Universidade Federal Fluminense / Instituto de Computação
Disciplina: Análise e Projeto de Algoritmos (Graduação) **Prof.** Carlos Martinhon

Complexidade de Algoritmos – Lista 1

01 - Seja x um número real e $S=\{a_1,a_2,...,a_n\}$ um conjunto de n números reais.

(a) Faça um algoritmo que determine se a soma de *2* elementos de S é igual a x. Construa um algoritmo $O(n^2)$ justificando sua resposta.

(b) Resolver o mesmo problema através de um algorimto de complexidade $O(nlogn)$. (**Sugestão:** Ordene S e depois busque o elemento x-z (onde $z \in S$).

(c) Construa um algoritmo de complexidade $O(nlogn)$ que indique se S contém ou não elementos repetidos.

(d) Considere dois subconjuntos S_1 *e* S_2, e um número real x. Construa um algoritmo de ordem $O(nlogn)$ que verifique se a soma de *2* elementos de S_1 e S_2 respectivamente é igual a x. Considere $n=n_1+n_2$ é o tamanho total de ambos os conjuntos.

(e) Resolva o mesmo problema supondo agora que S já esteja ordenada. Construa um algoritmo de complexidade $O(n)$.

(f) Novamente, considere S ordenada. Suponha ainda que os elementos de S sejam distintos entre si. Construa um algoritmo de complexidade $O(logn)$ que retorne um índice i, caso exista, tal que $S[i]=i$.

02 - Dados dois vetores $X=(x_1,...,x_n)$, $Y=(y_1,...,y_n)$. Dizemos que X<Y se existe i (onde $1 \le i \le n$) tal que $x_j = y_j$, *para* $1 \le j < i$ *e* $x_i < y_i$. Dados m vetores (cada um de tamanho n), escreva um algoritmo que determine o menor vetor segundo a definição apresentada. Calcule sua complexidade.

03 - Considere uma lista A com n elementos. Construa um algoritmo de complexidade O($n$log$n$) que indique se A contém ou não elementos repetidos.

04 – Faça um algoritmo que converta um número escrito na base 10 para a base 2. Qual a complexidade desse algoritmo?

05 - Resolva as séries abaixo:

a) $\sum_{i=1}^{n} i$ **b)** $\sum_{i=1}^{n} a^i$ **c)** $\sum_{i=1}^{n} ia^i$ **d)** $\sum_{i=1}^{n} \binom{n}{i}$

06 - Calcule as seguintes recorrências (**Nota:** considere T(1)=1).

a) $T(n) = 2+T(n-1)$ **c)** $T(n) = 2T(n/2) + 2$

b) $T(n) = 2T(n/2) + cn$ **d)** $T(n) = 2T(n-1)+1$

07 – Resolva as seguinte recorrências (**Sugestão**: considere inicialmente n potência de 2 e depois n qualquer):

(a) $\begin{cases} T(1) = a, \\ T(n) = 8T(n/2) + bn^2, \quad p/\ n > 1 \end{cases}$ (b) $\begin{cases} T(2) = 1, \\ T(n) = 2T\left(n^{1/2}\right) + \log n, \quad p/\ n > 1 \end{cases}$

08 - Considere dois algoritmos A_1 e A_2 de complexidade local $T_1(n) = 100n^2$ e $T_2(n) = 2^n$ respectivamente. Qual o menor valor de n tal que $T_1(n) < T_2(n)$?

09 - Responda às seguintes questões:

a) $(n+1)! \in O(n!)$?
b) $\log(n) \in O(\sqrt{n})$?
c) $a^{\log b} = b^{\log a}$?
d) $2^{n+1} \in O(2^n)$?
e) $2^{2n} \in O(2^n)$?
f) $O(log\ n) = O(ln\ n)$?

10 - a) - Prove que:
Se $p(n) = a_k n^k + \ldots + a_1 n + a_0$ é um polinômio em n, então $O(\log(p(n)) = O(\log n)$.
b) - Mostre que $p(n)$ é $O(n^k)$. Podemos dizer que 2^n é de ordem $O(p(n))$?

11 - a) Elaborar um algoritmo para resolver o problema das Torres de Hanói com n discos e 4 pinos. O número de movimentos deve ser inferior ao Hanói de 3 pinos. Calcule a complexidade e compare com o algoritmo com o Hanói de 3 pinos.

b) Construa agora um algoritmo eficiente para o problema das Torres de Hanói com 3 pinos considerando-se agora $2n$ discos de n tamanhos distintos e de maneira que sempre tenhamos 2 discos de mesmo tamanho. Qual a complexidade deste algoritmo?

c) Elaborar um algoritmo (com o menor número de movimentos) para o problema das Torres de Hanói com n discos e 3 pinos satisfazendo a seguinte restrição adicional: todo movimento deve ser realizado de um pino lateral para o pino do centro ou do pino do centro para um pino lateral (movimentos entre pinos laterais são proibídos!). O objetivo é mover os n discos de um pino lateral para outro pino lateral. Calcule também sua complexidade.

12 – Qual o número de *bits* (tamanho do problema) necessários para se armazenar os dados do problema de programação linear? E o problema da mochila?

13 – (Busca Binária Recursiva) Seja S uma lista ordenada com n elementos inteiros e k um inteiro qualquer. Construa um algoritmo recursivo de complexidade $O(log\ n)$ que responda se k pertence ou não à lista S.

14 – Mostre que:

a) Se $\lim_{n\to\infty} \frac{f(n)}{g(n)} = k > 0$ então $O(f(n)) = O(g(n))$.

b) Se $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$ então $O(f(n)) \subset O(g(n))$.

c) Se $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$ então $f(n) \in \Omega(g(n))$.

d) Mostre que $g(n)$ é $q(f(n))$ se, e somente se, $g(n)$ é $q(f(n))$.

15 - Seja *A* uma lista de *n* elementos. Na ordenação por seleção (*Selection Sort*), tomamos o menor elemento de *A* e o colocamos na primeira posição de uma lista auxiliar *B*. Em seguida, tomamos o segundo menor elemento de *A* e o colocamos na segunda posição de *B*. Repetimos o processo até que a ordenação esteja completa. Escreva um pseudo-código e calcule a complexidade deste algoritmo.