MC-102 — Aula 12
Ordenação – Selection Sort, Bubble Sort e Insertion Sort

Alexandre M. Ferreira

IC – Unicamp

12/4/2017

Roteiro

Ordenação

- Vamos estudar alguns algoritmos para o seguinte problema:

Dado uma coleção de elementos com uma relação de ordem entre si, devemos gerar uma saída com os elementos ordenados.

- Nos nossos exemplos usaremos um vetor de inteiros para representar tal coleção.
  - É claro que quaisquer inteiros possuem uma relação de ordem entre si.
- Apesar de usarmos inteiros, os algoritmos servem para ordenar qualquer coleção de elementos que possam ser comparados.

Ordenação

- O problema de ordenação é um dos mais básicos em computação.
  - Mas muito provavelmente é um dos problemas com o maior número de aplicações diretas ou indiretas (como parte da solução para um problema maior).
- Exemplos de aplicações diretas:
  - Criação de *rankings*, Definir preferências em atendimentos por prioridade, Criação de Listas etc.
- Exemplos de aplicações indiretas:
  - Otimizar sistemas de busca, manutenção de estruturas de bancos de dados etc.

Selection Sort

- Seja **vet** um vetor contendo números inteiros.
- Devemos deixar **vet** em ordem crescente.
- A idéia do algoritmo é a seguinte:
  - Ache o menor elemento a partir da posição 0. Troque então este elemento com o elemento da posição 0.
  - Ache o menor elemento a partir da posição 1. Troque então este elemento com o elemento da posição 1.
  - Ache o menor elemento a partir da posição 2. Troque então este elemento com o elemento da posição 2.
  - E assim sucessivamente...

Selection-Sort

Exemplo: (5,3,2,1,90,6).

Iteração 0. Acha menor: (5,3,2,1,90,6). Faz troca: (1,3,2,5,90,6).

Iteração 1. Acha menor: (1,3,2,5,90,6). Faz troca: (1,2,3,5,90,6).

Iteração 2. Acha menor: (1,2,3,5,90,6). Faz troca: (1,2,3,5,90,6).

Iteração 3. Acha menor: (1,2,3,5,90,6). Faz troca: (1,2,3,5,90,6).

Iteração 5: Acha menor: (1,2,3,5,90,6). Faz troca: (1,2,3,5,6,90).

Selection-Sort

Exemplo: (5,3,2,1,90,6).
Iteração 0. Acha menor: (5,3,2,1,90,6). Faz troca: (1,3,2,5,90,6).
Iteração 1. Acha menor: (1,3,2,5,90,6). Faz troca: (1,2,3,5,90,6).
Iteração 2. Acha menor: (1,2,3,5,90,6). Faz troca: (1,2,3,5,90,6).
Iteração 3. Acha menor: (1,2,3,5,90,6). Faz troca: (1,2,3,5,90,6).
Iteração 5: Acha menor: (1,2,3,5,90,6). Faz troca: (1,2,3,5,6,90).

Selection-Sort

Exemplo: (5,3,2,1,90,6).
Iteração 0. Acha menor: (5,3,2,1,90,6). Faz troca: (1,3,2,5,90,6).
Iteração 1. Acha menor: (1,3,2,5,90,6). Faz troca: (1,2,3,5,90,6).
Iteração 2. Acha menor: (1,2,3,5,90,6). Faz troca: (1,2,3,5,90,6).
Iteração 3. Acha menor: (1,2,3,5,90,6). Faz troca: (1,2,3,5,90,6).
Iteração 5: Acha menor: (1,2,3,5,90,6). Faz troca: (1,2,3,5,6,90).

Selection-Sort

Exemplo: (5,3,2,1,90,6).
Iteração 0. Acha menor: (5,3,2,1,90,6). Faz troca: (1,3,2,5,90,6).
Iteração 1. Acha menor: (1,3,2,5,90,6). Faz troca: (1,2,3,5,90,6).
Iteração 2. Acha menor: (1,2,3,5,90,6). Faz troca: (1,2,3,5,90,6).
Iteração 3. Acha menor: (1,2,3,5,90,6). Faz troca: (1,2,3,5,90,6).
Iteração 5: Acha menor: (1,2,3,5,90,6). Faz troca: (1,2,3,5,6,90).

