UFMS – Departamento de Computação e Estatística
Bacharelado em Ciência da Computação

Análise de Algoritmos

Prof. Marco Aurélio

Lista 3 – Entrega: até 31/10 às 19h na secretaria do DCT

1. Mostre que em uma árvore rubro-negra o caminho mais longo de um nó qualquer x a uma folha descendente tem comprimento no máximo duas vezes o comprimento do caminho mais curto de x a uma folha descendente

2. Seja $R(i, j)$ o número de vezes que a entrada $m[i, j]$ é acessada pelo algoritmo Matriz_chain_order para calcular as outras entradas. Mostre que o número total de acessos de m é:
$$\sum_{i=1}^{n}\sum_{j=1}^{n} = \frac{n^3 - n}{3}$$

3. a) Mostre como calcular o comprimento de uma Subsequência Comum Máxima usando apenas $2\min\{m, n\}$ entradas da matriz c mais $O(1)$ espaço adicional de memória. b) Mostre também como fazer isto usando $\min\{m, n\}$ entradas da matriz c mais $O(1)$ espaço adicional de memória.

4. Seja $S = \{a, b, c, d, e, f, g\}$ um conjunto de objetos com valores-pesos como segue: $a(12, 4)$, $b(10, 6)$, $c(8, 5)$, $d(11, 7)$, $e(14, 3)$, $f(7, 1)$, $g(9, 6)$. Sendo $W = 18$ a capacidade da mochila, a) qual a solução ótima para o problema da mochila fracionada para S? b) e para o problema da mochila 0-1? Mostre o desenvolvimento.

5. Mostre que durante a operações sobre conjuntos disjuntos, para qualqer x, $rank[x] \leq rank[p[x]]$, com desigualdade estrita se $x \neq p[x]$. O valor de $rank[x]$ inicia com 0 e aumenta áte que $x \neq p[x]$. A partir daí $rank[x]$ não muda.

6. Dê uma sequência de m operações *Make_Set, Union* e *Find_Set* que toma tempo $\Omega(m \log n)$ quando usamos apenas união por rank.

7. Mostre que, em uma busca em largura, o valor $d(u)$ atribuído ao nó u independe da ordem dos nós na lista de adjacência.

8. Reescreva o procedimento DFS, utilizando pilha para eliminar a recursão

9. Dado um digrafo acíclico, escreva uma algoritmo que rotule seus vértices com valores $r(v)$ entre $0..n - 1$ de forma que cada aresta vai do nó de menor rótulo para o nó de maior rótulo. Dica: isto equivale a encontrar uma ordem de execução de tarefas, onde cada tarefa pode possuir outras tarefas como pré-requisito.