Ordenação Topológica

Laboratório de Programação Competitiva - 2020

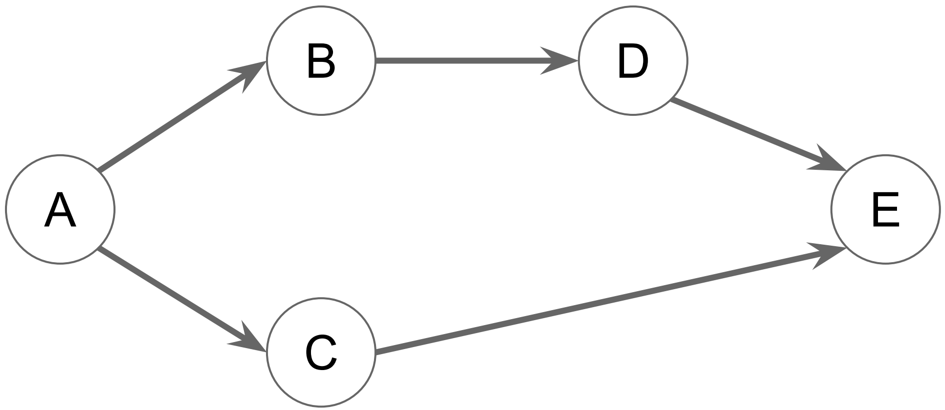
Pedro Henrique Paiola

Relações de dependência

- Motivação: dado um conjunto de N tarefas (dependentes entre si), em que ordem podemos executar estas tarefas?
- As relações de dependência podem ser modeladas através de um grafo direcionado.

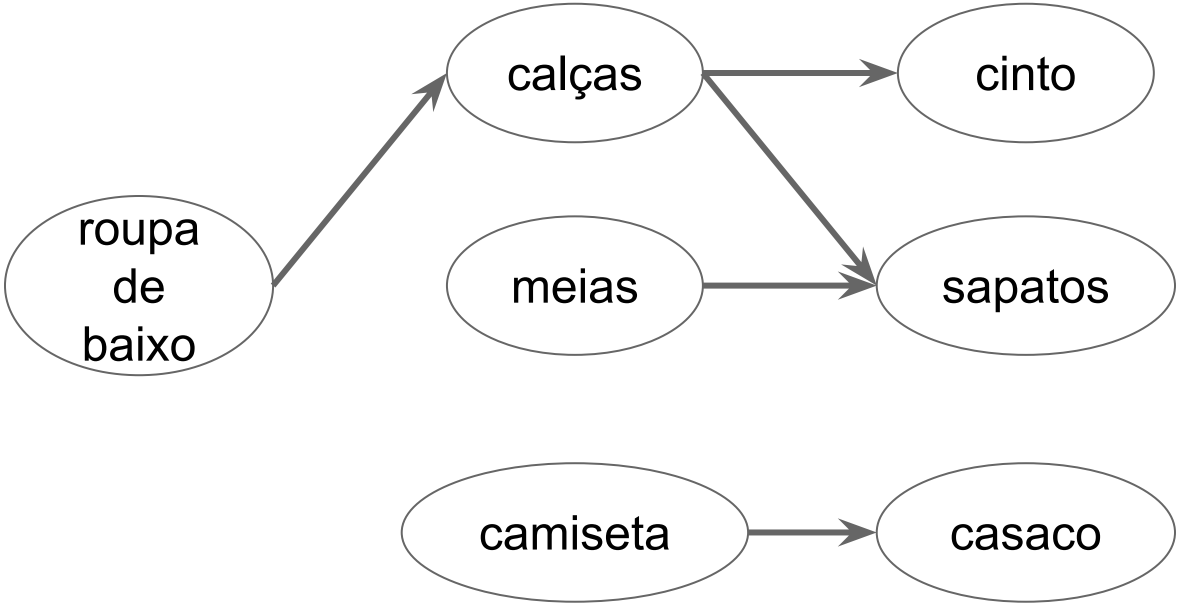
Relações de dependência

- Exemplo
  - B depende de A
  - C depende de A
  - D depende de B
  - E depende de C
  - E depende de D



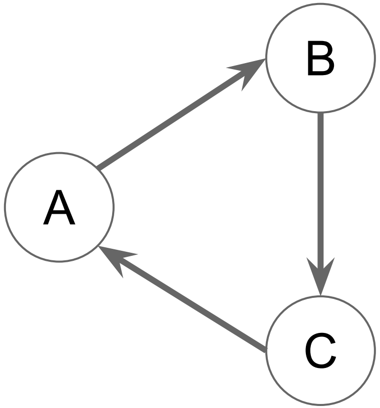
Relações de dependência

- Exemplo do processo de se vestir:



Ordenação Topológica

- Tendo isto como base, uma ordenação topológica é uma **permutação** dos vértices de um grafo direcionado que **respeita as relações de dependências** impostas.
- É fácil perceber que **grafos com ciclos** não admitem ordenação topológica.



Ordenação Topológica

- Formalizando: a ordenação topológica de um grafo direcionado acíclico (*dag*) G é uma ordem linear (sequência, lista) de vértices tal que se G contém uma aresta (u, v), então u aparece antes de v na ordem linear.

Ordenação Topológica

O grafo abaixo tem diversas ordenações topológicas possiveis:

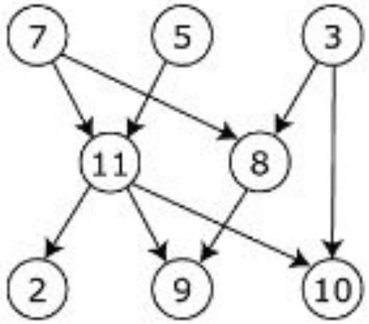


Figura por Derrick Coetzee

- 7, 5, 3, 11, 8, 2, 9, 10 (visual esquerda-para-direita, de-cima-para-baixo)
- 3, 5, 7, 8, 11, 2, 9, 10 (vértice de menor número disponível primeiro)
- 3, 7, 8, 5, 11, 10, 2, 9
- 5, 7, 3, 8, 11, 10, 9, 2 (menor número de arestas primeiro)
- 7, 5, 11, 3, 10, 8, 9, 2 (vértice de maior número disponível primeiro)
- 7, 5, 11, 2, 3, 8, 9, 10

