MAC 338 - Análise de Algoritmos

Departamento de Ciência da Computação

Primeiro semestre de 2011

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 com maior número de acesso 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 o número de acessos de cada palavra é dado entre parêntesis: `case` (15), `else` (5), `end` (5), `if` (35), `of` (5) e `then` (35).

4. Pode-se generalizar o problema da árvore de busca binária ótima para que inclua na sua descrição uma estimativa do número 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 estimativas de acessos de buscas bem e mal sucedidas:

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. 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 $c(1, n)$?

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

8. Escreva uma função que 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 $\mathrm{O}(n^2)$. Modifique a sua função (sem piorar a sua complexidade assintótica) para que ela devolva também uma subseqüência crescente da seqüência dada de soma máxima.

9. **Problema 15-2 do CLRS** (Como imprimir nitidamente) Considere o problema de imprimir nitidamente um parágrafo em uma impressora. O texto de entrada é uma seqüência de n palavras de comprimentos $l_1, l_2, \ldots, l_n$, medidos pelo número de caracteres. Queremos imprimir esse parágrafo com nitidez em uma série de linhas que contêm no máximo M caracteres cada uma. Nosso critério de "nitidez" é dado a seguir. Se uma determinada linha contém palavras de i até j, onde $i \leq j$, e deixamos exatamente um espaço entre as palavras, o número de espaços extras no final da linha é $M - j + i - \sum_{k=i}^{j} l_k$, que deve ser não-negativo para que as palavras caibam na linha. Desejamos minimizar a soma, sobre todas as linhas exceto a última, do cubo do número de espaços extras no final das linhas. Escreva um algoritmo de programação dinâmica para imprimir um parágrafo de n palavras nitidamente em uma impressora. Analise o tempo de execução e os requisitos de espaço do seu algoritmo.

10. **Problema 15-4 do CLRS** (Planejando uma festa da empresa) O professor Stewart presta consultoria ao presidente de uma corporação que está planejando uma festa da empresa. A empresa tem uma estrutura hierárquica; isto é, a relação de supervisores forma uma árvore com raiz no presidente. O pessoal do escritório classificou cada funcionário com uma avaliação de sociabilidade, que é um número real. Para tornar a festa divertida para todos os participantes, o presidente não deseja que um funcionário e seu supervisor imediato participem.

    O professor Stewart recebe a árvore que descreve a estrutura da corporação, usando a representação de filho da esquerda, irmão da direita, usada para o armazenamento de árvores enraizadas (olhe na seção 10.4 se precisar). Cada nó da árvore contém, além dos ponteiros, o nome de um funcionário e a ordem de sociabilidade desse funcionário. Escreva um algoritmo para compor uma lista de convidados que maximize a soma das avaliações de sociabilidade dos convidados. Analise o tempo de execução do seu algoritmo.

11. **PC 111105** (Cortes de tora) Você deve cortar uma tora de madeira em vários pedaços. A empresa mais em conta para fazer isso é a *Analog Cutting Machinery (ACM)*, que cobra de acordo com o comprimento da tora a ser cortada. A máquina de corte deles permite que apenas um corte seja feito por vez.

    Se queremos fazer vários cortes, é fácil ver que ordens diferentes destes cortes levam a preços diferentes. Por exemplo, considere uma tora com 10 metros de comprimento, que tem que ser cortada a 2, 4 e 7 metros de uma de suas extremidades. Há várias possibilidades. Podemos primeiramente fazer o corte dos 2 metros, depois dos 4 e depois dos 7. Tal ordem custa $10+8+6 = 24$, porque a primeira tora tinha comprimento 10, o que restou tinha 8 metros de comprimento e o último pedaço tinha comprimento 6. Se cortássemos na ordem 4, depois 2, depois 7, pagaríamos $10 + 4 + 6 = 20$, que é mais barato.

    Seu chefe encomendou um programa que, dado o comprimento l da tora e k pontos $p_1, \ldots, p_k$ de corte da tora, encontre o custo mínimo para executar esses cortes na ACM.

12. **UVA 10066** (Tôrres gêmeas) Havia em um império antigo duas torres de formatos diferentes, situadas cada uma em uma cidade diferente. As torres eram construídas de lajotas circulares postas uma sobre a outra. Cada uma das lajotas era da mesma altura e tinha um raio inteiro. Não era de se espantar no entanto, que apesar de terem formatos diferentes, as duas torres tinham muitas lajotas em comum.

    Mais de mil anos depois que elas foram construídas, o imperador ordenou que seus arquitetos removessem algumas das lajotas das duas torres de maneira que elas ficassem com a mesma forma e tamanho, e que ao mesmo tempo ficassem tão altas quanto possível. A ordem das lajotas nas novas torres deveria ser a mesma que nas torres originais. O imperador achou que, desta maneira, as torres seriam capazes de permanecer como símbolo da harmonia e igualdade entre as duas cidades. Ele decidiu chamá-las então de torres gêmeas.

    Agora, cerca de dois mil anos depois, desafiamos você a resolver um problema mais simples: dada a descrição das duas torres distintas, você deve encontrar o número de lajotas que haveria no mais alto par de torres gêmeas que pode ser construído delas.

13. **PC 111106** (Carregamento de balsa) Balsas são usadas para transportar carros para a outra margem de um rio ou outro trecho de água. Considere balsas que sejam largar o suficiente para acomodar duas faixas de carros em todo o seu comprimento. Os carros entram nas faixas por um lado da balsa e saem, na outra margem, do outro lado da balsa.

    A fila de carros para entrar na balsa é uma fila única e o operador direciona cada carro para uma das duas faixas da balsa — a faixa *esquerda* ou a faixa *direita* — de modo a balancear as duas faixas da balsa. Cada carro na fila tem um comprimento diferente, que é estimado pelo operador enquanto os carros estão na fila. Baseando-se nessas estimativas, o operador decide em qual das duas faixas cada carro deve embarcar, e embarca tantos carros da fila quanto possível. Escreva um programa que informe o operador para qual faixa ele deve direcionar cada carro de modo a maximizar o número de carros embarcados na balsa.