MAC 338 - Análise de Algoritmos

Departamento de Ciência da Computação
Primeiro semestre de 2005

Lista 5

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

2. 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.

3. 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).

4. Pode-se generalizar o problema da árvore de busca binária ótima para que inclua na sua descrição estimativa de buscas mal-sucedidas. Mais concretamente, nessa variante do problema, seriam dados não apenas os números de acessos a_i de cada chave i, mas também números b_i, para $i = 0, \ldots, n$, representando uma estimativa do número de buscas mal-sucedidas a elementos entre i e $i + 1$ (considere 0 como $-\infty$ e $n - 1$ como $+\infty$, para que o b_0 e o b_n sejam o que se espera).

   Encontre a recorrência para o custo de uma árvore de busca binária ótima para esta variante do problema. Escreva o algoritmo de programação dinâmica gerado a partir dessa recorrência. Quanto tempo consome o seu algoritmo?

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
a_i		4	6	8	2	10	12	14
b_i	6	6	6	6	5	5	5	5

6. Verifique que na árvore de expansão da recorrência para $c(i, j)$ existem duas chamadas que se repetem. Deduza disso uma recorrência que dê uma delimitação inferior exponencial para o tempo do algoritmo recursivo óbvio para calcular $c(i, j)$.

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

8. Escreva uma versão do algoritmo descrito 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)?

9. 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.