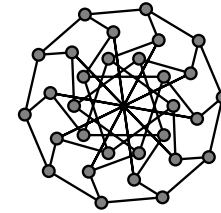


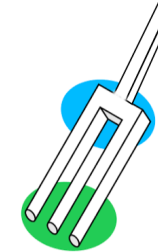
DFS e ordenação topológica

CLRS 22 Elementary Graph Algorithms
CLRS 22.3 e 22.4

Grafo conexo



Garfo sem nexos



Busca em profundidade

DFS

Outra **moldeira** de algoritmos que vão fazendo seu serviço enquanto percorrem o grafo.

Produz uma estrutura bem mais sofisticada que a BFS.

Busca em profundidade - elementos

Vértice branco: ainda não descoberto

Vértice cinzento: descoberto, em processamento

Vértice preto: processado

DFS termina descrevendo via π 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

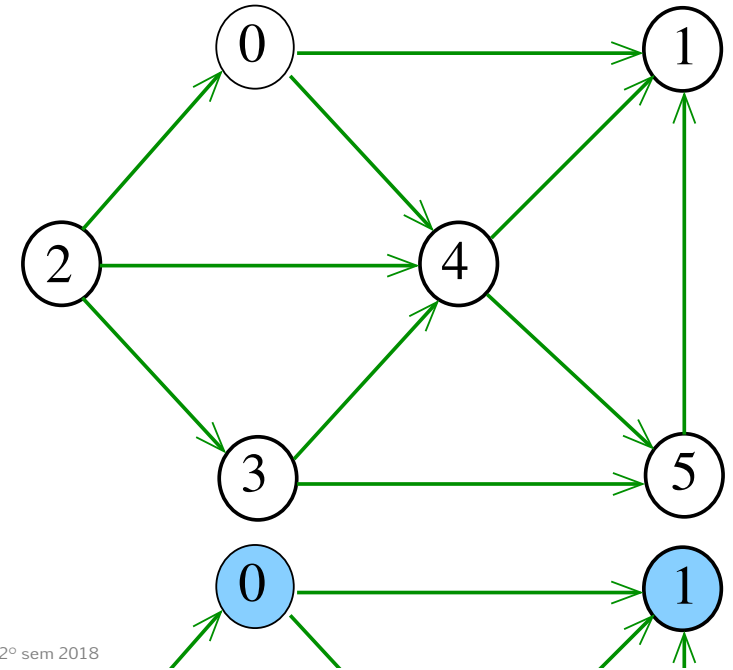
DFS(G)

- 1 para cada $u \in V(G)$ faça
- 2 $u.cor \leftarrow$ branco $u.\pi \leftarrow$ nil
- 3 tempo \leftarrow 0
- 4 para cada $u \in V(G)$ faça
- 5 se $u.cor =$ branco
- 6 então DFS-Visit(u)

DFS-Visit(G, u)

- 1 $u.cor \leftarrow$ cinzento $u.d \leftarrow$ tempo tempo \leftarrow tempo + 1
- 3 para cada $v \in u.Estrela$ faça
- 4 se $v.cor =$ branco
- 5 então $v.\pi \leftarrow u$
- 6 DFS-Visit(G, v)
- 7 $u.cor \leftarrow$ preto
- 8 $u.f \leftarrow$ tempo tempo \leftarrow tempo + 1

DFS(G)

