UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS DE CRATEÚS

Ordenação Topológica e Conectividade

Algoritmos em grafos

Professor: Rennan Dantas

Universidade Federal do Ceará
Campus de Crateús

06 de abril de 2022

Ordenação Topológica

Introdução

- Veremos como podemos usar busca em profundidade para executar uma ordenação topológica de um grafo acíclico dirigido (ou "gad")
- Uma ordenação topológica de um gad $G = (V, E)$ é uma ordenação linear de todos os seus vértices, tal que se G contém uma aresta (u, v), então u aparece antes de v na ordenação (se o grafo contém um ciclo, nenhuma ordenação topológica é possível)
- Podemos ver uma ordenação topológica de um grafo como uma ordenação de seus vértices ao longo de uma linha horizontal de modo que todas as arestas dirigidas vão da esquerda para a direita

Exemplo

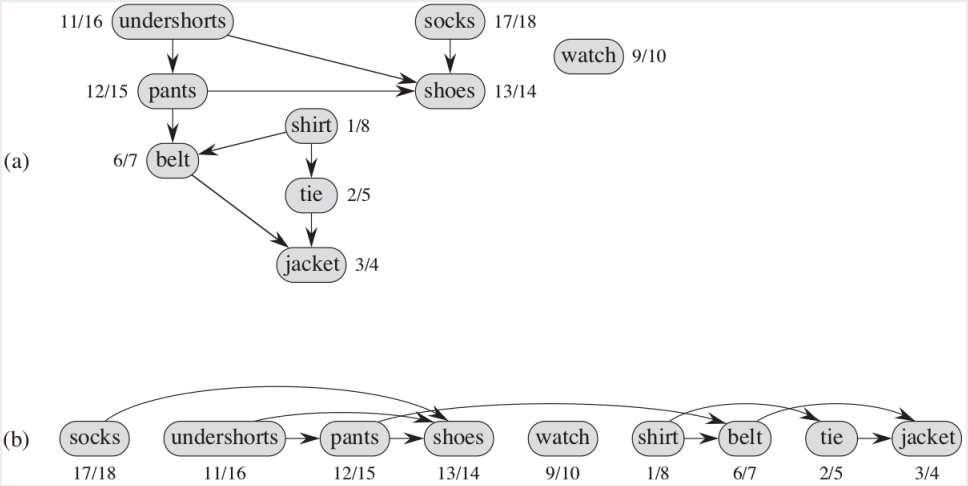

Figura: Fonte: Livro Algoritmos - Cormen

Ordenação Topológica

Algoritmo

O seguinte algoritmo ordena topologicamente um gad:

TOPOLOGICAL-SORT(G)

1 call DFS(G) to compute finishing times $v.f$ for each vertex v
2 as each vertex is finished, insert it onto the front of a linked list
3 **return** the linked list of vertices

Figura: Fonte: Livro Algoritmos - Cormen

Ordenação Topológica

- Podemos executar uma ordenação topológica no tempo $\Theta(V + E)$, já que a busca em profundidade demora o tempo $\Theta(V + E)$ e que inserir cada um dos $|V|$ vértices à frente da lista encadeada leva o tempo $O(1)$
- Demonstramos a correção desse algoritmo utilizando o seguinte lema fundamental que caracteriza grafos acíclicos dirigidos

Ordenação Topológica

Lema

Um grafo dirigido G é acíclico se e somente se uma busca em profundidade de G não produz nenhuma aresta de retorno

Prova

- $\Rightarrow$ Suponha que uma busca em profundidade produza uma aresta de retorno (u, v). Então o vértice v é um ancestral do vértice u na floresta em profundidade
- Assim, G contém um caminho de v a u, e a aresta de retorno (u, v) completa um ciclo

Prova

- ($\Leftarrow$) Suponha que G contenha um ciclo c
- Mostramos que uma busca em profundidade de G produz uma aresta de retorno
- Seja v o primeiro vértice a ser descoberto em c e seja (u, v) a aresta precedente em c
- No tempo $v.d$, os vértices de c formam um caminho de vértices brancos de v a u
- Pelo teorema do caminho branco, o vértice u se torna um descendente de v na floresta em profundidade
- Então, (u, v) é uma aresta de retorno $\square$

Ordenação Topológica

Teorema

Topologic-Sort produz uma ordenação topológica de um grafo acíclico dirigido dado como sua entrada.

