

19. Aninhamento de laços

DIM0320

2015.1

Sumário

- 1 Interseção de conjuntos
- 2 Ordenação por seleção
- 3 Ordenação por inserção
- 4 Exercícios

1 Interseção de conjuntos

2 Ordenação por seleção

3 Ordenação por inserção

4 Exercícios

Lembrete

Elementos vistos

- leia, escreva
- <-
- condicionais:
 - ▶ se entao senao fimse
 - ▶ se entao fimse
 - ▶ escolha
- laços
 - ▶ enquanto..faca
 - ▶ repita..ate
 - ▶ para..faca

Sobre laços

- Qualquer outro comando pode ocorrer no corpo de um laço
- Se esse outro comando for um laço, os dois laços são **aninhados**.

Interseção de conjuntos

Observações

- O programa usa vetores para representar conjuntos
- Um conjunto **não** pode ter duas vezes o mesmo elemento

Assunto

Escrever um programa que

- lê dois conjuntos A e B de 25 elementos inteiros.
- calcula um terceiro conjunto $C = A \cap B$
- escreve como saída o conjunto C

- 1 Interseção de conjuntos
- 2 Ordenação por seleção
- 3 Ordenação por inserção
- 4 Exercícios

O problema da ordenação

Definição

O **problema de ordenação** tem a definição seguinte:

Entrada Uma sequência de n números $(a_1, a_2, \dots, a_n)$

Saída Uma permutação (re-ordenação) $(a'_1, a'_2, \dots, a'_n)$ da sequência de entrada tal que $a'_1 \leq a'_2 \leq \dots \leq a'_n$

Observação

- Em Portugal, sequência = **vetor**

Ideia

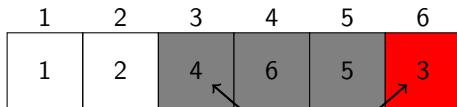
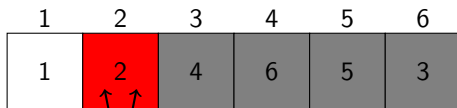
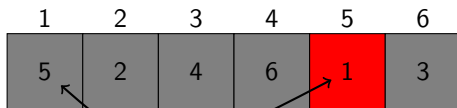
Procedimento

- Pegue o **menor elemento** da sequência e excluí-lo da
- Troque esse elemento com o primeiro elemento da sequência
- Pegar o **segundo menor elemento** da sequência (i.e. o menor do resto da sequência)
- Troque esse elemento com o segundo elemento da sequência
- ...
- Assim para os $n - 1$ primeiros elementos da sequência

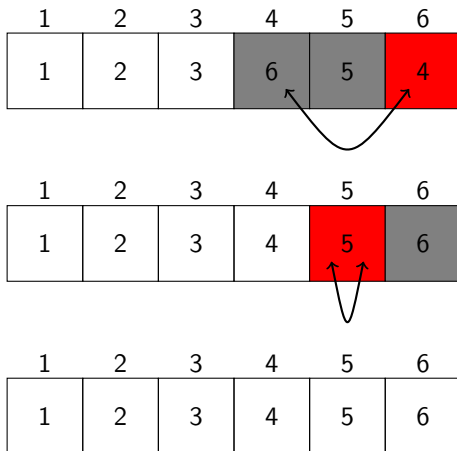
Pergunta

- Porque considerar só os $n - 1$ primeiros elementos e não os n primeiros elementos ?

Execução



Execução



Portugol

```
1 algoritmo "Ordenação por selecao"
2 var v : vetor [1..100] de inteiro
3     n : inteiro // n < 100
4     i, j, min, temp : inteiro
5 inicio
6     repita
7         leia(n)
8     ate (n > 0) e (n <= 100)
9     para i de 1 ate n faca
10         v[i] <- randi(100) // valor aleatorio inteiro no v[i]
11     fimpara
12
13     // Ordenacao
14     para i de 1 ate n faca
15         min <- i // indice do menor elemento de [i..n]
16         para j de i + 1 ate n faca
17             se v[j] < v[min] entao
18                 min <- j
19         fimse
20     fimpara
21     //troca v[i] e o min
22     temp <- v[i]
23     v[i] <- v[min]
24     v[min] <- temp
25 fimpara
26
27 para i de 1 ate n faca
28     escreva(v[i], " ")
29 fimpara
30 fimalgoritmo
```

Corretude do algoritmo

Invariante

- O sub-vetor $v[1..i]$ está ordenado.
- Todos os elementos do sub-vetor $v[1..i-1]$ são menores que os elementos do sub-vetor $v[i..n]$

Para mostra a corretude

- Mostrar que o invariante é verdadeiro **antes** da primeira
- Se o invariante é verdadeiro antes do uma iteração do laço, ele **fica** verdadeiro antes da próxima iteração.

- 1 Interseção de conjuntos
- 2 Ordenação por seleção
- 3 Ordenação por inserção**
- 4 Exercícios

Ideia

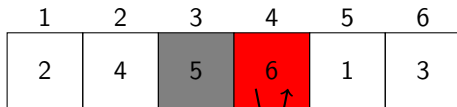
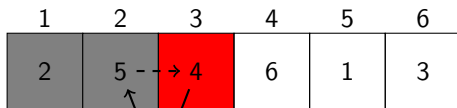
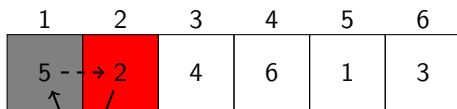
Problema

Imagine que tem que ordenar uma mão de cartas (em ordem crescente). Inicialmente elas estão face para baixo e a sua mão esquerda está vazia. Como fazer ?

