Lista de Exercícios 5

Universidade Federal do Ceará - Campus Quixadá
Projeto e Análise de Algoritmo — QXD0041 – 2024.2
Prof. Fabio Dias

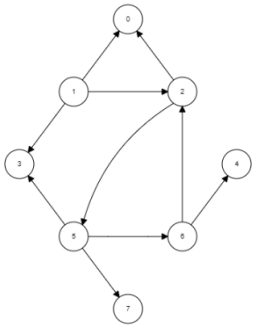
Grafos e Busca em Grafos

1. O transposto de um grafo direcionado $G = (V, E)$ é o grafo direcionado $G_T = (V, E_T)$, onde $E_T = (v, u) \in VV : (u, v) \in E$. Assim, G_T é G com todas as suas arestas invertidas. Descreva algoritmos eficientes para calcular G_T a partir de G, para a representação por lista de adjacências e também para a representação por matriz de adjacências de G. Analise os tempos de execução de seus algoritmos.

2. Mostre como determinar se um grafo direcionado G contém um **sorvedouro universal** — um vértice com grau de entrada $|V|$–1 e grau de saída 0 — no tempo $O(V)$, dada uma matriz de adjacências para G.

3. **Quando um grafo possui mais de uma aresta interligando os mesmo dois vértices diz-se que este grafo possui arestas múltiplas(ou arestas paralelas). Ele é chamado de multigrafo ou grafo múltiplo.** Dada uma representação por lista de adjacências de um multigrafo $G = (V, E)$, descreva um algoritmo de tempo $O(V + E)$ para calcular a representação por lista de adjacências do grafo não direcionado "equivalente" $G' = (V, E')$, onde E' consiste nas arestas em E onde todas as arestas múltiplas entre dois vértices foram substituídas por uma aresta única e onde todos os laços foram removidos. **Um laço é uma aresta incidente ao mesmo vértice. Por exemplo, a aresta (v,v).** Laços somente ocorrem em multigrafos.

4. Aplique a Busca em Largura no grafo abaixo, mostrado a cada passo, o crescimento da árvore de busca em largura.



5. Modifique a Busca em Largura para que, além de calcular o tamanho do menor caminho da origem a todo vértice, ela também calcule o número de caminhos distintos de menor tamanho.

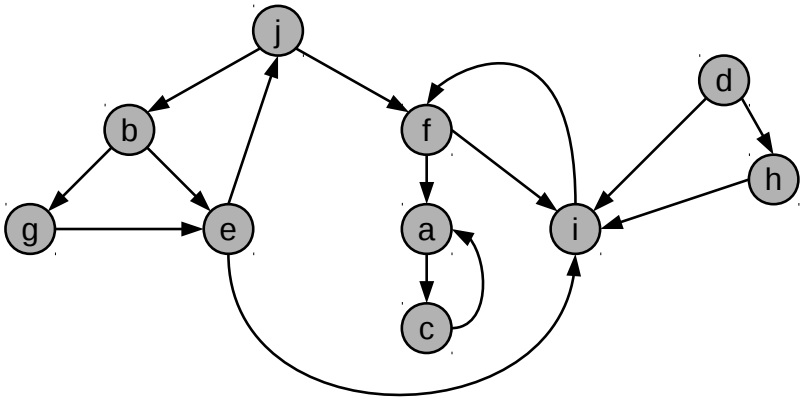
6. Em muitas aplicações de redes de computadores, são necessário definir redundâncias na rede, para evitar que um certo servidor fique fora do ar devido a uma falha em um link. Dado um grafo $G = (V, E)$ conexo e não direcionado, precisamos calcular a quantidade de caminhos distintos de um vértice origem a todos os demais vértices do grafo. Modifique a Busca em Largura para resolver esse problema.

7. Modifique a Busca em Largura para funcionar em um grafo representado por uma matriz de adjacências. Qual é o tempo de execução dessa versão?

8. Na busca em Largura, cada vértice é pintado de cinza imediatamente antes de ser adicionado na Fila Q, indicando que esse vértice foi descoberto. O que acontece com a busca em Largura se pintamos o vértice de cinza apenas quando o vértice é removido da Fila Q? O algoritmo ainda funciona? Sua complexidade é alterada?

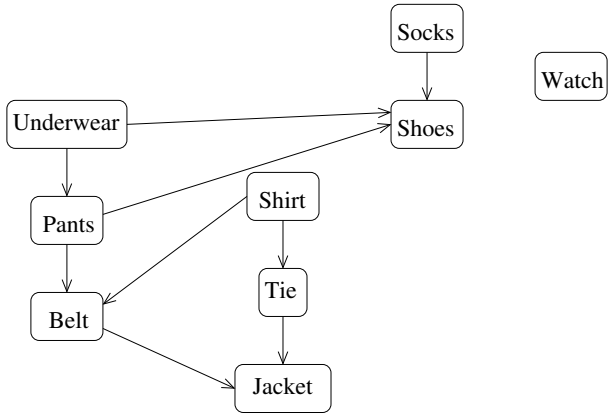
9. Dado um grafo não direcionado, como podemos determinar se esse grafo possui ou não um ciclo no tempo $O(V + E)$?

10. Mostre como a busca em profundidade funciona no grafo da figura abaixo. Presuma que o laço da rotina principal DFS considera os vértices em ordem alfabética e presuma que cada lista de adjacências está ordenada alfabeticamente. Mostre o tempo de descobrimento (**d**) e o tempo de término (**f**) de cada vértice na caixa abaixo.

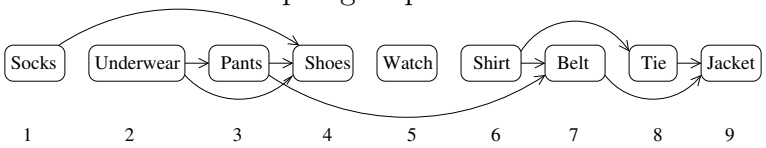


	a	b	c	d	e	f	g	h	i	j
d										
f										

11. Uma ordenação topológica de um grafo direcionado acíclico é uma ordem linear dos vértices tal que $\forall(u, v) \in E \Rightarrow u$ aparece antes v na ordem linear. Normalmente é usada em problema de agendamento e similares. Por exemplo, ordem de Roupas (a seta implica "deve vestir antes"). Nós queremos obter a ordem que de vestir.



A ordem abaixo é uma ordem topológica possível:



Implemente um algoritmo de complexidade $O(V+E)$ que dado um grafo direcionado acíclico, devolve a ordenação topológica do grafo.

12. Implemente uma busca em profundidade (DFS) usando uma pilha (de forma a eliminar a recursão). O seu algoritmo deverá *devolver uma floresta de busca em profundidade* representada por um vetor π e deve executar em tempo $O(V+E)$. Você pode utilizar as sub-rotinas de uma pilha como caixas-pretas: CriarPilha(Q), Topo(Q), Desempilhar(Q), Empilhar($Q,\ v$).