

Universidade Federal do Piauí
Campus SHNB – Picos -PI
Curso de Sistemas de Informação

Algoritmos e Programação II

Glauber Dias Gonçalves
gdgnew@gmail.com



Conteúdo

- Ordenação
 - Critérios de ordenações
 - Métodos elementares
 - Método da Bolha
 - Método da Seleção
 - Método da Inserção

Ordenação

- Objetivo:
 - Rearranjar os itens de um vetor (ou lista) de modo que suas chaves estejam ordenadas de acordo com alguma regra



- Pode-se pensar em uma estrutura para ordenações:

```
typedef int chave_t;  
struct item {  
    chave_t chave;  
    /* outros componentes */  
};
```

Critérios de Ordenação

- Localização dos dados
 - Ordenação interna
 - Todos os dados estão em memória principal (RAM)
 - Ordenação externa
 - Memória principal não cabe os dados
 - Dados armazenados em memória secundaria (disco)

Critérios de Ordenação

- Estabilidade
 - Método é estável se a ordem relativa dos registros com a mesma chave não se altera após a ordenação.

Adams	A
Black	B
Brown	D
Jackson	B
Jones	D
Smith	A
Thompson	D
Washington	B
White	C
Wilson	C

Adams	A
Smith	A
Washington	B
Jackson	B
Black	B
White	C
Wilson	C
Thompson	D
Brown	D
Jones	D

Adams	A
Smith	A
Black	B
Jackson	B
Washington	B
White	C
Wilson	C
Brown	D
Jones	D
Thompson	D

Critérios de Ordenação

- Adaptabilidade:
 - Um método é adaptável quando a sequencia de operações realizadas depende da entrada
 - Um método que sempre realiza as mesmas operações, independente da entrada, é não adaptável.
 - Exemplo:
 - Os dados de entrada já estão (quase) ordenados?

Critérios de Ordenação

- Uso da memória:
 - *In place*: ordena sem usar memória adicional ou usando uma quantidade constante de memória adicional
 - Alguns métodos precisam duplicar os dados para tornar a ordenação mais rápida

Critérios de Avaliação

- Sendo n o número de elementos em um vetor, utilizaremos as seguintes medidas de avaliação:
 - Número de comparações $C(n)$ entre chaves.
 - Número de movimentações $M(n)$ de itens

Critérios de Avaliação

- Número de Comparações

```
if (v[j].chave < v[j-1].chave)
    troca(v[j-1], v[j]);
```

Critérios de Avaliação

- Número de Movimentações

```
// struct item a;  
// struct item b;  
struct item aux = a;  
a = b;  
b = aux;
```

Método da Bolha

Método da Bolha

- Ideia
 - 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 (n): subvetor $[n]$ está “ordenado”
 - Na segunda passada, o segundo maior será movido para a penúltima posição do vetor ($n-1$): subvetor $[n-1, n]$ está ordenado
 - Sucessivas passagens pelo vetor até o menor elemento estar na posição 1: “subvetor” $[1, n]$ estará ordenado

Método da Bolha

- Ordenando o vetor 5, 4, 3, 2, 1

5	4	3	2	1
4	5	3	2	1
4	3	5	2	1
4	3	2	5	1
4	3	2	1	5

1ª. passagem

4	3	2	1	5
3	4	2	1	5
3	2	4	1	5
3	2	1	4	5

2ª. passagem

3	2	1	4	5
2	3	1	4	5
2	1	3	4	5

3ª. passagem

2	1	3	4	5
1	2	1	4	5

4ª. passagem

Método da Bolha

```
void bolha(struct item *v, int n) {  
    int i, j;  
    for(i = 0; i < n-1; i++) {  
        for(j = 1; j < n-i; j++) {  
            if(v[j].chave < v[j-1].chave)  
                troca(v[j-1], v[j]);  
        }  
    }  
}
```

