

05 – Grafos: ordenação topológica

SCC0503 – Algoritmos e Estruturas de Dados II

Prof. Moacir Ponti Jr.
www.icmc.usp.br/~moacir

Instituto de Ciências Matemáticas e de Computação – USP

2011/1

- 1 Recordando
- 2 Ordenação Topológica
 - Definição
 - Aplicações
- 3 Algoritmos para Ordenação Topológica
- 4 Caminhos Hamiltonianos e Ordenação Topológica

- 1 Recordando
- 2 Ordenação Topológica
 - Definição
 - Aplicações
- 3 Algoritmos para Ordenação Topológica
- 4 Caminhos Hamiltonianos e Ordenação Topológica

uso de DFS e BFS em digrafos permite

- obter para um dado vértice v em G , o subgrafo alcançável a partir de v .
- calcular os componentes fortemente conexos em G .
- testar se G é fortemente conexo — com duas execuções: em G e G^T .
- encontrar um ciclo direcionado em G .
- obter um caminho com o menor número de arestas entre um vértice inicial v e qualquer outro vértice alcançável no em G .

Grafo direcionado acíclico (DAG)

Digrafo que não tem ciclos, como;

- Hierarquia de herança entre classes em orientação a objetos.
 - Pré-requisitos entre disciplinas.
 - Restrições de cronograma entre tarefas de um projeto.
-
- Toda árvore direcionada é um digrafo acíclico.
 - Todo caminho num digrafo acíclico é simples, não tem repetição de vértices.
 - Como saber se um digrafo é acíclico?

1 Recordando

2 Ordenação Topológica

- Definição
- Aplicações

3 Algoritmos para Ordenação Topológica

4 Caminhos Hamiltonianos e Ordenação Topológica

- Uma permutação dos vértices de um digrafo é uma sequência em que cada vértice aparece uma só vez.
- Uma ordenação topológica é uma permutação dos vértices v_1, \dots, v_n , de um digrafo acíclico, de forma que para qualquer aresta (v_i, v_j) , $i < j$.
 - caminhos orientados percorrem os vértices em ordem crescente
 - caminhos orientados percorrem os vértices em ordem crescente
 - qualquer caminho entre v_i e v_j não passa por v_k se $k < i$ ou $k > j$.
- Digrafos com ciclos não admitem ordenação topológica, enquanto todo DAG admite ordenação topológica.

Ordenação Topológica

O grafo abaixo tem diversas ordenações topológicas possíveis:

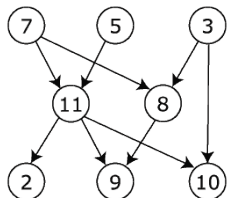


Figura por Derrick Coetzee

- 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

Dependência entre tarefas

Os vértices do digrafo podem representar tarefas a serem realizadas, e os arcos restrições de dependência entre as tarefas.

- uma ordenação topológica é uma sequência válida de tarefas.

Pré-requisitos de disciplinas

Os vértices do digrafo podem representar disciplinas de um curso, e os arcos pré-requisitos entre as disciplinas.

- uma ordenação topológica é uma sequência válida para se cursar as disciplinas.

- 1 Recordando
- 2 Ordenação Topológica
 - Definição
 - Aplicações
- 3 Algoritmos para Ordenação Topológica
- 4 Caminhos Hamiltonianos e Ordenação Topológica

Ao receber um digrafo G , um algoritmo de ordenação topológica deve devolver:

- 1 uma ordenação topológica de G , ou
- 2 um ciclo em G .

Há alguns algoritmos possíveis que retornam uma ordenação topológica. Alguns devem ser adaptados para retornar um ciclo, caso o digrafo não seja um DAG.

Um desses algoritmos, descrito originalmente por Kahn[7], tem a seguinte estratégia:

- 1 encontra os vértices “fonte” (com grau de entrada zero), e os insere em um conjunto S (uma fila)
 - ao menos um vértice desses deve existir se o grafo é acíclico.
- 2 partindo do princípio que, se os vértices “fonte” e seus arcos de saída forem removidos, o grafo remanecente é também um DAG:
 - 1 remove da fila sucessivamente os vértices fontes,
 - 2 rotula-os em ordem de remoção, e remove seus arcos.

Algoritmos de ordenação topológica

```
int TopologicalSort(G, tsL) {
    S = newQueue(); // fila S com vertice fonte
    for (all vertices v in G) {
        if (deg(v) == 0)
            enqueue(S,v); // insere na fila S
    }
    t = 0; // inicia contador
    while (!isEmpty(S)) {
        v = dequeue(S); // retira fonte de S
        v.topsort = t; // marca ordenacao
        tsL[t++] = v; // insere em arranjo ordenado
        for (all incident vertices (v,w)) {
            w.inDegree--; // reduz grau de entrada dos adjacentes
            if (w.inDegree == 0)
                enqueue(S,w) // quando forem fontes, inserir em S
        }
    }
}
```

Observações sobre o algoritmo de eliminação de fontes

- pode ser implementado também com uma pilha, o que gera uma ordenação topológica diferente (mas válida).
- o que significa é verdadeira uma condição ao final da execução do algoritmo: `if (t != G->V)?`
 - o digrafo não é um DAG, t vértices foram inseridos na lista ordenada e existiam $G \rightarrow V$ vértices no digrafo.





DFS para ordenação topológica

A busca em profundidade (DFS) pode também ser utilizada para obter uma ordenação topológica, adaptando o algoritmo conforme descrito por Robert Tarjan [8]. Para isso:

- 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, adicione-o na ordenação topológica.

- 1 Recordando
- 2 Ordenação Topológica
 - Definição
 - Aplicações
- 3 Algoritmos para Ordenação Topológica
- 4 Caminhos Hamiltonianos e Ordenação Topológica

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

-  SEDGEWICK, R.
Algorithms in C: part 5, 3.ed., Addison-Wesley, 2002.
Digraphs and DAGs (Seção 19.6)
-  ZIVIANI, N.
Projeto de Algoritmos, 3.ed. Cengage, 2004.
(Capítulo 7)
-  CORMEN, T. H. et al.
Algoritmos: teoria e prática. Campus-Elsevier, 2002.
Ordenação Topológica (Seção 22.4).
-  TENEMBAUM, A.M. et al.
Estruturas de Dados Usando C. Pearson Makron, 1995.
Grafos e suas aplicações (Capítulo 8).



KNUTH, D.

The Art of Computer Programming: fundamental algorithms, v.1.
Addison-Wesley, 1969.

Basic Mathematical Properties of Trees (Seção 2.3.4)



FEOFILOFF, P.

Algoritmos de Ordenação Topológica.

[http:](http://www.ime.usp.br/~pf/algoritmos_para_grafos/aulas/toposorting.html)

[//www.ime.usp.br/~pf/algoritmos_para_grafos/aulas/toposorting.html](http://www.ime.usp.br/~pf/algoritmos_para_grafos/aulas/toposorting.html)



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://www.springerlink.com/content/k5633403j221763p>