TÉCNICO LISBOA

Análise e Síntese de Algoritmos

Ordenação topológica.

CLRS Cap. 22

Prof. Pedro T. Monteiro

IST - Universidade de Lisboa

2024/2025

Contexto

- Revisão [CLRS, Cap.1-13]
  - Fundamentos; notação; exemplos
- Técnicas de Síntese de Algoritmos [CLRS, Cap.15-16]
  - Programação dinâmica [CLRS, Cap.15]
  - Algoritmos greedy [CLRS, Cap.16]
- Algoritmos em Grafos [CLRS, Cap.21-26]
  - Algoritmos elementares
  - Caminhos mais curtos [CLRS, Cap.22,24-25]
  - Árvores abrangentes [CLRS, Cap.23]
  - Fluxos máximos [CLRS, Cap.26]
- Programação Linear [CLRS, Cap.29]
  - Algoritmos e modelação de problemas com restrições lineares
- Tópicos Adicionais
  - Complexidade Computacional [CLRS, Cap.34]

Resumo

Ordenação Topológica

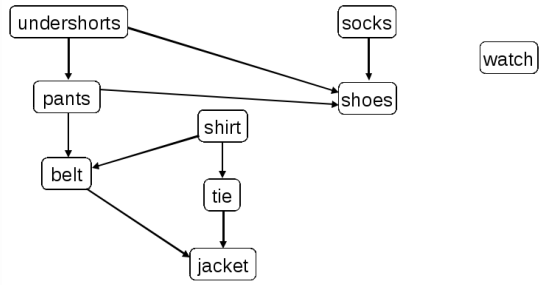
Algoritmo de Kahn

Algoritmo de baseado na DFS

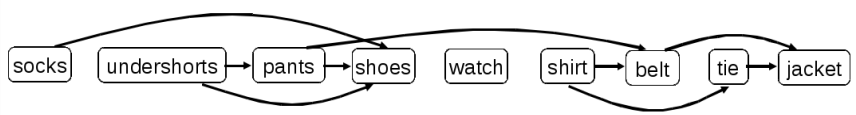
Ordenação Topológica

Motivação

Grafo que representa um conjunto de dependências ou precedências:



Ordenação Topológica:



Caminhos em Grafos

Dado um grafo $G = (V, E)$, um caminho p é uma sequência $< v_0, v_1, ..., v_k >$ tal que para todo o i, $0 \leq i \leq k-1$, $(v_i, v_{i+1}) \in E$

- Se existe um caminho p de u para v, então v diz-se atingível a partir de u usando p
- Um ciclo num grafo $G = (V, E)$ é um caminho $< v_0, v_1, ..., v_k >$, tal que $v_0 = v_k$
- Um grafo dirigido $G = (V, E)$ se não tem ciclos diz-se acíclico (Directed Acyclic Graph – DAG)

Ordenação Topológica

Dado um **DAG** $G = (V, E)$ é uma ordenação de todos os vértices tal que se $(u, v) \in E$ então u aparece antes de v na ordenação

Aplicações

- Gestão dependências pacotes
- Avaliação de células em folhas de cálculo
- Resolução dependências símbolos em linkers
- ...

Soluções Algorítmicas

- Eliminação de vértices
- Utilizando informação de DFS

Algoritmo eliminação de vértices (Kahn's)

Propriedades DAG

Dado que não contém ciclos:

- Existe pelo menos um nó com `indegree=0`
- Existe pelo menos um nó com `outdegree=0`

Algoritmo eliminação de vértices (Kahn's)

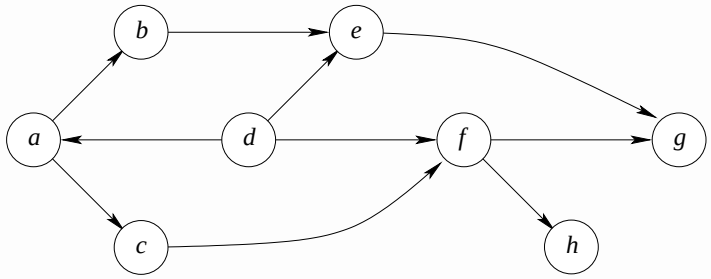
Topological-Sort-Kahn(G)

```
L ← ∅
Q ← ∅
for each v ∈ G.V do
   if (w, v) ∉ G.E then
      enqueue(Q, v)
   end if
end for
while Q ≠ ∅ do
   u = dequeue(Q)
   eliminar todos os arcos (u,v)
   if (w, v) ∉ G.E then
      enqueue(Q, v)
   end if
   L ← L + u
end while
return L
```

Complexidade

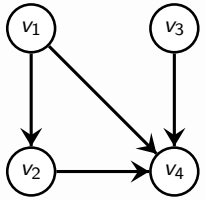
- $\Theta(V + E)$

Exemplo



Ordenação? d, a, b, c, e, f, g, h

Algoritmo baseado na DFS: Intuição



Depois de executar a DFS:

- $f[v_3]$ é sempre $> f[v_4]$
- $f[v_2]$ é sempre $> f[v_4]$
- $f[v_1]$ é sempre $> f[v_2], f[v_4]$

Num DAG, se existe caminho de u para v, então $f[u] > f[v]$!

Logo, basta ordenar os vértices de forma decrescente dos tempos de fim

Algoritmo baseado na DFS

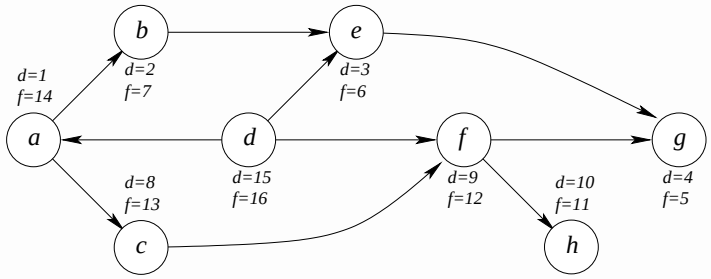
Topological-Sort-DFS(G)

```
DFS(G) para calculo do tempo de fim f[v], para cada v ∈ G.V
Quando um vértice é terminado, inserir numa pilha
return pilha
```

Complexidade

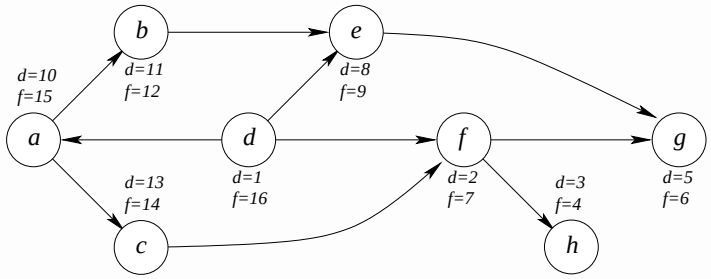
- $\Theta(V + E)$

Exemplo



Ordenação? d, a, c, f, h, b, e, g

Exemplo



Ordenação? d, a, c, b, e, f, g, h

Dúvidas?