

Ex: 9 -5 – 7 – 8 – 10 (5 elementos – 4 passos) – Intuitivo, fazer por selecao tambem

|  |  |
| --- | --- |
| 9  5  9 e 5 🡪 menor, incluir 5 na frente. 5-9  7  7 e 5 🡪 maior, nada  7 e 9 🡪 menor, incluir na frente 5 – 7 – 9  8  5 e 8 🡪 maior, nada  7 e 8 🡪 maior, nada  9 e 8 🡪 menor, inlcuir na frente. 5-7-8-9  10  5 e 10 🡪 maior, nada  7 e 10 🡪 maior, nada  8 e 10🡪 maior, nada  9 e 10 🡪 maior, é o ultimo, entra atrás. 5-7-8-9-10 | **Tal qual o algoritmo**  **9**  **5**  **9 e 5 🡪 menor, incluir 5 na primeira posição. 5 – 9**  **7**  **7 e 9 🡪 9 é maior – desloca 9 em 1 posicao**  **5 e 7 🡪 5 é menor – incluir 7. 5-7-9**  **8**  **9 e 8 🡪 9 é maior. Desloca 9 em 1 posicao**  **7 e 8 🡪 7 é menor. Incluir 8. 5-7-8-9**  **10**  **9 e 10🡪 9 é menor. Incluir 10. 5-7-8-9-10** |

Pedir para alunos simularem com os valores 6 – 4 – 7- 5 – 1 (5 elementos – 4 passos), pedir para fazerem por seleção também

|  |  |
| --- | --- |
| 6  4  4 e 6 menor, 4-6  7  4 e 7 🡪 maior, nada  6 e 7 🡪 maior, ultimo. 4-6-7  5  4 e 5🡪 maior, nada  6 e 5 🡪 menor, entra na frente. 4-5-6-7  1  1 e 4 🡪 menor, entra na frente. 1-4-5-6-7 | 6  4  4 e 6 🡪 6 é maior, desloca em uma posição. 4-6  7  6 e 7 🡪 6 é menor, incluir 7 no final da lista. 4-6-7  5  5 e 7🡪 7 é maior. Desloca em uma posição  5 e 6 🡪 6 é maior. Desloca em uma posição  5 e 4🡪 4 é menor. Incluir 5. 4-5-6-7  1  1 e 7 🡪 7 é maior. Desloca em 1 posicao  1 e 6 🡪 6 é maior. Desloca em 1 posicao  1 e 5 🡪 5 é maior. Desloca em 1 posicao  1 e 4 🡪 4 é maior. Ultima posição. Incluir 1. 1-4-5-6-7 |

Dar mais exemplos: 7 – 8 – 4 – 6– 2– 5– 3 – 1 – 10 (9 elementos – 8 passos), pedir para fazerem por seleção também

Mostrar exemplo 9 -5 – 7 – 8 – 10 de modo algorítmico.

**Mostrar Ordem = O(n2).**

**Estavel**

Melhor Caso: **ordenado** – somente uma comparação por passo – n-1 passos, O (n).

Pior Caso: **inversamente ordenado** – n-1 comparações por passo – n-1 passos, O(n2)

Caso Médio: (n-1)/2 comparações por passo - n-1 passos, O(n2)

Algoritmo

|  |  |
| --- | --- |
| Insercao(ent[]: inteiro, tot: inteiro)  i,j,aux: inteiro;  para i = 2 até tot faça  aux 🡨 ent[i]  j 🡨 i- 1  enquanto (j>=1 E aux< ent[j]) faça  ent[j+1] 🡨 ent[j]  j 🡨 j-1  fim\_enquanto  ent[j+1] 🡨 aux  fim\_para  fim | void insercao(int ent[],int tot)  {  int i,j,aux;  for (i=2; i<=tot; i++)  {  aux = ent[i];  j=i-1;  while ((j>=1) && (aux< ent[j]) )  {  ent[j+1] = ent[j];  j--;  }  ent[j+1] = aux;  }  } |

**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)

{

}

Mostrar quadro resumo

|  |  |  |
| --- | --- | --- |
|  | SELEÇÃO | INSERÇÃO |
| Estavel x Instavel | Instavel | Estavel |
| Melhor caso | O(n2) | O(n) - ordenado |
| Caso Médio | O(n2) | O(n2) |
| Pior Caso | O(n2) | O(n2) – inversamente ordenado |