Selection-Sort

Exemplo: (5,3,2,1,90,6).
Iteração 0. Acha menor: (5,3,2,1,90,6). Faz troca: (1,3,2,5,90,6).
Iteração 1. Acha menor: (1,3,2,5,90,6). Faz troca: (1,2,3,5,90,6).
Iteração 2. Acha menor: (1,2,3,5,90,6). Faz troca: (1,2,3,5,90,6).
Iteração 3. Acha menor: (1,2,3,5,90,6). Faz troca: (1,2,3,5,90,6).
Iteração 5: Acha menor: (1,2,3,5,90,6). Faz troca: (1,2,3,5,6,90).

Selection-Sort

- Como achar o menor elemento a partir de uma posição inicial?
- Vamos achar **o índice** do menor elemento em um vetor, a partir de uma posição inicial **ini**:

```
int min = ini, j;
for(j=ini+1; j<tam; j++){
  if(vet[min] > vet[j])
    min = j;
}
```

Selection-Sort

- Como achar o menor elemento a partir de uma posição inicial?
- Vamos achar **o índice** do menor elemento em um vetor, a partir de uma posição inicial **ini**:

```
int min = ini, j;
for(j=ini+1; j<tam; j++){
  if(vet[min] > vet[j])
    min = j;
}
```

Selection-Sort

- O trecho de código abaixo busca o índice do elemento mínimo de um vetor, a partir de uma posição **ini** passada por parâmetro:

```
// A variável min guarda o índice do menor elemento
int min = ini, j;
for(j=ini+1; j<tam; j++){
  if(vet[min] > vet[j])
    min = j;
}
```

Selection-Sort

- Dado o código anterior para achar o índice do menor elemento, como implementar o algoritmo de ordenação?
- Ache o menor elemento a partir da posição 0, e troque com o elemento da posição 0.
- Ache o menor elemento a partir da posição 1, e troque com o elemento da posição 1.
- Ache o menor elemento a partir da posição 2, e troque com o elemento da posição 2.
- E assim sucessivamente...

Selection-Sort

```
int i, j, min, aux;
for(i=0; i<tam; i++){

  //Acha posição do menor elemento a partir de i
  min = i;
  for(j=i+1; j<tam; j++){
    if(vet[min] > vet[j])
      min = j;
  }

  aux = vet[i];
  vet[i] = vet[min];
  vet[min] = aux;
}
```

Selection-Sort

Com os códigos anteriores implementadas podemos executar o exemplo:

```
int main(){
  int vetor[10]={14,7,8,34,56,4,0,9,-8,100};
  int i, j, min, aux, tam=10;
  printf("\nVetor Antes: ");
  for(i=0;i<10;i++)
    printf("%d, ",vetor[i]);

  for(i=0; i<tam; i++){
    min = i;
    for(j=i+1; j<tam; j++){
      if(vetor[min] > vetor[j])
        min = j;
    }
    aux = vetor[i];
    vetor[i] = vetor[min];
    vetor[min] = aux;
  }

  printf("\n\nVetor Depois: ");
  for(i=0;i<10;i++)
    printf("%d, ",vetor[i]);
  return 0;
}
```

Bubble-Sort

- Seja **vet** um vetor contendo números inteiros.
- Devemos deixar **vet** em ordem crescente.
- O algoritmo faz algumas iterações repetindo o seguinte:
  - Compare $vet[0]$ com $vet[1]$ e troque-os se $vet[0] > vet[1]$.
  - Compare $vet[1]$ com $vet[2]$ e troque-os se $vet[1] > vet[2]$.
  - .....
  - Compare $vet[tam-2]$ com $vet[tam-1]$ e troque-os se $vet[tam-2] > vet[tam-1]$.

  Após uma iteração repetindo estes passos o que podemos garantir???
  - O maior elemento estará na posição correta!!!

Bubble-Sort

- Seja **vet** um vetor contendo números inteiros.
- Devemos deixar **vet** em ordem crescente.
- O algoritmo faz algumas iterações repetindo o seguinte:
  - Compare $vet[0]$ com $vet[1]$ e troque-os se $vet[0] > vet[1]$.
  - Compare $vet[1]$ com $vet[2]$ e troque-os se $vet[1] > vet[2]$.
  - .....
  - Compare $vet[tam-2]$ com $vet[tam-1]$ e troque-os se $vet[tam-2] > vet[tam-1]$.

