Ordenação – SelectionSort

Instituto de Computação – Unicamp

27 de Abril de 2012

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 como a 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).
Após a iteração 0: (1,3,2,5,90,6).
Após a iteração 1: (1,2,3,5,90,6).
Após a iteração 2: (1,2,3,5,90,6).
Após a iteração 3: (1,2,3,5,90,6).
Após a iteração 4: (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:

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

Criamo então uma função que retorna o índice do elemento mínimo de um vetor:

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

Selection-Sort

- Dado 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...

```
void selectionSort(int vet[], int tam){

  int i, min, aux;

  for(i=0; i<tam; i++){
    min = indiceMenor(vet, tam, i);
    aux = vet[i];
    vet[i] = vet[min];
    vet[min] = aux;
  }
}
```

```
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

- 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 elementos das posicoes min e i */
```

Antes:

```
void selectionSort(int vet[], int tam){

  int i, min, aux;

  for(i=0; i<tam; i++){
    min = indiceMenor(vet, tam, i);
    aux = vet[i];
    vet[i] = vet[min];
    vet[min] = aux;
  }
}
```

Depois:

```
void selectionSort2(int vet[], int tam){
  int i, j, min, aux;
  for(i=0; i<tam; 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;
  }
}
```

Exercício

O laço principal não precisa ir até o último elemento. Por que?

```
void selectionSort2(int vet[], int tam){
  int i, j, min, aux;
  for(i=0; i<tam-1; 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;
  }
}
```