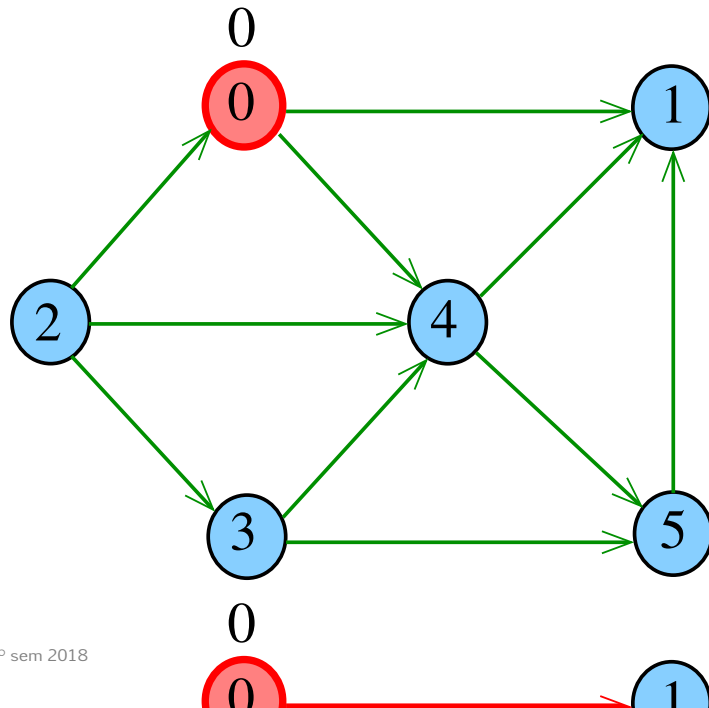


5 / 9

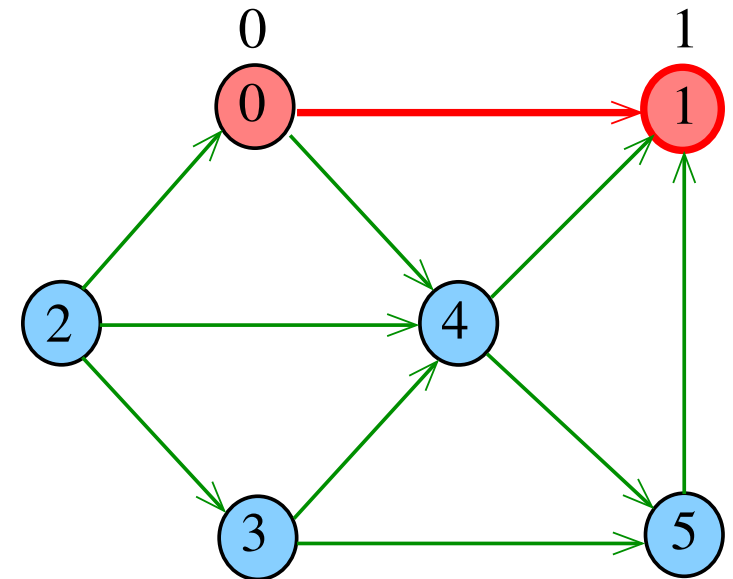
Análise de Algoritmos — 2º sem 2018

6 / 9

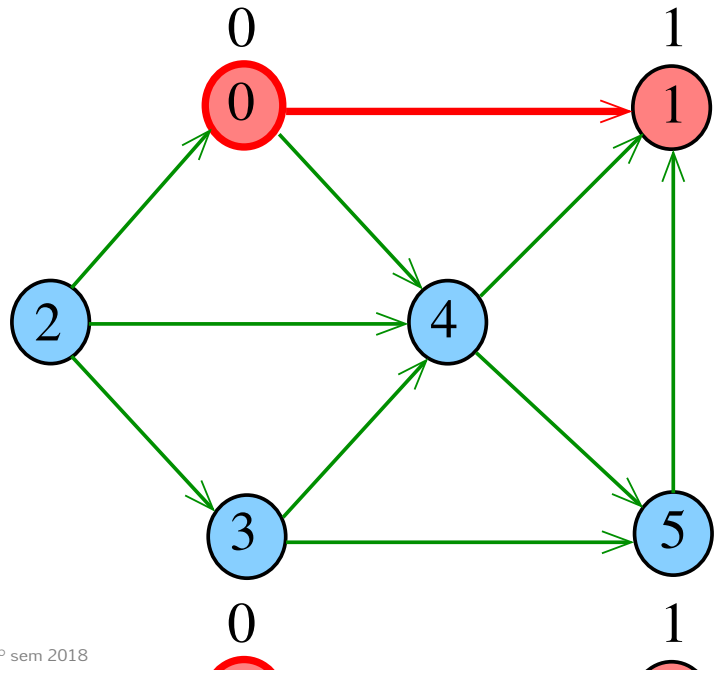
DFS-Visit($G, 0$)



