MCTA003-17 - ANÁLISE DE ALGORITMOS
CENTRO DE MATEMÁTICA, COMPUTAÇÃO E COGNIÇÃO
UNIVERSIDADE FEDERAL DO ABC
PROF. MAYCON SAMBINELLI

**Ordenação**

**1.** Podemos afirmar que "*o tempo de execução do* INSERTIONSORT *é* $\Theta(n^2)$"? Justifique.

**2.** Sempre é melhor utilizar o MERGESORT ao invés do INSERTIONSORT? Justifique.

**3.** Simule passo a passo a execução dos algoritmos INSERTIONSORT, MERGESORT e HEAPSORT no vetor $A = (6, 1, 8, 7, 3, 9, 5, 2, 4)$.

**4.** Seja $A[1..n]$ um vetor qualquer de inteiros de tamanho $n$ e seja $k$ um inteiro qualquer. Resolva o problema de verificar se existem posições $i$ e $j$ tais que $A[i] + A[j] = k$ em tempo $O(n \log n)$. Justifique corretamente o funcionamento e o tempo de execução do seu algoritmo.

**5.** Escreva um algoritmo que recebe um vetor com $n$ letras $A$s e $B$s e, *por meio de trocas*, move todos os $A$s para o início do vetor. Sua solução deve levar tempo $O(n)$.

*Observação:* resolver esse problema por meio de trocas significa que você não pode apenas contar a quantidade de $A$s e escrever essa mesma quantidade no início do vetor.

**6.** Seu amigo Morty Smith afirma que criou um algoritmo baseado com comparação que resolve o problema da ordenação e tem tempo $\Omega(n)$ no pior caso. Em seguida ele disse que errou a análise e que na verdade o algoritmo leva tempo $O(n)$ no pior caso. Ele está certo em alguma das situações? Justifique.

**7.** Prove que o algoritmo HEAPSORT está correto, isto é, que ele corretamente ordena qualquer vetor dado na entrada. Para isso, use a frase $P(x) =$ "Antes da $x$-ésima iteração começar, vale que (a) $i = n - x + 1$, (b) o vetor $A[x + 1..n]$ está ordenado de modo não-decrescente e contém os $n - x$ maiores elementos de $A$; (b) $A.tamanho = x$ e o vetor $A[1..A.tamanho]$ é um heap.". Considere que as funções da estrutura heap estão corretas.

**8.** Mostre como implementar uma fila usando heap. Especificamente, implemente funções ENFILEIRA e DESENFILEIRA considerando que você já possui as funções implementadas para heap (REMOVEDAHEAP, INSERENAHEAP, ALTERAPRIORIDADE e CONSTROI). Você não deve mexer nas implementações das funções de heap (use-as como caixa preta). Analise os tempos de execução dos seus algoritmos.

**9.** Mostre como implementar uma fila de prioridades usando um vetor ordenado pelas prioridades dos elementos. Considere que um elemento $x$ tem prioridade dada por $x.prioridade$. Especificamente, apresente o pseudocódigo de uma função REMOVE($A$, $n$), que sempre retorna o elemento de maior prioridade, INSERE($A$, $x$, $n$), que insere um elemento $x$ novo, ALTERA($A$, $i$, $k$, $n$), que altera a prioridade do elemento que está na posição $i$ para o valor $k$, e CONSTROI($A$, $n$), que recebe um vetor $A$ qualquer e o inicializa para ser uma fila de prioridades (no caso, um vetor ordenado). O parâmetro $n$ sempre indica a quantidade de elementos armazenados na estrutura. Não há necessidade de formalizar invariantes de laço neste exercício.

**10.** Descreva um algoritmo que, dados inteiros $n$ e $k$, juntamente com $k$ vetores ordenados que em conjunto tenham $n$ elementos, produza um único vetor ordenado contendo todos os elementos dos vetores dados (isto é, faça uma intercalação). O seu algoritmo deve ter complexidade $O(n \log k)$. Note que isto se transforma em $O(n \log n)$ no caso de $n$ listas de 1 elemento, e em $O(n)$ se só houver duas listas (no total com $n$ elementos).

**11.** Descreva um algoritmo que, dados $n$ inteiros no intervalo de 1 a $k$, pré-processe a entrada de alguma forma para que depois disso possa-se responder em $O(1)$ qualquer consulta sobre quantos dos $n$ inteiros dados estão contidos em um intervalo $[a..b]$. O pré-processamento efetuado pelo seu algoritmo deve consumir tempo $O(n + k)$.

**12.** Escreva um algoritmo que ordena uma lista de $n$ itens dividindo-a em três sublistas de aproximadamente $n/3$ itens, ordenando cada sublista recursivamente e intercalando as três sublistas ordenadas. Mostre que seu algoritmo está correto e analise seu tempo de execução.

**13.** O COMBINA usado pelo MERGESORT recebe um vetor $A$ e três inteiros $ini$, $meio$ e $fim$, considera que $A[ini..meio]$ e $A[meio + 1..fim]$ estão ordenados, e devolve $A[ini..fim]$ totalmente ordenado com os mesmos elementos contidos ali inicialmente.

Crie um algoritmo COMBINA++, que recebe um vetor $A$ e quatro inteiros $ini$, $umT$, $doisT$ e $fim$, considera que $A[ini..umT]$, $A[umT + 1..doisT]$ e $A[doisT + 1..fim]$ estão ordenados, e devolve $A[ini..fim]$ totalmente ordenado.

Prove que este algoritmo está correto adaptando a prova do COMBINA.