Universidade Federal do Ceará
Centro de Ciências
Departamento de Computação
Bacharelado em Ciência da Computação

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

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

- $T(n) = 2 \cdot T(n-2) + 1$
- $T(n) = T(0.9 \cdot n) + 7$
- $T(n) = 3 \cdot T(n/2) + n^2$
- $T(n) = 4 \cdot T(n/2) + n^2$
- $T(n) = 5 \cdot T(n/2) + n^2$
- $T(n) = T(\sqrt{n}) + \log_2 n$
- $T(n) = 2 \cdot T(\sqrt{n}) + \log_2 n$
- $T(n) = 2 \cdot T(n/5) + 3 \cdot T(n/6) + n$

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. Seja $X[1 \ldots n]$ um vetor qualquer (os elementos desse vetor não são necessariamente inteiros ou caracteres; podem ser objetos quaisquer, como frutas ou arquivos executáveis). Suponha que você possui apenas um operador "=" que permite comparar se um objeto é igual a outro. Dizemos que X tem um elemento **majoritário** x se mais da metade de seus elementos são iguais a x. Escreva um algoritmo de tempo $\Theta(n \log n)$ que diz se X possui ou não um elemento majoritário. Caso sim, devolva o seu valor. **Dica:** Se x é majoritário em X, então x é majoritário na primeira ou na segunda metade de X (explique porquê).

6. 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.

7. Faça um algoritmo de tempo $\Theta(n \log n)$ 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.

8. 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. **Dica:** Tenta fazer essa contagem enquanto ordena o vetor no Merge-Sort.

9. Altere o algoritmo HeapSort para trabalhar com Heaps mínimos (ao invés de Heaps máximos) e considerando que a raiz está na última posição do vetor (ao invés da primeira posição).

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. Faça um 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 9 submatrizes quadradas. Calcule a complexidade de tempo em notação assintótica.

12. Seja $X[1 \ldots n]$ um vetor de **números reais**. Dizemos que X tem um elemento **popular** x se mais de **um terço** de seus elementos são iguais a x. Escreva um algoritmo de tempo linear $\Theta(n)$ que diz se X possui ou não um elemento popular. Caso sim, devolva o seu valor. **Dica:** Use o algoritmo de Seleção do k-ésimo mínimo de tempo linear no pior caso.

13. Dizemos que um algoritmo é de **quase-ordenação** se, para qualquer vetor $A[1 \ldots n]$, o algoritmo rearranja os valores do vetor A de modo que $i < j$ implica $A[i] < A[j] + 0.1$. Por exemplo, o vetor $A = [1.5\ 1.45\ 2.4\ 2.35\ 3]$ está quase-ordenado. Sabendo que os valores do vetor de entrada são números reais menores que 100, faça um algoritmo de tempo $O(n)$ para quase-ordenar o vetor.