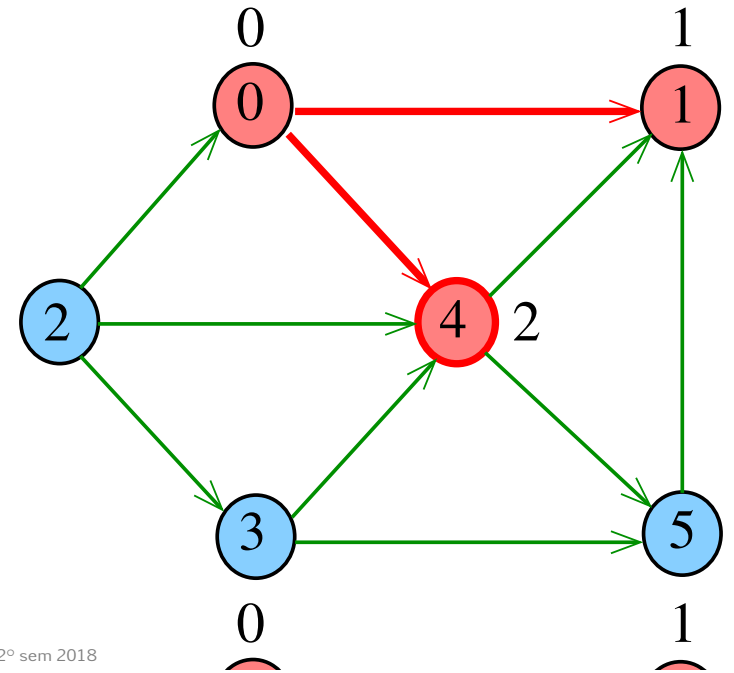
DFS-Visit($G, 1$)



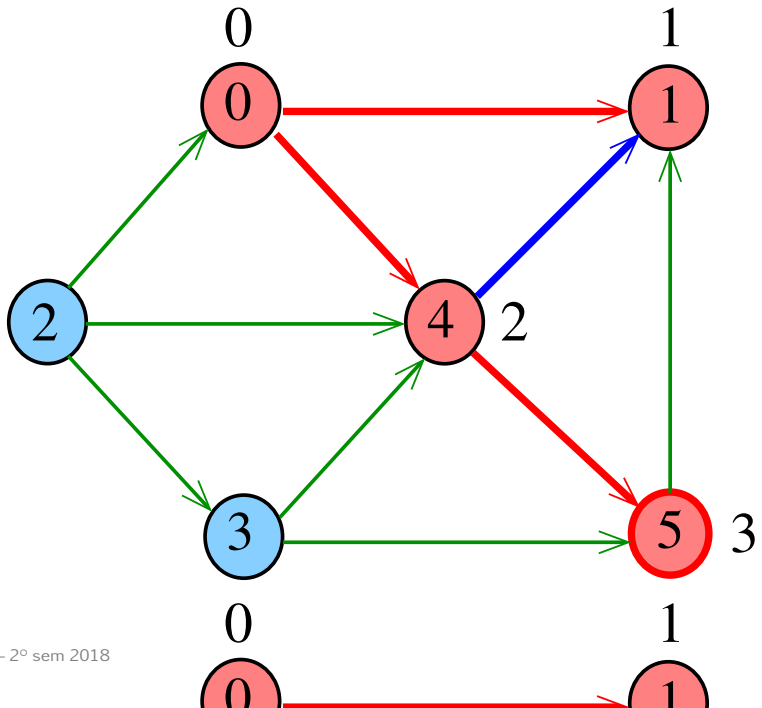
DFS-Visit($G,0$)



