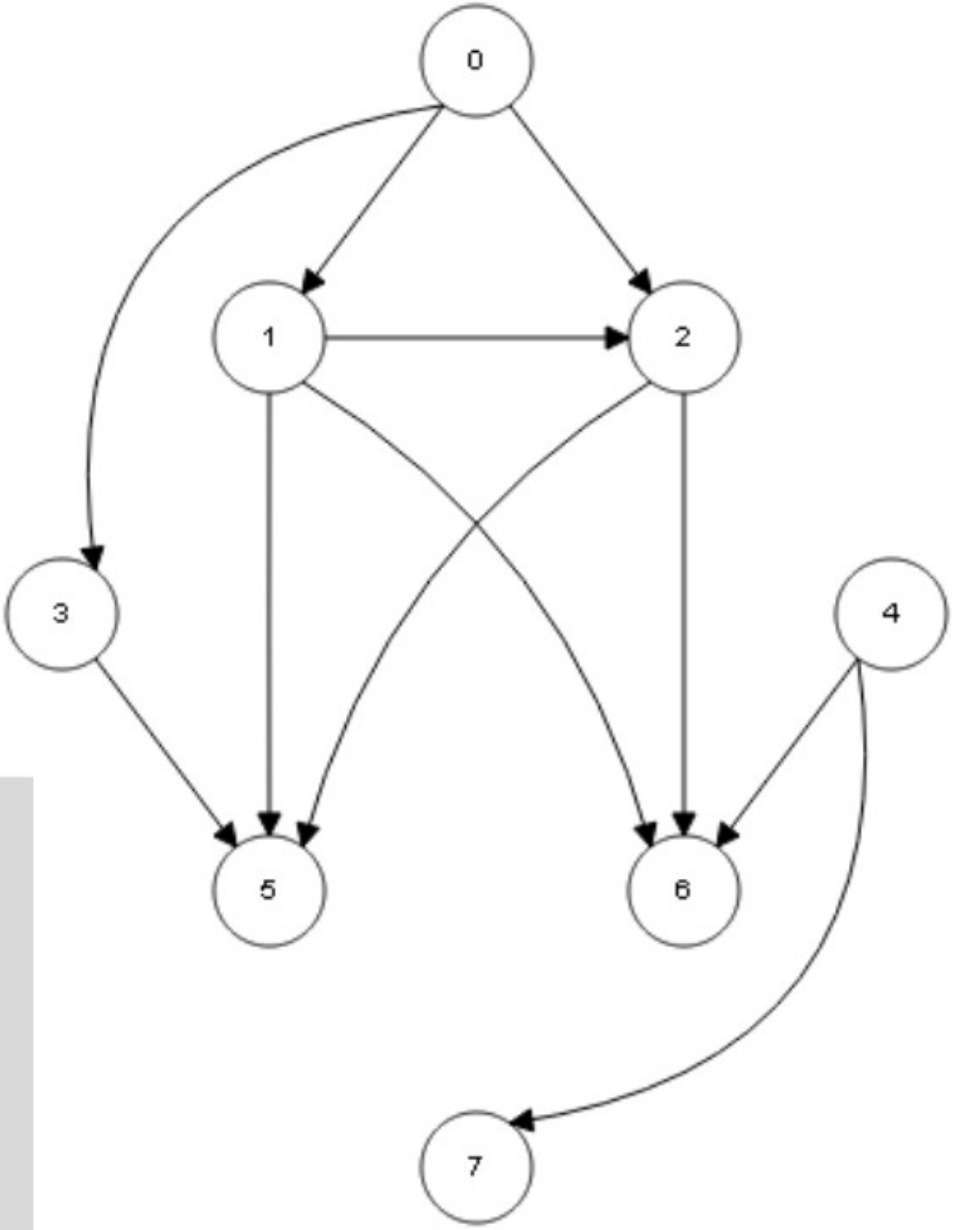

```
while S ≠ ∅
  v ← unstack(S)
  stack(v, L)
  for each u ∈ Adj[v]
    I[u] ← I[u] - 1
    if I[u] = 0
      stack(u, S)
```

Algoritmo de Kahn

	I
0	0
1	0
2	1
3	0
4	0
5	3
6	2
7	0

S
1

L
4
7
0
3



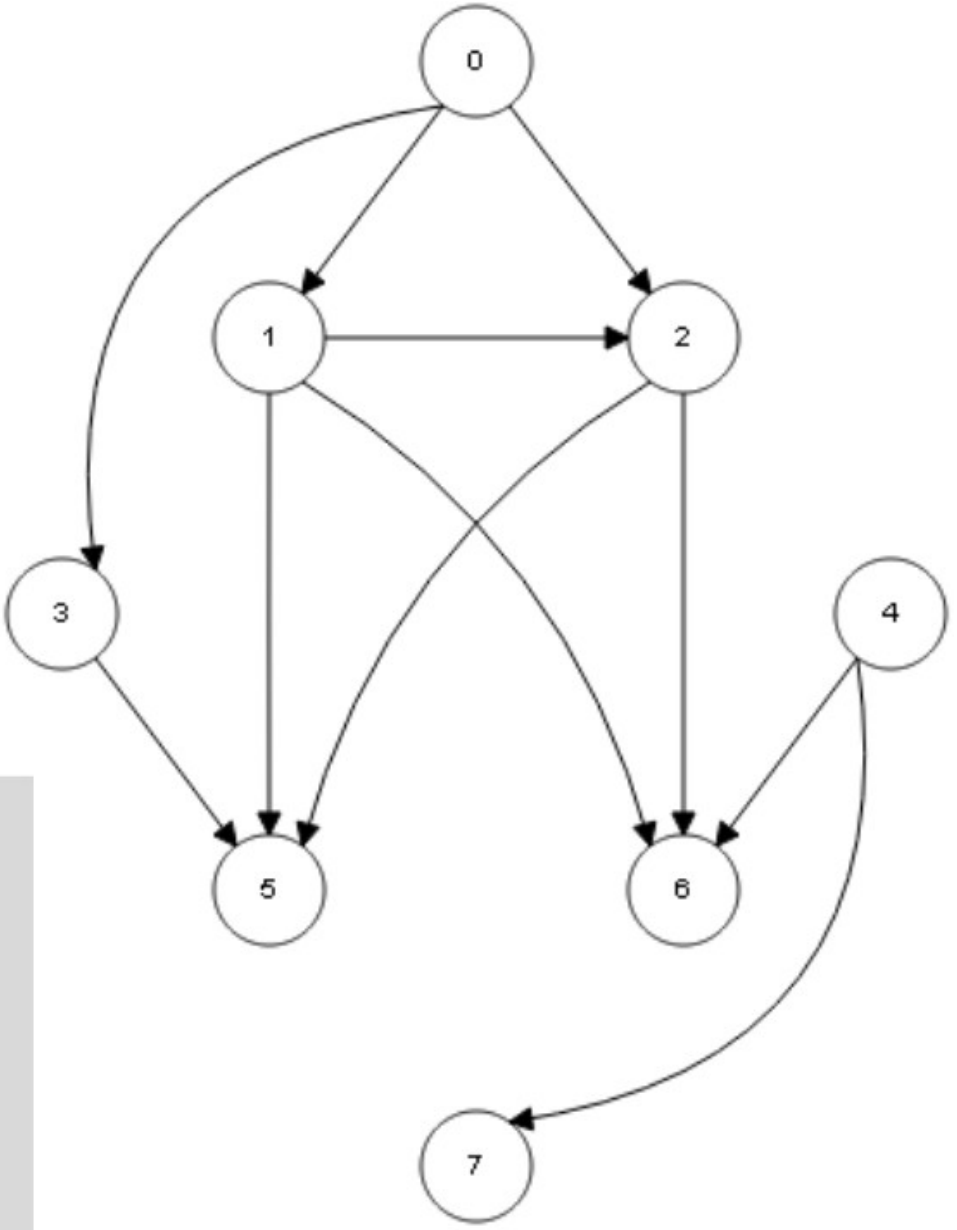

```
while S ≠ Ø
  v ← unstack(S)
  stack(v, L)
  for each u ∈ Adj[v]
    I[u] ← I[u] - 1
    if I[u] = 0
      stack(u, S)
```

