

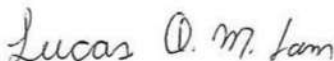
MAC0323 ALGORITMOS E ESTRUTURAS DE DADOS II

FOLHA DE SOLUÇÃO

Nome: Lucas Quaresma Medina Lam

Número USP: 11796399

Assinatura



Sua assinatura atesta a autenticidade e originalidade de seu trabalho e que você se compromete a seguir o código de ética da USP em suas atividades acadêmicas, incluindo esta atividade.

Exercício: 4

Data: 05/07/2021

SOLUÇÃO

Descrição de um algoritmo para encontrar uma ordenação de remoção de vértices de um grafo conexo da forma que os grafos resultantes continuem sendo conexos:

Para encontrar essa ordenação, é usada a busca em profundidade 'dfs' e pilhas.. Como começamos com um grafo já conexo podemos pegar qualquer vértice e fazer nossa busca em profundidade. Utilizando o algoritmo 'dfs' mostrado em aula, digamos que utilizamos um dado grafo G , começamos a busca em um qualquer vértice x_i , e o empilhamos. A busca passa pelo grafo encontrando uma direção para o mesmo. Ao passar pelos vértices, se ele não estiver marcado, é empilhado. Ao final da busca, desempilhamos os vertices x_i que estão na pilha, e iremos ter encontrado a ordenação buscada.

Vemos que o algoritmo descrito está correto, pois começamos a busca em x_v-1 marcamos e o empilhamos, e vamos marcando e empilhando os proximos vértices, $x_{v-2}, x_{v-3}, \dots, x_2, x_1$. Ao fazer a busca em profundidade para um vértice, sabemos que estamos criando caminhos de subgrafos conexos (visto em aula), sendo eles $[x_{v-1}, x_{v-2}], [x_{v-1}, x_{v-2}, x_{v-3}], \dots, [x_{v-1}, x_{v-2}, x_{v-3}, \dots, x_2, x_1]$. Ao retirarmos o último elemento desse grafo, o subgrafo resultante ainda seria um subgrafo conexo, e ao repetirmos 'v' vezes, chegamos ao resultado pedido. Para sair na ordem de retirada dos vértices, os colocamos em uma pilha, e após toda busca, o desempilhamos, o que daria a ordem de retirada do ultimo elemento adicionado ao subgrafo, até o primeiro.

Sabemos que o algoritmo descrito tem eficiência $O(V+E)$, pois utilizamos como base o algoritmo 'dfs' (que como visto em aula, tem desempenho $O(V+E)$), com apenas algumas alterações. As alterações feitas no algoritmo foram adicionar elementos em uma em uma pilha (tempo constante a cada adição), e desempilhar esses elementos ao final, que sabemos também que leva tempo proporcional ao tamanho da pilha, ou seja, o número de vertices. Vemos então que no final, o algoritmo consegue ter desempenho $O(V+E)$ também.