Estruturas de Dados II

G6

Grafos

1 – Escreva um algoritmo e um programa em C para determinar se um grafo orientado é um dag.

2 – Escreva um programa recursivo em C para imprimir os nós de um dag em ordem topológica reversa.

3 – Escreva um programa não-recursivo em C para imprimir os nós de um dag em ordem topológica reversa.

4 – Um nó *nd* num grafo conexo é um *ponto de articulação* se a remoção de *nd* e de todos os arcos adjacentes a *nd* resultar num grafo não-conexo. Sendo assim, a "conexidade" do grafo depende de *nd.* Um grafo sem pontos de articulação é chamado *biconexo.*

a. Demonstre que a raiz da árvore geradora em profundidade de um grafo biconexo tem somente um filho.

b. Demonstre que, se *nd* não for a raiz de uma árvore geradora em profundidade *t, nd* será um ponto de articulação se apenas *t* não contiver uma aresta para trás a partir de um descendente de *nd* até um ancestral de *nd.*

e. Modifique o algoritmo recursivo de percurso em profundidade de modo a determinar se um grafo conexo é biconexo.

5 – Escreva uma rotina em C para criar uma floresta geradora em largura de um grafo.

6 – Escreva rotinas em C que usem um percurso em largura para determinar se um grafo orientado e um grafo não-orientado são cíclicos.

7 – Escreva uma rotina em C para produzir o menor caminho a partir do nó *x* até o nó *y* num grafo não-ponderado, se existir um caminho, ou uma indicação de que não existe um caminho entre os dois nós.

8 – Demonstre que os algoritmos para encontrar um ciclo usando a busca em profundidade ou a busca em largura devem ser O(n).

9 – Implemente o algoritmo de Prim usando uma matriz de adjacência e utilizando listas de adjacência e uma fila de prioridade.

10 – Implemente **o** algoritmo **de** Kruskal **como uma** rotina **em C.**