Algoritmo de Kahn

	I
0	0
1	0
2	1
3	0
4	0
5	3
6	2
7	0

S
1

L
4
7
0
3



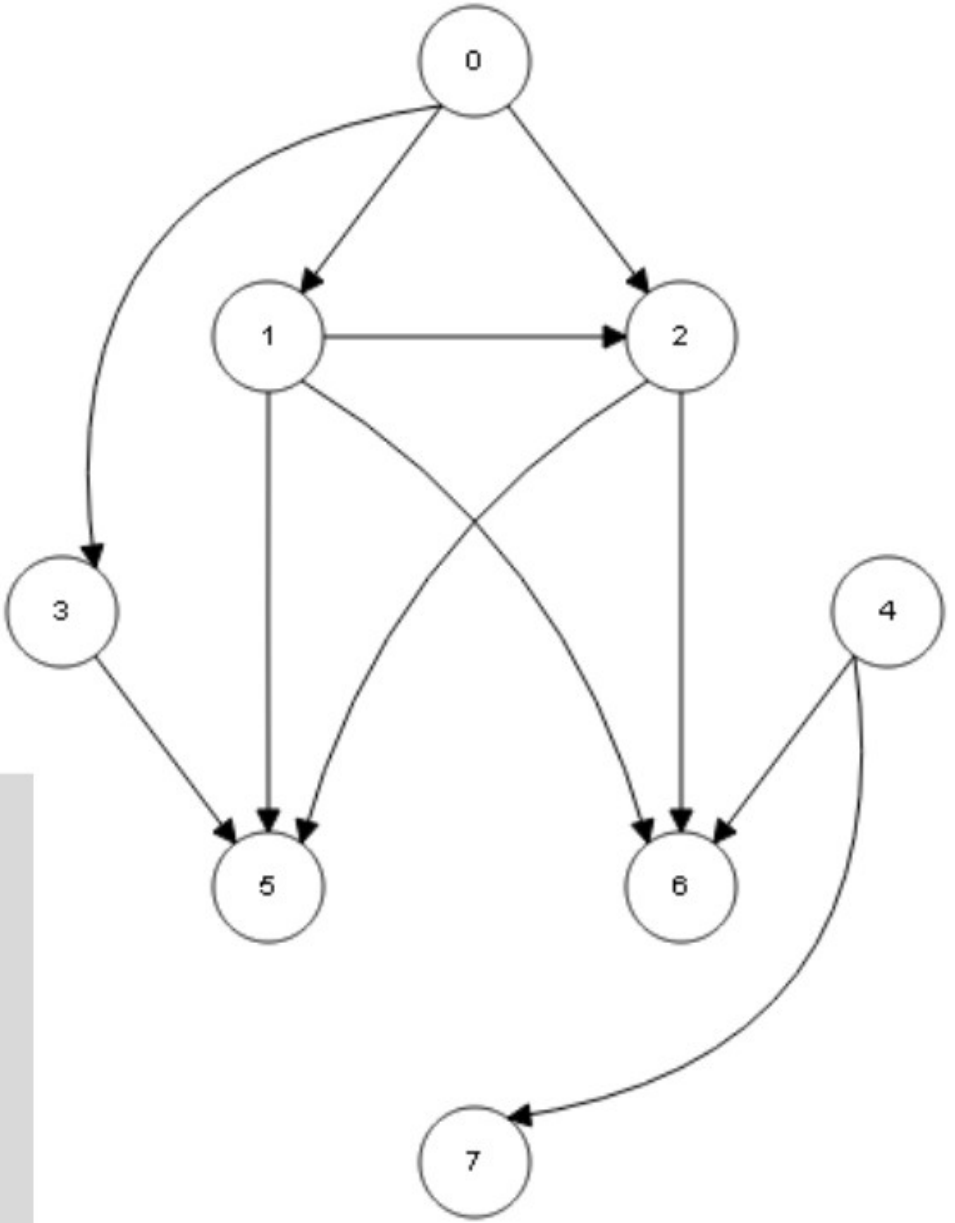

```
while S ≠ ∅
  v ← unstack(S)
  stack(v, L)
  for each u ∈ Adj[v]
    I[u] ← I[u] - 1
    if I[u] = 0
      stack(u, S)
```

Algoritmo de Kahn

	I
0	0
1	0
2	1
3	0
4	0
5	2
6	2
7	0

S
1

L
4
7
0
3



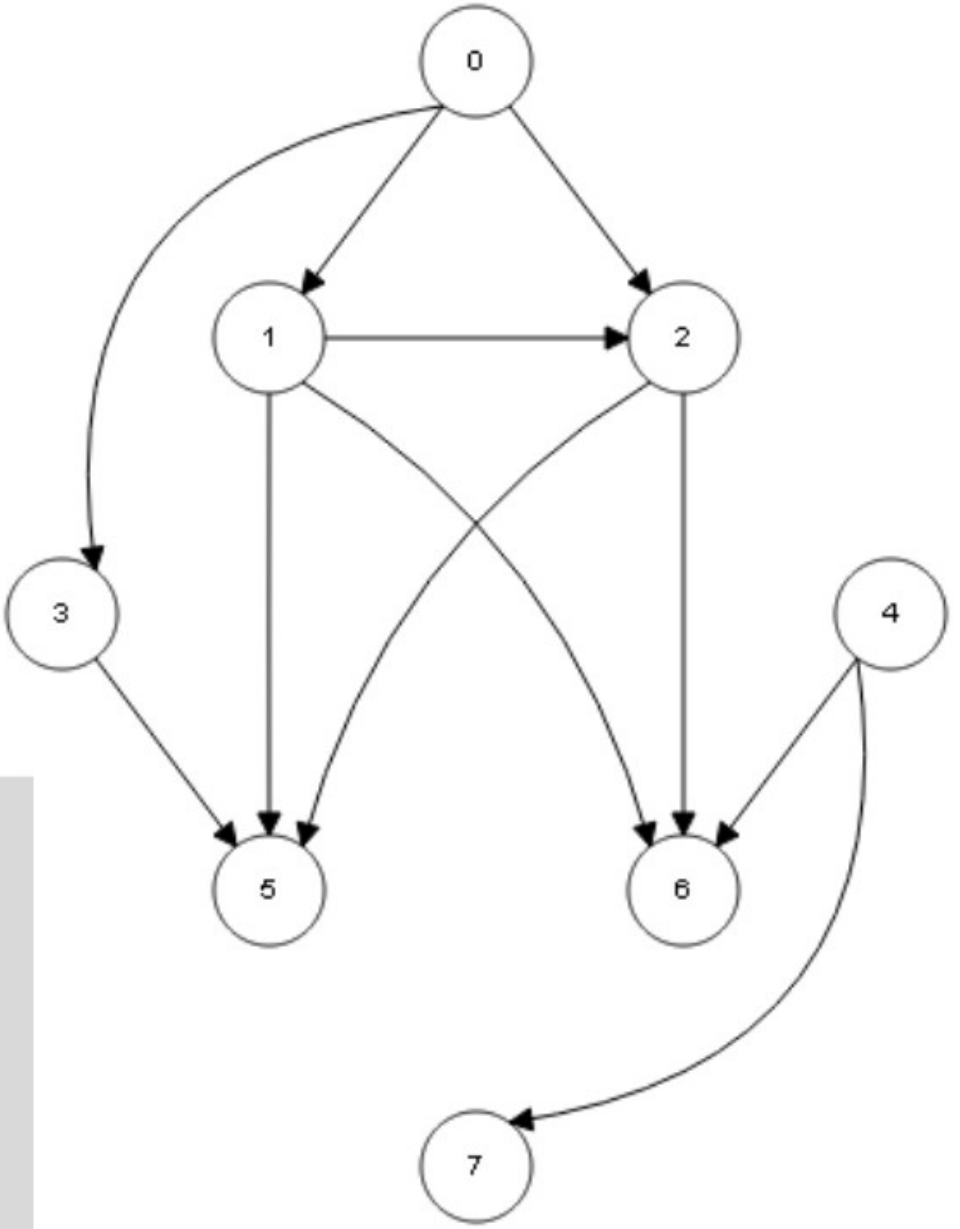

```
while S ≠ ∅
  v ← unstack(S)
  stack(v, L)
  for each u ∈ Adj[v]
    I[u] ← I[u] - 1
    if I[u] = 0
      stack(u, S)
```

