Ordenação

Luciano Nascimento Moreira

Conceitos básicos

- Ordenar: processo de rearranjar um conjunto de objetos em uma ordem ascendente ou descendente.
- A ordenação visa facilitar a recuperação posterior de itens do conjunto ordenado.
  - Dificuldade de se utilizar um catálogo telefônico se os nomes das pessoas não estivessem listados em ordem alfabética.
- A maioria dos métodos de ordenação é baseada em comparações das chaves.
- Existem métodos de ordenação que utilizam o princípio da distribuição.

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.

Vetor	10	20	30	40	50	60	70	80	90
	R$ 100,00	R$ 100,00	R$ 200,00	R$ 400,00	R$ 500,00	R$ 600,00	R$ 600,00	R$ 500,00	R$ 400,00
Estável	10	20	30	40	90	50	80	60	70
	R$ 100,00	R$ 100,00	R$ 200,00	R$ 400,00	R$ 400,00	R$ 500,00	R$ 500,00	R$ 600,00	R$ 600,00
Instável	20	10	30	90	40	50	80	70	60
	R$ 100,00	R$ 100,00	R$ 200,00	R$ 400,00	R$ 400,00	R$ 500,00	R$ 500,00	R$ 600,00	R$ 600,00

Adaptabilidade

- Um método é adaptável quando a sequencia de operações realizadas depende da entrada
- Um método que sempre realiza as mesmas operações, independende da entrada, é não adaptável.

Características

- Ordenação interna
  - arquivo a ser ordenado cabe todo na memória principal
- Ordenação externa
  - arquivo a ser ordenado não cabe na memória principal
  - Dados armazenados em memória secundária (disco)
- Diferenças entre os métodos:
  - Em um método de ordenação interna, qualquer registro pode ser imediatamente acessado.
  - Em um método de ordenação externa, os registros são acessados sequencialmente ou em grandes blocos.

Ordenação Interna

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

Ordenação Interna

- 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.

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.

Conceitos básicos

- Qualquer tipo de chave sobre o qual exista uma regra de ordenação bem definida pode ser utilizado.
- Em Java pode-se criar um tipo de registro genérico chamado interface

```
public interface Item {
   public int compara ( Item i t ) ;
   public void alteraChave ( Object chave) ;
   public Object recuperaChave ( ) ;
}
```

Conceitos básicos

- A classe **MeuItem** define o tipo de dados int para a chave e implementa os métodos **compara**, **alteraChave** e **recuperaChave**.
- O método compara retorna um valor menor do que zero se a < b, um valor maior do que zero se a > b, e um valor igual a zero se a = b.

Conceitos básicos

```
public class MeuItem implements Item {
   private int chave;
   // outros componentes do registro

   public MeuItem(int chave) {
      this.chave = chave ;
   }

   public int compara(Item it) {
      MeuItem item = (MeuItem) it ;
      if (this.chave < item.chave)
         return -1;
      else if (this.chave > item.chave)
         return 1;
      return 0;
   }
```

Conceitos básicos

```
    public void alteraChave (Object chave) {
       Integer ch = (Integer) chave;
       this.chave = ch.intValue ( ) ;
    }

    public Object recuperaChave() {
       return new Integer(this.chave );
    }
}
```

Métodos de ordenação

Name	Best	Average	Worst	Memory	Stable	Method
Quicksort	$n \log n$	$n \log n$	n^2	$\log n$	Depends	Partitioning
Merge sort	$n \log n$	$n \log n$	$n \log n$	Depends; worst case is n	Yes	Merging
In-place Merge sort	—	—	$n \left(\log n\right)^2$	1	Yes	Merging
Heapsort	$n \log n$	$n \log n$	$n \log n$	1	No	Selection
Insertion sort	n	n^2	n^2	1	Yes	Insertion
Introsort	$n \log n$	$n \log n$	$n \log n$	$\log n$	No	Partitioning & Selection
Selection sort	n^2	n^2	n^2	1	No	Selection
Timsort	n	$n \log n$	$n \log n$	n	Yes	Insertion & Merging
Shell sort	n	$n(\log n)^2$ or $n^{3/2}$	Depends on gap sequence; best known is $n(\log n)^2$	1	No	Insertion
Bubble sort	n	n^2	n^2	1	Yes	Exchanging
Binary tree sort	n	$n \log n$	$n \log n$	n	Yes	Insertion
Cycle sort	—	n^2	n^2	1	No	Insertion
Library sort	—	$n \log n$	n^2	n	Yes	Insertion
Patience sorting	—	—	$n \log n$	n	No	Insertion & Selection
Smoothsort	n	$n \log n$	$n \log n$	1	No	Selection
Strand sort	n	n^2	n^2	n	Yes	Selection
Tournament sort	—	$n \log n$	$n \log n$	n[6]		Selection
Cocktail sort	n	n^2	n^2	1	Yes	Exchanging
Comb sort	n	$n \log n$	n^2	1	No	Exchanging
Gnome sort	n	n^2	n^2	1	Yes	Exchanging
Bogosort	n	$n \cdot n!$	$n \cdot n! \rightarrow \infty$	1	No	Luck

