MC102 - Algoritmos e Progração de Computador

Prof. Alexandre Xavier Falcão

10° Aula: Ordenação de Vetores

1 Ordenação

Em muitas situações, tal como na busca binária, nós desejamos ordenar vetores. Nesta aula vamos aprender três algoritmos básicos de ordenação: ordenação por seleção, ordenação por inserção e ordenação por permutação. Suponha, por exemplo, que desejamos colocar em ordem crescente o vetor de 10 notas da aula anterior.

1.1 Ordenação por seleção

O algoritmo mais intuitivo é o de ordenação por seleção (*selection sort*).A idéia básica é percorrer o vetor várias vezes, selecionando o maior elemento, e trocando-o com o da última posição ainda não usada.

```
#include <stdio.h>

int main()
{
  float nota[10];
  int i,j,jm;

  printf("Entre com 10 notas\n");
  for (i=0; i < 10; i++) {
    scanf("%f",&nota[i]);
  }

  /* Coloque-as em ordem crescente por Seleção */

  for (j=9; j >= 2; j--){
  /* seleciona a maior nota entre j notas e troca-a com a da posição j */
    jm    = j;
    for (i=0; i < j; i++){
      if (nota[i] > nota[jm]){
        jm    = i;
      }
    }
```

```
  /* troca */
    if (j != jm){
      nota[j]  = nota[j] + nota[jm];
      nota[jm] = nota[j] - nota[jm];
      nota[j]  = nota[j] - nota[jm];
    }
  }

  /* imprime as notas ordenadas */

  for (i=0; i < 10; i++) {
    printf("%f ",nota[i]);
  }
  printf("\n");

  return 0;
}
```

Observe que no pior caso teremos $\frac{n(n-1)}{2}$ comparações entre notas e $n-1$ trocas, onde n é o tamanho do vetor. Dizemos então que o algoritmo executa em tempo proporcional ao quadrado do número de elementos e adotamos a notação $O(n^2)$. O processo de ordenação também não usa nenhum vetor auxiliar, portanto dizemos que a ordenação é *in place*. Esta observação se aplica aos outros métodos abaixo.

1.2 Ordenação por inserção

A ordenação por inserção (*insertion sort*) se assemelha ao processo que nós usamos para ordenar cartas de um baralho. A cada passo nós ordenamos parte da seqüência e a idéia para o passo seguinte é inserir a próxima nota na posição correta da seqüência já ordenada no passo anterior.

```
#include <stdio.h>

int main()
{
  float nota[10];
  int i,j;

  printf("Entre com 10 notas\n");
  for (i=0; i < 10; i++) {
    scanf("%f",&nota[i]);
  }

  /* Coloque-as em ordem crescente por Inserção */

  for (j=1; j < 10; j++){
    /* insere a nota da posição j na posição correta da
```

```
       seqüência já ordenada da posição 0 até j-1 */
    i = j;
    while ((i > 0)&&(nota[i] < nota[i-1])){
    /* troca as notas das posições i e i-1 */
      nota[i]   = nota[i] + nota[i-1];
      nota[i-1] = nota[i] - nota[i-1];
      nota[i]   = nota[i] - nota[i-1];
      i--;
    }
  }

  /* imprime as notas ordenadas */

  for (i=0; i < 10; i++) {
    printf("%f ",nota[i]);
  }
  printf("\n");

  return 0;
}
```

Observe que no pior caso, o número de comparações e de trocas é $\frac{n(n-1)}{2}$, onde n é o tamanho do vetor. Portanto, o algoritmo é também $O(n^2)$, apesar do número maior de trocas.

1.3 Ordenação por permutação

A idéia básica é simular o processo de ebolição de uma bolha de ar dentro d'água (*bubble sort*). A cada passo, a maior nota, que representa a bolha, deve subir até a superfície. Ou seja, trocas são executadas do início para o final do vetor, até que a maior nota fique na última posição ainda não usada.

```
#include <stdio.h>

int main()
{
  float nota[10];
  int i,j;

  printf("Entre com 10 notas\n");
  for (i=0; i < 10; i++) {
    scanf("%f",&nota[i]);
  }

  /* Coloque-as em ordem crescente por Permutação */

  for (j=9; j > 0; j--){
    /* seleciona a maior nota entre cada par de notas consecutivas,
```

```
      faz a troca se necessário, até que a maior nota seja inserida
      na posição j+1 */
    for (i=0; i < j; i++) {
      if (nota[i] > nota[i+1]){ /* troca as notas */
        nota[i]   = nota[i+1] + nota[i];
        nota[i+1] = nota[i]   - nota[i+1];
        nota[i]   = nota[i]   - nota[i+1];
      }
    }
  }

  /* imprime as notas ordenadas */

  for (i=0; i < 10; i++) {
    printf("%f ",nota[i]);
  }
  printf("\n");

  return 0;
}
```

No pior caso, temos também $\frac{n(n-1)}{2}$ comparações e trocas, onde n é o tamanho do vetor. Portanto, o algoritmo é $O(n^2)$ com número de trocas equivalente ao da ordenação por inserção.

2 Exercícios

1. Refaça o programa de busca binária usando um dos algoritmos acima para ordenar o vetor.
2. Faça um programa que ler dois vetores ordenados em ordem crescente, com números n_1 e n_2 de elementos diferentes, e gera um terceiro vetor mantendo a ordem por intercalação de valores dos outros dois.