Algoritmo de Kahn

	I
0	0
1	0
2	1
3	0
4	0
5	2
6	2
7	0

S
/

L
4
7
0
3
1



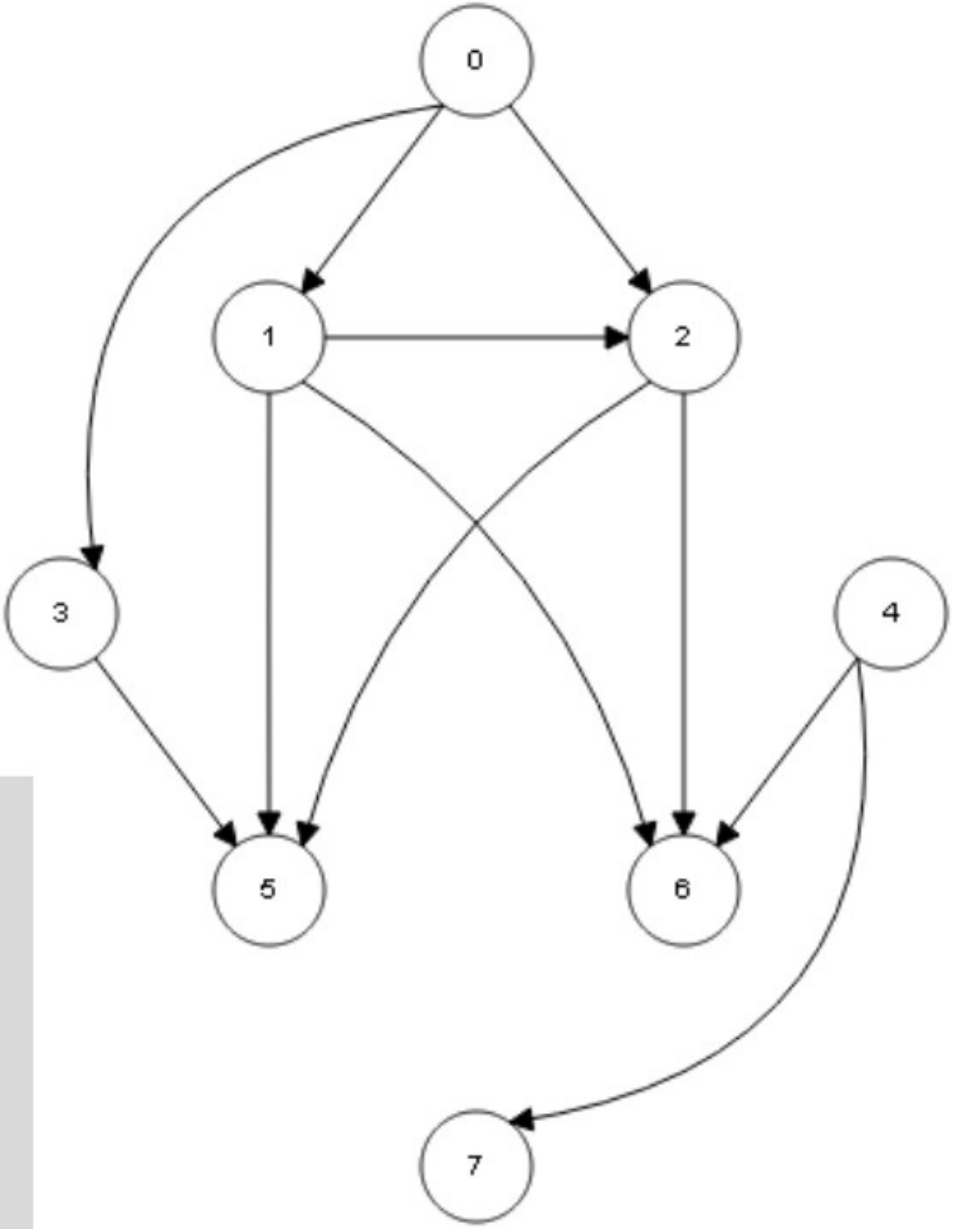

```
while S ≠ ∅
  v ← unstack(S)
  stack(v, L)
  for each u ∈ Adj[v]
    I[u] ← I[u] - 1
    if I[u] = 0
      stack(u, S)
```

