Ordenação Topológica
5189-32

Rodrigo Calvo
rcalvo@uem.br

Departamento de Informática – DIN
Universidade Estadual de Maringá – UEM

1° semestre de 2016

Introdução

- Uma ordenação topológica de grafo direcionado acíclico $G = (V, A)$ é uma ordenação linear de todos os vértices, tal que para toda aresta $(u, v) \in A$, u aparece antes de v na ordenação
- Se os vértices forem dispostos em uma linha horizontal, todas as arestas devem ter a orientação da esquerda para direita
- Os grafos direcionados acíclicos são usados para indicar precedências entre eventos.
- Uma aresta direcionada (u, v) indica que a atividade u tem que ser realizada antes da atividade v.

Grafo Direcionado Acíclico

- Grafo Direcionado também é chamado de **Dígrafo**
- Um **Grafo Direcionado Acíclico** é um **Dígrafo** que não tem ciclos
    - Hierarquia de herança entre classes em orientação a objetos
    - Pré-requisitos em disciplinas
    - Restrições de cronograma entre tarefas de um projeto

- Toda **árvore direcionada** é um **dígrafo acíclico**
- Todo caminho num dígrafo acíclico simples não tem repetição de vértices
- Como saber se um dígrafo possui ciclos?

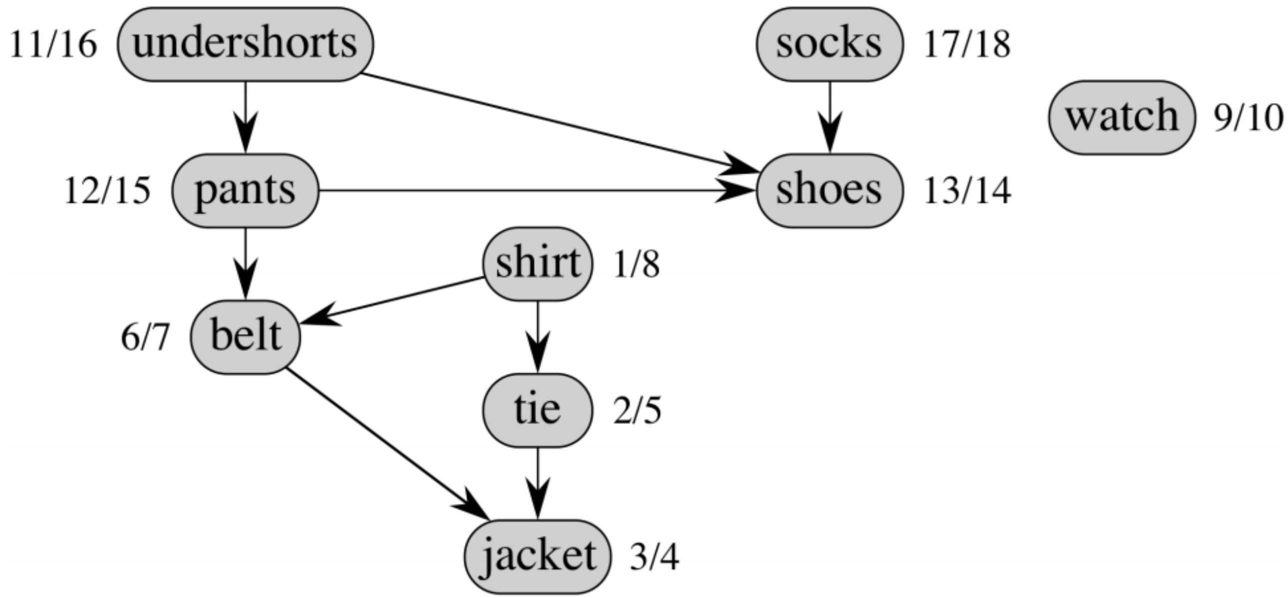
Aplicações

- Dependência entre tarefas
  - Os vértices do dígrafo podem representar tarefas a serem realizadas e as arestas restrições de dependência entre as tarefas
    - Uma ordenação topológica é uma sequência válida de tarefas

- Pré-requisitos de disciplinas
  - Os vértices do dígrafo podem representar as disciplinas de um curso e as arestas pré-requisitos entre as disciplinas
    - Uma ordenação topológica é uma sequência válida para cursar as disciplinas

Exemplo de aplicação

- Exemplo: O professor Bumstead deve se vestir pela manhã



Ordenação Topológica

- Há dois algoritmos para determinar a ordenação topológica de um dígrafo:
  - Eliminação de fontes;
  - Adaptação do algoritmo de busca em profundidade

