UNIVERSIDADE FEDERAL DE JATAÍ

Instituto de Ciências Exatas e Tecnológicas
Bacharelado em Ciência da Computação
Estrutura de Dados II

LISTA 2

- Escreva uma versão recursiva do método de ordenação por trocas sucessivas.
- Escreva uma versão recursiva do método de ordenação por seleção.
- Escreva uma versão recursiva do método de ordenação por inserção.

1) Defina formalmente o problema de ordenação.

2) Forneça um exemplo de aplicação real que envolva o problema de ordenação e de encontrar o menor valor.

3) Considere a ordenação de n números armazenados no arranjo A, localizando primeiro o menor elemento de A e permutando esse elemento contido em A[1]. Em seguida, encontre o segundo menor elemento de A e o troque pelo elemento A[2]. Continue dessa maneira para os primeiros n − 1 elementos de A. Escreva um programa para esse algoritmo conhecido como ordenação por seleção.

4) Escreva um algoritmo que receba valores em um vetor e imprima “ORDENADO” se o vetor estiver em ordem crescente.

5) Escreva um algoritmo que ordene de maneira decrescente (do maior para o menor).

6) Qual é o vetor resultante após as 4 primeiras trocas ao executar ordenação por inserção com o seguinte vetor inicial 26 65 45 73 10 18 78 93 70 49 23 22 ?

7) Faça um programa para criar um vetor capaz de armazenar 100 nomes de, no máximo, 50 caracteres cada. O programa deve conter os subprogramas abaixo relacionados. O programa deverá ter também um menu principal que permita ao usuário executar os subprogramas quantas vezes desejar e só deverá terminar sua execução se o usuário solicitar.

1. Procedimento para acrescentar ao vetor um nome digitado pelo usuário. Este nome deve ser colocado sempre após a última posição preenchida do vetor. Associado ao vetor deve existir um flag indicando se o vetor está ou não ordenado. O procedimento de inserção deve atualizar este flag para falso;

2. Procedimento para exibir os nomes que estão armazenados no vetor em um dado momento;
3. Procedimento para classificar em ordem crescente os nomes armazenados no vetor em um dado momento, usando o algoritmo selection sort.
4. Procedimento para classificar em ordem crescente os nomes armazenados no vetor em um dado momento, usando o algoritmo bubble sort.
5. Função para fazer uma busca no vetor. Um dos parâmetros dessa função deve ser o nome a ser procurado. Caso encontre, a função deve retornar para o programa a posição do vetor onde encontrou o elemento. Caso contrário deve retornar -1 se não encontrar, e -2 se o vetor estiver vazio. O programa deverá utilizar um algoritmo de busca sequencial simples se o vetor não estiver ordenado e o algoritmo de busca binária, caso o vetor esteja ordenado.

8) Crie um programa que permita realizar cadastros de alunos. Cada cadastro de alunos deve ser composto pelos seguintes dados: ano de ingresso, nome, CPF e matrícula. A matrícula deve ser gerada pelo próprio programa concatenando o sufixo do ano de ingresso com o CPF. Por exemplo, se um aluno entrou em 2019 na universidade e seu CPF for 12345678910 então o número de matrícula será 1912345678910. O seu programa deve permitir ao usuário:

a) Cadastrar novos alunos;
b) Excluir o cadastro de um aluno pelo número de matrícula;
c) Realizar a ordenação dos cadastros pelo número de matrícula;
d) Exibir uma lista com todos os cadastros;
e) Realizar a ordenação dos cadastros pelo nome.

9) Uma ordenação por contagem de um vetor x de tamanho n é executada da seguinte forma: declare um vetor count e defina count[i] como o número de elementos menores que x[i]. Em seguida, coloque x[i] na posição count[i] de um vetor de saída (leve em consideração a possibilidade de elementos iguais). Escreva uma função para ordenar um vetor x de tamanho n usando esse método.

10) Um vetor v[p,r] está “arrumado” se existe j pertence [p, r] tal que v[p..j - 1]<v[j]< v[j + 1.. r]. Escreva um algoritmo que decida se v[p,r] está arrumado. Em caso afirmativo, o seu algoritmo deve devolver o valor de j.

11) Faça um programa em linguagem de programação C que seja capaz de ordenar um baralho de cartas usando o método de ordenação por distribuição. Você pode considerar que o baralho possuirá apenas as seguintes cartas:

As < 2 < 3 < J < Q < K com os naipes

paus < ouro < copas < espada

O seu algoritmo deve ser capaz de gerar o baralho, embaralhar as cartas e em seguida ordená-las. Todas as operações realizadas pelo seu programa devem ser exibidas na tela para o usuário.

12) Implemente o algoritmo MergeSort para fazer ordenações decrescente. Exiba na tela para o usuário quantas comparações foram necessárias fazer usando o MergeSort para ordenar o conjunto de dados.

13) Crie um programa que dado uma string, coloque as letras dela em ordem decrescente usando o algoritmo quick sort. Considere a seguinte estrutura:

```
struct pessoa{
int Matricula;
char Nome[30];
float Nota;
};
```

Faça uma função que dado um vetor de tamanho N dessa estrutura, ordene o vetor pelo campo escolhido pelo usuário.

14) Exercício retirado de http://edirlei.3dgb.com.br/aulas/paa_2017_1/ListaExercicios10.pdf

O arquivo senhas.txt contêm dados sobre a frequência de uso de senhas semelhantes de 430 mil usuários de um determinado sistema. Este arquivo pode ser acessado no seguinte link: http://www.inf.puc-rio.br/~elima/paa/senhas.txt

No arquivo, cada linha representa uma determinada senha. O primeiro número de cada linha indica o tamanho da senha (número de caracteres) e o segundo número indica a quantidade de ocorrências dessa senha (frequência). Por exemplo, se a senha "vogel1282", de tamanho 9, foi utilizada por 25 usuários, haverá no arquivo uma linha como esta:

9 25 vogel1282

O seguinte programa faz a leitura do arquivo de senhas e as armazena em um vetor de ponteiros: http://www.inf.puc-rio.br/~elima/paa/senhas.c

a) Implemente e utilize todos os algoritmos de ordenação estudados (Bubble Sort, Selection Sort, Insertion Sort, Merge Sort e Quick Sort) para ordenar as senhas em ordem alfabética.
b) Para cada algoritmo, registre o tempo necessário para ordenar o vetor de senhas (em milissegundos).

c) Elabore uma tabela para comparar o tempo de execução dos algoritmos.