# MAC 5711 – Análise de Algoritmos

Segundo Semestre de 2002

Primeira Prova – 16 de setembro

1. [2 pontos]
   Se $T(0) = T(1) = 1$, cada uma das seguintes recorrências define uma função $T$ nos inteiros não-negativos.

   (a) $T(n) = 3T(\lfloor n/2 \rfloor) + n^2$

   (b) $T(n) = 2T(n-2) + 1$

   (c) $T(n) = T(n-1) + n^2$

   Quais delas não podem ser limitadas por uma função polinomial? Justifique a sua resposta.

2. [2 pontos]
   Considere um conjunto armazenado em uma lista ligada (ou seja, cada elemento do conjunto está armazenado em uma das células da lista ligada). Descreva um algoritmo que ordene o conjunto, ou seja, que rearranje as células da lista ligada de forma que os elementos do conjunto apareçam em ordem na lista ligada rearranjada. Seu algoritmo deve ter complexidade de pior caso $O(n \log n)$, onde $n$ é o número de elementos do conjunto, e deve consumir espaço extra $O(1)$. Não escreva código. Apenas descreva o algoritmo e explique porque a sua complexidade é $O(n \log n)$.

3. [2 pontos]
   Considere um conjunto $S$ com $n$ elementos distintos. A *mediana* de $S$, quando $n$ é ímpar, é o $\left(\frac{n+1}{2}\right)$-ésimo menor elemento de $S$. Quando $n$ é par, $S$ tem duas medianas: o $\frac{n}{2}$-ésimo menor elemento de $S$ e o $\left(\frac{n}{2}+1\right)$-ésimo menor elemento de $S$.

   (a) Descreva um algoritmo que determine uma mediana de um conjunto $S$ dado em um vetor $v[1\,..\,n]$. A complexidade de pior caso do seu algoritmo deve ser $O(n \log n)$. Não escreva código. Apenas descreva o algoritmo e explique porque sua complexidade é $O(n \log n)$.

   (b) Considere a variante da função `particiona` usada no algoritmo `quicksort` que computa uma mediana do vetor a ser particionado e usa essa mediana como pivô da partição.

   Escreva a recorrência para a complexidade de pior caso do `quicksort` para os dois seguintes casos: (1) o algoritmo usado para o cálculo da mediana no `particiona` tem complexidade $O(n \log n)$ e (2) o algoritmo usado para o cálculo da mediana no `particiona` tem complexidade $O(n)$.

   Resolva as duas recorrências e conclua qual é a complexidade de pior caso dessas duas variantes do `quicksort`. (Assuma que $n$ é uma potência de 2, para facilitar.) Compare a complexidade dessas variantes com a complexidade do `quicksort` original.

4. [2 pontos]
   Descreva informalmente (não escreva código) um algoritmo que consome $O(n)$ unidades de tempo para imprimir os $\lfloor \sqrt{n} \rfloor$ maiores elementos de um vetor $v[1\,..\,n]$. Explique porque o algoritmo funciona e a análise de tempo.

5. [2 pontos]
   Considere o *problema da busca*:

   > dado um vetor $v[1\,..\,n]$ e um número $x$, determinar, caso exista, um índice $i$ em $\{1, \ldots, n\}$ tal que $v[i] = x$.

   Dizemos que uma função resolve o problema da busca se ela recebe $v$ e $x$ como parâmetros e devolve 0, caso $x$ não apareça em $v$, e um $i$ em $\{1, \ldots, n\}$ tal que $x = v[i]$, caso $x$ apareça em $v$.

   Prove que toda função que implementa um algoritmo baseado em comparações e resolve o problema da busca tem complexidade $\Omega(\log n)$.

   **Sugestão:** Considere a árvore de decisão de uma função que implementa um algoritmo de busca baseado em comparações. Coloque como rótulo de cada folha o valor devolvido pela função em cada caso.