  Após uma iteração repetindo estes passos o que podemos garantir???
  - O maior elemento estará na posição correta!!!

Bubble-Sort

- Após uma iteração de trocas, o maior elemento estará na última posição.
- Após outra iteração de trocas, o segundo maior elemento estará na posição correta.
- E assim sucessivamente.
- Quantas iterações repetindo estas trocas precisamos para deixar o vetor ordenado?

Bubble-Sort

Exemplo: (5,3,2,1,90,6).
Valores sublinhados estão sendo comparados:

$(\underline{5}, \underline{3}, 2, 1, 90, 6)$
$(3, \underline{5}, \underline{2}, 1, 90, 6)$
$(3, 2, \underline{5}, \underline{1}, 90, 6)$
$(3, 2, 1, \underline{5}, \underline{90}, 6)$
$(3, 2, 1, 5, \underline{90}, \underline{6})$
$(3, 2, 1, 5, 6, 90)$

- Isto termina a primeira iteração de trocas. Temos que repetir todo o processo mais 4 vezes!!!
- Mas notem que não precisamos mais avaliar a última posição!

Bubble-Sort

Exemplo: (5,3,2,1,90,6).
Valores sublinhados estão sendo comparados:
$(\underline{5}, \underline{3}, 2, 1, 90, 6)$
$(3, \underline{5}, \underline{2}, 1, 90, 6)$
$(3, 2, \underline{5}, \underline{1}, 90, 6)$
$(3, 2, 1, \underline{5}, \underline{90}, 6)$
$(3, 2, 1, 5, \underline{90}, \underline{6})$
$(3, 2, 1, 5, 6, 90)$

- Isto termina a primeira iteração de trocas. Temos que repetir todo o processo mais 4 vezes!!!
- Mas notem que não precisamos mais avaliar a última posição!

Bubble-Sort

Exemplo: (5,3,2,1,90,6).
Valores sublinhados estão sendo comparados:
$(\underline{5}, \underline{3}, 2, 1, 90, 6)$
$(3, \underline{5}, \underline{2}, 1, 90, 6)$
$(3, 2, \underline{5}, \underline{1}, 90, 6)$
$(3, 2, 1, \underline{5}, \underline{90}, 6)$
$(3, 2, 1, 5, \underline{90}, \underline{6})$
$(3, 2, 1, 5, 6, 90)$

- Isto termina a primeira iteração de trocas. Temos que repetir todo o processo mais 4 vezes!!!
- Mas notem que não precisamos mais avaliar a última posição!

Bubble-Sort

Exemplo: (5,3,2,1,90,6).
Valores sublinhados estão sendo comparados:
$(\underline{5}, \underline{3}, 2, 1, 90, 6)$
$(3, \underline{5}, \underline{2}, 1, 90, 6)$
$(3, 2, \underline{5}, \underline{1}, 90, 6)$
$(3, 2, 1, \underline{5}, \underline{90}, 6)$
$(3, 2, 1, 5, \underline{90}, \underline{6})$
$(3, 2, 1, 5, 6, 90)$

- Isto termina a primeira iteração de trocas. Temos que repetir todo o processo mais 4 vezes!!!
- Mas notem que não precisamos mais avaliar a última posição!

Bubble-Sort

Exemplo: (5,3,2,1,90,6).
Valores sublinhados estão sendo comparados:
$(\underline{5}, \underline{3}, 2, 1, 90, 6)$
$(3, \underline{5}, \underline{2}, 1, 90, 6)$
$(3, 2, \underline{5}, \underline{1}, 90, 6)$
$(3, 2, 1, \underline{5}, \underline{90}, 6)$
$(3, 2, 1, 5, \underline{90}, \underline{6})$
$(3, 2, 1, 5, 6, 90)$

- Isto termina a primeira iteração de trocas. Temos que repetir todo o processo mais 4 vezes!!!
- Mas notem que não precisamos mais avaliar a última posição!

