Ordenação topológica

Marco A L Barbosa
malbarbo.pro.br

Departamento de Informática
Universidade Estadual de Maringá

Vamos considerar a execução de algum as atividades:

- Montagem de um computador
- Criação de um arquivo para distribuição de um programa a partir do código fonte
- Vestimento da roupa pela manhã

Montagem de um produto

Quais são as etapas necessárias na montagem de um computador?

- Colocação do processador na placa mãe
- Colocação do dissipador
- Colocação dos fans
- Fixação da placa mãe no gabinete
- Colocação da memória na placa mãe
- Fixação do disco no gabinete
- . . .

Estas tarefas podem ser executadas em qualquer ordem? Não!

Distribuição de programas

Quais ações (execução de comandos) são necessárias para criar um arquivo para distribuição de um programa a partir do código fonte?

- Compilação de cada arquivo de código fonte (possivelmente em linguagens diferentes) para gerar os códigos objeto
- Ligação dos códigos objetos em arquivos executáveis ou bibliotecas dinâmicas
- Remoção dos símbolos de arquivos executáveis
- Criação dos diretórios onde serão armazenados os arquivos produzidos
- Execução dos testes automatizados
- . . .

Estas tarefas podem ser executadas em qualquer ordem? Não!

Colocar a roupa pela manhã

Quais são as etapas necessárias para colocar a roupa pela manhã?

- Colocação das meias
- Colocação da calça
- Colocação das roupas íntimas
- Colocação da blusa
- Colocação do cinto
- ...

Estas tarefas podem ser executadas em qualquer ordem? Não!

Problema

Muitos tarefas requerem a realização de diversas etapas para serem executas. Em geral, existem algumas precedências entre as etapas (algumas precisam ser executas antes do que outras).

Nesse contexto, queremos determinar uma ordem de execução das tarefas que seja válida (nenhum tarefa pode ser executada antes das tarefas que a precedem)

Como podemos modelar este problema usando grafos?

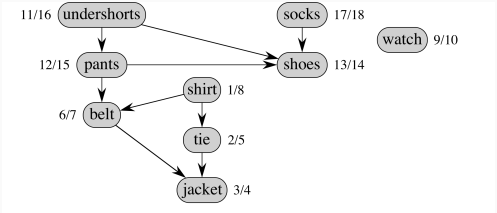
- Representando cada etapa por um vértice
- E cada precedência do tipo “a etapa 1 deve ser realizada antes da etapa 2” com uma aresta direcionada que saí do vértice que representa a etapa 1 e entra no vértice que representa a etapa 2.

Exemplo

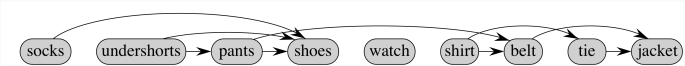
Por exemplo, considere a colocação de cada peça de roupa como uma etapa:

- Camisa
- Meia
- Calça
- Roupa íntima
- ...

Nós podemos representar estas etapas e suas precedências com o seguinte grafo:



Em que ordem podemos vestir as roupas para que elas “fiquem certas”?



Qualquer sequência para colocar as roupas de “forma certa” terá a seguinte característica: para cada aresta (u, v) do grafo o vértice u aparece antes que o vértice v na sequência.

Uma sequência de vértices com essas características é chamada de ordenação topológica.

Encontrar uma ordenação topológica dos vértices de um grafo orientado acíclico é um problema clássico da computação. Vamos ver como resolver este problema.

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.

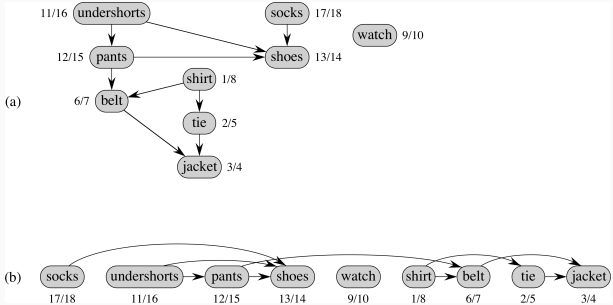
- Se os vértices forem dispostos em uma linha horizontal, todas as arestas devem ter a orientação da esquerda para direita

Problema

Projete um algoritmo que receba como entrada um grafo acíclico orientado e encontre uma ordenação topológica dos seus vértices.

Observe que formulamos o problema em termos abstratos, de certa forma a ideia de tarefas e precedência não é mais necessária.