Método da Bolha

- Comparações – $C(n)$
- Movimentações – $M(n)$

Método da Bolha

- Comparações - $C(n)$

$$\begin{aligned} C(n) &= \sum_{i=2}^n (i-1) = \sum_{i=1}^n (i-1) = \sum_{i=1}^n i - \sum_{i=1}^n 1 \\ &= \frac{n(n+1)}{2} - n = \frac{n^2 - n}{2} \end{aligned}$$

- Movimentações - $M(n)$

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

Método da Bolha

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

Método da Seleção

Método da Seleção

- Ideia
 - Procura o i -ésimo menor elemento do vetor
 - Troca do i -ésimo menor elemento com o elemento na i -ésima posição:
 - subvetor $[1, i]$ está ordenado
 - Repete até ter colocado todos os elementos em suas posições
 - Elementos são movimentados apenas uma vez

Método da Seleção

- Ordenando o vetor 5, 4, 3, 2, 1

5	4	3	2	1
5	4	3	2	1
5	4	3	2	1
5	4	3	2	1
5	4	3	2	1

1ª. passagem

1	4	3	2	5
1	4	3	2	5
1	4	3	2	5
1	4	3	2	5

2ª. passagem

1	2	3	4	5
1	2	3	4	5
1	2	3	4	5

3ª. passagem

1	2	3	4	5
1	2	3	4	5

4ª. passagem

Método da Seleção

```
void selecao(struct item *v, int n) {  
    int i, j, min;  
    for(i = 0; i < n - 1; i++) {  
        min = i;  
        for(j = i + 1 ; j < n; j++) {  
            if(v[j].chave < v[min].chave)  
                min = j;  
        }  
        troca(v[i], v[min]);  
    }  
}
```

Método da Seleção

- Comparações – $C(n)$
- Movimentações – $M(n)$

Método da Seleção

- Comparações – $C(n)$

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

- Movimentações – $M(n)$

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

Método da 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.
- Desvantagens:
 - O fato de o arquivo já estar ordenado não ajuda em nada, pois o custo continua quadrático.
 - O algoritmo não é estável.
 - **Que exemplo de entrada (chaves) pode demonstrar isso?**

Método da Inserção

Método da Inserção

- Ideia:
 - n cartas de baralho na mesa
 - cartas na mão esquerda estão ordenadas
 - mão direita pega uma carta e ordena



Método da Inserção

- Ordenando o vetor 5, 4, 3, 2, 1

1ª. passagem

5	4	3	2	1
5	4	3	2	1

2ª. passagem

4	5	3	2	1
4	5	3	2	1
4	3	5	2	1

3ª. passagem

3	4	5	2	1
3	4	5	2	1
3	4	2	5	1
3	2	4	5	1

4ª. passagem

2	3	4	5	1
2	3	4	5	1
2	3	4	1	5
2	3	1	4	5
2	1	3	4	5

Quadro cinza é a chave a ser ordenada (mão direita)

Quadros à esquerda do cinza são chaves ordenadas (mão esquerda)

Quadros à direita do cinza são chaves desordenadas (mesa)

Resultado:

1	2	3	4	5	
---	---	---	---	---	--

Método da Inserção

- Ordenando o vetor 1, 2, 3, 4, 5 (algoritmo adaptável?)

1ª. passagem

1	2	3	4	5
1	2	3	4	5

2ª. passagem

1	2	3	4	5
1	2	3	4	5

3ª. passagem

1	2	3	4	5
1	2	3	4	5

4ª. passagem

1	2	3	4	5
1	2	3	4	5

Quadro cinza é a chave a ser ordenada (mão direita)

Quadros à esquerda do cinza são chaves ordenadas (mão esquerda)

Quadros à direita do cinza são chaves desordenadas (mesa)

Resultado:

1	2	3	4	5	
---	---	---	---	---	--

Método da Inserção

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

Método da Inserção

- Número de comparações – $C(n)$
 - Melhor caso
 - Pior caso
 - Caso médio
- Número de movimentações – $M(n)$
 - $M(n) = C(n)$ (aproximadamente)