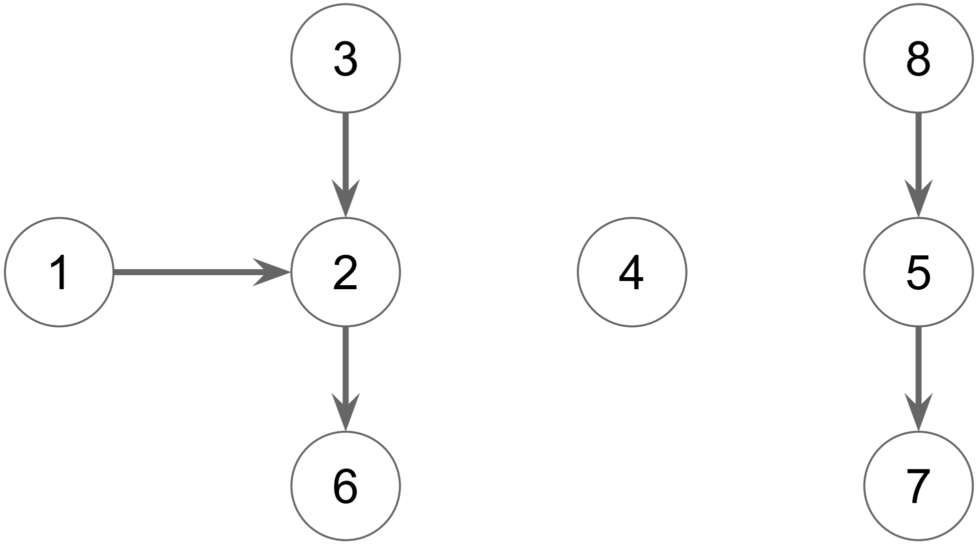
Algoritmo de Kahn

- Um possível algoritmo para encontrar uma ordenação topológica é o desenvolvido por Kahn, baseado na estratégia de eliminação de fontes. Da seguinte forma:
  - Encontra-se os vértices “fontes” (com grau de entrada zero), e os insere em um conjunto S (uma fila ou pilha)
  - Partindo do princípio que, se os vértices “fontes” e seus arcos de saída forem removidos, o grafo remanescente é também um DAG.
    - Remove da fila sucessivamente os vértices fontes.
    - Rotula-os em ordem de remoção, e remove seus arcos

Algoritmo de Kahn

- Um possível algoritmo para encontrar uma ordenação topológica é o desenvolvido por Kahn, baseado na estratégia de eliminação de fontes. Da seguinte forma:
    - Pegar um vértice de grau de entrada zero e acrescentar o vértice a ordem de execução
    - Remover todas as arestas que partem desse vértice e atualizar os graus do vértice ligados a essas arestas
    - Repetir o processo até não haver mais vértices de grau zero (ou acabarem todos os vértices)

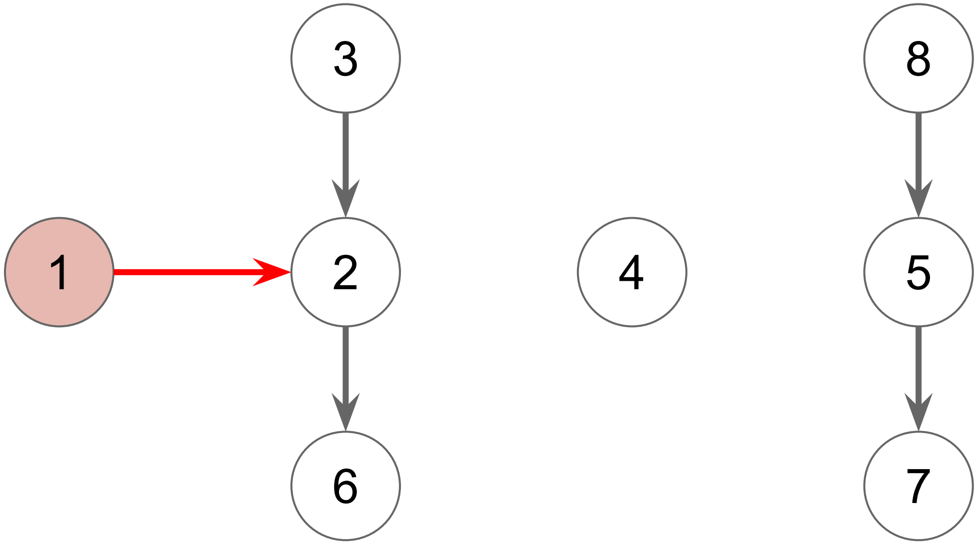
Algoritmo de Kahn



Vértice	Grau
1	0
2	2
3	0
4	0
5	1
6	1
7	1
8	0

Ordem:

