Análise de algoritmos - Lista 1
Professor: Guilherme Oliveira Mota

Entrega: 23h55 de 25/6/2018 (**Somente pelo Tidia**)

(1) Defina precisamente as notações O, Ω e Θ.

(2) Para as funções $f(n)$ e $g(n)$ dada em cada um dos itens a seguir, prove que
$f(n) = O(g(n))$ ou $f(n) \neq O(g(n))$, e
$f(n) = \Omega(g(n))$ ou $f(n) \neq \Omega(g(n))$,
onde $a > 1$ e $b > 1$ são constantes, lembrando que $\log n$ significa $\log_2(n)$.

- $f(n) = \sqrt{n}$ e $g(n) = n^{2/3}$.
- $f(n) = n \log n$ e $g(n) = 20n \log(100n)$.
- $f(n) = n/1000$ e $g(n) = 50^{100}$.
- $f(n) = 2000 \log n$ e $g(n) = \log(n^3)$.
- $f(n) = 100^{n+a}$ e $g(n) = 100^n$.
- $f(n) = 99^{n+a}$ e $g(n) = 100^n$.
- $f(n) = 4^n$ e $g(n) = 3^n$.
- $f(n) = \log \sqrt{n}$ e $g(n) = \log(100n)$.
- $f(n) = n!$ e $g(n) = 4^{\log(100n)}$.

(3) Seja F_n o n-ésimo número da sequência de Fibonacci, i.e., $F_1 = 1$, $F_2 = 1$ e $F_n = F_{n-1} + F_{n-2}$ para $n \geq 3$. Use indução para provar que $F_n \geq 2^{n/2}$ para todo $n \geq 6$.

(4) Prove que o algoritmo MERGE SORT funciona corretamente. Primeiro, prove que o algoritmo COMBINA funciona corretamente, utilizando uma invariante de laço para isso. Depois, utilize esse fato para provar por indução que o algoritmo MERGE SORT funciona corretamente.

(5) Prove que o algoritmo QUICKSORT funciona corretamente. Primeiro, prove que o algoritmo PARTIÇÃO funciona corretamente, utilizando uma invariante de laço para isso. Depois, utilize esse fato para provar por indução que o algoritmo QUICKSORT funciona corretamente.

(6) Modifique o algoritmo PARTIÇÃO para que funcione também em vetores que contém repetição de elementos (seu algoritmo deve ter tempo linear, i.e., $O(n)$). Feito isso, modifique o algoritmo QUICKSORT para que funcione com o novo PARTIÇÃO.

(7) No que segue assuma que $T(1) = 1$. Utilize o método indicado para estimar $T(n)$ por cima (i.e., utilizando a notação O).

- $T(n) = 4T(n/2) + n$ (método iterativo).
- $T(n) = T(n/3) + n$ (método iterativo).
- $T(n) = aT(n/a) + n$, onde a é um inteiro positivo (método iterativo).
- $T(n) = 4T(n/2) + \sqrt{n}$ (método da substituição. Palpite: $O(n^2)$).
- $T(n) = T(n-1) + T(n-2) + 1$ (método da substituição. Palpite: $O(2^n)$).
- $T(n) = 7T(n/3) + n^2$ (método da árvore e Teorema Mestre).
- $T(n) = 16T(n/4) + 17n^{1,99}$ (Teorema Mestre).
- $T(n) = T(n/4) + 1$ (Teorema Mestre).

(8) O algoritmo de *Busca Binária* encontra um elemento x em um vetor A com n elementos que está ordenado em ordem **não-decrescente**. Esse algoritmo verifica o elemento central de A (i.e., $A[\lfloor n/2 \rfloor]$) e retorna $\lfloor n/2 \rfloor$ caso $A[\lfloor n/2 \rfloor] = x$. Se $A[\lfloor n/2 \rfloor] > x$, então o algoritmo faz a busca recursivamente no vetor $A[1..(\lfloor n/2 \rfloor - 1)]$, e se $A[\lfloor n/2 \rfloor] < x$, então o algoritmo faz a busca recursivamente no vetor

$A[(\lfloor n/2 \rfloor+1)..n]$. Caso o algoritmo não encontre x, retorna um aviso que x não foi encontrado juntamente com o último índice que foi consultado no vetor.

Escreva o pseudo-código para o algoritmo (recursivo) *Busca Binária* explicado acima e prove, usando o método iterativo, que seu tempo de execução é $\Theta(\log n)$ no pior caso.

(9) Seja $T(n) = 2T(n/2) + n$. Vimos que $T(n) = \Theta(n \log n)$, mas assumimos que n era potência de 2. Utilizando isso, mostre que para qualquer $n \geq 3$ (mesmo que n não seja potência de 2) ainda vale que $T(n) = \Theta(n \log n)$.
Dica: se n não é potência de 2, então existe um inteiro k tal que $2^{k-1} < n < 2^k$.

(10) Seja $A[1..n]$ um vetor de inteiros e k um inteiro qualquer. Explique como verificar se existem posições i e j tais que $A[i] + A[j] = k$ em tempo $O(n \log n)$.

(11) Considere o problema de ordenação em que a entrada pode conter elementos com mesmo valor. Um algoritmo de ordenação é dito *estável* se a ordem inicial entre quaisquer dois elementos com mesmo valor é mantida na ordenação final.

Diga quais dos algoritmos vistos em sala são estáveis (Insertion sort, Merge sort, Quicksort). Justifique sua resposta.