MAC 5711 - Análise de Algoritmos

Departamento de Ciência da Computação

Primeiro semestre de 2004

Lista 5

Devolução dos exercícios 11 e 12: 18 de maio.

1. Os números de Fibonacci podem ser definidos pela seguinte recorrência

$$F_n = \begin{cases} 1 & \text{se } n = 1, 2, \\ F_{n-1} + F_{n-2} & \text{se } n > 2. \end{cases}$$

   Escreva uma função recursiva para calcular F_n. Demonstre que o tempo de execução de tal função é exponencial em n. Explique como você poderia aplicar programação dinâmica para obter um algoritmo mais eficiente. Como é o algoritmo resultante? Quanto tempo ele consome em função do n?

2. Quantas árvores de busca binária ótima existem que guardam 6 diferentes chaves?

3. Encontre uma instância do problema da árvore de busca binária ótima o mais simples que você puder cuja solução não tenha o elemento de maior probabilidade na raiz. Dica: há uma instância assim em que S tem apenas 4 elementos e os q_i's são todos nulos.

4. Construa uma árvore de busca binária ótima para os seguintes itens, onde a probabilidade de cada palavra é dada entre parêntesis: `case` (.15), `else` (.05), `end` (.05), `if` (.35), `of` (.05) e `then` (.35). Considere $q_i = 0$ para todo i.

5. **Exercício 15.5-2 do CLRS** Determine o custo e uma árvore de busca binária ótima num conjunto com 7 elementos com as seguintes probabilidades:

i	0	1	2	3	4	5	6	7
p_i		0,04	0,06	0,08	0,02	0,10	0,12	0,14
q_i	0,06	0,06	0,06	0,06	0,05	0,05	0,05	0,05

6. Considere a recorrência dada em aula para a determinação de uma árvore de busca binária ótima. No algoritmo dado em aula que calcula a solução da recorrência, usamos uma matriz para guardar os valores de $w(i, j)$, ou seja, gastamos espaço $\Theta(n^2)$ para guardar todos estes valores. Encontre um jeito mais eficiente para calcular $w(i, j)$, que consuma espaço $O(n)$.

7. Na aula, mostramos duas chamadas que se repetem na árvore de recursão gerada a partir da recorrência para calcular $c(1, n)$. Deduza disso uma recorrência que dê uma delimitação inferior exponencial para o tempo do algoritmo recursivo simples (sem memoização).

8. Escreva uma versão recursiva com memoização do algoritmo dado em aula para a determinação do custo de uma árvore de busca binária ótima. Quanto tempo consome essa versão do algoritmo consome para calcular $c(1, n)$?

9. Escreva uma versão do algoritmo dado em aula para a determinação do custo de uma árvore de busca binária ótima que consuma espaço $O(n)$. Você é capaz de calcular também uma árvore ótima gastando espaço $O(n)$ (sem piorar a complexidade de tempo do algoritmo)?

10. Há 7 problemas no capítulo 15 do CLRL que podem ser resolvidos por programação dinâmica. Resolva todos os que puder destes. Vou escolher pelo menos um deles para vocês entregarem.

11. Escreva uma função recebe como parâmetros um inteiro $n > 0$ e um vetor que armazena uma seqüência de n inteiros e devolve o comprimento de uma subseqüência crescente da seqüência dada de soma máxima. Sua função deve consumir tempo $O(n^2)$. Modifique a sua função (sem piorar a sua complexidade assintótica) para que ela devolva também uma subseqüência crescente da seqüência dada de soma máxima.

12. Escreva um programa para o problema 111101 *Is Bigger Smarter* do *programming challenges* que consuma tempo $O(n^2)$, onde n é o número de elefantes. (Não precisa submeter ao PC, até porque o teste deles parece não estar funcionando.)