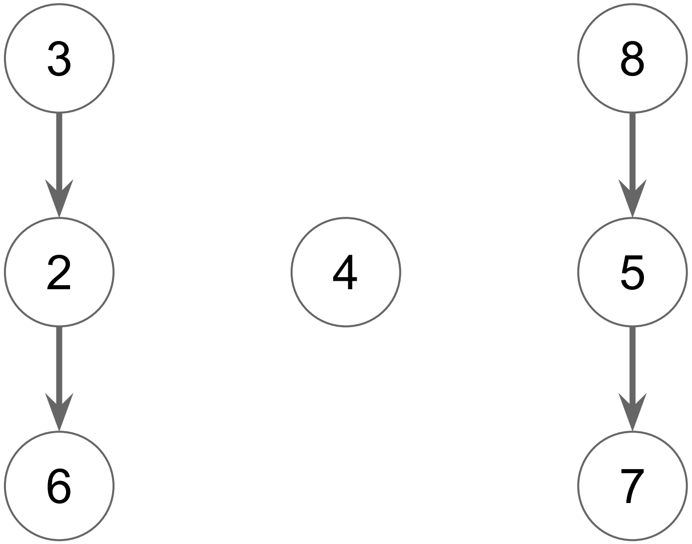
Algoritmo de Kahn



Vértice	Grau
1	0
2	1
3	0
4	0
5	1
6	1
7	1
8	0

Ordem: 1

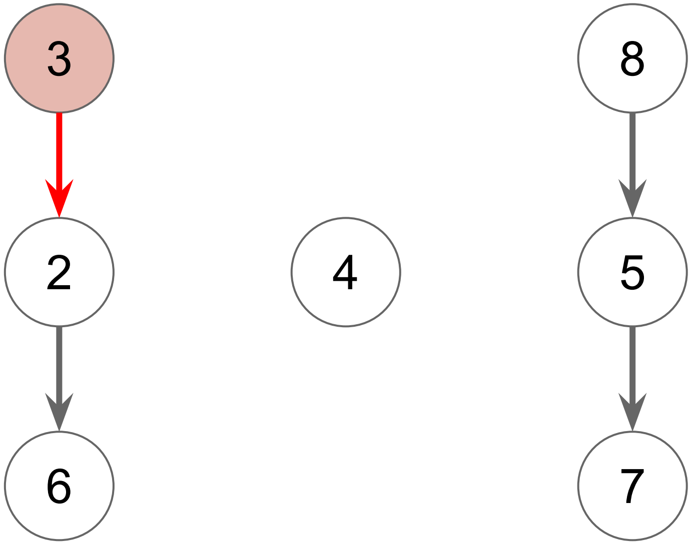
Algoritmo de Kahn



Vértice	Grau
1	-
2	1
3	0
4	0
5	1
6	1
7	1
8	0

Ordem: 1

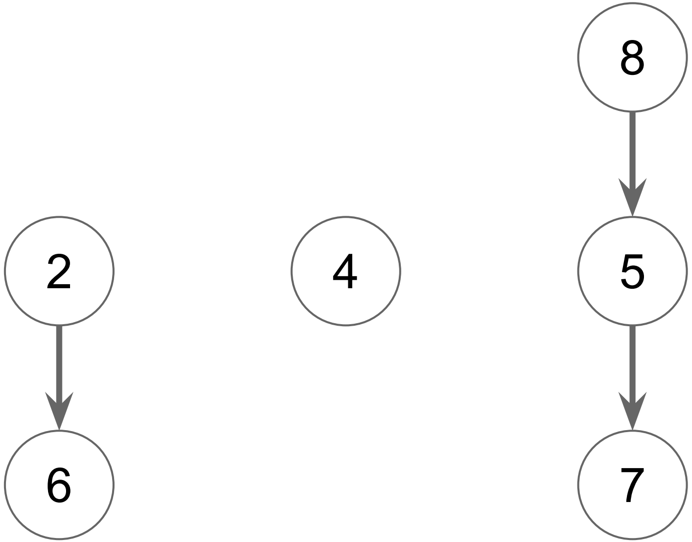
Algoritmo de Kahn



Ordem: 1 3

Vértice	Grau
1	-
2	0
3	0
4	0
5	1
6	1
7	1
8	0

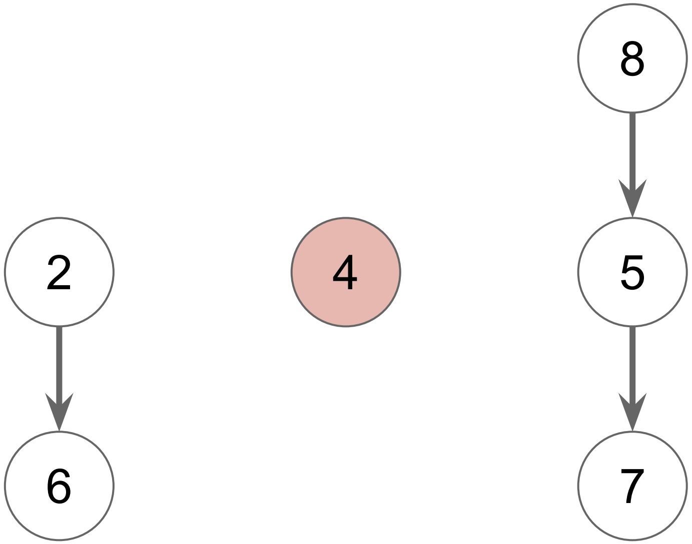
Algoritmo de Kahn



Vértice	Grau
1	-
2	0
3	-
4	0
5	1
6	1
7	1
8	0

Ordem: 1 3

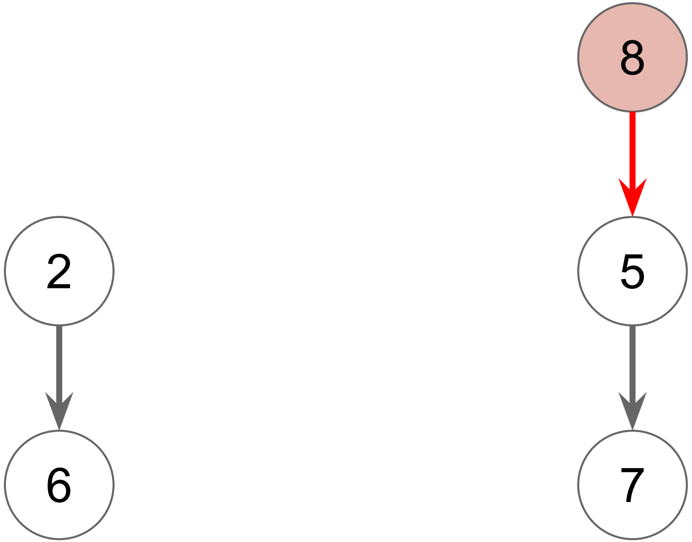
Algoritmo de Kahn



Ordem: 1 3 4

Vértice	Grau
1	-
2	0
3	-
4	0
5	1
6	1
7	1
8	0

Algoritmo de Kahn



Vértice	Grau
1	-
2	0
3	-
4	-
5	0
6	1
7	1
8	0

Ordem: 1 3 4 8

Algoritmo de Kahn

Ordem: 1 3 4 8

Vértice	Grau
1	-
2	0
3	-
4	-
5	0
6	1
7	1
8	-

Algoritmo de Kahn

Ordem: 1 3 4 8 2

Vértice	Grau
1	-
2	0
3	-
4	-
5	0
6	0
7	1
8	-

Algoritmo de Kahn

Ordem: 1 3 4 8 2

Vértice	Grau
1	-
2	-
3	-
4	-
5	0
6	0
7	1
8	-

Algoritmo de Kahn

Ordem: 1 3 4 8 2 5

Vértice	Grau
1	-
2	-
3	-
4	-
5	0
6	0
7	0
8	-

Algoritmo de Kahn

Ordem: 1 3 4 8 2 5

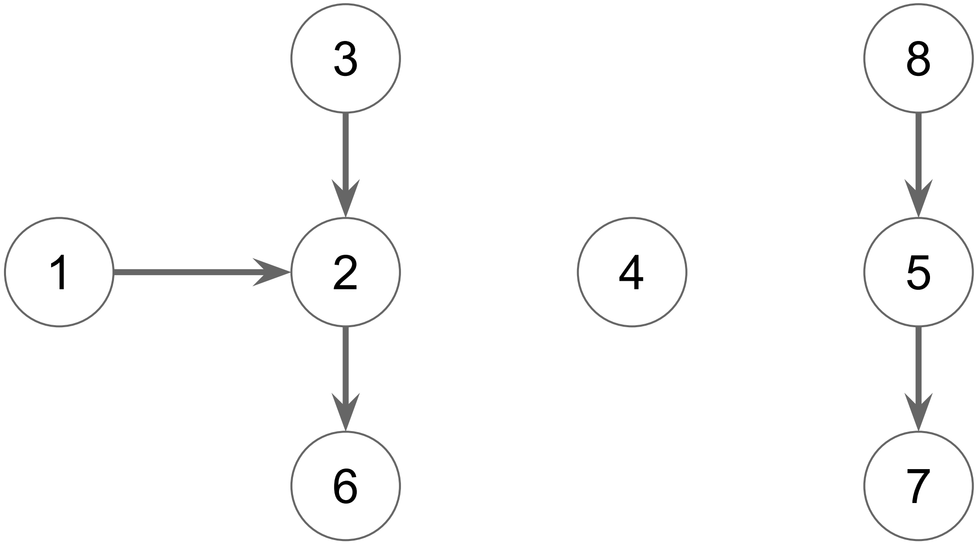
Vértice	Grau
1	-
2	-
3	-
4	-
5	-
6	0
7	0
8	-

Algoritmo de Kahn

Vértice	Grau
1	-
2	-
3	-
4	-
5	-
6	-
7	-
8	-

Ordem: 1 3 4 8 2 5 6 7

Algoritmo de Kahn



Ordem: 1 3 4 8 2 5 6 7

Vértice	Grau
1	-
2	-
3	-
4	-
5	-
6	-
7	-
8	-

```cpp
bool top_sort(){
    queue<int> q;
    for(int i = 0; i < adj_list.size(); i++)
        if (indegree[i] == 0)
            q.push(i);
    while(!q.empty()){
        int u = q.front(); q.pop();
        top_order.push_back(u);
        removed[u] = true;