Algoritmo de Kahn

	I
0	0
1	0
2	1
3	0
4	0
5	2
6	2
7	0

S
/

L
4
7
0
3
1



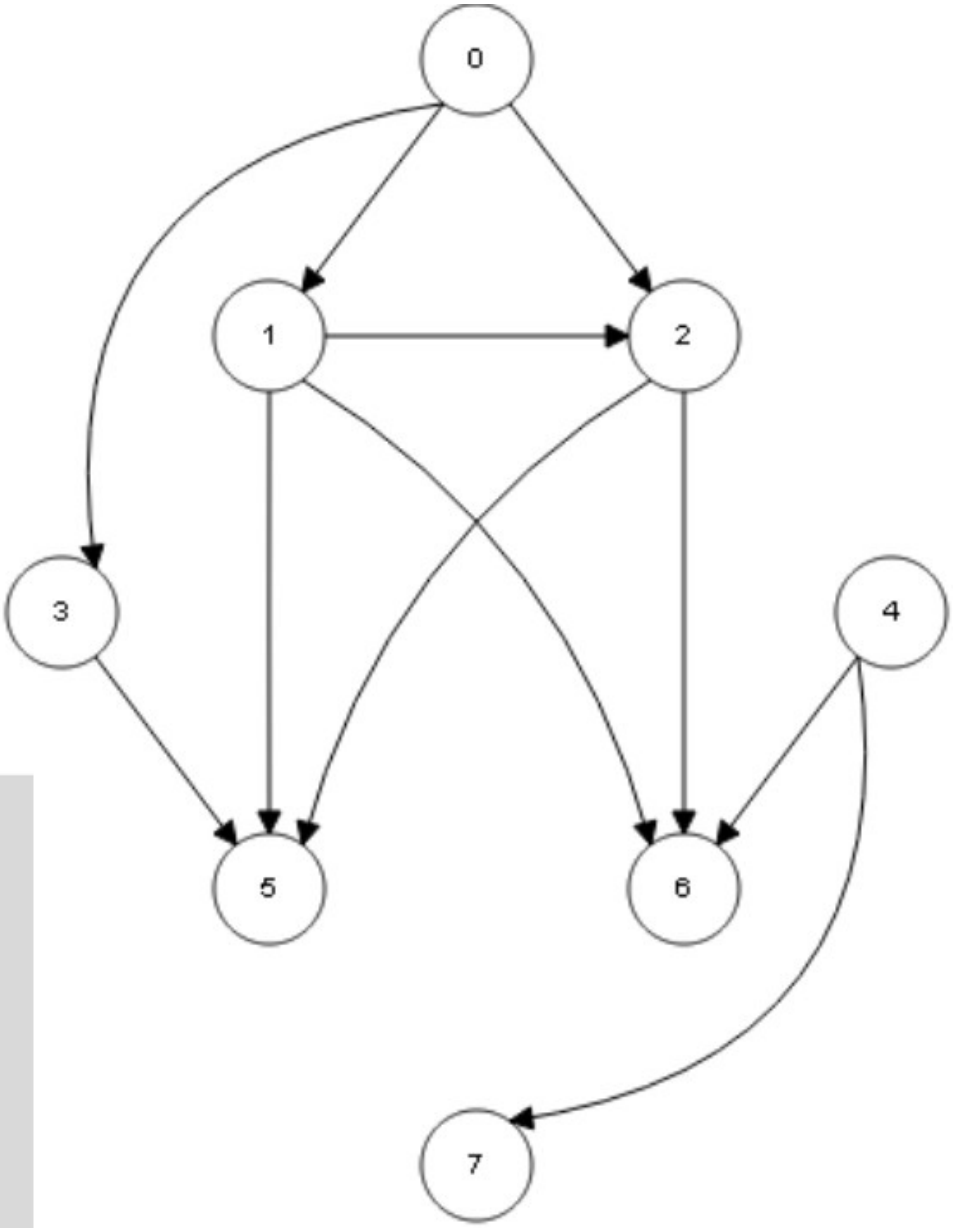

```
while S ≠ Ø
  v ← unstack(S)
  stack(v, L)
  for each u ∈ Adj[v]
    I[u] ← I[u] - 1
    if I[u] = 0
      stack(u, S)
```

Algoritmo de Kahn

	I
0	0
1	0
2	0
3	0
4	0
5	1
6	1
7	0

S
/

L
4
7
0
3
1



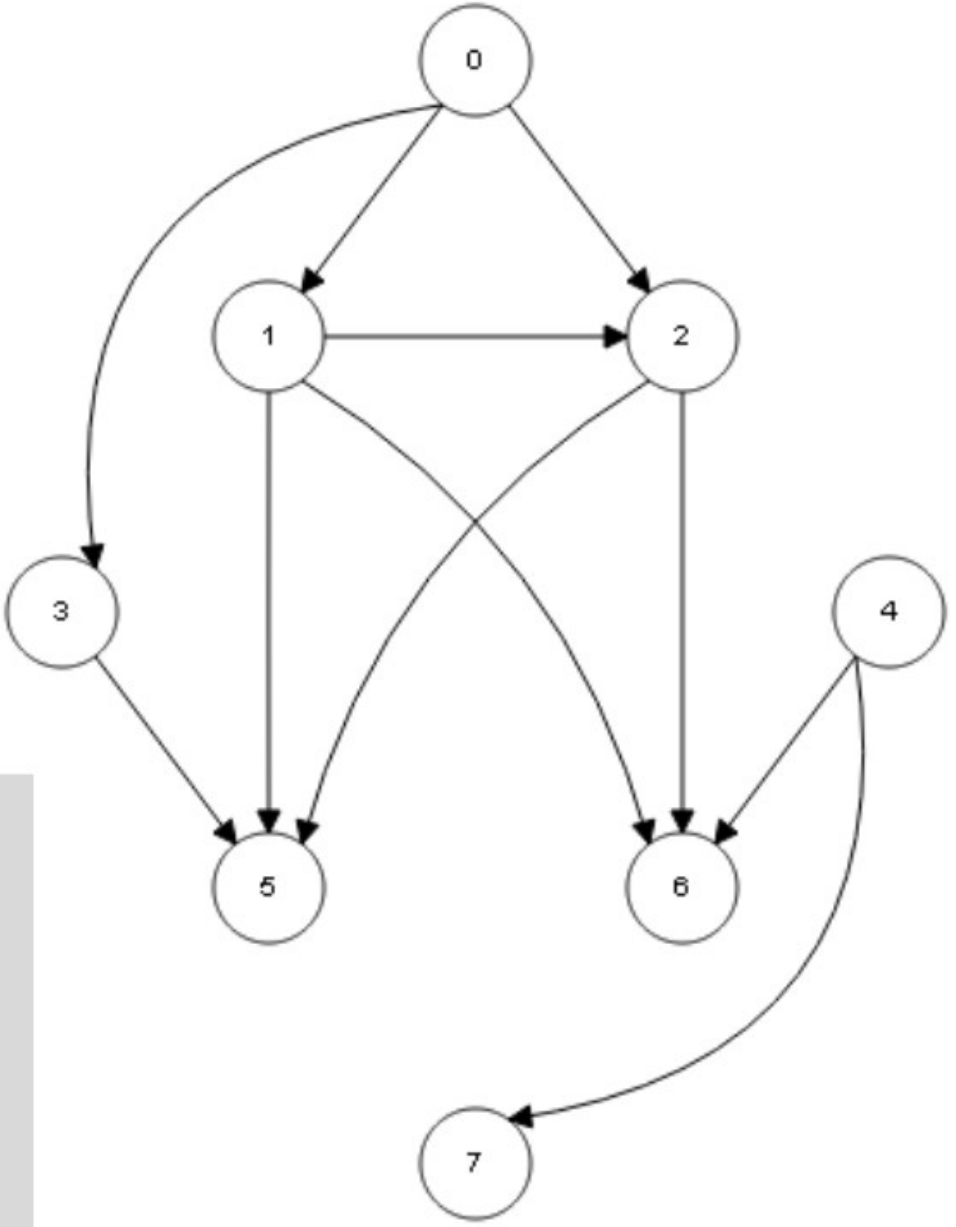

```
while S ≠ ∅
  v ← unstack(S)
  stack(v, L)
  for each u ∈ Adj[v]
    I[u] ← I[u] - 1
    if I[u] = 0
      stack(u, S)
```