Eliminação de fontes

- Descrito por Kahn, tem a seguinte estratégia:

1. Encontra vértices “fontes” (com grau de entrada zero) e os insere em um conjunto S (fila ou pilha)
   - Ao menos um vértice de grau de entrada zero deve existir, pois o grafo é acíclico
2. Sabendo que a remoção dos vértices “fontes” juntamente com suas arestas de saída, o grafo remanescente é um dígrafo acíclico:
   - Remove da fila (ou pilha) sucessivamente os vértices fontes
   - Rotula-os em ordem de remoção, e remove suas arestas

Eliminação de fontes
Atributos e Funções

- *v.inDegree* → grau de entrada do vértice *v*
- *v.topsort* → posição do vértice *v* na ordenação topológica

- *newQueue()* → cria uma fila de vértices “fontes”
- *enqueue(S,v*) → insere o vértice *v* na fila *S*
- *dequeue(S*) → remove o primeiro elemento da fila *S*
- *isEmpty(s*) → verifica se a fila *S* está vazia
    - Verdadeiro: a fila está vaiza
    - Falso: a fila não está vazia

Algoritmo

```
topological-sort(G)
S = newQueue() // cria fila S com os vértices fontes
for cada vértice v em G.V
 if v.inDegree = 0
   enqueue(S,v)
t = 0 // inicializa contador
while !isEmpty(S)
 v = dequeue(s) // retira um vértice fonte
 v.topsort = t
 ts[t] = v
 t = t + 1
 for cada vértice u em v.adj
   u.inDegree = u.indegree - 1
   if (u.inDegree = 0)
     enqueue(S,u)
```

Eliminação de fontes

Observações

1. Se uma pilha for utilizada ao invés de uma fila?

Eliminação de fontes

<u>Observações</u>

1. Se uma pilha for utilizada ao invés de uma fila ?
    - Uma ordenação topológica distinta é gerada

Eliminação de fontes

Observações

1. Se uma pilha for utilizada ao invés de uma fila ?
   - Uma ordenação topológica distinta é gerada

2. O que significa o caso em que o contador `t` for maior que o número de vértices ?

Eliminação de fontes

<u>Observações</u>

1. Se uma pilha for utilizada ao invés de uma fila ?
   - Uma ordenação topológica distinta é gerada

2. O que significa o caso em que o contador `t` for maior que o número de vértices ?
   - O dígrafo não á acíclico

Eliminação de fontes: Exemplo

- Um grafo pode ter diversas ordenações topológicas possíveis

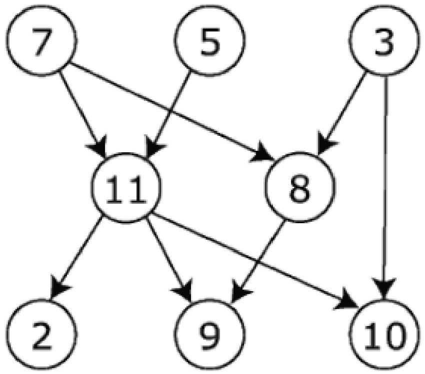


Figura por Derrick Coetzee

- 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

Busca em Profundidade

- A Busca em Profundidade também pode ser utilizada para obter uma ordenação topológica, adaptando o algoritmo **`dfs(G)`** descrito por Tarjan:

1. Iniciando a visita em *v*, visite todos os seus adjacentes (*v, u*), chamando a função **`dfs-visit`** recursivamente para u.
2. Após finalizar a lista de adjacências de cada vértice *v* sendo processado, adicione-o na ordenação topológica

- Utilizando a Busca em Profundidade, é possível também que um dígrafo acíclico possua várias ordenações topológicas possíveis

