MCTA003-17 -- Análise de Algoritmos

Prof. Ronaldo C. Prati

lista compilada pela Profa. Carla Negri Lintzmayer

Lista 2

1. A ordenação por inserção pode ser expressa sob a forma de um procedimento recursivo como a seguir. Para ordenar $A[1..n]$, ordenamos recursivamente $A[1..n-1]$ e depois inserimos $A[n]$ no arranjo ordenado $A[1..n-1]$. Escreva o pseudocódigo desse algoritmo, uma recorrência para seu tempo de execução e resolva a recorrência pelo método de substituição.
2. Suponha que $T(1) = c$, em que c é uma constante positiva (você pode assumir que $c = 1$ se preferir).
   Resolva as seguintes recorrências da forma mais justa possível com notação O:
   a. $T(n) = T(\frac{n}{3}) + n$ (método de iteração)
   b. $T(n) = 2T(n-1) + n$ (método de iteração)
   c. $T(n) = 4T(\frac{n}{2}) + n$ (método de substituição, suponha $T(n) = \Theta(n^2)$)
   d. $T(n) = T(n-1) + T(n-2) + 3$ (método de substituição, suponha $T(n) = O(2^n)$)
   e. $T(n) = 4T(\frac{n}{2}) + \sqrt{n}$ (árvore de recursão e método de substituição)
   f. $T(n) = 7T(\frac{n}{3}) + n^2$ (árvore de recursão e Teorema Mestre)
   g. $T(n) = 2T(n/4) + 1$ (Teorema Mestre)
   h. $T(n) = 2T(n/4) + \sqrt{n}$ (Teorema Mestre)
   i. $T(n) = 2T(n/4) + n$ (Teorema Mestre)
   j. $T(n) = 2T(n/4) + n^2$ (Teorema Mestre)
3. Qual é a diferença entre Heap e Heapsort?
4. Mostre como implementar uma fila usando heap. Isto é, implemente funções `Enfileira` e `Desenfileira` considerando funções já existentes para heap. Analise os tempos de execução dos seus algoritmos.
5. Mostre como implementar uma fila de prioridades (isto é, funções `RemoveDaHeap(H)`, `InsereNaHeap(H, x)`, `AlteraHeap(H, i, k)` e `ConstroiHeap(H)`) usando um vetor ordenado. Analise os tempos de execução dos seus algoritmos.
6. Escreva uma função que recebe um vetor com n letras As e Bs e, por meio de trocas, move todos os As para o início do vetor.
   Sua função deve consumir tempo $O(n)$.
7. Simule passo a passo a execução do algoritmo `Heapsort` no vetor $A = (6, 1, 8, 7, 3, 9, 5, 2, 4)$.

8. Execute o algoritmo de partição sobre o vetor $A = (13, 19, 9, 5, 12, 8, 7, 4, 21, 2, 6, 11)$. Escolha o elemento 8 como pivô.
9. Seu amigo Morty Smith afirma que criou um algoritmo de tempo $\Omega(n)$ no pior caso que resolve o problema de ordenação. Em seguida ele disse que errou a análise e que na verdade o algoritmo leva tempo $O(n)$ no pior caso. Ele está certo em alguma das situações? Justifique.
10. Suponha que $T(1) = c$, onde c é uma constante positiva (você pode assumir que $c = 1$ se preferir). Resolva as seguintes recorrências da forma mais justa possível com notação O (use o método que lhe for mais conveniente):
    a. $T(n) = T(\sqrt{n}) + 1$ (faça uma mudança de variáveis, usando $m = \log n$, e escreva uma recorrência equivalente)
    b. $T(n) = 64T(\frac{n}{8}) + 7n^3$
    c. $T(n) = T(\frac{n}{2}) + T(\frac{n}{4}) + n$
    d. $T(n) = T(n - a) + T(a) + n$ com $a \geq 1$
    e. $T(n) = T(n/2) + T(n/4) + T(n/8) + n$
11. Mostre que qualquer árvore binária tem altura $\Omega(\log n)$.
12. Mostre por indução que em qualquer árvore binária o número de nós com dois filhos é exatamente um a menos do que o número de folhas.
13. Prove que o tempo de execução do algoritmo `ConstroiHeap` é $O(n)$.
14. Prove que o algoritmo `Heapsort` está correto, isto é, que ele corretamente ordena qualquer vetor dado na entrada.
15. É fácil provar que $T(n) = 2T(\frac{n}{2}) + \Theta(n)$ é $\Theta(n \log n)$ sempre que n é potência de 2, por exemplo usando árvore de recursão. Suponha agora que $n \geq 3$ não é potência de 2. Prove que, ainda assim, $T(n) = \Theta(n \log n)$. *Dica:* se n não é potência de 2, então existe um inteiro $k \geq 2$ tal que $2^{k-1} < n < 2^k$.