        for(int k = 0; k < adj_list[u].size(); k++){
            int v = adj_list[u][k];
            if (!removed[v] && --indegree[v] == 0)
                q.push(v);
        }
    }
    return adj_list.size() == top_order.size();
}   //Complexidade O(V + A)
```

DFS para ordenação topológica

- Outro algoritmo possível para encontrar uma ordenação topológica foi proposta por Robert Tarjan, que utiliza uma busca em profundidade. O estratégia do algoritmo é a seguinte:
    - Iniciando a visita em v, visite todos os seus adjacentes (v, w) chamando a função DFS recursivamente para w.
    - Após finalizar a lista de adjacências de cada vértice v, sendo processado, adiciona-se v ao começo da lista

DFS para ordenação topológica

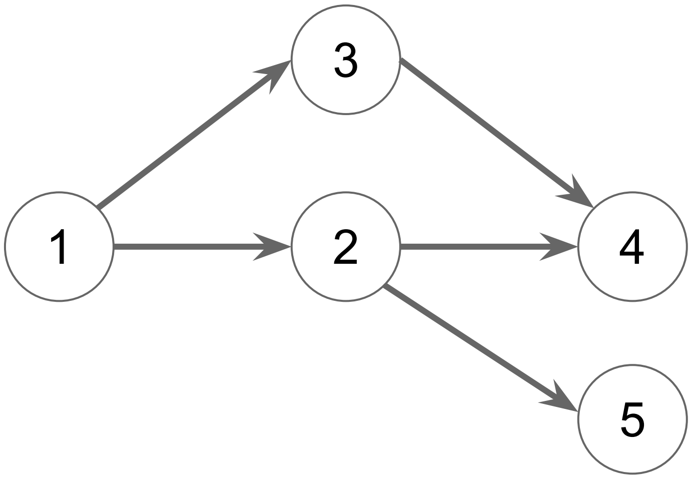
```
//Se for garantido que o grafo é acíclico
TopSort(G)
    Para cada v em V(G)
        se !visitado[v]
            dfs(v)

dfs(v)
    para cada w em adj[v]
        se !visitado[w]
            dfs(w)
    insertFront(L, v)
```

DFS para ordenação topológica

```
//Caso contrário
TopSort(G)
    para cada v em V(G)
        cor[v] = branco
        pai[v] = -1
    tempo = 0
    para cada v em V(G)
        se cor[v] == branco
            dfs(v)
```

```
dfs(v)
    cor[v] = cinza
    d[v] = ++tempo
    para cada w em adj[v]
        se cor[w] == branco
            dfs(w)
            pai[w] = u
        se cor[w] == cinza
            CICLO!
    cor[v] = preto
    f[v] = tempo++
    insertFront(L, v)
```

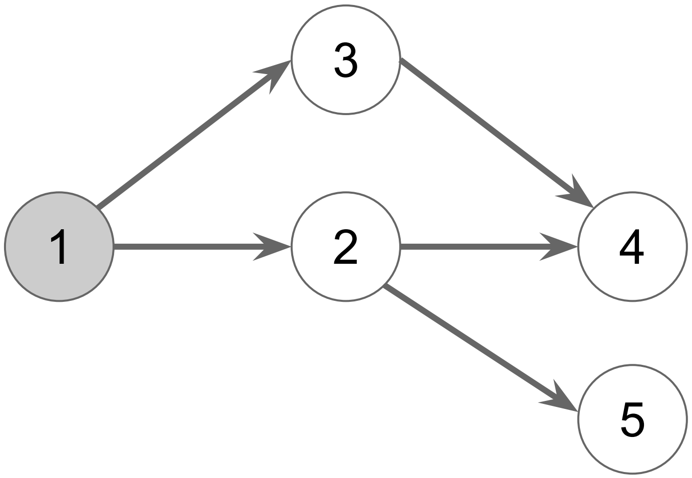
DFS para ordenação topológica



Vértice	pai	d	f
1	-1		
2	-1		
3	-1		
4	-1		
5	-1		

Ordem:

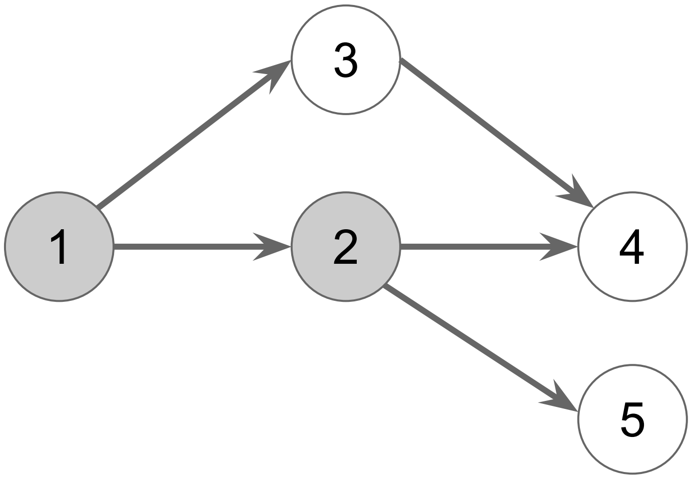
DFS para ordenação topológica



Vértice	pai	d	f
1	-1	1	
2	-1		
3	-1		
4	-1		
5	-1		

Ordem:

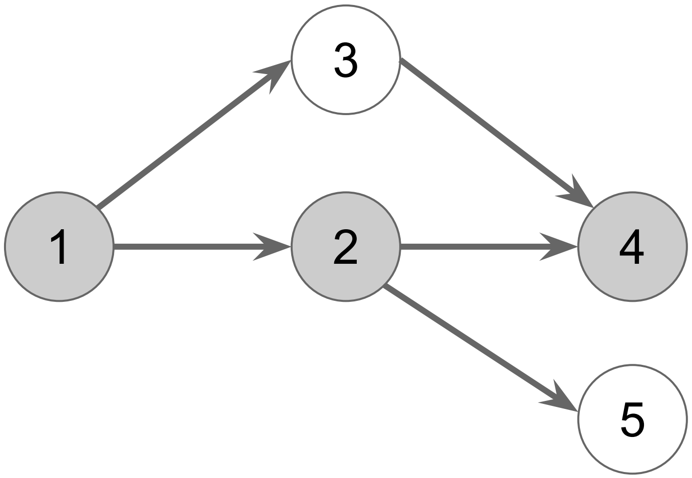
DFS para ordenação topológica



Vértice	pai	d	f
1	-1	1	
2	1	2	
3	-1		
4	-1		
5	-1		

Ordem:

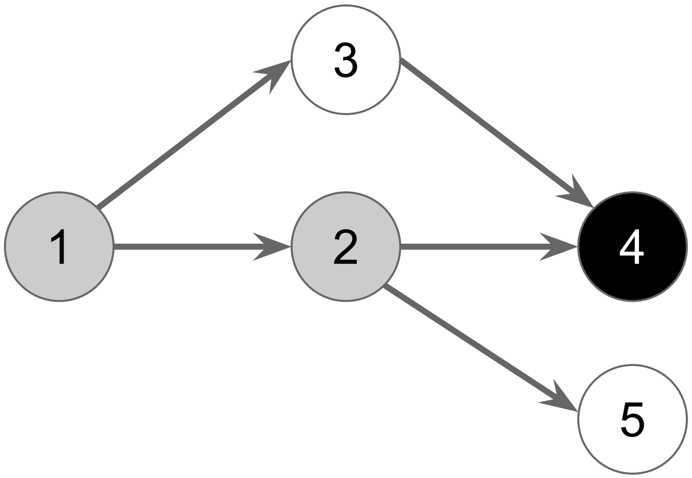
DFS para ordenação topológica



Vértice	pai	d	f
1	-1	1	
2	1	2	
3	-1		
4	2	3	
5	-1		

Ordem:

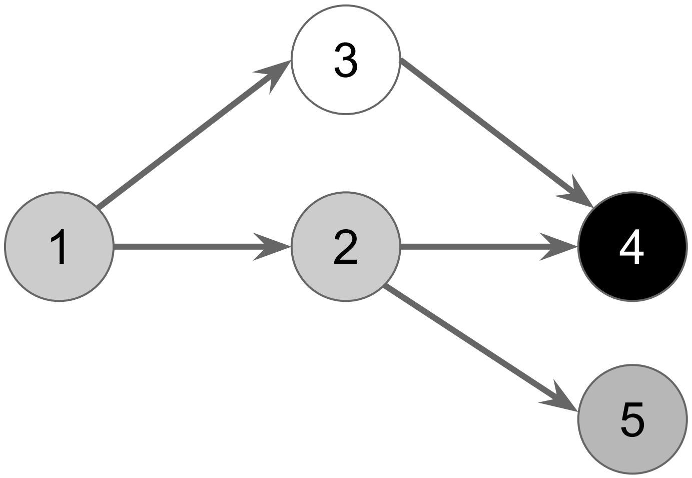
DFS para ordenação topológica



Vértice	pai	d	f
1	-1	1	
2	1	2	
3	-1		
4	2	3	4
5	-1		

Ordem: 4

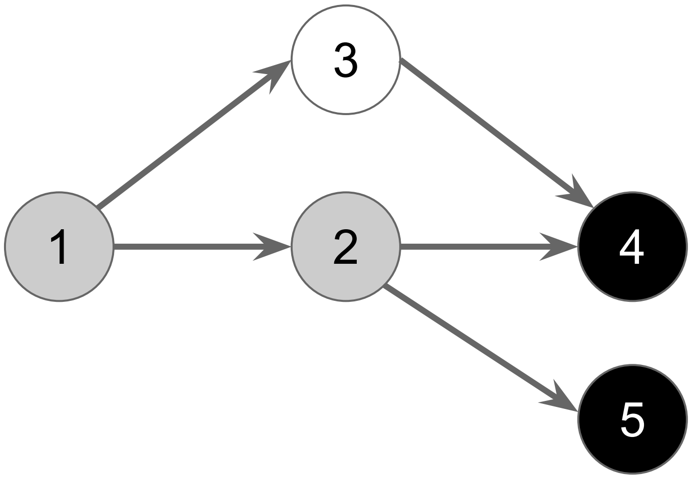
DFS para ordenação topológica



Vértice	pai	d	f
1	-1	1	
2	1	2	
3	-1		
4	2	3	4
5	2	5	

Ordem: 4

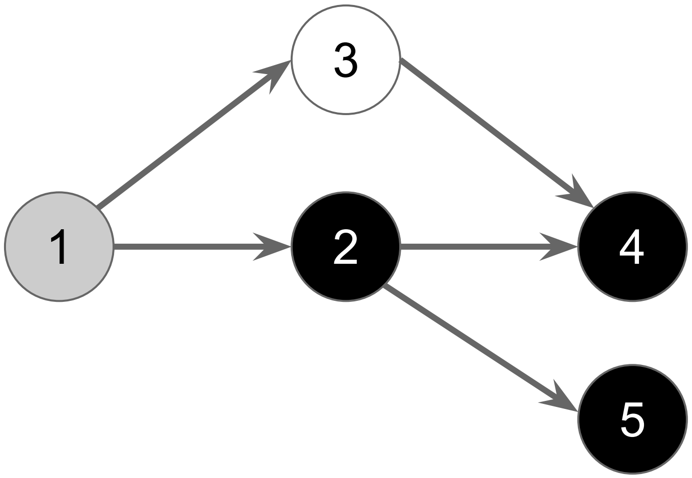
DFS para ordenação topológica



Vértice	pai	d	f
1	-1	1	
2	1	2	
3	-1		
4	2	3	4
5	2	5	6

Ordem: 5 4

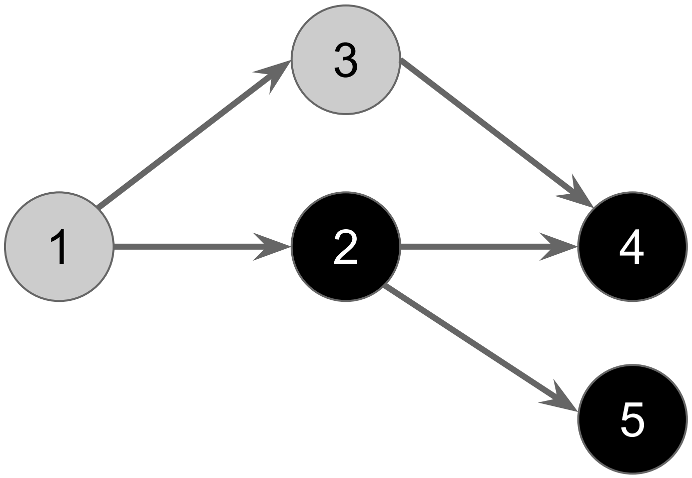
DFS para ordenação topológica



Vértice	pai	d	f
1	-1	1	
2	1	2	7
3	-1		
4	2	3	4
5	2	5	6

Ordem: 2 5 4

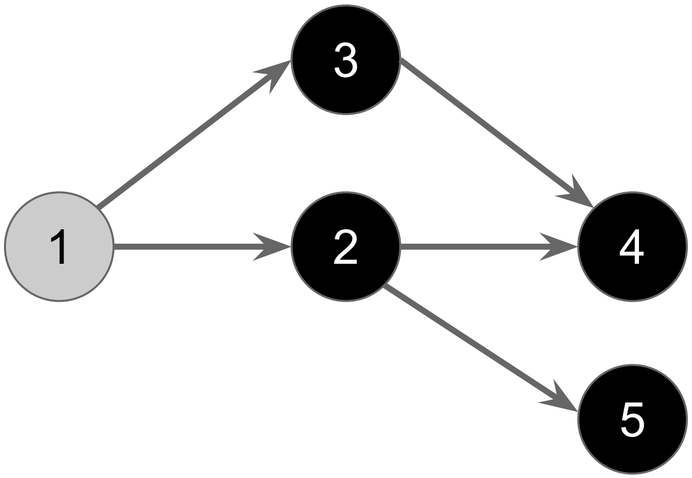
DFS para ordenação topológica



Vértice	pai	d	f
1	-1	1	
2	1	2	7
3	1	8	
4	2	3	4
5	2	5	6

Ordem: 2 5 4

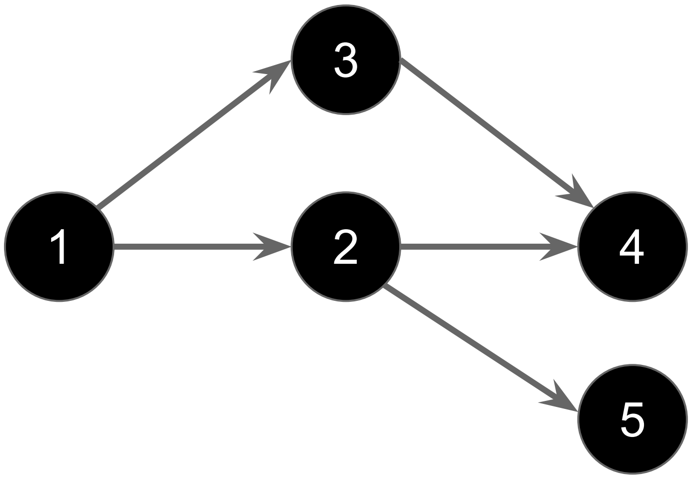
DFS para ordenação topológica



Vértice	pai	d	f
1	-1	1	
2	1	2	7
3	1	8	9
4	2	3	4
5	2	5	6

Ordem: 3 2 5 4

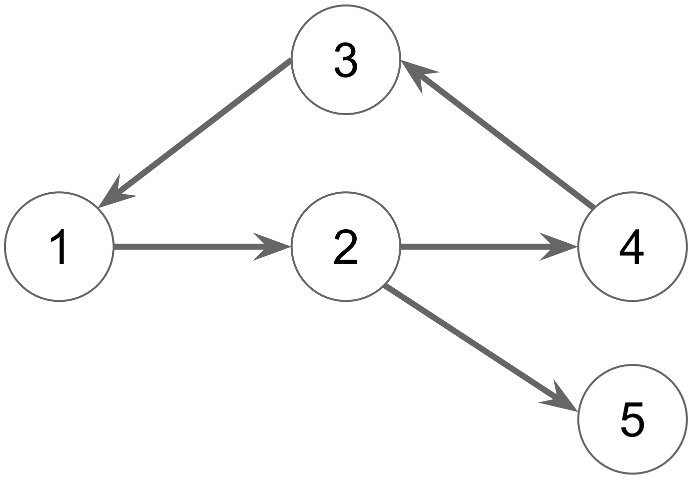