Algoritmo

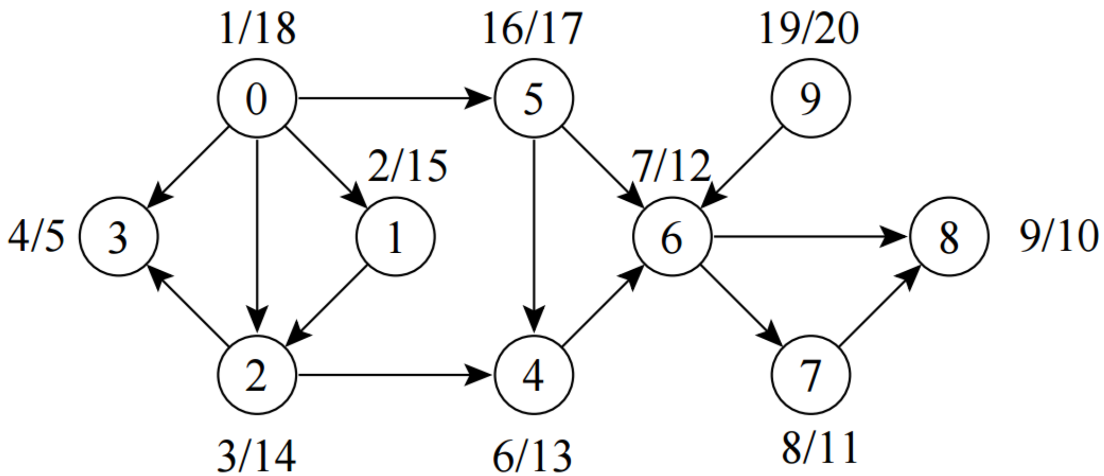
`topological-sort(G)`

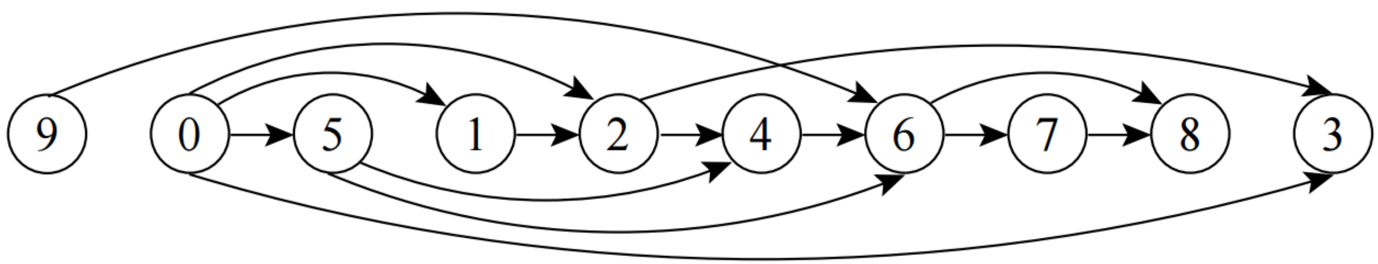
1 chamar **`dfs(G)`** para calcular o tempo de término *v.f* para cada vértice *v*

2 à medida que cada vértice é terminado, inserir o vértice à frente de uma lista ligada

