DCA0214.0 ESTRUTURAS DE DADOS - TEORIA: Lista de Exercícios sobre Métodos de Ordenação

Professor: Islame Felipe da Costa Fernandes

20 de Agosto de 2019

1. Qual a complexidade do algoritmo de ordenação por inserção se todas as chaves forem iguais?

2. Um algoritmo de ordenação é dito estável se ele preserva a ordem original dos elementos iguais. Por exemplo, seja o vetor $V = [4[a], 9[b], 2[c], 5[d], 2[e]]$ de chaves numéricas, onde as letras dentro de [ ] indicam apenas um registro para fins didáticos. Então, uma ordenação estável produziria: $V = [2[c], 2[e], 4[a], 5[d], 9[b]]$. Quais dos algoritmos de ordenação, vistos em aula são estáveis?

3. O algoritmo de ordenação por inserção pode ser escrito recursivamente do seguinte modo: para ordenar o vetor $V[1...n]$, ordena-se recursivamente $V[1...n-1]$ e então insere-se o elemento $V[n]$ no vetor ordenado $V[1...n-1]$. Escreva a equação de recorrência correspondente ao insertionSort recursivo e obtenha sua complexidade.

4. Escreva um algoritmo que recebe dois argumentos: (1) um vetor A com n inteiros, (2) um inteiro x. Seu algoritmo deve determinar se existem ou não dois elementos em A cuja soma é exatamente x. A complexidade do seu algoritmo deve ser $\Theta(nlogn)$ no pior caso.

5. Mostre que o QuickSort é $\Theta(n^2)$ quando o vetor V, contendo elementos distintos, está ordenado em ordem decrescente (ordem inversa).

6. Seja $A[1...n]$ um arranjo com n números distintos. Dizemos que o par (i, j) é uma inversão se, e somente se, $i < j$ e $A[i] > A[j]$. Responda:

   (a) Seja $C = \{1, 2, 3, ..., n\}$ um conjunto com n números. Qual arranjo dos elementos de C possui a maior quantidade de inversões? Quantas inversões ele tem?

   (b) Qual a relação entre o tempo de execução do Insertion Sort e a quantidade de inversões do arranjo de entrada? Seja d a quantidade de inversões do arranjo de entrada. Formule a complexidade do InsertionSort em função de d. Justifique.

(c) Explique como modificar o MergeSort a fim de obter um algoritmo que calcula a quantidade de inversões em um vetor $A[1...n]$.

7. Crie um algoritmo chamado **quickfind** baseado no quicksort para que, em vez de ordenar um vetor de números inteiros, ele nos retorne o k-ésimo menor elemento desse vetor. O procedimento **quickfind** deve ter a seguinte interface: **quickfind**(V, p, r, k), onde V é um vetor de inteiros, p é o menor índice de V, r o maior índice, e $1 \leq k \leq n$ um inteiro positivo. Por exemplo: para $V = [7, 1, 3, 10, 17, 2, 21, 9]$, a chamada **quickfind**$(V, 1, 8, 5)$ deverá retornar o número 9 (quinto menor elemento). Escreva o procedimento **quickfind** recursivamente (pseudo-código), modificando o procedimento quicksort visto em aula. Obs.: Você não deve simplesmente ordenar todo o vetor e depois tomar o k-ésimo elemento.

8. Utilizando invariante de laço, prove formalmente a corretude dos algoritmos BubbleSort e SelectionSort (se basear nos pseudos-códigos vistos em aula).

9. Seja um vetor A com n inteiros, onde cada inteiro está entre 0 e k. Sejam ainda os parâmetros a e b entre 0 e k. Escreva um algoritmo que determine, em $O(n + k)$, a quantidade de inteiros de A que estão entre a e b.

10. Suponha que modificássemos a linha 10 do algoritmo CountingSort (ver psudo-código dos slides) de tal modo que o j fosse incrementado de 1 até $A.length$. O CountingSort continuaria correto? Continuaria estável?