DFS para ordenação topológica



Vértice	pai	d	f
1	-1	1	10
2	1	2	7
3	1	8	9
4	2	3	4
5	2	5	6

Ordem: 1 3 2 5 4

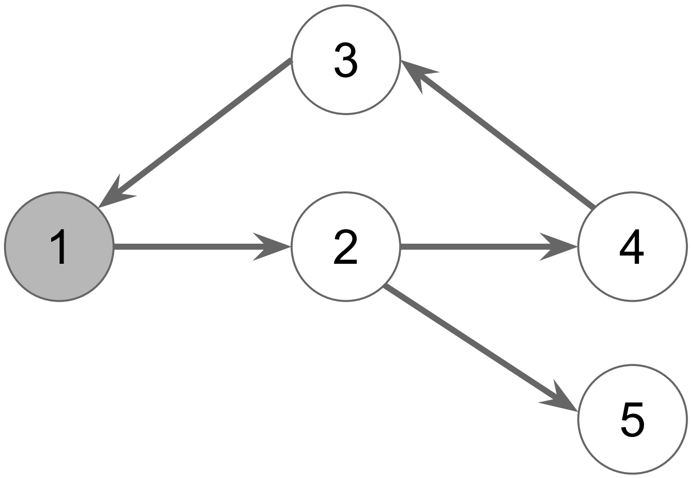
DFS para ordenação topológica



Vértice	pai	d	f
1	-1		
2	-1		
3	-1		
4	-1		
5	-1		

Ordem:

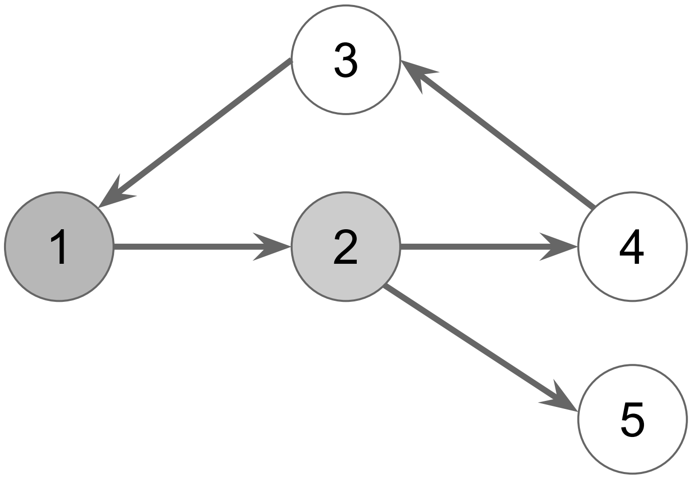
DFS para ordenação topológica



Vértice	pai	d	f
1	-1	1	
2	-1		
3	-1		
4	-1		
5	-1		

Ordem:

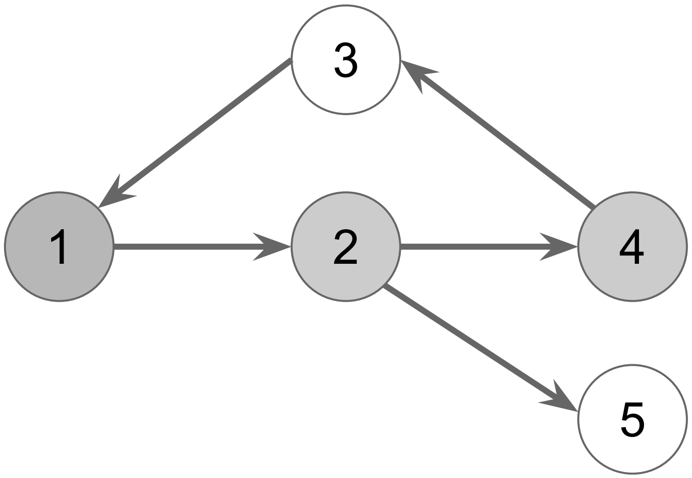
DFS para ordenação topológica



Vértice	pai	d	f
1	-1	1	
2	1	2	
3	-1		
4	-1		
5	-1		

Ordem:

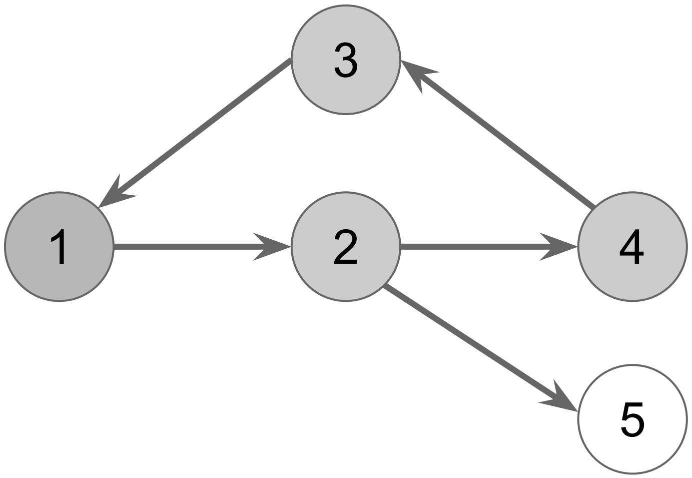
DFS para ordenação topológica



Vértice	pai	d	f
1	-1	1	
2	1	2	
3	-1		
4	2	3	
5	-1		

Ordem:

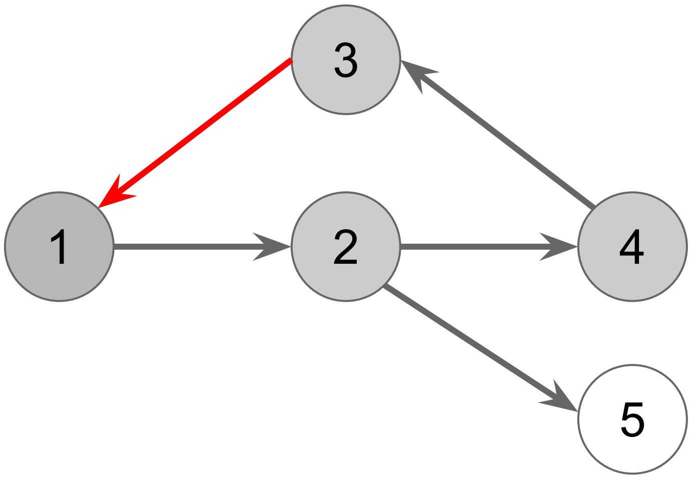
DFS para ordenação topológica



Vértice	pai	d	f
1	-1	1	
2	1	2	
3	4	4	
4	2	3	
5	-1		

Ordem:

DFS para ordenação topológica



Vértice	pai	d	f
1	-1	1	
2	1	2	
3	4	4	
4	2	3	
5	-1		

CICLO ENCONTRADO!