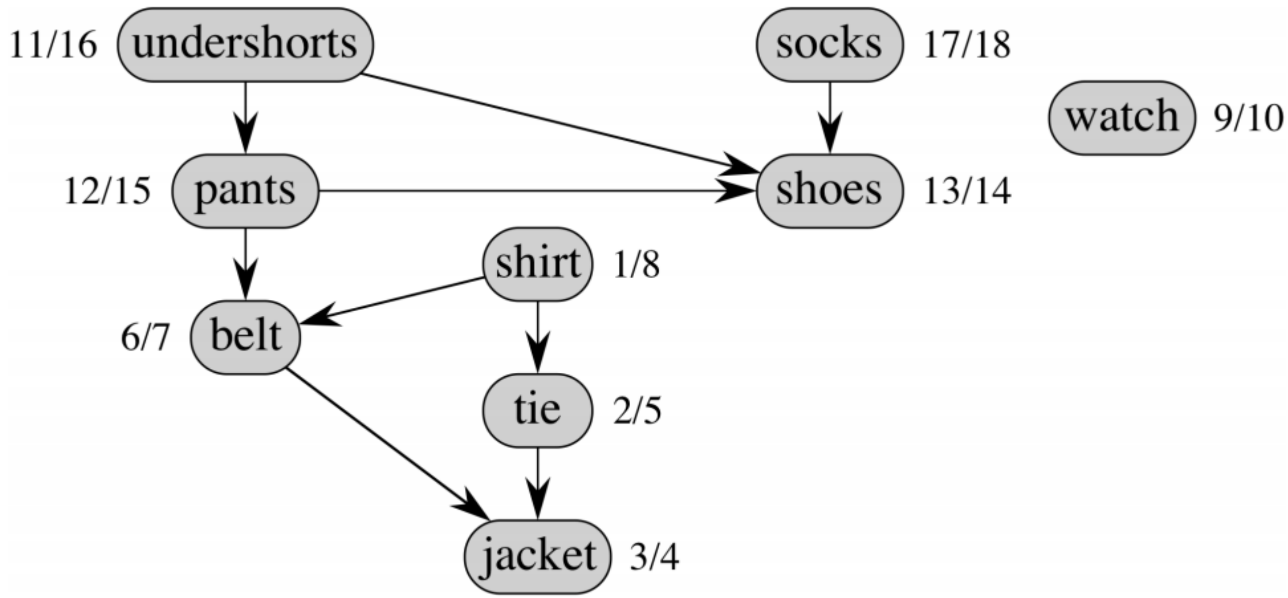
3 devolver a lista ligada de vértices

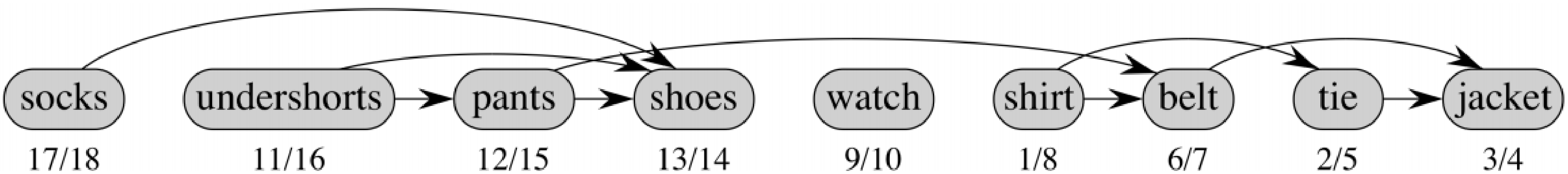
Busca em Profundidade: Exemplo





Busca em Profundidade: Exemplo





Análise do tempo de execução

- O tempo de execução da busca em profundidade é $\Theta(V + A)$
- O tempo para inserir cada vértice na lista de saída é $O(1)$, cada vértice é inserido apenas uma vez e portanto o tempo total gasto em operações de inserção é $\Theta(V)$
- Portanto o tempo de execução do algoritmo é $\Theta(V + A)$

Corretude do `topological_sort`

- É necessário mostrar que se $(u, v) \in A$, então $v.f < u.f$
- Quando a aresta (u, v) é explorada, quais são as cores de u e v?

Corretude do `topological_sort`

- É necessário mostrar que se $(u, v) \in A$, então $v.f < u.f$
- Quando a aresta (u, v) é explorada, quais são as cores de u e v?
- u é cinza
- v é cinza também?

Corretude do `topological_sort`

- É necessário mostrar que se $(u, v) \in A$, então $v.f < u.f$
- Quando a aresta (u, v) é explorada, quais são as cores de u e v?
- u é cinza
- v é cinza também?
  - Não, porque isto implicaria que v é ancestral de u, e portando a aresta (u, v) seria uma aresta de retorno. Grafos direcionados acíclicos não contém arestas de retorno

Corretude do `topological_sort`

- *v* é branco?

Corretude do `topological_sort`

- *v* é branco?
  - Então *v* torna-se um descendente de *u*. Pelo teorema do parênteses

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

Corretude do `topological_sort`

- v é branco?
  - Então v torna-se um descendente de u. Pelo teorema do parênteses

    $u.d < v.d < v.f < u.f$
- v é preto?

Corretude do `topological_sort`

- v é branco?
  - Então v torna-se um descendente de u. Pelo teorema do parênteses $u.d < v.d < v.f < u.f$
- v é preto?
  - Então v já foi finalizado. Como a aresta (u, v) está sendo explorada, u não foi finalizado. Logo $v.f < u.f$

Caminhos Hamiltonianos e Ordenação Topológica

- Se numa ordenação, todos os pares de vértices consecutivos forem conectados por arestas, então essas arestas formam um caminho **Hamiltoniano** direcionado no dígrafo acíclico.
- Se existe um caminho **Hamiltoniano**, a ordenação topológica é única.
- Inversamente, se uma ordenação topológica não formar um caminho **Hamiltoniano**, do dígrafo acíclico terá mais que uma ordenação, pois é possível trocar dois vértices consecutivos que não estão ligados por uma aresta.
- Assim, é possível testar em tempo polinomial se existe uma única ordenação, e portanto um caminho **Hamiltoniano** em um dígrafo

Bibliografia

- Kahn, A.B. Topological sorting of large networks. Communications of the ACM, v.5, n.11, p.558-562, 1962.

  http://portal.acm.org/citation,cfm?id=369025

- Tarjan, R.E. Edge-disjoint spanning trees and depth-first search. Algorithmica v.6, n.2, p.171-185, 1976.

  http://springerlink.com/content/k5633403j221763p

- Thomas H. Cormen et al. Introduction to Algorithms. 3rd edition. Capítulo 22.4.

- Nivio Ziviani. Projeto de Algoritmos com Implementações em Pascal e C. 3a Edição Revista e Ampliada, Cengage Learning, 2010. Capítulo 7. Seção 7.5