Algoritmo de Kahn

	I
0	0
1	0
2	0
3	0
4	0
5	1
6	1
7	0

S
2

L
4
7
0
3
1



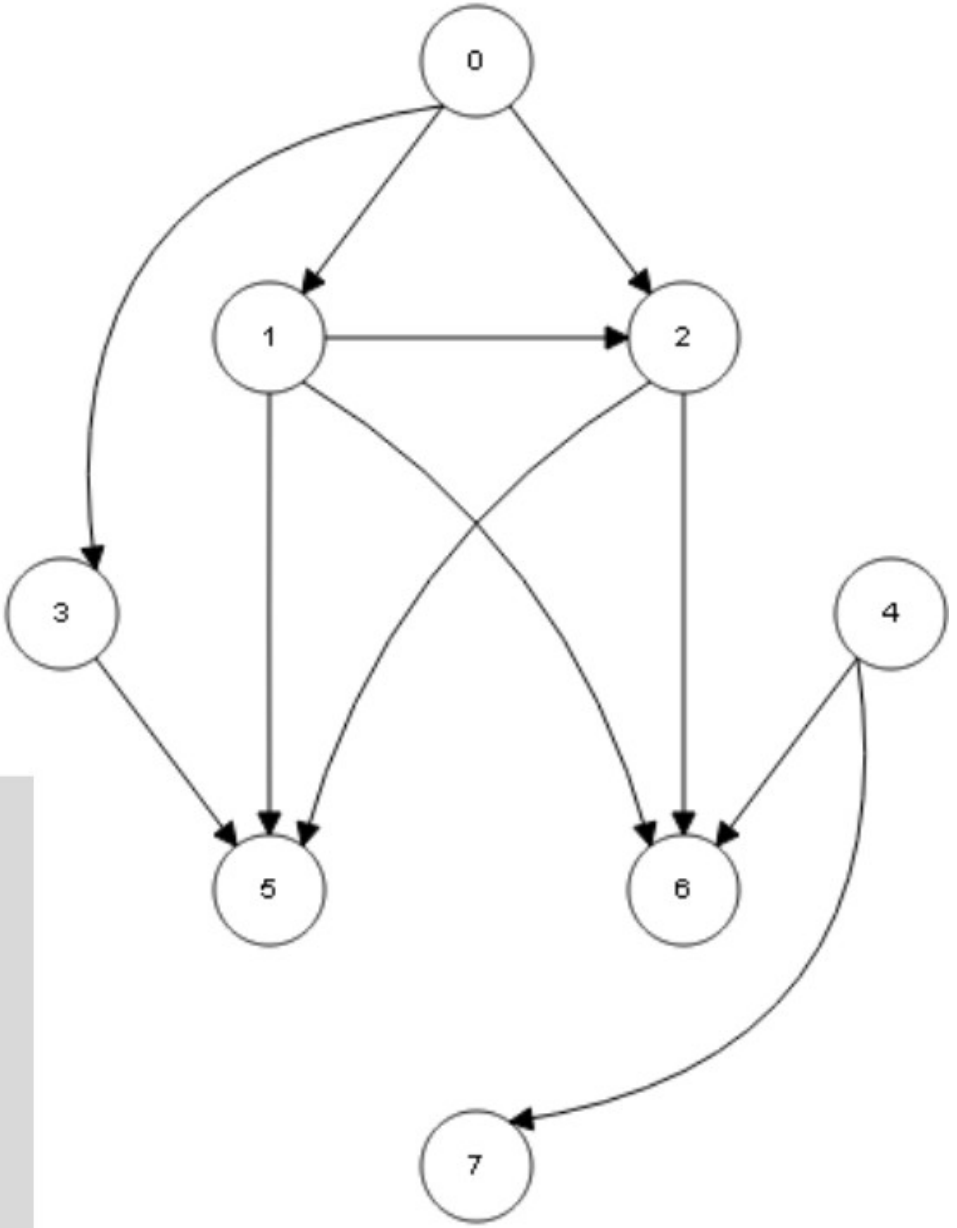

```
while S ≠ ∅
  v ← unstack(S)
  stack(v, L)
  for each u ∈ Adj[v]
    I[u] ← I[u] - 1
    if I[u] = 0
      stack(u, S)
```

Algoritmo de Kahn

	I
0	0
1	0
2	0
3	0
4	0
5	1
6	1
7	0

S
/

L
4
7
0
3
1
2

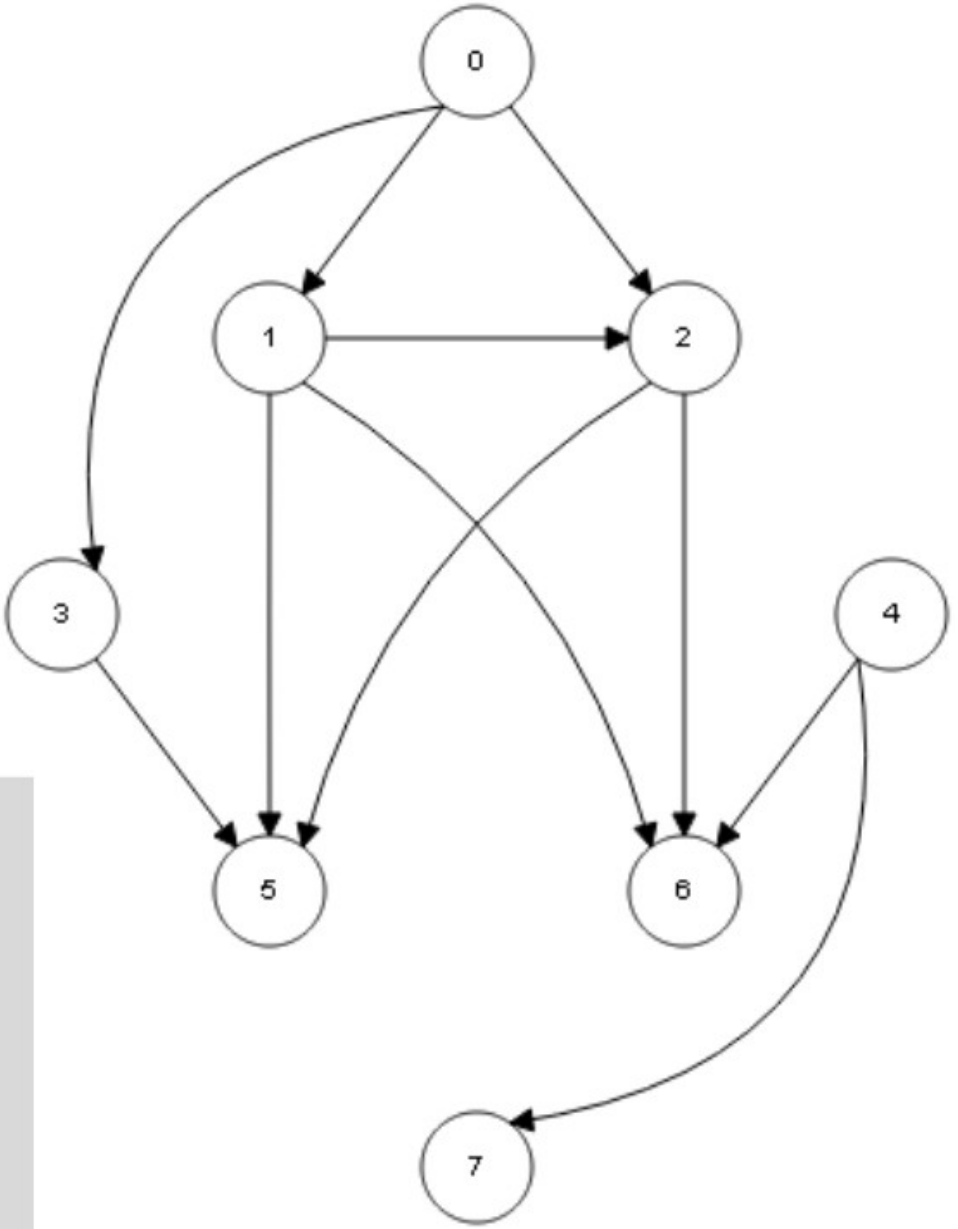



```
while S ≠ Ø
  v ← unstack(S)
  stack(v, L)
  for each u ∈ Adj[v]
    I[u] ← I[u] - 1
    if I[u] = 0
      stack(u, S)
```

Algoritmo de Kahn

	I	S	L
0	0	/	4
1	0		7
2	0		0
3	0		3
4	0		1
5	1		2
6	1		
7	0		

```
while S ≠ ∅
  v ← unstack(S)
  stack(v, L)
  for each u ∈ Adj[v]
    I[u] ← I[u] - 1
    if I[u] = 0
      stack(u, S)
```

