Classificação por Troca - bubble sort

Se aplicarmos o algoritmo apresentado sobre o vetor $v = \{ 2, 3, 4, 5, 6, 1 \}$, serão necessárias quantas passagens?

Serão necessárias cinco passagens para que ocorra a classificação.

Como minimizar este problema?

Um aluno atento, poderia sugerir que o conceito proposto pelo método da troca simples poderia ser aplicado efetuando-se uma varredura no vetor do último ao primeiro elemento, resultando assim em uma classificação na primeira passagem.

Mas, o que ocorreria se adotarmos esta segunda abordagem e o vetor de entrada fosse $v2 = \{ 6, 1, 2, 3, 4, 5 \}$?

O problema persistiria!

Como resolver esta questão?

UNIVASF

Classificação por Troca - bubble sort

Existem outras formas de aprimorar o *bubble sort*.

Uma delas é fazer com que as passagens sucessivas percorram o vetor em sentidos opostos, de modo que os elementos de menor magnitude se desloquem mais rapidamente para o início da lista da mesma forma que os de maior magnitude se desloquem para o final (no caso de uma ordenação crescente).

Construa agora, como exercício, um módulo que implemente este método.

UNIVASF

```
procedimento shaker_sort (var v: vetor [1..100] de inteiro;
    n: inteiro)
var aux, esq, dir, i: inteiro
    trocou: logico
Inicio
   esq <-1
   dir <- n
   repita
      trocou <- falso
      para i de esq ate dir-1 faca
         se (v[i]>v[i+1]) entao
            aux <- v[i]
            v[i] <- v[i+1]
            v[i+1] <- aux
            trocou <- verdadeiro
         fimse
      fimpara
      dir <- dir - 1
```

```
      se (trocou) entao
        trocou <- falso
        para i de dir ate esq+1 passo -1 faca
          se (v[i]<v[i-1]) entao
            aux <- v[i]
            v[i] <- v[i-1]
            v[i-1] <- aux
            trocou <- verdadeiro
          fimse
        fimpara
      fimse
      esq <- esq + 1
    ate ((esq>=dir) ou (nao trocou))
fimprocedimento
```

Classificação por Troca - shaker sort

Esta variante é conhecida como troca alternada ou *shaker sort,* devido a propiciar, por assim dizer, acomodação por "agitação" das chaves.

O ganho obtido se deve a que:

a) vão se formando dois subvetores ordenados que podem sair do processo de comparação e trocas;

b) para vetores pouco desordenados, as chaves menores também são logo posicionadas, detectando-se em seguida o fim do processo.

Classificação por Troca - shaker sort

De qualquer forma, o ganho é apenas em relação ao número de comparações: as trocas a serem efetuadas são sempre lado a lado, e todos os elementos terão que se movimentar no mesmo número de casas que no método *bubble sort*.

Como as comparações são menos onerosas que as trocas, estas continuam a ditar o desempenho.

UNIVASF

Classificação

Devemos ter em mente que os métodos de classificação podem ser aplicados não apenas sobre listas de inteiros, podemos aplicar sobre listas de reais, de caracteres, etc..

Outro detalhe a ser ressaltado é que estas listas podem ser constituídas não apenas por vetores de elementos dos tipos primitivos, ou seja, podemos classificar, por exemplo, uma lista de valores reais que se encontra armazenada nos elementos de um determinado campo dos registros contidos em um vetor de registros.

UNIVASF

Classificação

Exercício 55:

Defina um tipo de dado capaz de armazenar o nome, o número de inscrição e o percentual de acerto de um candidato a um concurso.

Em seguida construa um algoritmo capaz de manipular uma lista com no máximo 100 registros de candidatos, onde cada registro é um elemento do tipo de dado definido. A manipulação da lista é feita através dos seguintes módulos: incluir candidato, excluir candidato, classificar lista com base no desempenho dos candidatos (implementar bubble sort), classificar lista com base no nome dos candidatos (implementar shaker sort), e imprimir lista. O algoritmo deve se utilizar de forma satisfatória dos módulos mencionados.

UNIVASF

Classificação por Seleção - selection sort

Outro método, também simples, de ordenação é a ordenação por seleção.

Princípio de funcionamento:

1. Selecione o menor item do vetor (ou o maior).
2. Troque-o com o item que está na primeira posição do vetor.

Repita estas duas operações com os n-1 itens restantes, depois com os n-2 itens, até que reste apenas um elemento.

UNIVASF

Classificação por Seleção - selection sort

A ordenação por seleção consiste, em cada etapa, em selecionar o maior (ou o menor) elemento e colocá-lo em sua posição correta dentro da futura lista ordenada.

Durante a aplicação do método de seleção a lista com *n* registros fica decomposta em duas sub listas, uma contendo os itens já ordenados e a outra com os restantes ainda não ordenados.

- No início a sub lista ordenada é vazia e a outra contém todos os demais.
- No final do processo a sub lista ordenada apresentará (n-1) itens e a outra apenas 1.

UNIVASF

Classificação por Seleção - selection sort

As etapas (ou varreduras) como já descrito consistem em buscar o maior (ou menor) elemento da lista não ordenada e colocá-lo na lista ordenada.

Para uma melhor compreensão trabalharemos com um exemplo, visando demonstrar o resultado das etapas da ordenação de um vetor de inteiros, consideraremos o ordenamento crescente dos elementos e selecionaremos em cada etapa o maior elemento do subvetor não ordenado.

UNIVASF

Classificação por Seleção - selection sort

Resultado das Etapas					
Etapa	X[1]	X[2]	X[3]	X[4]	X[5]
0	5	9	1	4	3
1	{5	3	1	4}	{9}
2	{4	3	1}	{5	9}
3	{1	3}	{4	5	9}
4	{1}	{3	4	5	9}

Classificação por Seleção - selection sort

Exemplo:

	1	2	3	4	5	6
Chaves Iniciais	O	R	D	E	N	A
I=1	A	R	D	E	N	O
I =2	A	D	R	E	N	O
I=3	A	D	E	R	N	O
I=4	A	D	E	N	R	O
I=5	A	D	E	N	O	R

Nota: As Chaves em vermelho sofreram uma troca entre si

Classificação por Seleção - selection sort

Com base no que foi visto, construa um módulo, que receba como parâmetros um vetor de inteiros com no máximo cem elementos e o número de elementos neste vetor. Este módulo deve ordenar o vetor implementando a classificação por seleção.

UNIVASF

Classificação por Seleção - selection sort

```
procedimento selecao(var vet:vetor[1..100] de inteiro; n:
    inteiro)
var aux, j, i, maior: inteiro
inicio
  para i de 0 ate n-2 faca
    maior <- 1
    para j de 2 ate n-i faca
      se (vet[j]>vet[maior]) entao
        maior <- j
      fimse
    fimpara
    aux<-vet[n-i]
    vet[n-i]<-vet[maior]
    vet[maior]<-aux
  fimpara
fimprocedimento
```