Custo da Inserção no Pior Caso

- Exemplo: $n=5$ chaves decrescentes: 5, 4, 3, 2, 1

Quadro cinza é a chave a ser ordenada e números em vermelho são as chaves comparadas

1ª. passagem

5	4	3	2	1
5	4	3	2	1

1

2ª. passagem

4	5	3	2	1
4	5	3	2	1
4	3	5	2	1

2

3ª. passagem

3	4	5	2	1
3	4	5	2	1
3	4	2	5	1
3	2	4	5	1

3

4ª. passagem

2	3	4	5	1
2	3	4	5	1
2	3	4	1	5
2	3	1	4	5
2	1	3	4	5

4

=

10 comparações

Resultado:

1	2	3	4	5
---	---	---	---	---

Custo da Inserção no Pior Caso

- Exemplo: $n=5$ chaves decrescentes: 5, 4, 3, 2, 1

Quadro cinza é a chave a ser ordenada e números em vermelho são as chaves comparadas

1ª. passagem

5	4	3	2	1
5	4	3	2	1

2ª. passagem

4	5	3	2	1
4	5	3	2	1
4	3	5	2	1

3ª. passagem

3	4	5	2	1
3	4	5	2	1
3	4	2	5	1
3	2	4	5	1

4ª. passagem

2	3	4	5	1
2	3	4	5	1
2	3	4	1	5
2	3	1	4	5
2	1	3	4	5

Resultado:

1	2	3	4	5
---	---	---	---	---

1 + 2 + 3 + 4 = 10 comparações

Generalizar o número de comparações para vetor ou lista de tamanho n :

$$S_n = n/2 * (a_1 + a_n)$$

Soma de Progressões aritméticas

$$C(n) = (n-1)/2 * (1 + n-1) = (n^2 - n)/2$$

Custo da Inserção no Melhor Caso

- Exemplo: $n=5$ chaves crescentes: 1, 2, 3, 4, 5 (algoritmo adaptável?)

Quadro cinza é a chave a ser ordenada e números em vermelho são as chaves comparadas

1ª. passagem

1	2	3	4	5
1	2	3	4	5

1

+

2ª. passagem

1	2	3	4	5
1	2	3	4	5

1

+

3ª. passagem

1	2	3	4	5
1	2	3	4	5

1

+

4ª. passagem

1	2	3	4	5
1	2	3	4	5

1

=

4 comparações

Resultado:

1	2	3	4	5
---	---	---	---	---

Generalizar o número de comparações para vetor ou lista de tamanho n :

$$C(n) = n - 1$$

Custo da Inserção no Caso Médio

- Generalizar número de comparações para vetor ou lista de tamanho n

$$\frac{\begin{array}{c} \text{pior caso} \\ (n^2 - n)/2 \end{array} + \begin{array}{c} \text{melhor caso} \\ (n-1) \end{array}}{2} = n^2/4 + n/4 - 1/2$$

Classe de custos quadráticos

$O(n^2)$

notação assintótica

Método da Inserção

- Vantagens:
 - Adequado para
 - ordenar vetores pequenos
 - quando o arquivo está “quase” ordenado
 - quando se deseja adicionar poucos itens a um arquivo ordenado, pois o custo é linear
 - O algoritmo de ordenação por inserção é estável
- Desvantagens:
 - Número de comparações tem crescimento quadrático
 - Alto custo de movimentação de elementos no vetor

Sumário

- Ordenação: rearranjar itens (chaves) em vetor ou lista para ordená-los de acordo com alguma regra
- Algoritmos de ordenação elementares (custo quadrático):
 - Algoritmo da Bolha – método mais simples
 - Algoritmo da Seleção – eficiente para movimentações
 - Algoritmo Inserção – ordenação adaptável
- Lista de atividades 07 e 08

Bibliografia

- ZIVIANI, N. Projeto de Algoritmos, 3 ed. Cengage Learning 2010 (Capítulo 4)
- CORMEN, T. H. Algoritmos: Teoria e Prática. 3 ed. Rio de Janeiro: Campus, 2012. (Capitulo 2)