Bubble-Sort

Exemplo: (5,3,2,1,90,6).
Valores sublinhados estão sendo comparados:
$(\underline{5}, \underline{3}, 2, 1, 90, 6)$
$(3, \underline{5}, \underline{2}, 1, 90, 6)$
$(3, 2, \underline{5}, \underline{1}, 90, 6)$
$(3, 2, 1, \underline{5}, \underline{90}, 6)$
$(3, 2, 1, 5, \underline{90}, \underline{6})$
$(3, 2, 1, 5, 6, 90)$

- Isto termina a primeira iteração de trocas. Temos que repetir todo o processo mais 4 vezes!!!
- Mas notem que não precisamos mais avaliar a última posição!

Bubble-Sort

Exemplo: (5,3,2,1,90,6).
Valores sublinhados estão sendo comparados:
$(\underline{5}, \underline{3}, 2, 1, 90, 6)$
$(3, \underline{5}, \underline{2}, 1, 90, 6)$
$(3, 2, \underline{5}, \underline{1}, 90, 6)$
$(3, 2, 1, \underline{5}, \underline{90}, 6)$
$(3, 2, 1, 5, \underline{90}, \underline{6})$
$(3, 2, 1, 5, 6, 90)$

- Isto termina a primeira iteração de trocas. Temos que repetir todo o processo mais 4 vezes!!!
- Mas notem que não precisamos mais avaliar a última posição!

Bubble-Sort

Exemplo: (5,3,2,1,90,6).
Valores sublinhados estão sendo comparados:
$(\underline{5}, \underline{3}, 2, 1, 90, 6)$
$(3, \underline{5}, \underline{2}, 1, 90, 6)$
$(3, 2, \underline{5}, \underline{1}, 90, 6)$
$(3, 2, 1, \underline{5}, \underline{90}, 6)$
$(3, 2, 1, 5, \underline{90}, \underline{6})$
$(3, 2, 1, 5, 6, 90)$

- Isto termina a primeira iteração de trocas. Temos que repetir todo o processo mais 4 vezes!!!
- Mas notem que não precisamos mais avaliar a última posição!

Bubble-Sort

- O código abaixo realiza as trocas de uma iteração.
- São comparados e trocados, os elementos das posições: 0 e 1; 1 e 2; ...; $i-1$ e i.
- Assumimos que de $(i+1)$ até $(tam-1)$, o vetor já tem os maiores elementos ordenados.

```
for(j=0; j < i; j++) {
    if( vet[j] > vet[j+1] ) {
        aux = vetor[j];
        vetor[j] = vetor[j+1];
        vetor[j+1] = aux;
    }
}
```

Bubble-Sort

- O código abaixo realiza as trocas de uma iteração.
- São comparados e trocados, os elementos das posições: 0 e 1; 1 e 2; ...; $i-1$ e i.
- Assumimos que de $(i+1)$ até $(tam-1)$, o vetor já tem os maiores elementos ordenados.

```
for(j=0; j < i; j++) {
    if( vet[j] > vet[j+1] ) {
        aux = vetor[j];
        vetor[j] = vetor[j+1];
        vetor[j+1] = aux;
    }
}
```

Bubble-Sort

```
int i,j;
for(i=tam-1; i>0; i--) {
  for(j=0; j < i; j++) {   //Faz trocas até posição i
    if(vet[j] > vet[j+1]) {
          aux = vetor[j];
          vetor[j] = vetor[j+1];
          vetor[j+1] = aux;
    }
  }
}
```

Bubble-Sort

- Note que as trocas na primeira iteração ocorrem até a última posição.
- Na segunda iteração ocorrem até a penúltima posição.
- E assim sucessivamente.
- Por que?

Insertion Sort

- Seja **vet** um vetor contendo números inteiros, que devemos deixar ordenado.
- A idéia do algoritmo é a seguinte:
  - A cada passo, uma porção de 0 até $i - 1$ do vetor já está ordenada.
  - Devemos inserir o item da posição i na posição correta para deixar o vetor ordenado até a posição i.
  - No passo seguinte consideramos que o vetor está ordenado até i.