Métodos de ordenação

- Métodos Simples
  - Bolha (BubbleSort)
  - Seleção (SelectSort)
  - Inserção (InsertSort)
- Métodos Eficientes
  - Shellsort
  - Quicksort
  - Heapsort

Método Bolha

- Ideia
  - Passa no arquivo e troca elementos adjacentes que estão fora de ordem
  - Repete esse processo até que o arquivo esteja ordenado
- Algoritmo
  - Compara dois elementos adjacentes e troca de posição se estiverem fora de ordem
  - Quando o maior elemento do vetor for encontrado, ele será trocado até ocupar a última posição
  - Na segunda passada, o segundo maior será movido para a penúltima posição do vetor, e assim sucessivamente

Método Bolha

```
void bolha (Item v[], int n ) {
  Item aux;
  for(int i = 0; i < n-1; i++ )
    for(int j = 1; j < n-i; j++ )
      // v[j] < v[j-1]
      if (v[j].compara(v[j-1]) < 0) {
        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)

$$\sum_{i=0}^{n-2}\sum_{j=1}^{n-i-1} 1 = \sum_{i=0}^{n-2} n-i-1 = (n-1)+(n-2)+(n-3)+\cdots+2+1$$

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

- Movimentações – M(n)

$M(n) = 3C(n)$ Pior caso

Ordenação por Bolha

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

Método Seleção

- Seleção do n-ésimo menor (ou maior) elemento da lista
- Troca do n-ésimo menor (ou maior) elemento com a n-ésima posição da lista
- Repete até ter colocado todos os elementos em suas posições
- Uma única troca por vez é realizada

Método Seleção

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

Método da Inserção

	0	1	2	3	4	5
i	O	R	D	E	N	A
1	O	**R**	D	E	N	A
2	**D**	**O**	**R**	E	N	A
3	D	**E**	**O**	**R**	N	A
4	D	E	**N**	**O**	**R**	A
5	**A**	**D**	**E**	**N**	**O**	**R**

Análise de Complexidade

- Comparações – C(n)

- Movimentações – M(n)

Análise de Complexidade

- Comparações – C(n)

$$\sum_{i=0}^{n-2}\sum_{j=1}^{n-i-1} 1 = \sum_{i=0}^{n-2} n-i-1 = (n-1)+(n-2)+(n-3)+\cdots+2+1$$

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

- 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:
  - Não adaptável
    - Não importa se o arquivo está parcialmente ordenado, pois o custo continua quadrático.
  - O algoritmo não é estável.

Método da Inserção

- Algoritmo utilizado pelo jogador de cartas: Quando compra ou recebe uma nova carta, o jogador encontra qual posição ela deve ocupar em sua mão
  - As cartas são ordenadas da esquerda para direita uma por uma.
  - O jogador escolhe a segunda carta e verifica se ela deve ficar antes ou na posição que está.
  - Depois a terceira carta é classificada, deslocando-a até sua correta posição
  - O jogador realiza esse procedimento até ordenar todas as cartas
- Alto custo em remover uma carta de uma posição e colocá-la em outra quando a representação é por arranjos

Método da Inserção

- Implementação para vetores:
  - Mantemos os elementos entre zero e i-1 ordenados
    - Note que um vetor de 1 elemento está ordenado por definição
  - Achamos a posição do i-ésimo elemento e inserimos ele entre os i-1 que já estavam ordenados
  - O programa repete esse passo até ordenar todos os elementos

Método da Inserção

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

Método da Inserção

	0	1	2	3	4	5
i	O	R	D	E	N	A
1	O	**R**	D	E	N	A
2	**D**	**O**	**R**	E	N	A
3	D	**E**	**O**	**R**	N	A
4	D	E	**N**	**O**	**R**	A
5	**A**	**D**	**E**	**N**	**O**	**R**

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 médio : $C_i(n) = 1/i (1 + 2 + ... + i) = (i+1)/2$
  - Assumindo que todas as permutações de n são igualmente prováveis no caso médio, temos:
    - melhor caso: $C(n) = (1 + 1 + ... + 1) = n - 1$
    - pior caso : $C(n) = (2 + 3 + ... + n) = n^2/2 + n/2 - 1$
    - caso médio : $C(n) = ½ (3 + 4 + ... + n + 1) = n^2/4 + 3n/4 - 1$

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 médio : $M(n) = ½ (5 + 6 + \quad + n + 3)$
      $= n^2/4 + 11n/4 - 3$

Ordenação por Inserção

- O número mínimo de comparações e movimentos ocorre quando os itens estão originalmente em ordem.
- O número máximo comparações ocorre quando os itens estão originalmente na ordem reversa.
- Estável
- Adaptável
  - 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.