Universidade Federal do Ceará
Campus Russas

Construção e Análise de Algoritmos

Lista de exercícios 1

1. Uma pessoa sobe uma escada composta de n degraus, com passos que podem alcançar entre 1 e $k \leq n$ degraus. Escrever equações de recorrência que permitem determinar o número de modos distintos da pessoa subir a escada.

2. Prove as seguintes afirmações sobre notação assintótica:

- $n^3/100 - 25n^2 - 100n + 7$ é $\Omega(n^2)$ e $\Theta(n^3)$
- $77n^3 - 13n^2 + 29n - 5$ é $O(n^4)$ e $\Omega(n^3)$
- $34n\ log_7 n^2 + 13n$ é $\Omega(n)$ e $O(n^2)$

3. Resolva as seguintes equações de recorrência segundo o método da árvore de recursão:

- $T(n) \ = \ 2 \cdot T(n-1) \ + \ 5^n$
- $T(n) \ = \ 2 \cdot T(n/3) \ + \ 1$
- $T(n) \ = \ 17 \cdot T(n/4) \ + \ n^2$
- $T(n) \ = \ 3 \cdot T(n/9) \ + \ \sqrt{n}$
- $T(n) \ = \ T(0.99 \cdot n) \ + \ 7$
- $T(n) \ = \ T(\sqrt{n}) \ + \ 1$
- $T(n) \ = \ T(\sqrt{n}) \ + \ \log_2 n$
- $T(n) \ = \ 2 \cdot T(n/5) \ + 3 \cdot T(n/6) + \ n$

4. Suponha que você está tentando escolher entre os três algoritmos abaixo. Qual o tempo de cada um em notação assintótica e qual você escolheria?

- Algoritmo A resolve o problema dividindo a entrada em cinco subproblemas com a metade do tamanho, resolve cada subproblema recursivamente e depois combina-os em tempo linear.
- Algoritmo B resolve o problema dividindo a entrada em dois subproblemas de tamaho $n-1$ (onde n é o tamanho da entrada), resolve cada subproblema recursivamente e depois combina-os em tempo constante.
- Algoritmo C resolve o problema dividindo a entrada em nove subproblemas com um terço do tamanho, resolve cada subproblema recursivamente e depois combina-os em tempo quadrático.

5. Faça dois algoritmos de divisão e conquista para multiplicação dos inteiros x e y com n dígitos cada (suponha que n é potência de 2, isto é, $n = 2^k$, onde k é natural). Os algoritmos devem ter os tempos $\Theta(n^2)$ e $\Theta(n^{\log_2 3})$.

6. Suponha que você tem k vetores ordenados de tamanho n e deseja combiná-los em um único vetor ordenado de tamanho kn.

(a) Uma ideia é usar o algoritmo Intercala, intercalando o primeiro e o segundo, depois intercalando o resultado com o terceiro, depois com o quarto e etc... Qual a complexidade desse procedimento em termos de k e n?

(b) Mostre uma solução mais eficiente usando divisão e conquista.

7. Uma subsequência contígua de uma sequência S é uma subsequência de elementos consecutivos de S. Por exemplo, se $S = (5\ 15\ {-30}\ 10\ {-5}\ 40\ 10)$, então $(15\ {-30}\ 10)$ é uma subsequência contígua de S, mas $(5\ 15\ 40)$ não é. Escreva um algoritmo de Divisão e Comquista $\Theta(n \log n)$ que tem como entrada uma sequência de números $(a_1, a_2, \ldots, a_n)$ e devolva a subsequência contígua cuja soma é máxima (uma subsequência de tamanho zero tem soma zero). No exemplo anterior, a resposta seria a subsequência $(10\ \ {-5}\ 40\ 10)$ cuja soma é 55.

8. Faça dois algoritmo de divisão e conquista para multiplicar duas matrizes quadradas (ou seja, o número de linhas é igual ao número de colunas), dividindo cada matriz em 4 submatrizes quadradas. Suponha que o número de linhas de uma matriz é potência de 2. Os algoritmos devem ter os tempos $\Theta(n^3)$ e $\Theta(n^{\log_2 7})$.

9. Altere os algoritmos Intercala e MergeSort para resolver o seguinte problema: dado um vetor com n números inteiros positivos e um outro número inteiro positivo x, determine se existem ou não dois elementos cuja soma é igual a x.

10. Elabore um algoritmo $O(n)$ de decomposição de um vetor S em três subvetores. Esse algoritmo recebe como entrada, além do vetor S, um valor *piv* pertencente a S, e os índices p e r, $1 \leq p \leq r$. O algoritmo deve rearrumar os elementos em $S[p \ldots r]$ e retornar dois índices q_1 e q_2 satisfazendo as seguintes propriedades:

(a) se $p \leq k \leq q_1$, então $S[k] < piv$;

(b) se $q_1 < k \leq q_2$, então $S[k] = piv$;

(c) se $q_2 < k \leq r$, então $S[k] > piv$.

11. Sejam $X[1 \ldots n]$ e $Y[1 \ldots n]$ dois vetores ordenados. Escreva um algoritmo $\Theta(log\ n)$ para encontrar a mediana de todos os $2n$ elementos nos vetores X e Y. Prove esta complexidade.

12. Suponha que você possui dois vetores ordenados de tamanhos m e n. Você deseja obter o k-ésimo menor elemento da união desses dois vetores. Mostre um algoritmo de ordem $\Theta(\log m + \log n)$ que faça isso.

13. Seja $X[1 \ldots n]$ um vetor de inteiros. Dados $i < j$ em $\{1, \ldots, n\}$, dizemos que (i, j) é uma inversão de X se $X[i] > X[j]$. Escreva um algoritmo $\Theta(n\ log\ n)$ que devolva o número de inversões em um vetor X.

14. Altere o algoritmo HeapSort para trabalhar com Heaps mínimos, ao invés de Heaps máximos. Argumente porque é melhor trabalhar com Heaps máximos ao invés de Heaps mínimos.

15. Prove usando invariantes de laço que o algoritmo HeapSort e seu algoritmo da questão anterior estão corretos.