Prova

- Suponha que DFS seja executado em determinado gad $G = (V, E)$ para determinar tempos de término para seus vértices
- É suficiente mostrar que, para qualquer par de vértices distintos $u, v \in V$, se G contém uma aresta de u a v, então $v.f < u.f$
- Considere qualquer aresta (u, v) explorada por $DFS(G)$
- Quando essa aresta é explorada, v não pode ser cinzento, já que nesse caso v seria um ancestral de u e (u, v) seria uma aresta de retorno, o que contradiz o Lema anterior
- Portanto, v deve ser branco ou preto

Prova

- Se v é branco, ele se torna um descendente de u e, assim, $v.f < u.f$
- Se v é preto, ele já terminou, de modo que $v.f$ já foi definido
- Como ainda estamos explorando u, ainda temos de atribuir um carimbo de tempo a $u.f$ e, tão logo o façamos, também teremos $v.f < u.f$
- Assim, para qualquer aresta (u, v) no gad, temos $v.f < u.f$, o que prova o teorema □

Componentes fortemente conexas

- Consideraremos agora uma aplicação clássica de busca em profundidade: a decomposição de um grafo dirigido em suas componentes fortemente conexas
- Veremos como fazer isso usando duas buscas em profundidade
- Muitos algoritmos que funcionam com grafos dirigidos começam por uma decomposição desse tipo
- Após a decomposição do grafo em componentes fortemente conexas, tais algoritmos são executados separadamente em cada uma delas e combinados em soluções de acordo com a estrutura das conexões entre componentes

Componentes fortemente conexas

Uma componente fortemente conexa de um grafo dirigido $G = (V, E)$ é um conjunto máximo de vértices $C \subseteq V$ tal que, para todo par de vértices u e v em C, temos $u \rightsquigarrow v$ e $v \rightsquigarrow u$; isto é, u pode ser alcançado a partir do vértice v e vice-versa.

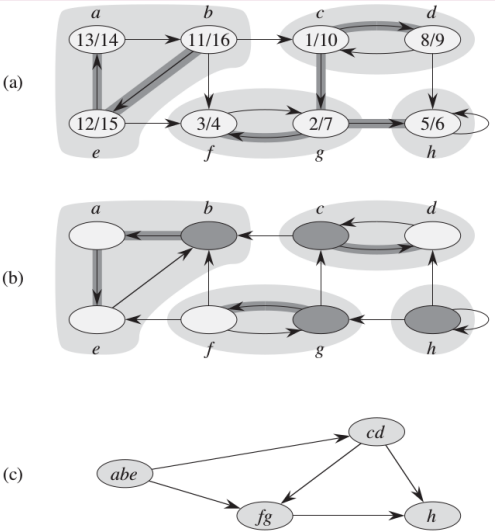

Figura: Fonte: Livro Algoritmos - Cormen

Componentes fortemente conexas

- Nosso algoritmo para encontrar componentes fortemente conexas de um grafo $G = (V, E)$ usa o transposto de G, que é definida como o grafo $G^T = (V, E^T)$, onde $E^T = \{(u, v) : (v, u) \in E\}$
- Isto é, E^T consiste nas arestas de G com suas direções invertidas
- É interessante observar que G e G^T têm exatamente as mesmas componentes fortemente conexas: u e v, podem ser alcançados um a partir do outro em G se e somente se puderem ser alcançados um a partir de outro em G^T

Componentes fortemente conexas

O algoritmo de tempo linear (isto é $\Theta(V + E)$) apresentado a seguir calcula as componentes fortemente conexas de um grafo dirigido $G = (V, E)$ usando duas buscas em profundidade, uma em G e uma em G^T

STRONGLY-CONNECTED-COMPONENTS(G)

1 call DFS(G) to compute finishing times $u.f$ for each vertex u
2 compute G^{T}
3 call DFS(G^{T}), but in the main loop of DFS, consider the vertices in order of decreasing $u.f$ (as computed in line 1)
4 output the vertices of each tree in the depth-first forest formed in line 3 as a separate strongly connected component

Figura: Fonte: Livro Algoritmos - Cormen

Lema 1

Sejam C e C' componentes fortemente conexas distintas em um grafo dirigido $G = (V, E)$, seja $u, v \in C$, seja $u', v' \in C'$ e suponha que G contenha um caminho $u \rightsquigarrow u'$. Então, G não pode conter também um caminho $v' \rightsquigarrow v$

Prova

- Se G contém um caminho $v' \rightsquigarrow v$, então contém caminhos $u \rightsquigarrow u' \rightsquigarrow v'$ e $v' \rightsquigarrow v \rightsquigarrow u$ em G
- Assim, u e v' podem ser visitados um a partir do outro, o que contradiz a hipótese de que C e C' são componentes fortemente conexas distintas $\square$

Componentes fortemente conexas

- Veremos que, considerando vértices na segunda busca em profundidade em ordem decrescente dos tempos de término que foram calculados na primeira busca em profundidade, estamos em essência, visitando os vértices do grafo de componentes (cada um dos quais corresponde a uma componente fortemente conexa de G) em sequência ordenada topologicamente
- Como o procedimento Strongly-Connected-Components executa duas buscas em profundidade, há potencial para ambiguidade quando discutimos $u.d$ ou $u.f$
- Esses valores sempre correspondem aos tempos de descoberta e término calculados pela primeira chamada de DFS, na linha 1

