Algoritmos e Estruturas de Dados II
Quarta Lista de Exercícios – Métodos de Ordenação

Métodos de Ordenação

1. Escreva um algoritmo que receba um vetor ordenado e um número extra e insira esse número na sua posição correta no vetor ordenado, deslocando os outros números se necessário.

2. Implemente o algoritmo de ordenação por inserção visto em aula e conte o número total de cópias de valores do vetor dentro do while ao executar no seguinte array: 72 12 62 69 27 67 41 56 33 74.

3. Faça um programa que leia n nomes inserindo-os em uma lista de forma ordenada utilizando a ideia do algoritmo InsertionSort. No final, o programa deve mostrar todos os nomes ordenados alfabeticamente.

4. Crie um programa que dado uma string, coloque as letras dela em ordem crescente pelo algoritmo BubbleSort.

5. Crie um programa que dado uma string, coloque as letras dela em ordem decrescente usando o algoritmo QuickSort.

6. Faça um programa que leia n nomes e ordene-os pelo tamanho utilizando o algoritmo SelectionSort.

7. Considere a seguinte estrutura:

```
typedef struct pessoa{
        int matricula;
        char nome[30];
        float nota;
}pessoa;
```

Faça uma função que dado um array de tamanho N dessa estrutura, ordene o array pelo campo escolhido pelo usuário. Utilize os seguintes algoritmos de ordenação:

a) BubbleSort;
b) SelectionSort;
c) InsertionSort;
d) MergeSort;
e) QuickSort;

Compare o tempo gasto para a ordenação, dentre os métodos, de modo a avaliar quais os algoritmos mais eficientes.

8. Um amigo lhe diz que é capaz de ordenar qualquer conjunto de 6 números com no máximo 8 comparações. O seu amigo está falando a verdade ou mentindo? Justifique.

9. Modifique a função Particiona do método QuickSort de modo que o pivô seja escolhido usando as seguintes regras:

a) Mediana de três elementos aleatórios do arranjo;
b)
b) Elemento aleatório do arranjo;

Faça a mensuração de tempo nas 3 funções Particiona, a original (mostrada em sala de aula) e as duas propostas aqui, e verifique qual delas é mais eficiente. Teste para diversos, tamanhos de arranjos, lembrem-se de usar os mesmos arranjos nos testes.

10. Faça uma função de ordenação mista, que use o algoritmo InsertionSort ao invés da função Particiona para sub-arranjos de tamanho menores do que 100.

11. Faça um algoritmo de ordenação por MergeSort que não seja recursivo.

12. Apresente o funcionamento do método ShellSort, que é uma generalização do método de ordenação por inserção.