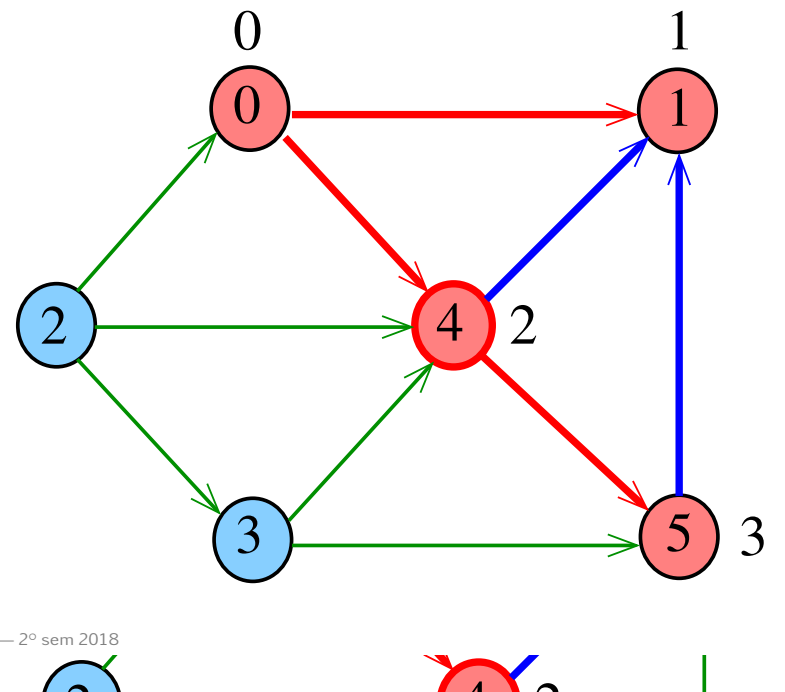
DFS-Visit($G,4$)



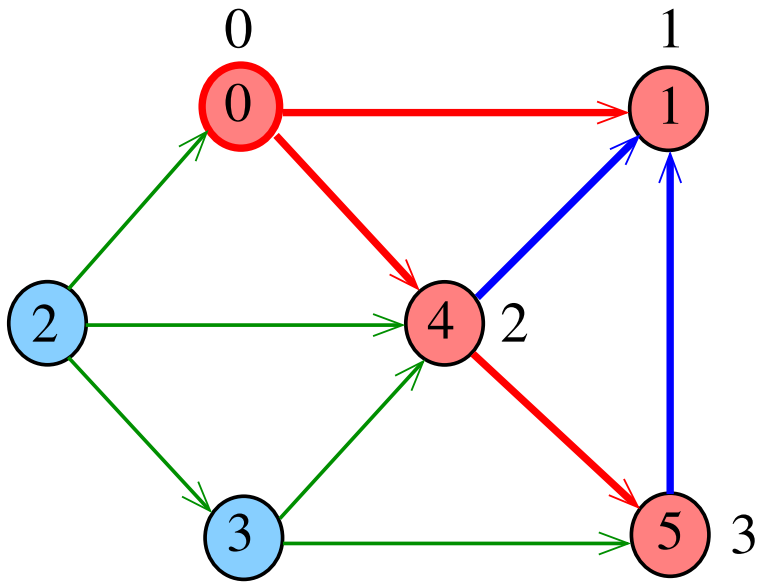
DFS-Visit($G,5$)



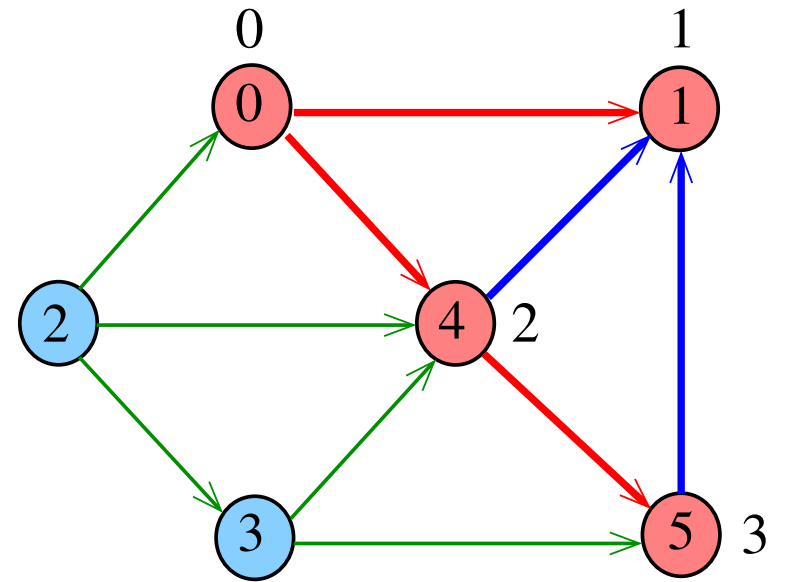
DFS-Visit($G,4$)



