AULA 5

ORDENAÇÃO: MÉTODOS ELEMENTARES

Além da busca, a ordenação é outra operação elementar em computação. Nesta aula revisaremos os métodos de ordenação elementares que já tivemos contato em aulas anteriores: o método das trocas sucessivas, da seleção e da inserção. Esses métodos foram projetados a partir de idéias simples e têm, como característica comum, tempo de execução de pior caso quadrático no tamanho da entrada. A aula é baseada no livro de P. Feofiloff [2].

5.1 Problema da ordenação

Lembrando da aula 4, um vetor $v[0..n-1]$ é **crescente** se $v[0] \leqslant v[1] \leqslant \cdots \leqslant v[n-1]$. O problema da ordenação de um vetor consiste em rearranjar, ou permutar, os elementos de um vetor $v[0..n-1]$ de tal forma que se torne crescente. Nesta aula nós discutimos três funções simples para solução desse problema. Nas aulas 5 e 6 examinaremos funções mais sofisticadas e eficientes.

5.2 Método das trocas sucessivas

O método das trocas sucessivas, popularmente conhecido como método da bolha, é um método simples de ordenação que, a cada passo, posiciona o maior elemento de um subconjunto de elementos do vetor de entrada na sua localização correta neste vetor. A função `trocas_sucessivas` implementa esse método.

```
/* Recebe um número inteiro n >= 0 e um vetor v de números inteiros
   com n elementos e rearranja o vetor v de modo que fique crescente */
void trocas_sucessivas(int n, int v[MAX])
{
   int i, j;

   for (i = n - 1; i > 0; i--)
      for (j = 0; j < i; j++)
         if (v[j] > v[j+1])
            troca(&v[j], &v[j+1]);

}
```

Um exemplo de chamada da função `trocas_sucessivas` é dado a seguir:

```
trocas_sucessivas(n, v);
```

Para entender a função `trocas_sucessivas` basta observar que no início de cada repetição do `for` externo vale que:

- o vetor $v[0..n-1]$ é uma permutação do vetor original,
- o vetor $v[i+1..n-1]$ é crescente e
- $v[j] \leqslant v[i+1]$ para $j = 0, 1, \ldots, i$.

Além disso, o consumo de tempo da função `trocas_sucesivas` é proporcional ao número de execuções da comparação "$v[j] > v[j+1]$", que é proporcional a n^2 no pior caso.

5.3 Método da seleção

O método de ordenação por seleção é baseado na idéia de escolher um menor elemento do vetor, depois um segundo menor elemento e assim por diante. A função `selecao` implementa esse método.

```
/* Recebe um número inteiro n >= 0 e um vetor v de números inteiros
   com n elementos e rearranja o vetor v de modo que fique crescente */
void selecao(int n, int v[MAX])
{
   int i, j, min;

   for (i = 0; i < n - 1; i++) {
      min = i;
      for (j = i+1; j < n; j++)
         if (v[j] < v[min])
            min = j;
      troca(&v[i], &v[min]);
   }

}
```

Um exemplo de chamada da função `selecao` é dado a seguir:

```
selecao(n, v);
```

Para entender como e por que o a função `selecao` funciona, basta observar que no início de cada repetição do `for` externo valem os seguintes invariantes:

- o vetor $v[0..n-1]$ é uma permutação do vetor original,
- o vetor $v[0..i-1]$ está em ordem crescente e

- $v[i-1] \leqslant v[j]$ para $j = i, i+1, \ldots, n-1$.

O terceiro invariante pode ser assim interpretado: $v[0..i-1]$ contém todos os elementos "pequenos" do vetor original e $v[i..n-1]$ contém todos os elementos "grandes". Os três invariantes garantem que no início de cada iteração os elementos $v[0], \ldots, v[i-1]$ já estão em suas posições definitivas.

Uma análise semelhante à que fizemos para a função `trocas_sucessivas` mostra que o método da inserção implementado pela função `selecao` consome n^2 unidades de tempo no pior caso.

5.4 Método da inserção

O método da ordenação por inserção é muito popular. É freqüentemente usado quando alguém joga baralho e quer manter as cartas de sua mão em ordem. A função `insercao` implementa esse método.

```
/* Recebe um número inteiro n >= 0 e um vetor v de números inteiros
   com n elementos e rearranja o vetor v de modo que fique crescente */
void insercao(int n, int v[MAX])
{
   int i, j, x;

   for (i = 1; i < n; i++) {
      x = v[i];
      for (j = i - 1; j >= 0 && v[j] > x; j--)
         v[j+1] = v[j];
      v[j+1] = x;
   }

}
```

Um exemplo de chamada da função `insercao` é dado a seguir:

```
insercao(n, v);
```

Para entender a função `insercao` basta observar que no início de cada repetição do `for` externo, valem os seguintes invariantes:

- o vetor $v[0..n-1]$ é uma permutação do vetor original e
- o vetor $v[0..i-1]$ é crescente.

O consumo de tempo da função `insercao` é proporcional ao número de execuções da comparação "$v[j] > x$", que é proporcional a n^2.

Exercícios

5.1 Escreva uma função que verifique se um dado vetor $v[0..n-1]$ é crescente.

```
int verifica_ordem(int n, int v[MAX])
{

   for (i = 0; i < n - 1; i++)
      if (v[i] > v[i+1])
         return 0;
   return 1;
}
```

5.2 Que acontece se trocarmos a realação `i > 0` pela relação `i >= 0` no código da função `trocas_sucessivas`? Que acontece se trocarmos `j < i` por `j <= i`?

5.3 Troque a relação `v[j] > v[j + 1]` pela relação `v[j] >= v[j + 1]` no código da função `trocas_sucessivas`. A nova função continua produzindo uma ordenação crescente de $v[0..n-1]$?

5.4 Escreva uma versão recursiva do método de ordenação por trocas sucessivas.

5.5 Que acontece se trocarmos `i = 0` por `i = 1` no código da função `selecao`? Que acontece se trocarmos `i < n-1` por `i < n`?

5.6 Troque `v[j] < v[min]` por `v[j] <= v[min]` no código da função `selecao`. A nova função continua produzindo uma ordenação crescente de $v[0..n-1]$?

5.7 Escreva uma versão recursiva do método de ordenação por seleção.

5.8 No código da função `insercao`, troque `v[j] > x` por `v[j] >= x`. A nova função continua produzindo uma ordenação crescente de $v[0..n-1]$?

5.9 No código da função `insercao`, que acontece se trocarmos `i = 1` por `i = 0`? Que acontece se trocarmos `v[j+1] = x` por `v[j] = x`?

5.10 Escreva uma versão recursiva do método de ordenação por inserção.

5.11 Escreva uma função que rearranje um vetor $v[0..n-1]$ de modo que ele fique em ordem estritamente crescente.

5.12 Escreva uma função que permute os elementos de um vetor $v[0..n-1]$ de modo que eles fiquem em ordem decrescente.