Insertion Sort

Exemplo: (5,3,2,1,90,6).
O valor sublinhado representa onde está o índice i

$(5, \underline{3}, 2, 1, 90, 6)$: vetor ordenado de $0 - 0$.
$(3, 5, \underline{2}, 1, 90, 6)$: vetor ordenado de $0 - 1$.
$(2, 3, 5, \underline{1}, 90, 6)$: vetor ordenado de $0 - 2$.
$(1, 2, 3, 5, \underline{90}, 6)$: vetor ordenado de $0 - 3$.
$(1, 2, 3, 5, 90, \underline{6})$: vetor ordenado de $0 - 4$.
$(1, 2, 3, 5, 6, 90)$: vetor ordenado de $0 - 5$.

Insertion Sort

Exemplo: (5,3,2,1,90,6).
O valor sublinhado representa onde está o índice i
$(5, \underline{3}, 2, 1, 90, 6)$: vetor ordenado de $0 - 0$.
$(3, 5, \underline{2}, 1, 90, 6)$: vetor ordenado de $0 - 1$.
$(2, 3, 5, \underline{1}, 90, 6)$: vetor ordenado de $0 - 2$.
$(1, 2, 3, 5, \underline{90}, 6)$: vetor ordenado de $0 - 3$.
$(1, 2, 3, 5, 90, \underline{6})$: vetor ordenado de $0 - 4$.
$(1, 2, 3, 5, 6, 90)$: vetor ordenado de $0 - 5$.

Insertion Sort

Exemplo: (5,3,2,1,90,6).
O valor sublinhado representa onde está o índice i
$(5, \underline{3}, 2, 1, 90, 6)$: vetor ordenado de $0 - 0$.
$(3, 5, \underline{2}, 1, 90, 6)$: vetor ordenado de $0 - 1$.
$(2, 3, 5, \underline{1}, 90, 6)$: vetor ordenado de $0 - 2$.
$(1, 2, 3, 5, \underline{90}, 6)$: vetor ordenado de $0 - 3$.
$(1, 2, 3, 5, 90, \underline{6})$: vetor ordenado de $0 - 4$.
$(1, 2, 3, 5, 6, 90)$: vetor ordenado de $0 - 5$.

Insertion Sort

Exemplo: (5,3,2,1,90,6).
O valor sublinhado representa onde está o índice i

$(5, \underline{3}, 2, 1, 90, 6)$: vetor ordenado de $0 - 0$.
$(3, 5, \underline{2}, 1, 90, 6)$: vetor ordenado de $0 - 1$.
$(2, 3, 5, \underline{1}, 90, 6)$: vetor ordenado de $0 - 2$.
$(1, 2, 3, 5, \underline{90}, 6)$: vetor ordenado de $0 - 3$.
$(1, 2, 3, 5, 90, \underline{6})$: vetor ordenado de $0 - 4$.
$(1, 2, 3, 5, 6, 90)$: vetor ordenado de $0 - 5$.

Insertion Sort

Exemplo: (5,3,2,1,90,6).
O valor sublinhado representa onde está o índice i

$(5, \underline{3}, 2, 1, 90, 6)$: vetor ordenado de $0 - 0$.
$(3, 5, \underline{2}, 1, 90, 6)$: vetor ordenado de $0 - 1$.
$(2, 3, 5, \underline{1}, 90, 6)$: vetor ordenado de $0 - 2$.
$(1, 2, 3, 5, \underline{90}, 6)$: vetor ordenado de $0 - 3$.
$(1, 2, 3, 5, 90, \underline{6})$: vetor ordenado de $0 - 4$.
$(1, 2, 3, 5, 6, 90)$: vetor ordenado de $0 - 5$.

Insertion Sort

Exemplo: (5,3,2,1,90,6).
O valor sublinhado representa onde está o índice i
$(5, \underline{3}, 2, 1, 90, 6)$: vetor ordenado de $0 - 0$.
$(3, 5, \underline{2}, 1, 90, 6)$: vetor ordenado de $0 - 1$.
$(2, 3, 5, \underline{1}, 90, 6)$: vetor ordenado de $0 - 2$.
$(1, 2, 3, 5, \underline{90}, 6)$: vetor ordenado de $0 - 3$.
$(1, 2, 3, 5, 90, \underline{6})$: vetor ordenado de $0 - 4$.
$(1, 2, 3, 5, 6, 90)$: vetor ordenado de $0 - 5$.