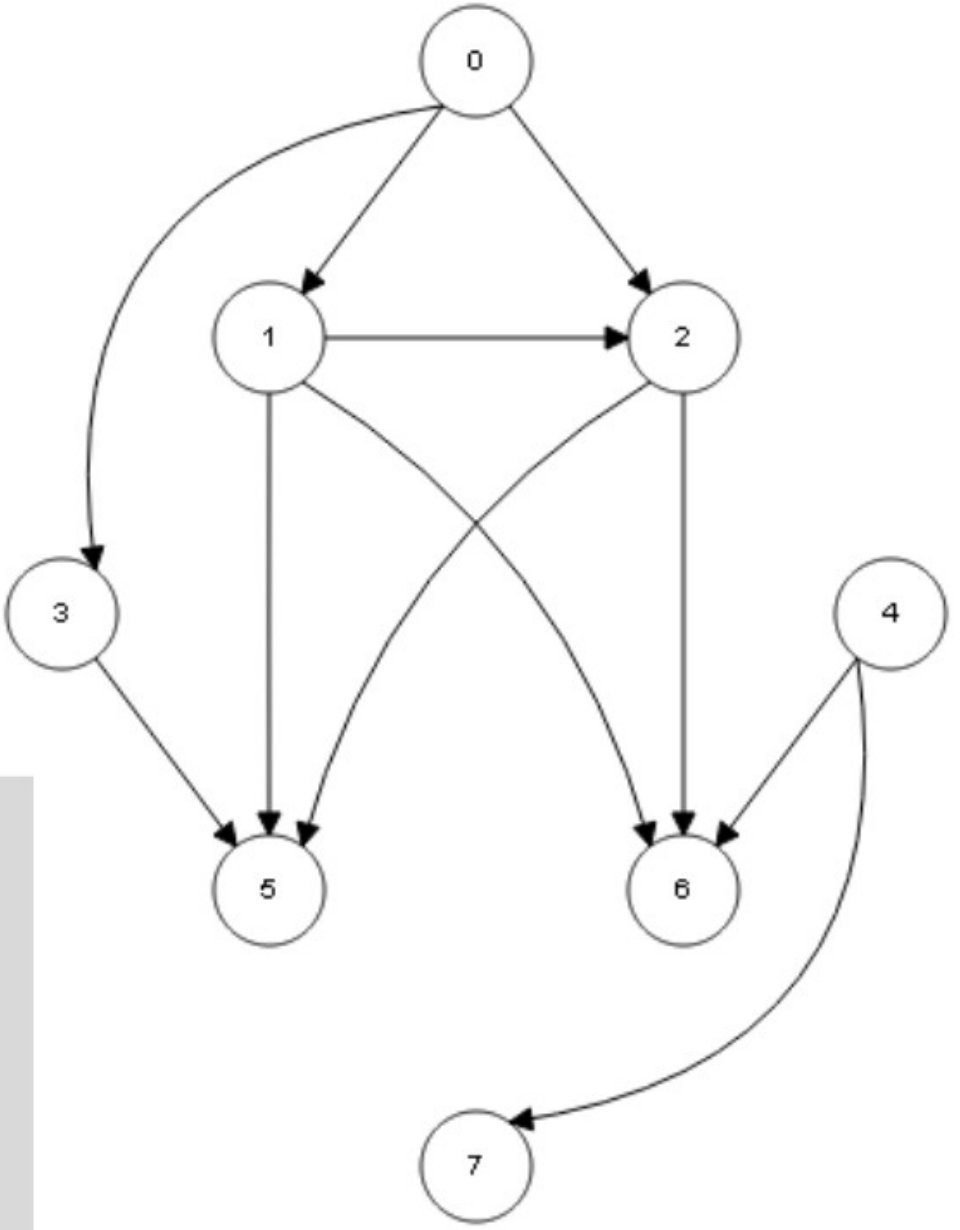


Algoritmo de Kahn

	I
0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	0

S
/

L
4
7
0
3
1
2



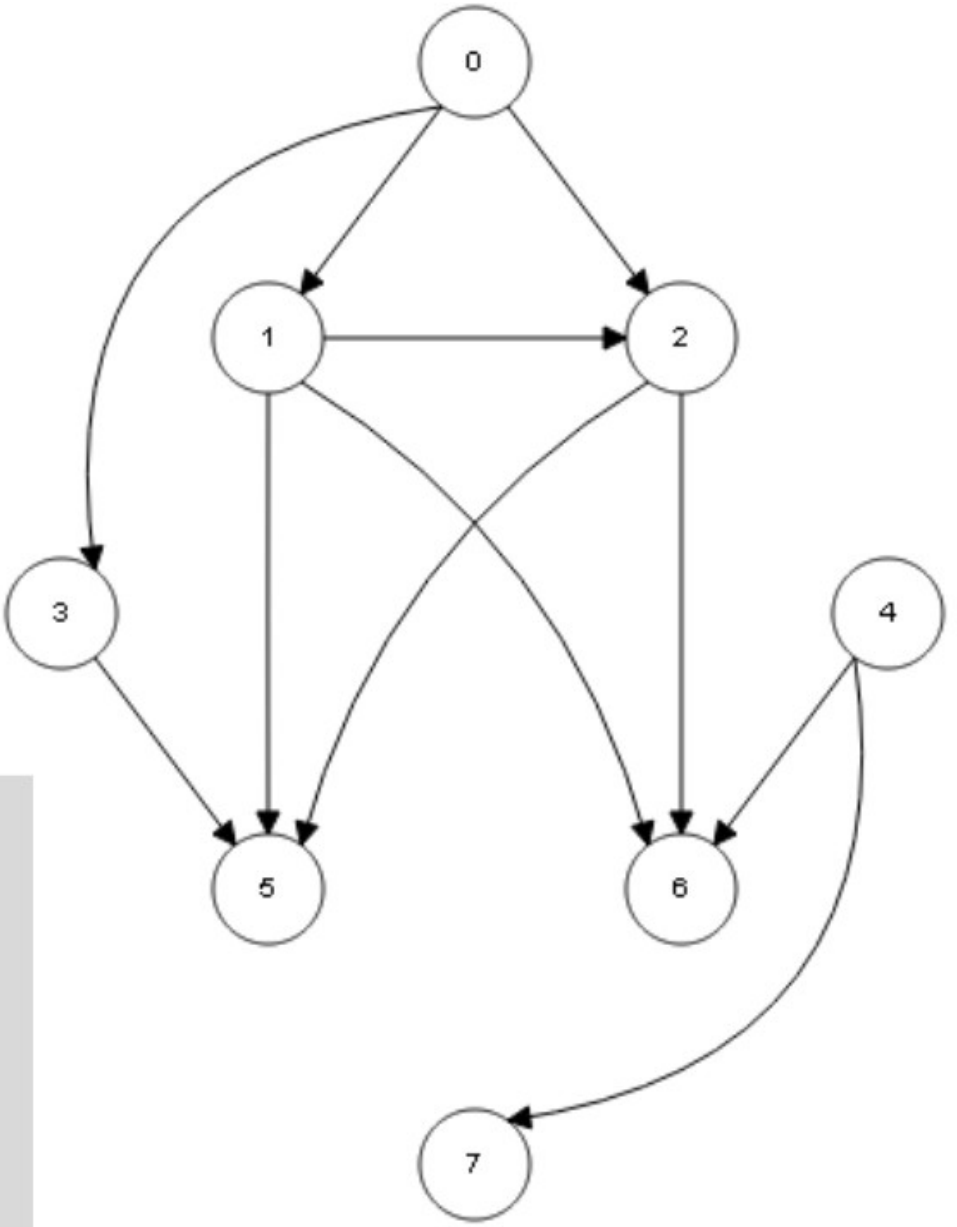

```
while S ≠ ∅
  v ← unstack(S)
  stack(v, L)
  for each u ∈ Adj[v]
    I[u] ← I[u] - 1
    if I[u] = 0
      stack(u, S)
```

