Aula 26 – Métodos de Ordenação

Algoritmos e Programação de Computadores

Profs: Ronaldo Castro de Oliveira – ronaldo.co@ufu.br

Anilton Joaquim da Silva – anilton@ufu.br

Introdução

- Uma das aplicações mais estudadas e realizadas sobre arranjos é a ordenação. Ordenar um arranjo significa permutar seus elementos de tal forma que eles fiquem em ordem crescente, ou seja:

  v[0] <= v[1] <= v[2] <= : : : <= v[n-1].

- Por exemplo, suponha o vetor

  v = [5, 6, -9, 9, 0, 4]

- Uma ordenação desse vetor resultaria em um rearranjo de seus elementos:

  v = [-9, 0, 4, 5, 6, 9]

- Existem diversos algoritmos de ordenação para vetores. Eles variam em relação à dificuldade de implementação e desempenho. Usualmente algoritmos mais fáceis de serem implementados apresentam desempenho inferior. Veremos 3 algoritmos diferentes de ordenação:

  1. Algoritmo de Ordenação por Troca (Bubble Sort).
  2. Algoritmo de Inserção (Insertion Sort);
  3. Algoritmo de Seleção (Selection Sort); e

Ordenação por Troca (Bubble Sort)

- Algoritmo simples, muito útil para ordenação de vetores pequenos, mas não indicadopara vetores maiores devido ao seu baixo desempenho computacional. Idéia básica:
  - Compare o primeiro elemento definido como PIVO com o segundo elemento definido como RESTO = PIVO +1. Se estiverem desordenados, então efetue a troca de posição. Compare o PIVO com o terceiro elemento (incremente RESTO) e efetue a troca de posição, se necessário;
  - Repita a operação anterior até que seja comparado o PIVO com último elemento. Ao final desta repetição o elemento de menor valor estará na sua posição correta (PIVO);
  - Continue a ordenação incrementando o PIVO e executando o mesmo procedimento anterior até que PIVO seja igual a posição do numero de elementos N -1.

Ordenação por Troca (Bubble Sort)

P=0	R=1	2	3	4	5
10	9	8	7	6	5
P=0	**1**	**R=2**	**3**	**4**	**5**
9	10	8	7	6	5
P=0	**1**	**2**	**R=3**	**4**	**5**
8	10	9	7	6	5
P=0	**1**	**2**	**3**	**R=4**	**5**
7	10	9	8	6	5
P=0	**1**	**2**	**3**	**4**	**R=5**
6	10	9	8	7	5
P=0	**1**	**2**	**3**	**4**	**5**
5	10	9	8	7	6

Iteração 1

0	P=1	R=2	3	4	5
5	10	9	8	7	6
0	**P=1**	**2**	**R=3**	**4**	**5**
5	9	10	8	7	6
0	**P=1**	**2**	**3**	**R=4**	**5**
5	8	10	9	7	6
0	**P=1**	**2**	**3**	**4**	**R=5**
5	7	10	9	8	6
0	**1**	**2**	**3**	**4**	**5**
5	6	10	9	8	7

Iteração 2

0	1	P=2	R=3	4	5
5	6	10	9	8	7
0	**1**	**P=2**	**3**	**R=4**	**5**
5	6	9	10	8	7
0	**1**	**P=2**	**3**	**4**	**R=5**
5	6	8	10	9	7
0	**1**	**2**	**3**	**4**	**5**
5	6	7	10	9	8

Iteração 3

0	1	2	P=3	R=4	5
5	6	7	10	9	8
0	**1**	**2**	**P=3**	**4**	**R=5**
5	6	7	9	10	8
0	**1**	**2**	**3**	**4**	**5**
5	6	7	8	10	9

Iteração 4

0	1	2	3	P=4	R=5
5	6	7	8	10	9
0	**1**	**2**	**3**	**4**	**5**
5	6	7	8	9	10

Iteração 5

Ordenação por Troca (Bubble Sort)

```
void ordena_bolha(int V[ ], int N)
{
   int PIVO, RESTO, AUX;
   for (PIVO = 0; PIVO < N-1; PIVO++)
   {
      for (RESTO = PIVO+1; RESTO < N: RESTO++)
      {
          if (V[PIVO] > V[RESTO])
          {
              AUX = V[PIVO];
              V[PIVO] = V[RESTO];
              V[RESTO] = AUX
          }
      }
   }
}
```

Ordenação por Inserção

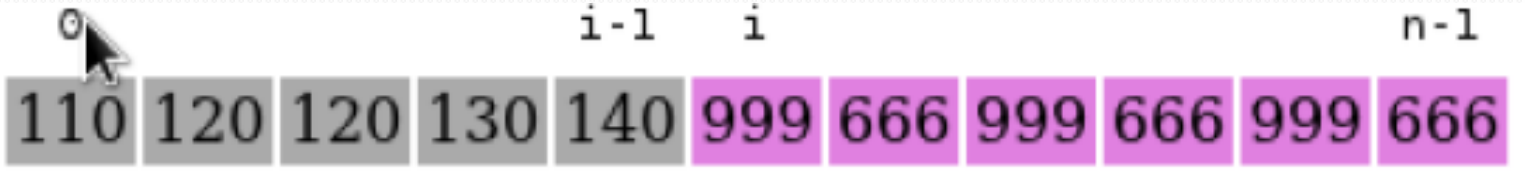
- Trata-se de um dos algoritmos de implementação mais simples. Seu método de ordenação semelhante ao que usamos para ordenar as cartas de um baralho. A idéia básica do algoritmo é descrita a seguir:
  - Compare a chave (x) com os elementos à sua esquerda, deslocando para direita cada elemento maior do que a chave;
  - Insira a chave na posição correta à sua esquerda, onde os elementos já estão ordenados;
  - Repita os passos anteriores atualizando a chave para a próxima posição à direita até o fim do vetor.

Ordenação por Inserção

```
void ordena_insercao(int v[], int n)
{
    int i, j, aux;
    for(i = 1; i < n; i++)
    {
            aux = v[i];
            j = i - 1;
            while( j >= 0 && v[ j ] > aux)
            {
                    v[ j+1 ] = v[ j ];
                    j --;
            }
            v[j+1] = aux;
    }
}
```

Ordenação por Seleção

- A implementação deste método de ordenação é muito simples. A idéia básica é descrita a seguir:
  - Selecione o menor elemento do vetor de tamanho n;
  - Troque esse elemento com o elemento da primeira posição do vetor;
  - Repita as duas operações anteriores considerando apenas os n-1 elementos restantes, em seguida repita com os n-2 elementos restantes; e assim sucessivamente até que reste apenas um elemento no vetor a ser considerado.



Ordenação por Seleção

```
void ordena_selecao(int v[ ], int n)
{
    int i, j, aux, min;
    for(i = 0; i < n-1; i++)
    {
            min = i;
            for(j = i+1; j < n; j++)
            {
                    if(v[ j ] < v[ min ])
                    {
                            min = j;
                    }
            }
            aux = v[ i ];
            v[ i ] = v[ min ];
            v[ min ] = aux;
            }
    }
```

Exercícios

1. Fazer um programa que leia um vetor de N posições de números float e imprima este vetor em ordem decrescente.
2. Fazer um programa que leia uma matriz de NxM elementos inteiros e imprima esta mesma matriz com cada uma das linhas ordenadas em ordem crescente.
3. Fazer um programa que leia o NOME e o TELEFONE de N pessoas. O programa deverá imprimir a lista de telefones com os nomes e os telefones em ordem alfabética dos nomes.