Componentes fortemente conexas

- Estendemos a notação de tempos de descoberta e término a conjunto de vértices
- Se $U \subseteq V$, então definimos d(U)=min$_{u \in U} u.d$ e f(U)=max$_{u \in U} u.f$
- Isto é, $d(U)$ e $f(U)$ são o tempo de descoberta mais antigo e o tempo de término mais recente, respectivamente, de qualquer vértice em U
- O lema a seguir e seu corolário dão uma propriedade fundamental que relaciona componentes formalmente conexas a tempos de término na primeira busca em profundidade

Lema

Sejam C e C' componentes fortemente conexas distintas no grafo dirigido $G = (V, E)$. Suponha que exista uma aresta $(u, v) \in E$, onde $u \in C$ e $v \in C'$. Então, $f(C) > f(C')$.

Prova

Livro/Quadro

Corolário

Sejam C e C' componentes fortemente conexas distintas no grafo dirigido $G = (V, E)$. Suponha que exista uma aresta $(u, v) \in E^T$, onde $u \in C$ e $v \in C'$. Então, $f(C) < f(C')$.

Prova

Como $(u, v) \in E^T$, temos $(v, u) \in E$. Visto que as componentes fortemente conexas de G e G^T são as mesmas, o lema anterior implica $f(C) < f(C')$ $\square$

Componentes fortemente conexas

- Esse corolário nos dá a chave para entender por que o algoritmo de componentes funciona
- Vamos examinar o que acontece quando executamos a segunda busca profundidade, que está em G^T
- Começamos com a componente fortemente conexa C cujo tempo de término $f(C)$ é máximo
- A busca começa em algum vértice $x \in C$ e visita todos os vértices em C
- Pelo corolário anterior, G^T não contém nenhuma aresta de C a qualquer outra componente fortemente conexa e, por isso, a busca iniciada em x não visitará vértices em qualquer outra componente

Componentes fortemente conexas

- Assim, a árvore com raiz em x contém exatamente os vértices de C
- Agora que as visitas a todos os vértices de C foram concluídas, a busca na linha 3 seleciona como raiz um vértice de alguma outra componente fortemente conexa C' cujo tempo de término $f(C')$ é máximo em relação a todas as outras componentes, exceto C
- Mais uma vez, a busca visitará todos os vértices em C' mas, pelo corolário, as únicas arestas em G^T que vão até C' a qualquer outra componente devem ir até C, que já visitamos
- Em geral, quando a busca em profundidade de G^T na linha 3 visita qualquer componente fortemente conexa, quaisquer arestas que saem dessa componente devem ir até componentes que a busca já visitou
- Então, cada árvore de busca em profundidade será exatamente uma componente fortemente conexa

O teorema a seguir formaliza esse argumento

Teorema

O procedimento Strongly-Connected-Components calcula corretamente as componentes fortemente conexas do grafo dirigido dado como sua entrada

Prova

- Mostramos por indução em relação ao número de árvores de busca encontradas na busca em profundidade de G^T na linha 3 que os vértices de cada árvore formam uma componente fortemente conexa
- A hipótese da indução é que as primeiras k árvores produzidas na linha 3 são componentes fortemente conexas
- A base para a indução, quando $k = 0$, é trivial

Componentes fortemente conexas

- No passo da indução, supomos que cada uma das k primeiras árvores em profundidade produzidas na linha 3 é uma componente fortemente conexa, e consideramos a $(k+1)$-ésima árvore produzida
- Seja o vértice u a raiz dessa árvore, e suponhamos que u esteja na componente fortemente conexa C
- Como resultado do modo como escolhemos raízes na busca em profundidade na linha 3, $u.f = f(C) > f(C')$ para qualquer componente fortemente conexa C' exceto C que ainda tenha de ser visitada
- Pela hipótese de indução, no momento em que a busca visita u, todos os outros vértices de C são brancos
- Então, pelo teorema do caminho branco, todos os outros vértices de C são descendentes de u nessa árvore em profundidade

Componentes fortemente conexas

- Além disso, pela hipótese de indução e pelo corolário, quaisquer arestas em G^T que saem de C devem ir até componentes conexas que já foram visitadas
- Assim, nenhum vértice em uma componente fortemente conexa exceto C será um descendente de u durante a busca em profundidade de G^T
- Portanto, os vértices da árvore de busca em profundidade em G^T enraizada em u formam exatamente uma componente fortemente conexa, o que conclui o passo de indução e a prova $\square$

Próxima Aula

O que vem por aí?

- Exercícios
- Teste 01
- Revisão/tira dúvida
- Prova 01

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS DE CRATEÚS

Ordenação Topológica e Conectividade

Algoritmos em grafos

Professor: Rennan Dantas

Universidade Federal do Ceará
Campus de Crateús

06 de abril de 2022