

MC-102 — Aula 22

Ordenação – Selection Sort e Bubble Sort

Instituto de Computação – Unicamp

16 de Maio de 2016

Roteiro

1 O problema da Ordenação

2 Selection Sort

3 BubbleSort

4 Exercício

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;  
}
```

Criamos então uma função que retorna o índice do elemento mínimo de um vetor, a partir de uma posição **ini** passada por parâmetro:

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

Selection-Sort

- Dada a função 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

```
void selectionSort(int vet[], int tam){  
  
    int i, min, aux;  
  
    for(i=0; i<tam; i++){  
        min = indiceMenor(vet, tam, i); //Acha posicao do menor elemento a partir de i  
        aux = vet[i];  
        vet[i] = vet[min];  
        vet[min] = aux;  
    }  
}
```

Selection-Sort

```
void selectionSort(int vet[], int tam){  
  
    int i, min, aux;  
  
    for(i=0; i<tam; i++){  
        min = indiceMenor(vet, tam, i); //Acha posicao do menor elemento a partir de i  
        aux = vet[i];  
        vet[i] = vet[min];  
        vet[min] = aux;  
    }  
}
```


Selection-Sort

```
void selectionSort(int vet[], int tam){  
  
    int i, min, aux;  
  
    for(i=0; i<tam; i++){  
        min = indiceMenor(vet, tam, i); //Acha posicao do menor elemento a partir de i  
        aux = vet[i];  
        vet[i] = vet[min];  
        vet[min] = aux;  
    }  
}
```

Selection-Sort

Com as funções anteriores implementadas podemos executar o exemplo:

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

    selectionSort(vetor,10);

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

    return 0;
}
```

Selection-Sort

- Uma operação muito comum nos algoritmos de ordenação é a troca do conteúdo entre duas variáveis do vetor.
- No algoritmo Selection Sort temos:

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

- Vamos criar uma função que faz a troca do conteúdo de duas variáveis inteiras:

```
void troca(int *a, int *b){  
    int aux;  
    aux = *a;  
    *a = *b;  
    *b = aux;  
}
```

Selection-Sort

Podemos então alterar o código do Selection Sort de

```
void selectionSort(int vet[], int tam){
    int i, min, aux;
    for(i=0; i<tam; i++){
        min = indiceMenor(vet, tam, i); //Acha posicao do menor elemento a partir de i
        aux = vet[i];
        vet[i] = vet[min];
        vet[min] = aux;
    }
}
```

para

```
void selectionSort2_1(int vet[], int tam){
    int i, min;
    for(i=0; i<tam; i++){
        min = indiceMenor(vet, tam, i); //Acha posicao do menor elemento a partir de i
        troca(&vet[i], &vet[min]);
    }
}
```

Selection-Sort

- O uso da função para achar o índice do menor elemento não é estritamente necessário.
- Podemos refazer a função selectionSort como segue:

```
for(i=0; i<tam; i++){  
    min = i;  
    for(j = i+1; j<tam; j++){  
        if(vet[min] > vet[j])  
            min = j;  
    }  
    troca(&vet[i], &vet[min]);  
}
```

Selection-Sort

- O uso da função para achar o índice do menor elemento não é estritamente necessário.
- Podemos refazer a função selectionSort como segue:

```
for(i=0; i<tam; i++){  
    min = i;  
    for(j = i+1; j<tam; j++){  
        if(vet[min] > vet[j])  
            min = j;  
    }  
    troca(&vet[i], &vet[min]);  
}
```

Selection-Sort

Antes:

```
void selectionSort(int vet[], int tam){
    int i, min;
    for(i=0; i<tam; i++){
        min = indiceMenor(vet, tam, i);
        troca(&vet[i], &vet[min]);
    }
}
```

Depois:

```
void selectionSort(int vet[], int tam){
    int i, j, min;
    for(i=0; i<tam; i++){
        min = i;
        for(j = i+1; j<tam; j++){//Acha posicao do menor elemento a partir de i
            if(vet[min] > vet[j])
                min = j;
        }
        troca(&vet[i], &vet[min]);
    }
}
```

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:

(5, 3, 2, 1, 90, 6)

(3, 5, 2, 1, 90, 6)

(3, 2, 5, 1, 90, 6)

(3, 2, 1, 5, 90, 6)

(3, 2, 1, 5, 90, 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:

(5, 3, 2, 1, 90, 6)

(3, 5, 2, 1, 90, 6)

(3, 2, 5, 1, 90, 6)

(3, 2, 1, 5, 90, 6)

(3, 2, 1, 5, 90, 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:

(5, 3, 2, 1, 90, 6)

(3, 5, 2, 1, 90, 6)

(3, 2, 5, 1, 90, 6)

(3, 2, 1, 5, 90, 6)

(3, 2, 1, 5, 90, 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:

(5, 3, 2, 1, 90, 6)

(3, 5, 2, 1, 90, 6)

(3, 2, 5, 1, 90, 6)

(3, 2, 1, 5, 90, 6)

(3, 2, 1, 5, 90, 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:

(5, 3, 2, 1, 90, 6)

(3, 5, 2, 1, 90, 6)

(3, 2, 5, 1, 90, 6)

(3, 2, 1, 5, 90, 6)

(3, 2, 1, 5, 90, 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:

(5, 3, 2, 1, 90, 6)

(3, 5, 2, 1, 90, 6)

(3, 2, 5, 1, 90, 6)

(3, 2, 1, 5, 90, 6)

(3, 2, 1, 5, 90, 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:

(5, 3, 2, 1, 90, 6)

(3, 5, 2, 1, 90, 6)

(3, 2, 5, 1, 90, 6)

(3, 2, 1, 5, 90, 6)

(3, 2, 1, 5, 90, 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:

(5, 3, 2, 1, 90, 6)

(3, 5, 2, 1, 90, 6)

(3, 2, 5, 1, 90, 6)

(3, 2, 1, 5, 90, 6)

(3, 2, 1, 5, 90, 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] )  
        troca(&vet[j], &vet[j+1])
```

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] )  
        troca(&vet[j], &vet[j+1])
```

Bubble-Sort

```
void bubbleSort(int vet[], int tam){
    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] )
                troca(&vet[j], &vet[j+1]);
    }
}
```

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?

Exercício

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

Exercício

No algoritmo SelectionSort, o laço principal é executado de $i=0$ até $i=tam-2$ e não $i=tam-1$. Por que?

```
void selectionSort2(int vet[], int tam){
    int i, j, min;
    for(i=0; i<tam-1; i++){condição não precisa ser i < tam. Por que?
        min = i;
        for(j = i+1; j<tam; j++){
            if(vet[min] > vet[j])
                min = j;
        }
        troca(&vet[i], &vet[min]);
    }
}
```