MAC 5711 - Análise de Algoritmos

Departamento de Ciência da Computação
Segundo semestre de 2005

Lista 7

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. Determine uma subseqüência comum de comprimento máximo das seqüências $\langle 1, 0, 0, 1, 0, 1, 0, 1\rangle$ e $\langle 0, 1, 0, 1, 1, 0, 1, 1, 0\rangle$.

3. Escreva a versão recursiva com memoização do algoritmo LCS-LENGTH visto em aula.

4. Escreva uma versão do algoritmo LCS-LENGTH que consuma espaço $O(\min\{m, n\})$ e que tenha a mesma complexidade de tempo que a versão vista em aula.

5. Escreva um algoritmo *recursivo* que recebe a tabela b gerada pelo algoritmo LCS-LENGTH e imprime uma subseqüência de comprimento máximo das seqüências $X[1\,..\,n]$ e $Y[1\,..\,m]$. Seu algoritmo deve consumir tempo $O(n + m)$.

6. Escreva um algoritmo *recursivo* que recebe a tabela b gerada pelo algoritmo LCS-LENGTH e imprime *todas* as subseqüências de comprimento máximo das seqüências $X[1\ldots n]$ e $Y[1\ldots m]$. Qual é a complexidade de tempo do seu algoritmo?

7. Escreva um algoritmo que produz uma subseqüência comum de comprimento máximo de $X[1\,..\,n]$ e $Y[1\,..\,m]$ a partir apenas da tabela c gerada pelo algoritmo LCS-LENGTH e das seqüências originais, sem usar a tabela b.

8. Dê um algoritmo $O(n^2)$ para encontrar uma subseqüência monotonicamente crescente de comprimento máximo de uma seqüência de n números.

9. Considere o problema da mochila, que tem como dados de entrada inteiros m e n e duas seqüências de inteiros positivos, $v_1, \ldots, v_n$ e $w_1, \ldots, w_n$, e busca um subconjunto S de $\{1, \ldots, n\}$ tal que $w(S) \leq m$ e $v(S)$ é máximo. Seja $\text{val}(i, t)$ o valor de uma solução ótima do problema que consiste apenas dos objetos $\{1, \ldots, i\}$ e cuja mochila tem capacidade t. Deduzimos na aula a seguinte recorrência para $\text{val}(i, t)$, para $i = 0, 1, \ldots, n$ e $t = 0, 1, \ldots, m$:

$$\text{val}(i, t) = \begin{cases} 0 & \text{se } i = 0 \text{ ou } t = 0 \\ \text{val}(i-1, t) & \text{se } w_i > t \\ \max\{\text{val}(i-1, t), v_i + \text{val}(i-1, t-w_i)\} & \text{se } w_i \leq t \end{cases}$$

   Escreva o algoritmo de programação dinâmica derivado da recorrência acima. Analise a sua complexidade de tempo. O algoritmo obtido é polinomial no tamanho da entrada do problema?

10. Escreva a versão recursiva memoizada do algoritmo do exercício anterior. Qual é a complexidade do algoritmo resultante?

11. Escreva um algoritmo que, da tabela val, obtem um subconjunto que seja uma solução ótima do problema da mochila. Qual é a complexidade do seu algoritmo?

12. Modifique primeiro o algoritmo do exercício 9 para que ele guarde mais informação útil para se obter um subconjunto que seja uma solução ótima do problema da mochila. Depois refaça o exercício 11 agora usando essa informação extra.