MCTA003-17 -- Análise de Algoritmos

Prof. Ronaldo C. Prati

lista compilada pela Profa. Carla Negri Lintzmayer

Lista 1

1. Seja F_n o n-ésimo número da sequência de Fibonacci. Temos $F_1 = 1$, $F_2 = 1$ e $F_n = F_{n-1} + F_{n-2}$ para $n \geq 3$. Use indução e prove que $F_n \geq 2^{0.5n}$ para todo $n \geq 6$.
2. Defina as notações O, Ω e Θ.
3. Ordene a lista de funções a seguir por ordem crescente de taxa de crescimento.
   Isto é, se a função $g(n)$ imediatamente segue a função $T(n)$ na sua lista, então deve valer que $T(n)$ é $O(g(n))$.

   $T_1(n) = n^{2.5}$, $T_2(n) = \sqrt{2n}$, $T_3(n) = n + 10$, $T_4(n) = 10^n$, $T_5(n) = 100^n$, $T_6(n) = n^2 \log n$

4. Podemos afirmar que o tempo de execução do *InsertionSort* é $\Theta(n^2)$? Justifique.
5. Em cada situação a seguir, prove se $T(n) = O(g(n))$ ou $T(n) \neq O(g(n))$, e se $T(n) = \Omega(g(n))$ ou $T(n) \neq \Omega(g(n))$. Comente quando $T(n) = \Theta(g(n))$.
   Considere que a e b são constantes positivas e que $\log$ está na base 2:
   a. $T(n) = n^{1/2}$ e $g(n) = n^{2/3}$
   b. $T(n) = \frac{n}{1000}$ e $g(n) = 50^{100}$
   c. $T(n) = \log_a n$ e $g(n) = \log_b n$
   d. $T(n) = 100^{n+a}$ e $g(n) = 100^n$
   e. $T(n) = 100^{an}$ e $g(n) = 100^n$
   f. $T(n) = n^{1.01}$ e $g(n) = n \log^2 n$
   g. $T(n) = \log \sqrt{n}$ e $g(n) = \log(100n)$
   h. $T(n) = n!$ e $g(n) = n \log n$
6. Sejam $f(n)$ e $g(n)$ funções assintoticamente não negativas. Usando a definição da notação Θ, prove que $\max\{f(n), g(n)\} = \Theta(f(n) + g(n))$.
7. Você provou que seu algoritmo tem tempo de execução $\Omega(n^2)$ no pior caso. O que isso diz sobre o tempo de execução do algoritmo sobre qualquer entrada? Você acredita que o tempo de execução no melhor caso é $\Omega(n^3)$. Isso é uma contradição com o resultado anterior?
8. Sempre é melhor utilizar o *MergeSort* ao invés do *InsertionSort*? Justifique.
9. Considere o problema da busca: dado um vetor A e um elemento x, se x estiver em A, devolva o índice i tal que $A[i] = x$; caso contrário, devolva -1. Um algoritmo de busca linear simplesmente percorre o vetor da posição 1 até a posição n em busca de x. Note que se A estiver ordenado, podemos comparar o elemento armazenado na posição média ($A[n/2]$) com x e eliminar metade do vetor do espaço de busca. A busca binária é um algoritmo que repete essa ideia, sempre dividindo ao meio o vetor restante a cada vez. Escreva um pseudocódigo para a busca binária (não recursivo) e

analise seu tempo de execução. Extra: prove que seu algoritmo está correto por meio de uma invariante de laço.

10. Seja $A[1..n]$ um vetor qualquer de inteiros de tamanho n e k um inteiro qualquer. Mostre como verificar se existem posições i e j tais que $A[i] + A[j] = k$ em tempo $O(n \log n)$. Justifique corretamente o tempo de execução do seu algoritmo.
11. Sejam $f(n)$ e $g(n)$ funções tais que $f(n)$ é $O(g(n))$. Para cada item a seguir, decida se é verdadeiro ou falso e dê uma prova ou contra-exemplo:
    a. $\log f(n)$ é $O(\log g(n))$
    b. $2^{f(n)}$ é $O(2^{g(n)})$
    c. $f(n)^2$ é $O(g(n)^2)$
12. Como podemos modificar praticamente qualquer algoritmo para ter um bom tempo de execução no melhor caso?
13. Explique por que a declaração "O tempo de execução do algoritmo *A* é no mínimo $O(n^2)$"" não faz sentido.
14. Em cada situação a seguir, prove se $T(n) = O(g(n))$ ou $T(n) \neq O(g(n))$, e se $T(n) = \Omega(g(n))$ ou $T(n) \neq \Omega(g(n))$. Comente quando $T(n) = \Theta(g(n))$.
    Considere que a e b são constantes positivas e que $\log$ está na base 2:
    1. $T(n) = 3^n$ e $g(n) = 2^n$
    2. $T(n) = (n + a)^b$ e $g(n) = \Theta(n^b)$ com $b > 0$
15. Considere o seguinte problema. Estamos observando o preço das ações de uma empresa por n dias consecutivos. Para cada dia $i = 1, 2, \ldots, n$, temos o preço $p(i)$ por ação naquele dia. Como escolher um dia i no qual comprar a ação e um dia posterior $j > i$ no qual vendê-la de forma a maximizar o lucro $p(j) - p(i)$? Descreva e analise um algoritmo para esse problema.