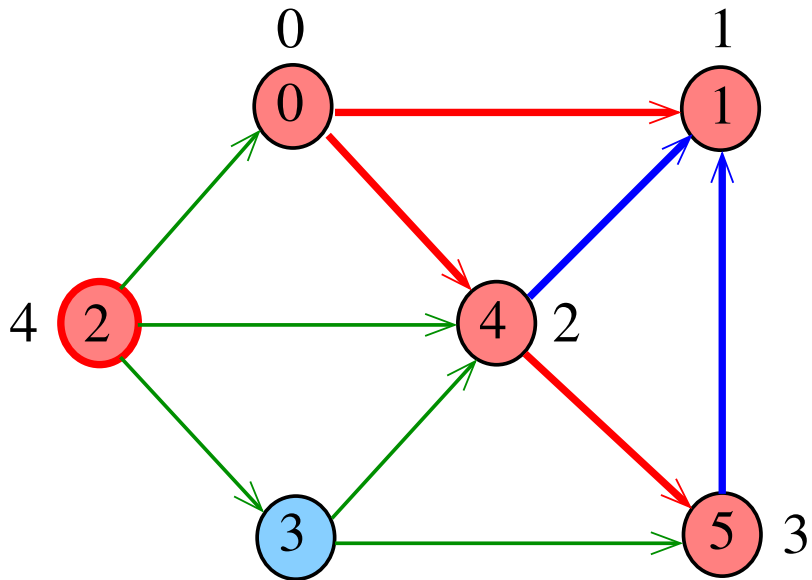
DFS-Visit($G,0$)



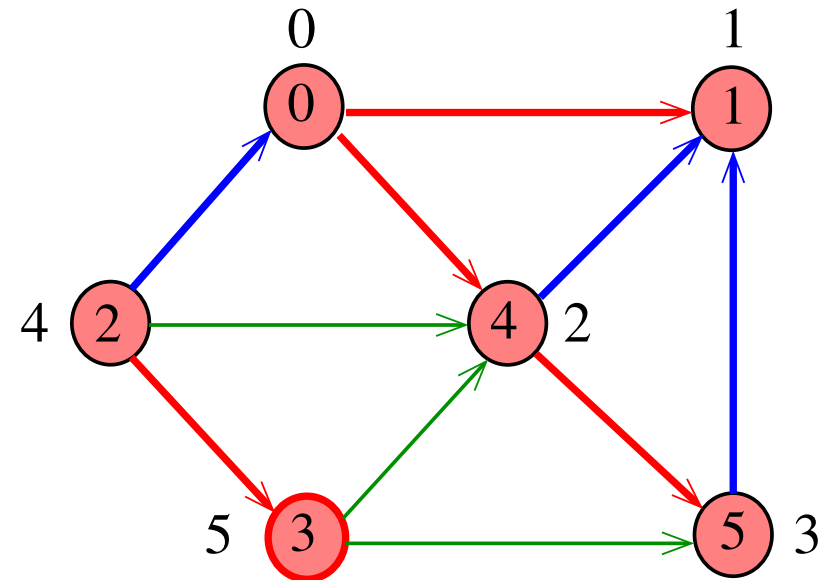
DIGRAPHdfs(G)