Fox and Names (CodeForces - 510C)

- Neste problema recebemos uma lista com N nomes em ordem lexicográfica.
- Porém, não na ordem alfabética tradicional! O objetivo é determinar qual a ordem de letras utilizadas para montar esta lista (se assim for possível)
- Exemplo:
  3
  rivest
  shamir
  adleman

  bcdefghijklmnopq**rsa**tuvwxyz

Fox and Names (CodeForces - 510C)

- Solução: a partir da análise da lista de nomes, vamos determinar a precedência das letras.
- Para um par de nomes consecutivos, vamos determinar qual a primeira letra em que eles diferem, e então determinar a precedência das letras que eles determinam.
- Ex:

  mar**y**on

  mar**i**ana

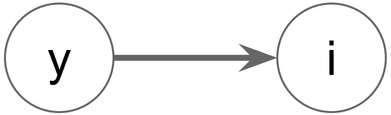
  Então y < i

Fox and Names (CodeForces - 510C)

- Vamos representar estas relações em um grafo, onde cada letra está associada a um vértice.
- Por fim, buscamos uma ordenação topológica, que é uma possível resposta para nosso problema.
- Exemplo:

  mar**y**on

  mar**i**ana



Relação com a Matemática Discreta

- **Ordem Parcial**
  - Uma relação R em um conjunto S (S, $\preccurlyeq$) com as seguintes propriedades:
    - Reflexiva ($a \preccurlyeq a,\ \forall a \in S$)
    - Anti-simétrica (Se $a \preccurlyeq b$ e $b \preccurlyeq a$, então $a = b$)
    - Transitiva (Se $a \preccurlyeq b$ e $b \preccurlyeq c$, então $a \preccurlyeq c$)
- **Conjunto Parcialmente Ordenado (poset)**
  - Um conjunto S juntamente com uma ordem parcial R: (S,R)

Relação com a Matemática Discreta

- Mostre que a relação de divisibilidade no conjunto dos inteiros positivos é uma ordem parcial. Ou seja, $(Z^+, |)$ é um poset.
  - Reflexiva
    $a|a$ para todo inteiro a
  - Anti-simétrica
    Se $a|b$ e $b|a$, então $a = b$
  - Transitiva
    Se $a|b$ e $b|c$ então $a|c$

Relação com a Matemática Discreta

- **Ordem total**
  - Se $(S, \preccurlyeq)$ é um poset e todos os pares de elementos de S são comparáveis, diz-se que S é um conjunto **totalmente ordenado** ou linearmente ordenado, e $\preccurlyeq$ é chamada de **ordem total** ou linear.
- O poset $(Z, \leq)$ é um conjunto totalmente ordenado
  - Para qualquer (a, b), ou $a \leq b$ ou $b \leq a$
- O poset $(Z^+, |)$ não é totalmente ordenado
  - Para os números 2 e 3, por exemplo, não vale nem $2|3$ e nem $3|2$

Relação com a Matemática Discreta

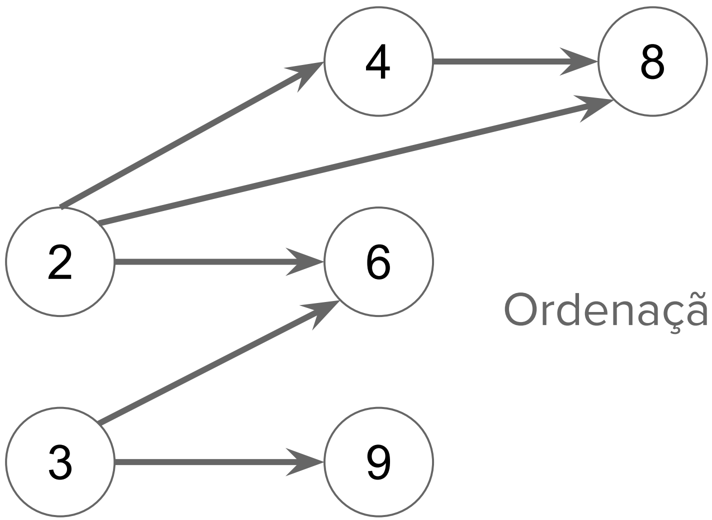
- **Ordenação topológica e Teoria da Ordem**
  - Dada uma ordem parcial $\leqslant$, a ordenação topológica consiste em obter uma ordem total $\leq$ que respeita (ou extende) $\leqslant$.
  - Uma DAG pode ser considerada uma visão “estrutural” de uma ordem parcial. E a ordenação topológica é uma visão estrutural da ordem total.

Relação com a Matemática Discreta

- Pode-se estabelecer relações entre diversas propriedades de posets com os dags que os representam. Por exemplo:
  - Um elemento $a \in S$ é chamado de um **elemento minimal** de S se não existe $b \in S$ tal que $b \prec a$ ($b \preccurlyeq a$, $b \neq a$).
  - No grafo subjacente à $(S, \preccurlyeq)$, um vértice com **grau de entrada zero** representa um elemento minimal.

Relação com a Matemática Discreta

- Exemplo: S = {2, 3, 4, 6, 8, 9}, poset = (S, |)



Ordenação topológica: 2 4 8 3 6 9

Referências

http://wiki.icmc.usp.br/images/9/93/Alg2_05.Grafos_ordenacaotopologica.pdf
https://algoritmosempython.com.br/cursos/algoritmos-python/algoritmos-grafos/ordenacao-topologica
http://edirlei.3dgb.com.br/aulas/paa/PAA_Aula_07_Ordenacao_Topologica.pdf
http://www.dimap.ufrn.br/~prolo/Disciplinas/131/DIM0111.0-AEDII/materiais/grafos/07%20Grafos%20Dirigidos%20--%20Ordenacao%20Topologica.pdf
https://www.cin.ufpe.br/~gdcc/matdis/aulas/ordensParciais.pdf
https://www2.dc.ufscar.br/~mario/ensino/2019s1/aed2/aula24.pdf
https://neps.academy/lesson/198
https://sites.google.com/site/ldsicufal/disciplinas/programacao-avancada/ordenacao-topologica---kahn