Insertion Sort

Exemplo: (5,3,2,1,90,6).
O valor sublinhado representa onde está o índice i

$(5, \underline{3}, 2, 1, 90, 6)$: vetor ordenado de $0 - 0$.
$(3, 5, \underline{2}, 1, 90, 6)$: vetor ordenado de $0 - 1$.
$(2, 3, 5, \underline{1}, 90, 6)$: vetor ordenado de $0 - 2$.
$(1, 2, 3, 5, \underline{90}, 6)$: vetor ordenado de $0 - 3$.
$(1, 2, 3, 5, 90, \underline{6})$: vetor ordenado de $0 - 4$.
$(1, 2, 3, 5, 6, 90)$: vetor ordenado de $0 - 5$.

Insertion Sort

- Vamos supor que o vetor está ordenado de 0 até $i - 1$.
- Vamos inserir o elemento da posição i no lugar correto.

```
j=i;
while(j>0) { //trocar v[i] com elementos anteriores
             //até achar sua posicao correta
  if(vet[j-1] > vet[j]) {
    aux = vet[j];
    vet[j] = vet[j-1];
    vet[j-1] = aux;
    j--;
  } else
    break;
}
```

Insertion Sort

- Vamos supor que o vetor está ordenado de 0 até $i - 1$.
- Vamos inserir o elemento da posição i no lugar correto.

```
j=i;
while(j>0) { //trocar v[i] com elementos anteriores
             //até achar sua posicao correta
  if(vet[j-1] > vet[j]) {
     aux = vet[j];
     vet[j] = vet[j-1];
     vet[j-1] = aux;
     j--;
  } else
     break;
}
```

Insertion Sort

Código completo:

```
    int i, j;
    for(i=1; i<tam; i++){
      j = i; //Colocar elemento v[i] na pos. correta
      while(j>0){ //trocar v[i] com elementos anteriores
                  //até achar sua posicao correta
        if(vet[j-1] > vet[j]){
            aux = vet[j];
            vet[j] = vet[j-1];
            vet[j-1] = aux;
            j--;
        } else
            break;
      }
    }
}
```

Insertion Sort

- Vamos apresentar uma forma alternativa de colocar **v[i]** na posição correta.
- Vamos supor que o vetor está ordenado de 0 até $i - 1$.
- Vamos inserir o elemento da posição i no lugar correto.

```
aux = vet[i];   //inserir aux na posição correta
j = i - 1; //analisar elementos das posições j anteriores


while( j >=0  && vet[j] > aux ){
    vet[j+1] = vet[j];      // enquanto vet[j] > aux empurra
     j--;                   // vet[j] para frente
}


//Quando terminar o laço:
//    OU j == -1, significando que você empurrou v[0] para frente
//    OU vet[j] <= aux.
// De qualquer forma (j+1) é a posição correta para v[i]
 vet[j+1] = aux;
}
```

Insertion Sort

- Vamos apresentar uma forma alternativa de colocar **v[i]** na posição correta.
- Vamos supor que o vetor está ordenado de 0 até $i - 1$.
- Vamos inserir o elemento da posição i no lugar correto.

```
aux = vet[i];   //inserir aux na posição correta
j = i - 1; //analisar elementos das posições j anteriores

while( j >=0  && vet[j] > aux ){
    vet[j+1] = vet[j];      // enquanto vet[j] > aux empurra
     j--;                   // vet[j] para frente
}


//Quando terminar o laço:
//    OU j == -1, significando que você empurrou v[0] para frente
//    OU vet[j] <= aux.
// De qualquer forma (j+1) é a posição correta para v[i]
 vet[j+1] = aux;
}
```

Insertion Sort

- Vamos apresentar uma forma alternativa de colocar **v[i]** na posição correta.
- Vamos supor que o vetor está ordenado de 0 até $i - 1$.
- Vamos inserir o elemento da posição i no lugar correto.

```
aux = vet[i];   //inserir aux na posição correta
j = i - 1; //analisar elementos das posições j anteriores

while( j >=0  && vet[j] > aux ){
     vet[j+1] = vet[j];      // enquanto vet[j] > aux empurra
      j--;                   // vet[j] para frente
}

//Quando terminar o laço:
//    OU j == -1, significando que você empurrou v[0] para frente
//    OU vet[j] <= aux.
// De qualquer forma (j+1) é a posição correta para v[i]
 vet[j+1] = aux;
}
```