Algoritmo de Kahn

	I
0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	0

S
5
6

L
4
7
0
3
1
2



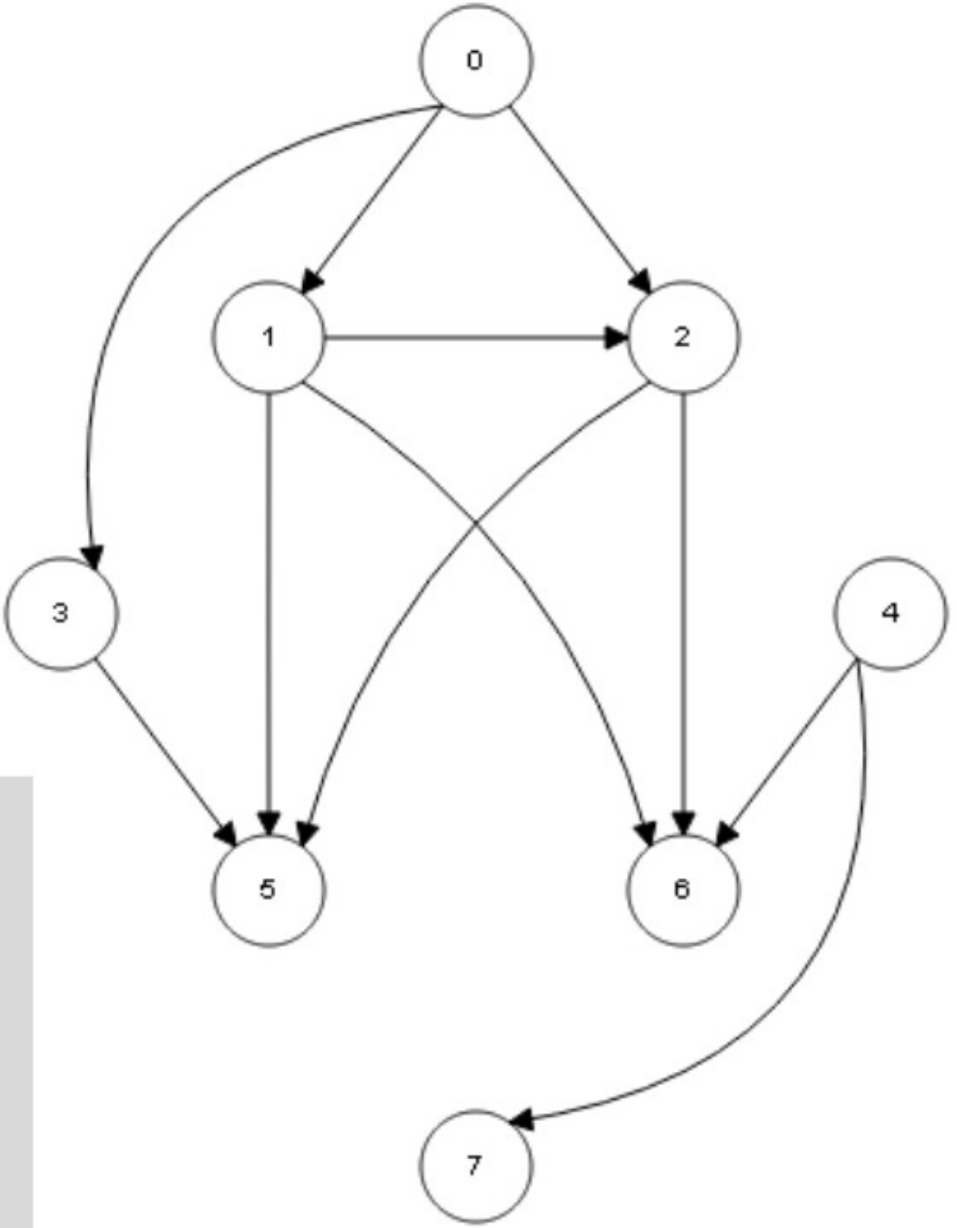

```
while S ≠ ∅
  v ← unstack(S)
  stack(v, L)
  for each u ∈ Adj[v]
    I[u] ← I[u] - 1
    if I[u] = 0
      stack(u, S)
```

Algoritmo de Kahn

	I
0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	0

S
5

L
4
7
0
3
1
2
6




```
while S ≠ ∅
  v ← unstack(S)
  stack(v, L)
  for each u ∈ Adj[v]
    I[u] ← I[u] - 1
    if I[u] = 0
      stack(u, S)
```

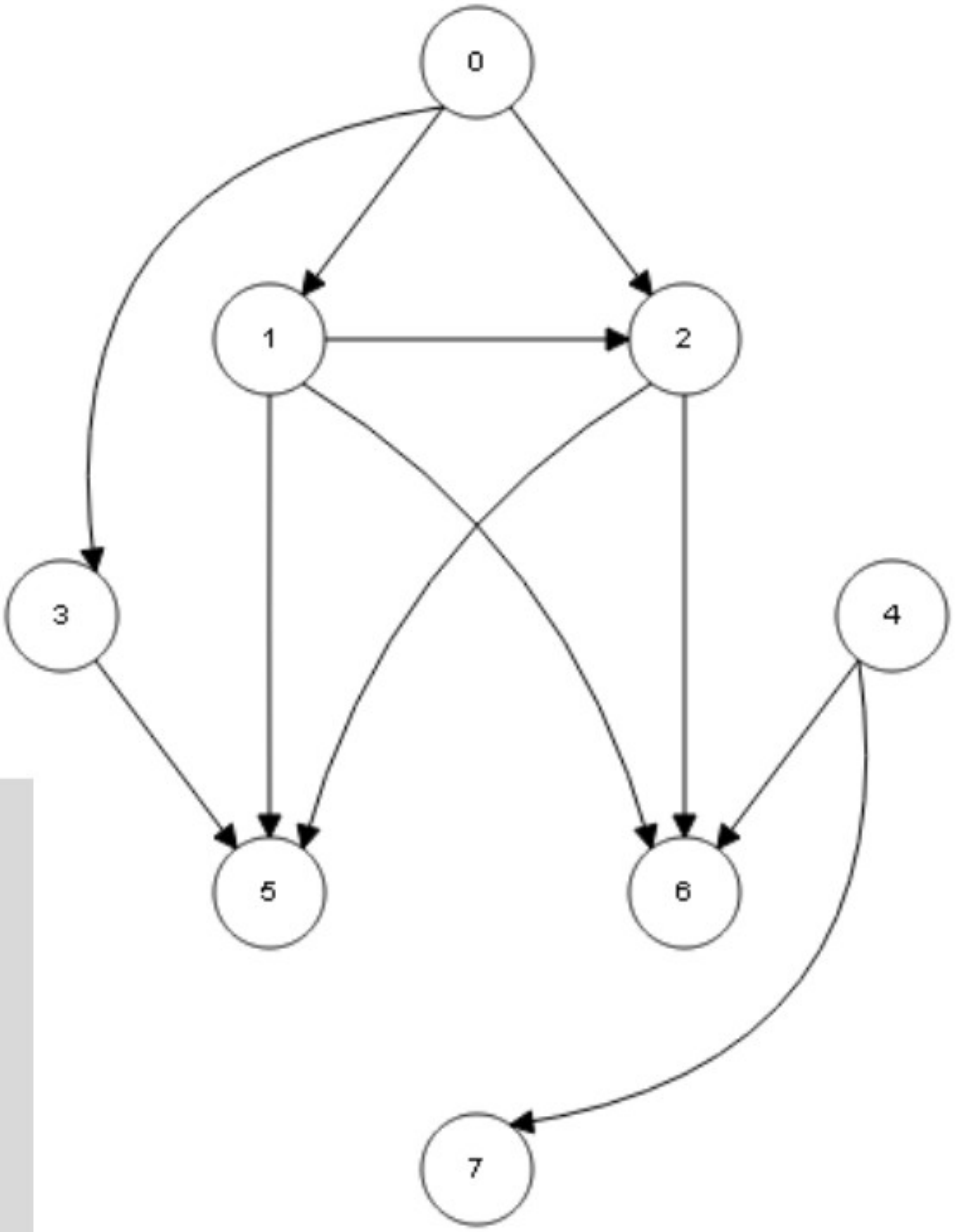
Algoritmo de Kahn

	I
0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	0

S
/

L
4
7
0
3
1
2
6
5

```
while S ≠ Ø
  v ← unstack(S)
  stack(v, L)
  for each u ∈ Adj[v]
    I[u] ← I[u] - 1
    if I[u] = 0
      stack(u, S)
```

