Ordenação – BubbleSort e InsertionSort

Instituto de Computação – Unicamp

10 de Maio de 2012

Roteiro

Ordenação

- Continuamos com o estudo de algoritmos para o problema de ordenação:

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

- Novamente usaremos um vetor de inteiros como exemplo de coleção a ser ordenada.

Bubble-Sort

- Seja **vet** um vetor contendo números inteiros.
- Devemos deixar **vet** em ordem crescente.
- O algoritmos 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 algoritmos 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 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

- 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 = vet[j];
       vet[j] = vet[j+1];
       vet[j+1] = aux;
   }
```


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 = vet[j];
       vet[j] = vet[j+1];
       vet[j+1] = aux;
   }
```

Bubble-Sort

```
void bubbleSort(int vet[], int tam){
   int i,j, aux;

    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 = vet[j];
              vet[j] = vet[j+1];
              vet[j+1] = aux;
          }
    }
}
```


Bubble-Sort

```
void bubbleSort(int vet[], int tam){
   int i,j, aux;

    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 = vet[j];
              vet[j] = vet[j+1];
              vet[j+1] = aux;
          }
    }
}
```


Bubble-Sort

```
void bubbleSort(int vet[], int tam){
   int i,j, aux;

    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 = vet[j];
              vet[j] = vet[j+1];
              vet[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.

```
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
}

vet[ j+1 ] = aux; // (j+1) é posição correta para vet[i]
}
```


Insertion-Sort

- 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
}

vet[ j+1 ] = aux; // (j+1) é posição correta para vet[i]
}
```

(Instituto de Computação – Unicamp) MC-102 — Aula 20 10 de Maio de 2012 12 / 18

Insertion-Sort

- 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
}

vet[ j+1 ] = aux; // (j+1) é posição correta para vet[i]
}
```

(Instituto de Computação – Unicamp) MC-102 — Aula 20 10 de Maio de 2012 12 / 18

Insertion-Sort

- 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
}

vet[ j+1 ] = aux; // (j+1) é posição correta para vet[i]
}
```

(Instituto de Computação – Unicamp) MC-102 — Aula 20 10 de Maio de 2012 12 / 18

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

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

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

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

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

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

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


```
void insertionSort(int vet[], int tam){
   int i,j, aux;

   for(i=1; i<tam; i++){

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

      while( (j>=0) && (vet[j] > aux)  ){
         vet[j+1] = vet[j];
         j--;
      }
      vet[j+1] = aux;
   }
}
```

```
void insertionSort(int vet[], int tam){
   int i,j, aux;

   for(i=1; i<tam; i++){

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

      while( (j>=0) && (vet[j] > aux)  ){
         vet[j+1] = vet[j];
         j--;
      }
      vet[j+1] = aux;
   }
}
```

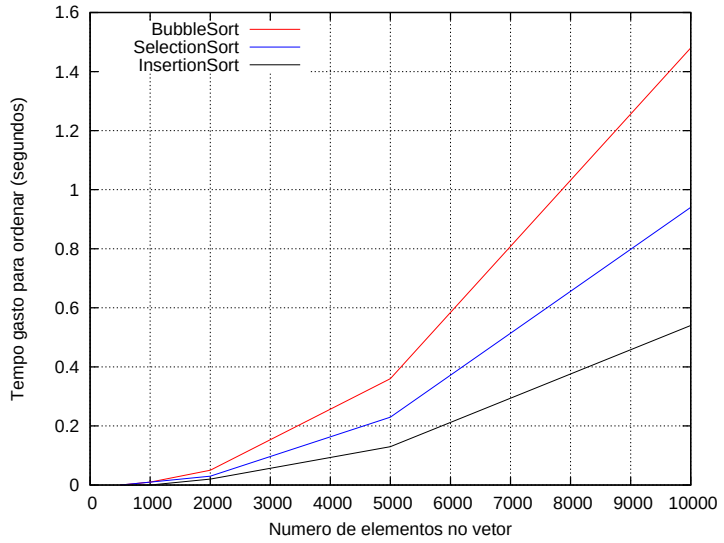


```
void insertionSort(int vet[], int tam){
   int i,j, aux;

   for(i=1; i<tam; i++){

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

      while( (j>=0) && (vet[j] > aux)  ){
         vet[j+1] = vet[j];
         j--;
      }
      vet[j+1] = aux;
   }
}
```

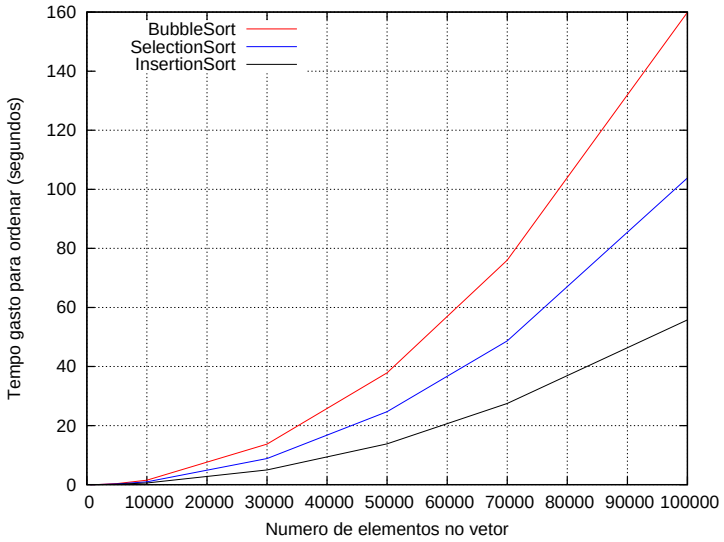


```
void insertionSort(int vet[], int tam){
   int i,j, aux;

   for(i=1; i<tam; i++){

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

      while( (j>=0) && (vet[j] > aux)  ){
         vet[j+1] = vet[j];
         j--;
      }
      vet[j+1] = aux;
   }
}
```


Comparação dos métodos estudados



Comparação dos métodos estudados



Pegando o tempo de execução de um programa...

http://en.cppreference.com/w/c/chrono/clock

No seu programa colocar:

```
#include <time.h>
int main(){
    clock_t start = clock(); //<- antes de qualquer comando


    double time = 0;
    todo seu código

    //antes do return
    time = ((double)clock() - start) / CLOCKS_PER_SEC)
    return(0)

}
```


Mais sobre ordenação...

http://pt.wikipedia.org/wiki/Algoritmo_de_ordenação