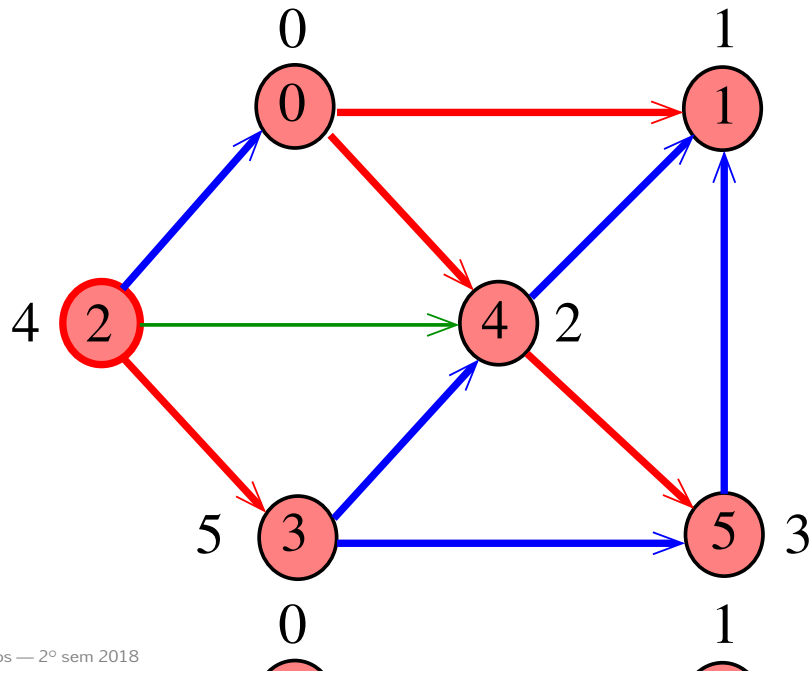
DFS-Visit($G,2$)



DFS-Visit($G,3$)



DFS-Visit($G, 2$)



Consumo de tempo

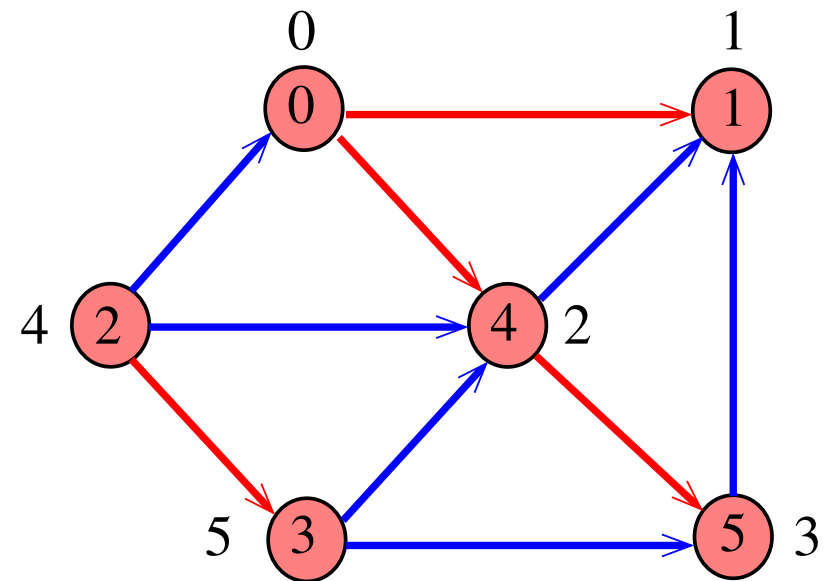
Cada vértice é descoberto uma única vez, pois é branco e, ao ser descoberto, passa a ser cinzento, e depois preto.

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.

DFS(G)



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:

- 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 .
- 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 $u.d < v.d < v.f < u.f$.

Teorema (do caminho branco): v é descendente de u sse, no momento $u.d$ em que u é descoberto, v pode ser alcançado de u por um caminho com apenas vértices brancos.

Classificação das arestas

Quatro tipos de arestas derivadas de uma DFS:

- Arestas da árvore:** arestas da floresta DF.
- Arestas de retorno:** de um vértice para um ascendente na floresta DF.
- Arestas de avanço:** de um vértice para um descendente na floresta DF.
- Arestas cruzadas:** todas as outras arestas.

Teorema: *Se o grafo não é dirigido, então só há arestas da árvore e arestas de retorno.*

Aplicação: ordenação topológica.

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 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, insira-o no início de uma lista ligada.

Devolva a lista ligada.

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, insira-o no início de uma lista ligada.

Devolva a lista ligada.

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.