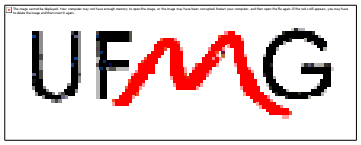

Ordenação

Critério de Ordenação

- Ordena-se de acordo com uma chave:

```
typedef int ChaveTipo;

typedef struct
{
  ChaveTipo Chave;
  /* outros componentes */
} Item;
```

Características

- Estabilidade: relativo à manutenção da ordem original de itens de chaves iguais
  - Um método de ordenação é **estável** se a ordem relativa dos itens com chaves iguais não se altera durante a ordenação.
- Ordenação interna: arquivo a ser ordenado cabe todo na memória principal.
- Princípio: comparação x distribuição

Critério de Avaliação

- Sendo n o número registros no arquivo, as medidas de complexidade relevantes são:
  - Número de comparações C(n) entre chaves.
  - Número de movimentações M(n) de itens

Outras Considerações

- O uso econômico da memória disponível é um requisito primordial na ordenação interna.
- Métodos de ordenação ***in situ*** são os preferidos.
- Métodos que utilizam listas encadeadas não são muito utilizados.
- Métodos que fazem cópias dos itens a serem ordenados possuem menor importância.

Software de Visualização

- Bolha
- Seleção
- Inserção

Método da Bolha

```
int Bolha(int *v, int n)
{
    int i, j, aux;

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

Análise de Complexidade

- Comparações – C(n)

- Movimentações – M(n)

Análise de Complexidade

- Comparações - C(n)

$$C(n) = \sum_{i=2}^{n} (i-1) = \sum_{i=1}^{n} (i-1) = \sum_{i=1}^{n} i - \sum_{i=1}^{n} 1$$

$$= \frac{n(n+1)}{2} - n = \frac{n^2 - n}{2}$$

- Movimentações - M (n)

M(n) = 3C(n)

Ordenação por Bolha

- Vantagens:
  - Algoritmo simples
  - Algoritmo estável
- Desvantagens:
  - O fato de o arquivo já estar ordenado não ajuda em nada, pois o custo continua quadrático.

Ordenação por Seleção

```
void Selecao (Item *A, Indice n)
{
  Indice i, j, Min;
  Item x;
  for (i = 1; i <= n - 1; i++){
    Min = i;
    for (j = i + 1; j <= n; j++)
      if (A[j].Chave < A[Min].Chave) Min = j;
    x = A[Min];
    A[Min] = A[i];
    A[i] = x;
  }
}
```

Análise de Complexidade

- Comparações – C(n)

- Movimentações – M(n)

Análise de Complexidade

- Comparações – C(n)

$$C(n) = n^2/2 - n/2$$

- Movimentações – M(n)

$$M(n) = 3(n - 1)$$

Ordenação por Seleção

- Vantagens:
  - Custo linear no tamanho da entrada para o número de movimentos de registros.
  - É o algoritmo a ser utilizado para arquivos com registros muito grandes.
  - É muito interessante para arquivos pequenos.
- Desvantagens:
  - O fato de o arquivo já estar ordenado não ajuda em nada, pois o custo continua quadrático.
  - O algoritmo não é **estável.**

Ordenação por Inserção

```
void Insercao(Item *A, Indice n)
{
  Indice i, j;
  Item x;
  for (i = 2; i <= n; i++){
    x = A[i]; j = i -1;
    A[0] = x; /* sentinela */
    while (x.Chave < A[j].Chave)
    {
      A[j+1] = A[j];
      j--;
    }
    A[j+1] = x;
  }
}
```

Análise de Complexidade

- Comparações – C(n)

- Movimentações – M(n)

Análise de Complexidade

- Comparações – C(n)
  - No anel mais interno, na i-ésima iteração, o valor de C_i é:
    - melhor caso : $C_i(n) = 1$
    - pior caso : $C_i(n) = i$
    - caso medio : $Ci(n) = 1/i\ (1 + 2 + ... + i) = (i+1)/2$

    (assumindo que todas as permutações de n são igualmente prováveis no caso médio)
  - Considerando o loop externo temos:
    - melhor caso : $C(n) = (1 + 1 + ... + 1) = n - 1$
    - pior caso : $C(n) = (2 + 3 + ... + n) = n^2/2 + n/2 + 1$
    - caso medio : $C(n) = ½\ (3 + 4 + ... + n + 1) = n^2/4 + 3n/4 - 1$

Análise de Complexidade

- Movimentações – M(n)

$M_i(n) = C_i(n) - 1 + 3 = C_i(n) + 2$

  - Logo, o número de movimentos é:
    - melhor caso : $M(n) = (3 + 3 + ... + 3) = 3(n-1)$
    - pior caso : $M(n) = (4 + 5 + ... + n + 2) = n^2/2 + 5n/2 - 3$
    - caso medio : $M(n) = ½ (5 + 6 + \quad + n + 3) = n^2/4 + 11n/4 - 3$

Ordenação por Inserção

- O número mínimo de comparações e movimentos ocorre quando os itens estão originalmente em ordem.
- O número máximo ocorre quando os itens estão originalmente na ordem reversa.
- É o método a ser utilizado quando o arquivo está “quase” ordenado.
- É um bom método quando se deseja adicionar uns poucos itens a um arquivo ordenado, pois o custo é linear.
- O algoritmo de ordenação por inserção é **estável.**

Ordenação Interna

- Classificação dos métodos de ordenação interna:
  - Métodos simples:
    - Adequados para pequenos arquivos.
    - Requerem $O(n^2)$ comparações.
    - Produzem programas pequenos.
  - Métodos eficientes:
    - Adequados para arquivos maiores.
    - Requerem O(n log n) comparações.
    - Usam menos comparações.
    - As comparações são mais complexas nos detalhes.
    - Métodos simples são mais eficientes para pequenos arquivos.