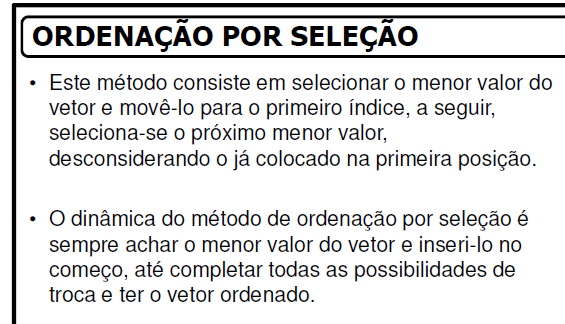
Ordenação por seleção



Mostrar exemplo prático: 12 – 30 – 10 – 8 – 20 – 5 – 3

Dar noção intuitiva e dar como algoritmo, onde indicamos posição do menor elemento (min)

Pedir para simularem com o exemplo:

9 – 7 – 10 – 1 – 5 – 20 – 15

Passo: 1

Posição (min) = 1 (MenorElemento = 9)

9 e 7 – Posição = 2 (MenorElemento = 7)

7 e 10 – nada

7 e 1 – Posicao = 4 (MenorElemento = 1)

1 e 5 – nada

1 e 20 – nada

1 e 15 nada

Trocar Passo (1) com posição do menor elemento (4)

1 – 7 – 10 – 9 – 5 – 20 – 15

Passo = 2

Posição = 2 (MenorElemento = 7)

7 e 10 – nada

7 e 9 – nada

7 e 5 –Posição = 5 (MenorElemento = 5)

5 e 20 – nada

5 e 15 – nada

Trocar Passo (2) com posição do menor elemento (5)

1 – 5 – 10 – 9 – 7 – 20 – 15

Passo = 3

Posicao = 3 (MenorElemento = 10)

10 e 9 –Posicao = 4 (MenorElemento = 9)

9 e 7 – Posicao = 5 (MenorElemento = 7)

7 e 20 – nada

7 e 15 – nada

Trocar Passo (3) com posição do menor elemento (5)

1 – 5 – 7 – 9 – 10 – 20 – 15

Passo = 4

Posicao = 4 (MenorElemento = 9)

9 e 10 – nada

9 e 20 – nada

9 e 15 – nada

Trocar Passo (4) com posição do menor elemento (4) – **não faz nada**

1 – 5 – 7 – 9 – 10 – 20 – 15

Passo = 5

Posicao = 5 (MenorElemento = 10)

10 e 20 – nada

10 e 15 – nada

Trocar Passo (5) com posição do menor elemento (5) – não faz nada

1 – 5 – 7 – 9 – 10 – 20 – 15

Passo = 6

Posicao = 6 (MenorElemento = 20)

20 e 15 –Posicao = 7 (MenorElemento = 15)

Trocar Passo (6) com posição do menor elemento (7)

1 – 5 – 7 – 9 – 10 – 15 – 20

Dar algoritmo da ordenação por seleção

Selecao(tot: inteiro; ent[]: inteiro)

i,j,min,aux: inteiro;

para i = 1 até tot-1 faça

min 🡨 i

para j = i+1 até tot faça

se ent[j] < ent[min] entao

min 🡨 j;

fim\_se

fim\_para

se (i <> min) entao

aux = ent[i];

ent[i] 🡨 ent[min];

ent[min] 🡨 aux;

fim\_se

fim\_para

fim

Pedir para fazerem ***(se houver tempo)****:*

8 – 10 – 12 – 4 – 3 – 7 – 6

MenorElemento 🡨1

Compara posição 1 (8) com 10 – nada

Compara posição 1 (8) com 12 – nada

Compara posição 1 (8) com 4 – muda. MenorElemento 🡨 4

Compara posição 4 (4) com 3 – muda. MenorElemento 🡨5

Compara posição 5 (3) com 7 –nada.

Compara posição 5 (3) com 6 – nada.

Troca posição 1 com posição 5

3 – 10 – 12- 4 – 8 – 7 – 6

Compara posição 2 (10) com .....

Troca posição 2 com posição 4

3 – 4 – 12 – 10 – 8 – 7 – 6

Posicao 3

3 – 4- 6- 10 – 8- 7 – 12

Posicao 4

3 – 4- 6- 7 – 8- 10 – 12

Posicao5

3 – 4- 6- 7 – 8- 10 – 12

Posicao 6

3 – 4- 6- 7 – 8- 10 – 12

**- Algoritmo Instavel –** caso exista ao menos uma situação onde elementos de mesmo valor troquem de posição em relação a originalmente informada.

- Algoritmo estável – aquele em que elementos de mesmo valor **NUNCA** trocam de posição.

*Impossível provar estável uma vez que é para todos os casos e todos os casos é infinito*

**Algoritmo INSTAVEL.Ex: sem troca: 3-1-4-3, com troca 3 – 3 – 1**

**Mostrar complexidade: O(n2) = n(n-1)/2**

**Pior Caso, Caso Médio e Melhor caso: COMPARAÇÃO (+ importante): todos iguais.**

**TROCA (menos relevante): melhor caso: ordenado.**

**Passar Base do programa C com alunos**

#include <stdio.h>

#include <stdlib.h>

void ordenar(int ent[],int tot);

int main(int argc, char \*argv[])

{

int entrada[101];

int totele;

int i;

printf("Total de elementos: ");

scanf("%d",&totele);

for (i=1;i<=totele;i++)

{

printf("Entrada %d = ",i);

scanf("%d",&entrada[i]);

}

ordenar(entrada,totele);

printf("Saida \n");

for (i=1;i<=totele;i++)

{

printf("%d - ",entrada[i]);

}

system("PAUSE");

return 0;

}

void ordenar(int ent[],int tot)

{

}

|  |  |
| --- | --- |
| #include <stdio.h>  #include <stdlib.h>  void ordenar(int ent[],int tot);  int main(int argc, char \*argv[])  {  int entrada[101];  int totele;  int i;  printf("Total de elementos: ");  scanf("%d",&totele);  for (i=1;i<=totele;i++)  {  printf("Entrada %d = ",i);  scanf("%d",&entrada[i]);  }  ordenar(entrada,totele);  printf("Saida \n");  for (i=1;i<=totele;i++)  {  printf("%d - ",entrada[i]);  }  system("PAUSE");  return 0;  } | void ordenar(int ent[],int tot)  {  int i,j,k,aux,min,contarpassos;  contarpassos=0;  for (i=1;i<=tot-1;i++)  {  min = i;  for (j=i+1;j<=tot;j++)  {  contarpassos++;  if (ent[j] < ent[min])  {  min=j;  }  }  if (i != min)  {  aux = ent[i];  ent[i]=ent[min];  ent[min] = aux;  }  for (k=1;k<=tot;k++)  {  printf("%d - ",ent[k]);  }  system("PAUSE");  }  printf("%d\n",contarpassos);  } |