# Estruturas de dados básicas II

Lista de exercícios – Complexidade assintótica & ordenação

1. Explique e exemplifique a diferença entre dominância das folhas e dominância da raiz na análise de algoritmos recursivos. Exemplifique, também, um caso onde não aconteça nenhum desses dois tipos de dominância.

2. Suponha um vetor com $n$ inteiros. Descreva algoritmos para determinar se existem elementos duplicados cujas complexidades assintóticas de pior caso sejam (i) $O(n^2)$, (ii) $O(n \cdot log\, n)$ e (iii) $O(n)$.

3. Defina tipo abstrato de dados que ofereça duas operações básicas:

   (a) Adicionar/remover elementos numéricos inteiros.

   (b) Dado um inteiro $x$, descobrir se existem dois números $a$ e $b$ armazenados nesta estrutura tal que $a + b = x$.

   ✓ Dica – ao escolher uma estrutura de dados adequada, é importante analisar a frequência esperada de uso das operações implementadas.

4. O *merge sort* é um algoritmo cuja complexidade depende da altura da árvore de recursão (de ordem $log_k\, n$) e do passo de conquista, isto é, a mescla de $k$ vetores em tempo linear (tipicamente, $k = 2$). Por sua vez, o *insertion sort* apresenta complexidade de pior caso $O(n^2)$ para ordenar vetores.

   (a) Analise a complexidade do *merge sort* para $k > 2$.

   (b) Analise a complexidade do *insertion sort* caso o vetor original seja dividido em $k$ subvetores.

   (c) Analise a complexidade de um procedimento que combine as características do *merge sort* e do *insertion sort*. Mais precisamente, o procedimento deve dividir o vetor original em $k$ subvetores, aplicar o *insertion sort* a cada subvetor e mesclar estes $k$ subvetores ordenados assim como no *merge sort*.

   (d) Dada a complexidade do procedimento acima, qual o valor ótimo de $k$? Compare a complexidade assintótica deste algoritmo neste caso com a complexidade do *merge sort* e do *insertion sort*.

✓ Dica – tente resolver esta questão por conta própria, e em seguida compare sua resposta com a explicação encontrada na lista de exercícios abaixo: https://www.cs.auckland.ac.nz/courses/compsci220s1t/lectures/lecturenotes/GG-lectures/220exercises2.pdf

**Questões extras para quem quiser se aprofundar na área**

1. Explique o teorema central.
2. Demonstre que algoritmos de ordenação por comparação não podem apresentar complexidade assintótica de pior caso melhor do que $n \cdot log\, n$.
3. Considere o vetor abaixo:

   $$150 - 13 - 25 - 597 - 421 - 66 - 200$$

   Discuta a eficiência dos algoritmos de ordenação em tempo linear tradicionais (*counting sort, bucket sort e radix sort*), explicando qual seria a melhor opção.