Insertion Sort

- Vamos apresentar uma forma alternativa de colocar **v[i]** na posição correta.
- Vamos supor que o vetor está ordenado de 0 até $i - 1$.
- Vamos inserir o elemento da posição i no lugar correto.

```
aux = vet[i];  //inserir aux na posição correta
j = i - 1; //analisar elementos das posições j anteriores

while( j >=0  && vet[j] > aux ){
     vet[j+1] = vet[j];     // enquanto vet[j] > aux empurra
      j--;                  // vet[j] para frente
}

//Quando terminar o laço:
//    OU j == -1, significando que você empurrou v[0] para frente
//    OU vet[j] <= aux.
// De qualquer forma (j+1) é a posição correta para v[i]
 vet[j+1] = aux;
}
```

Insertion Sort

Exemplo $(1, 3, 5, 10, 20, 2^*, 4)$ com $i = 5$.

$(1, 3, 5, 10, \underline{20}, 2, 4) : aux = 2; j = 4;$

$(1, 3, 5, \underline{10}, 20, 20, 4) : aux = 2; j = 3;$

$(1, 3, \underline{5}, 10, 10, 20, 4) : aux = 2; j = 2;$

$(1, \underline{3}, 5, 5, 10, 20, 4) : aux = 2; j = 1;$

$(\underline{1}, 3, 3, 5, 10, 20, 4) : aux = 2; j = 0;$

Aqui temos que $vet[j] < aux$ logo fazemos $vet[j + 1] = aux$

$(1, 2, 3, 5, 10, 20, 4) : aux = 2; j = 0;$

Insertion Sort

Exemplo $(1, 3, 5, 10, 20, 2^*, 4)$ com $i = 5$.

$(1, 3, 5, 10, \underline{20}, 2, 4) : aux = 2; j = 4;$

$(1, 3, 5, \underline{10}, 20, 20, 4) : aux = 2; j = 3;$

$(1, 3, \underline{5}, 10, 10, 20, 4) : aux = 2; j = 2;$

$(1, \underline{3}, 5, 5, 10, 20, 4) : aux = 2; j = 1;$

$(\underline{1}, 3, 3, 5, 10, 20, 4) : aux = 2; j = 0;$

Aqui temos que $vet[j] < aux$ logo fazemos $vet[j + 1] = aux$

$(1, 2, 3, 5, 10, 20, 4) : aux = 2; j = 0;$

Insertion Sort

Exemplo $(1, 3, 5, 10, 20, 2^*, 4)$ com $i = 5$.
$(1, 3, 5, 10, \underline{20}, 2, 4) : aux = 2; j = 4;$
$(1, 3, 5, \underline{10}, 20, 20, 4) : aux = 2; j = 3;$
$(1, 3, \underline{5}, 10, 10, 20, 4) : aux = 2; j = 2;$
$(1, \underline{3}, 5, 5, 10, 20, 4) : aux = 2; j = 1;$
$(\underline{1}, 3, 3, 5, 10, 20, 4) : aux = 2; j = 0;$
Aqui temos que $vet[j] < aux$ logo fazemos $vet[j + 1] = aux$
$(1, 2, 3, 5, 10, 20, 4) : aux = 2; j = 0;$

Insertion Sort

Exemplo $(1, 3, 5, 10, 20, 2^*, 4)$ com $i = 5$.

$(1, 3, 5, 10, \underline{20}, 2, 4)$: $aux = 2; j = 4;$

$(1, 3, 5, \underline{10}, 20, 20, 4)$: $aux = 2; j = 3;$

$(1, 3, \underline{5}, 10, 10, 20, 4)$: $aux = 2; j = 2;$

$(1, \underline{3}, 5, 5, 10, 20, 4)$: $aux = 2; j = 1;$

$(\underline{1}, 3, 3, 5, 10, 20, 4)$: $aux = 2; j = 0;$

Aqui temos que $vet[j] < aux$ logo fazemos $vet[j + 1] = aux$

$(1, 2, 3, 5, 10, 20, 4)$: $aux = 2; j = 0;$

Insertion Sort

Exemplo $(1, 3, 5, 10, 20, 2^*, 4)$ com $i = 5$.

