AULA 06 – APLICAÇÕES DE BUSCA EM PROFUNDIDADE

Prof. Daniel Kikuti

Universidade Estadual de Maringá

6 de abril de 2015

Sumário

- Introdução
- Ordenação topológica
- Exercícios

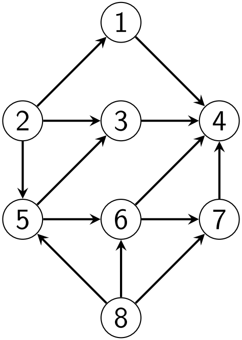
Introdução - Cursos

O problema

Dado o grafo G a seguir, onde:

- Cada vértice representa uma disciplina;
- Cada aresta (u, v) representa uma relação de pré-requisito, isto é, u deve ser cursada antes de v.

Identifique uma ordem para cursar as disciplinas.



Ordenação topológica

- Uma ordenação topológica de um grafo acíclico orientado $G = (V, E)$ é uma ordenação linear de todos os vértices, tal que para toda aresta $(u, v) \in E$, 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.
- Aplicações em problemas que exigem a definição de uma ordem baseada em relações de dependência.

Propriedade

Lema

Um grafo direcionado G é acíclico (DAG) se e somente se a execução do algoritmo de busca em profundidade em G não encontra nenhuma aresta de retorno.

Propriedade

Lema

Um grafo direcionado G é acíclico (DAG) se e somente se a execução do algoritmo de busca em profundidade em G não encontra nenhuma aresta de retorno.

Demonstração

($\Rightarrow$) Se G é acíclico $\Rightarrow$ não há aresta de retorno. [Contrapositiva] Suponha que exista uma aresta de retorno (u, v). Então v é ancestral de u na floresta da busca em profundidade. Portanto existe um caminho $(v \rightsquigarrow u)$, tal que $v \rightsquigarrow u \rightarrow v)$ é um ciclo.

($\Leftarrow$) Se não há uma aresta de retorno $\Rightarrow$ não há um ciclo em G. [Contrapositiva] Suponha que G contém um ciclo c. Seja v o primeiro vértice descoberto em c, e seja (u, v) a aresta precedente em c. No tempo de descoberta de v os vértices em c formam um caminho de vértices brancos $v \rightsquigarrow u$. Logo, u é descendente de v e, portanto, a aresta (u, v) é de retorno.

Algoritmo de ordenação topológica

```
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 finalizado, insera-o
   na cabeça de uma lista ligada
3  devolva a lista ligada de vértices
```

Análise do algoritmo

Consumo de tempo

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

Análise do algoritmo

Consumo de tempo

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

Conclusão

A complexidade do algoritmo `Topological-sort(G)` é $\Theta(V+E)$.

Análise do algoritmo

Correção

Precisamos mostrar que se $(u, v) \in G.E$, então $v.f < u.f$. Quando a aresta (u, v) é explorada, sabemos que u é cinza. E v?

- **Cinza?** Não, porque isto implicaria que u é ancestral de v. $\Rightarrow$ (u, v) é uma aresta de retorno. $\Rightarrow$ contradição com o Lema anterior (um DAG não possui arestas de retorno).
- **Branco?** Então v é descendente de u e portanto $u.d < v.d < \mathbf{v.f} < \mathbf{u.f}$. [ver demonstração do Teorema dos parênteses (Cormen)]
- **Preto?** Então v já foi finalizado. Como a aresta (u, v) está sendo explorada, u não foi finalizado e portanto $v.f < u.f$.

Exercício 1

[22.4-2] Mostre a ordenação de vértices produzida por `Topological-sort(G)` quando ele é executado sobre o DAG da figura abaixo. Considere os vértices em ordem alfabética, e suponha que cada lista de adjacência esteja em ordem alfabética.