Baseado no exemplo das roupas, podemos criar uma "hipótese" de como resolver o problema? (Ignore por enquanto os carimbos de tempo)



Método incremental

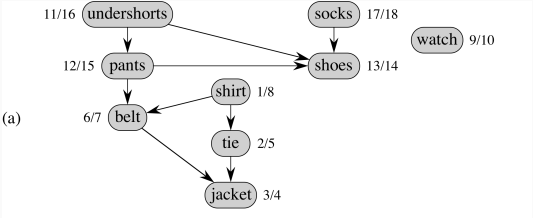
- Selecione um vértice com grau de entrada 0 e adicione no final da sequência
- Remova este vértice do grafo e repita o processo até que o grafo fique vazio

Este método funciona e foi descrito pela primeira vez por Kahn.

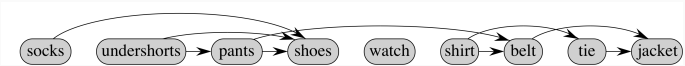
Exercício

Implemente o algoritmo de Kahn mas sem remover de fato os vértices do grafo.

Uma ordenação topológica também pode ser encontrada usando a busca em profundidade.



Considerando os eventos de descoberta e termino dos vértices na ordem que eles acontecem, podemos determinar uma ordenação topológica para os vértices?



TOPOLOGICAL-SORT(G)

1 chamar DFS(G) para calcular o tempo de término $v.f$ para cada vértice
2 à medida que cada vértice é finalizado, inserir o vértice à frente de uma lista ligada
3 **return** a lista ligada de vértices

Tempo de execução

- O tempo de execução da busca em profundidade é $\Theta(V + E)$
- 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 inserções é de $\Theta(V)$
- Portanto, o tempo de execução do algoritmo é $\Theta(V + E)$

Lema 22.11

Um grafo orientado G é acíclico se e somente se uma busca em profundidade de G não encontra arestas de retorno.

Prova

Veja o livro.

Teorema 22.12

TOPOLOGICAL-SORT(G) produz uma ordenação topológica do grafo acíclico orientado G.

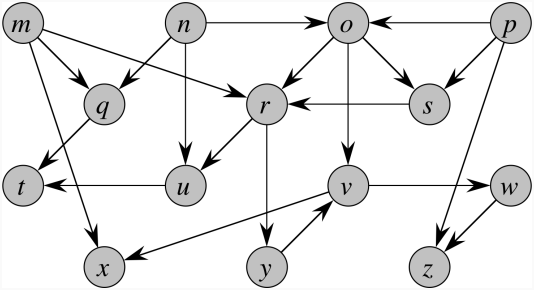
Precisamos mostrar que se $(u, v) \in E$, então $v.f < u.f$.

Quando a aresta (u, v) é explorada, quais são as cores de u e v?

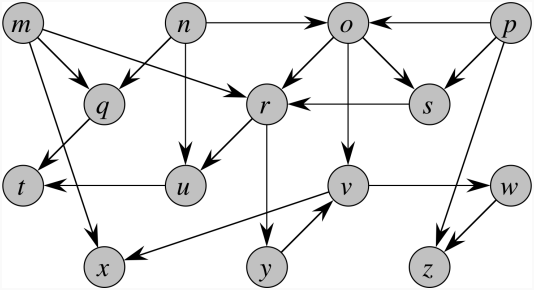
- u é cinza
- v é cinza também? Não, porque isto implicaria que v é ancestral de u, e portando a aresta (u, v) seria uma aresta de retorno. Gaos não contém arestas de retorno.
- v é branco? Então v torna-se um descendente de u. Pelo teorema do parênteses $u.d < v.d < \mathbf{v}.f < \mathbf{u}.f$.
- v é preto? Então v já foi finalizado. Como a aresta (u, v) está sendo explorada, u não foi finalizado, logo $v.f < u.f$.

□

22.4-1 Mostre a ordem do vértices produzido por TOPOLOGICAL-SORT quando executado no gao da Figura 22.8 (assuma que o for das linhas 5-7 do DFS considera os vertices em ordem alfabética e que os vértices nas listas de adjacência estão em ordem alfabética)



22.4-2 Dê um algoritmo de tempo linear que receba com entrada um grafo acíclico orientado $G = (V, E)$ e dois vértices s e t e retorne o número de caminhos simples de s para t em G. Por exemplo, o gao da Figura 22.8 contém exatamente quatro caminhos simples do vértice p para o vértice v: pov, $poryv$, $posryv$ e $psryv$.



Exercícios

Veja a lista de exercícios e algumas soluções na página da disciplina.

Referências

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