$(1, 3, 5, 10, \underline{20}, 2, 4)$: $aux = 2; j = 4;$

$(1, 3, 5, \underline{10}, 20, 20, 4)$: $aux = 2; j = 3;$

$(1, 3, \underline{5}, 10, 10, 20, 4)$: $aux = 2; j = 2;$

$(1, \underline{3}, 5, 5, 10, 20, 4)$: $aux = 2; j = 1;$

$(\underline{1}, 3, 3, 5, 10, 20, 4)$: $aux = 2; j = 0;$

Aqui temos que $vet[j] < aux$ logo fazemos $vet[j + 1] = aux$

$(1, 2, 3, 5, 10, 20, 4)$: $aux = 2; j = 0;$

Insertion Sort

Exemplo $(1, 3, 5, 10, 20, 2^*, 4)$ com $i = 5$.
$(1, 3, 5, 10, \underline{20}, 2, 4) : aux = 2; j = 4;$
$(1, 3, 5, \underline{10}, 20, 20, 4) : aux = 2; j = 3;$
$(1, 3, \underline{5}, 10, 10, 20, 4) : aux = 2; j = 2;$
$(1, \underline{3}, 5, 5, 10, 20, 4) : aux = 2; j = 1;$
$(\underline{1}, 3, 3, 5, 10, 20, 4) : aux = 2; j = 0;$
Aqui temos que $vet[j] < aux$ logo fazemos $vet[j + 1] = aux$
$(1, 2, 3, 5, 10, 20, 4) : aux = 2; j = 0;$

Insertion Sort

Exemplo $(1, 3, 5, 10, 20, 2^*, 4)$ com $i = 5$.
$(1, 3, 5, 10, \underline{20}, 2, 4) : aux = 2; j = 4;$
$(1, 3, 5, \underline{10}, 20, 20, 4) : aux = 2; j = 3;$
$(1, 3, \underline{5}, 10, 10, 20, 4) : aux = 2; j = 2;$
$(1, \underline{3}, 5, 5, 10, 20, 4) : aux = 2; j = 1;$
$(\underline{1}, 3, 3, 5, 10, 20, 4) : aux = 2; j = 0;$
Aqui temos que $vet[j] < aux$ logo fazemos $vet[j + 1] = aux$
$(1, 2, 3, 5, 10, 20, 4) : aux = 2; j = 0;$

Insertion Sort

Código completo.

```
int vet[], tam;
int i,j, aux;

for(i=1; i<tam; i++){ //Assume vetor ordenado de 0 ate i-1

  aux = vet[i];
  j=i-1;

  while(j>=0 && vet[j] > aux){ //Poe elementos v[j]>v[i]
    vet[j+1] = vet[j];         //para frente
    j--;
  }
  vet[j+1] = aux; //poe v[i] na pos. correta
 }
}
```

Insertion Sort

Código completo.

```
int vet[], tam;
int i,j, aux;

for(i=1; i<tam; i++){ //Assume vetor ordenado de 0 ate i-1

    aux = vet[i];
    j=i-1;

    while(j>=0 && vet[j] > aux){ //Poe elementos v[j]>v[i]
      vet[j+1] = vet[j];         //para frente
      j--;
    }
    vet[j+1] = aux; //poe v[i] na pos. correta
  }
}
```

Insertion Sort

Código completo.

```
int vet[], tam;
int i,j, aux;

for(i=1; i<tam; i++){ //Assume vetor ordenado de 0 ate i-1

  aux = vet[i];
  j=i-1;

  while(j>=0 && vet[j] > aux){ //Poe elementos v[j]>v[i]
    vet[j+1] = vet[j];         //para frente
    j--;
  }
  vet[j+1] = aux; //poe v[i] na pos. correta
 }
}
```

Insertion Sort

Código completo.

```
    int vet[], tam;
    int i,j, aux;

    for(i=1; i<tam; i++){ //Assume vetor ordenado de 0 ate i-1

      aux = vet[i];
      j=i-1;

      while(j>=0 && vet[j] > aux){ //Poe elementos v[j]>v[i]
        vet[j+1] = vet[j];         //para frente
        j--;
      }
      vet[j+1] = aux; //poe v[i] na pos. correta
    }
}
```

Exercício

Altere os algoritmos vistos nesta aula para que estes ordenem um vetor de inteiros em ordem decrescente ao invés de ordem crescente.