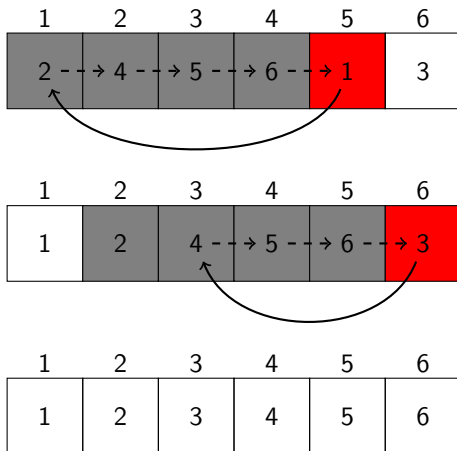
Um procedimento

- Pegue uma carta de cada vez, e insira-a na posição correta na mão esquerda.
- Para achar a posição correta na mão esquerda, vai comparar a nova carta com as presentes, da direita para esquerda.

Execução



Execução



Portugol

```
1 algoritmo "Ordenacao por insercao"
2 var v : vetor [1..100] de inteiro
3     n : inteiro // n < 100
4     i, j, chave : inteiro
5 inicio
6     repita
7         leia(n)
8     ate (n > 0) e (n <= 100)
9     para i de 1 ate n faca
10         v[i] <- randi() // valor aleatorio inteiro no v[i]
11     fimpara
12
13     // Ordenacao
14     para j de 2 ate n faca
15         chave <- v[j]
16         // inserir v[j] na sequencia ordenada v[1..j - 1]
17         i <- j - 1
18         enquanto (i > 0) e (v[i] > chave) faca
19             v[i + 1] <- v[i]
20             i <- i - 1
21         fimenquanto
22         v[i + 1] <- chave
23     fimpara
24
25     para i de 1 ate n faca
26         escreva(v[i], " ")
27     fimpara
28
29 fimalgoritmo
```

Corretude do algoritmo

Invariante

No começo de cada iteração do laço para j de 2 ate n faça, o sub-vetor $v[1..j-1]$ tem todos elementos dos elementos do sub-vetor de entrada $v[1..j-1]$, mas **ordenado**.

Perguntas ?



<http://dimap.ufrn.br/~richard/dim0320>

- 1 Interseção de conjuntos
- 2 Ordenação por seleção
- 3 Ordenação por inserção
- 4 Exercícios

Adição

Assunto

Dadas duas sequências com n números inteiros entre 0 e 9, podemos interpretá-las como dois números inteiros de n algarismos para, em seguida, calcular a sequência de números que representa a soma dos dois inteiros. Por exemplo, se $n = 8$ e $(8, 2, 4, 3, 4, 2, 5, 1)$ e $(3, 3, 7, 5, 2, 3, 3, 7)$ forem as duas sequências, então a soma dos dois “números” dados é igual a

$$\begin{array}{rcccccccc} & 8 & 2 & 4 & 3 & 4 & 2 & 5 & 1 \\ + & 3 & 3 & 7 & 5 & 2 & 3 & 3 & 7 \\ \hline 1 & 1 & 6 & 1 & 8 & 6 & 5 & 8 & 8 \end{array}$$

Escrever um algoritmo que leia um número inteiro positivo, n , com $n \leq 100$, e duas sequências com n números inteiros entre 0 e 9 e calcule e escreva a sequência que representa o número resultante da soma dos dois “números” dados pelas duas sequências de entrada (como acima)

Fusão de sequências

Assunto

Escreva um algoritmo que leia dois números inteiros positivos, n e m , e duas sequências ordenadas, em ordem não-decrescente, com n e m números inteiros, respectivamente.

Em seguida, o algoritmo deve criar e escrever como saída uma única sequência ordenada em ordem não-decrescente com todos os $m + n$ números das duas sequências de entrada.

Assuma que $n, m \leq 100$.

Segmento de soma máxima

Assunto

Escrever um algoritmo que lê um inteiro positivo, n , e uma sequência com n números inteiros e calcula e escreve o comprimento do maior segmento crescente da sequência.

Por exemplo, se a entrada for

9
5, 10, 3, 2, 4, 7, 9, 8, 5

então a saída é 4, pois 2, 4, 7, 9 é um segmento crescente com comprimento máximo e igual a 4 da sequência 5, 10, 3, 2, 4, 7, 9, 8, 5

Se a entrada for

5
10, 8, 7, 5, 2

então a saída é 1, pois todos os segmentos crescentes de tamanho máximo da sequência 10, 8, 7, 5, 2 possuem tamanho 1 e há exatamente cinco segmentos crescentes de tamanho máximo: 10, 8, 7, 5, 2.

Encontrar um sub-texto num texto

Objetivo

- Criar um algoritmo para achar uma subcadeia `stexto` numa cadeia `texto`.
- Os dois dados serão lidos da tela

Apoio

Pode-se usar a função primitiva:

- `compr(c)` retorna o comprimento (inteiro) do texto `c`