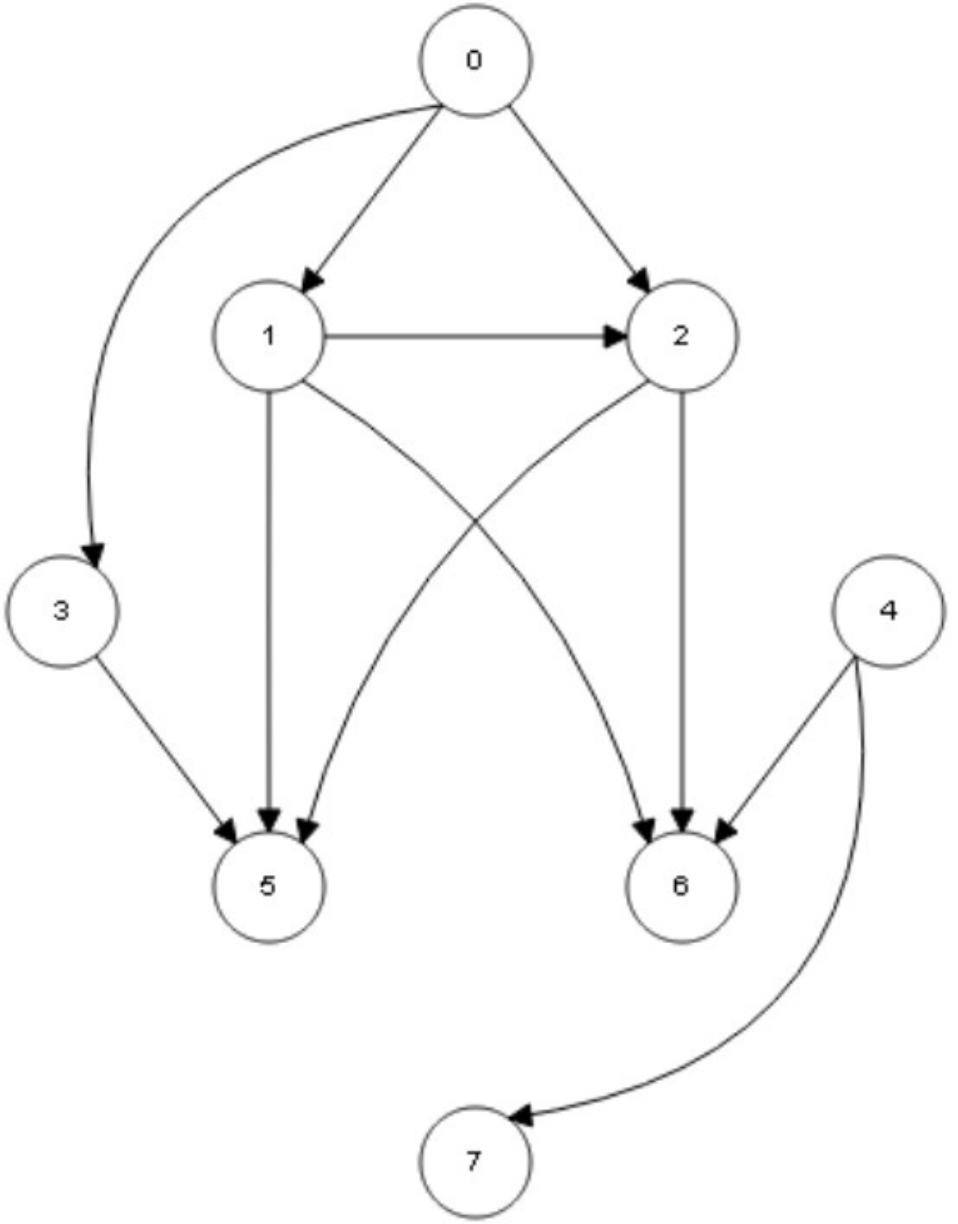


Algoritmo de Kahn

	I
0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	0

S
/

L
4
7
0
3
1
2
6
5



Busca em Profundidade (DFS)

Busca em Profundidade (DFS)

- Terminologia:

Busca em Profundidade (DFS)

- Terminologia:
  - L: lista que conterá os elementos da ordenação topológica.

Busca em Profundidade (DFS)

- Terminologia:
  - L: lista que conterá os elementos da ordenação topológica.
  - Um vértice pode ser **não marcado**, **temporariamente marcado** ou **definitivamente marcado**.

Busca em Profundidade (DFS)

- Terminologia:
  - L: lista que conterá os elementos da ordenação topológica.
  - Um vértice pode ser **não marcado**, **temporariamente marcado** ou **definitivamente marcado**.
  - Inicialmente, todos os vértices são não marcados.

Busca em Profundidade (DFS)

- Terminologia:
  - L: lista que conterá os elementos da ordenação topológica.
  - Um vértice pode ser **não marcado**, **temporariamente marcado** ou **definitivamente marcado**.
  - Inicialmente, todos os vértices são não marcados.
  - Ao serem atingidos pela primeira vez, os vértices são temporariamente marcados.

Busca em Profundidade (DFS)

- Terminologia:
  - L: lista que conterá os elementos da ordenação topológica.
  - Um vértice pode ser **não marcado**, **temporariamente marcado** ou **definitivamente marcado**.
  - Inicialmente, todos os vértices são não marcados.
  - Ao serem atingidos pela primeira vez, os vértices são temporariamente marcados.
  - Após terem suas dependências examinadas, os vértices são definitivamente marcados.

Busca em Profundidade (DFS)

- Terminologia:
  - L: lista que conterá os elementos da ordenação topológica.
  - Um vértice pode ser **não marcado**, **temporariamente marcado** ou **definitivamente marcado**.
  - Inicialmente, todos os vértices são não marcados.
  - Ao serem atingidos pela primeira vez, os vértices são temporariamente marcados.
  - Após terem suas dependências examinadas, os vértices são definitivamente marcados.
  - Caso um vértice temporariamente marcado seja examinado novamente, o grafo possui pelo menos um ciclo.

Busca em Profundidade (DFS)

```
   Entrada: Grafo G = (V, A)
1  L ← ∅;
2  enquanto existir vértice não marcado e sem arcos de entrada faça
3  |   selecione um vértice v não marcado;
4  |   visite(G, v, L);
5  fim
```

```
1  função visite(G,v,L)
2  se v é temporariamente marcado então  retorna Erro;//detecção de ciclo ;
3  se v é não marcado então
4  |   marque temporariamente v;
5  |   para cada arco (vw) faça
6  |   |   visite(G, w, L);//chamada recursiva da função
7  |   fim
8  |   marque definitivamente v;
9  |   adicione v ao final de L;
10 fim
```