ORDENAÇÃO DE ARQUIVOS

Vanessa Braganholo

Estruturas de Dados e Seus Algoritmos

BUSCA EM ARQUIVO BINÁRIO

Suponha que um banco mantém seus funcionários em um arquivo (mais de 10000 funcionários)

O banco deseja dar um aumento para o funcionário de código 305

Como encontrar o funcionário no arquivo?

ALTERNATIVA 1 – BUSCA SEQUENCIAL

Ler arquivo do início até encontrar o funcionário

- Muito custoso
- No pior caso (funcionário não existe ou é o último), lê o arquivo inteiro

BUSCA SEQUENCIAL POR FUNCIONÁRIO 305

Código	Nome	Salário
102	Joao Silva	1000
123	Carlos Albuquerque	1500
143	Ana Bueno	1500
200	Caio Gusmao	4000
239	Bianca Amarilo	3000
254	Arnaldo Souza	4300
305	Marisa Clara	5000
403	Bruno Simao	4500
410	Guilherme Santos	2000
502	Tatiana Andrade	2500

BUSCA SEQUENCIAL POR FUNCIONÁRIO 305

Código	Nome	Salário
102	Joao Silva	1000
123	Carlos Albuquerque	1500
143	Ana Bueno	1500
200	Caio Gusmao	4000
239	Bianca Amarilo	3000
254	Arnaldo Souza	4300
305	Marisa Clara	5000
403	Bruno Simao	4500
410	Guilherme Santos	2000
502	Tatiana Andrade	2500

BUSCA SEQUENCIAL POR FUNCIONÁRIO 305

Código	Nome	Salário
102	Joao Silva	1000
123	Carlos Albuquerque	1500
143	Ana Bueno	1500
200	Caio Gusmao	4000
239	Bianca Amarilo	3000
254	Arnaldo Souza	4300
305	Marisa Clara	5000
403	Bruno Simao	4500
410	Guilherme Santos	2000
502	Tatiana Andrade	2500

BUSCA SEQUENCIAL POR FUNCIONÁRIO 305

Código	Nome	Salário
102	Joao Silva	1000
123	Carlos Albuquerque	1500
143	Ana Bueno	1500
200	Caio Gusmao	4000
239	Bianca Amarilo	3000
254	Arnaldo Souza	4300
305	Marisa Clara	5000
403	Bruno Simao	4500
410	Guilherme Santos	2000
502	Tatiana Andrade	2500

BUSCA SEQUENCIAL POR FUNCIONÁRIO 305

Código	Nome	Salário
102	Joao Silva	1000
123	Carlos Albuquerque	1500
143	Ana Bueno	1500
200	Caio Gusmao	4000
239	Bianca Amarilo	3000
254	Arnaldo Souza	4300
305	Marisa Clara	5000
403	Bruno Simao	4500
410	Guilherme Santos	2000
502	Tatiana Andrade	2500

BUSCA SEQUENCIAL POR FUNCIONÁRIO 305

Código	Nome	Salário
102	Joao Silva	1000
123	Carlos Albuquerque	1500
143	Ana Bueno	1500
200	Caio Gusmao	4000
239	Bianca Amarilo	3000
254	Arnaldo Souza	4300
305	Marisa Clara	5000
403	Bruno Simao	4500
410	Guilherme Santos	2000
502	Tatiana Andrade	2500

BUSCA SEQUENCIAL POR FUNCIONÁRIO 305

Código	Nome	Salário
102	Joao Silva	1000
123	Carlos Albuquerque	1500
143	Ana Bueno	1500
200	Caio Gusmao	4000
239	Bianca Amarilo	3000
254	Arnaldo Souza	4300
305	Marisa Clara	5000
403	Bruno Simao	4500
410	Guilherme Santos	2000
502	Tatiana Andrade	2500

ALTERNATIVA 2

Se arquivo está ordenado, faz **busca binária**

BUSCA BINÁRIA POR FUNCIONÁRIO 305

	Código	Nome	Salário
1	102	Joao Silva	1000
2	123	Carlos Albuquerque	1500
3	143	Ana Bueno	1500
4	200	Caio Gusmao	4000
5	239	Bianca Amarilo	3000
6	254	Arnaldo Souza	4300
7	305	Marisa Clara	5000
8	403	Bruno Simao	4500
9	410	Guilherme Santos	2000
10	502	Tatiana Andrade	2500

Lê registro do meio e compara chave buscada com a chave do registro lido

início = 1
fim = 10
meio = trunc((inicio + fim)/2) = 5
305 > 239

BUSCA BINÁRIA POR FUNCIONÁRIO 305

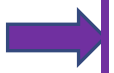
	Código	Nome	Salário
1	102	Joao Silva	1000
2	123	Carlos Albuquerque	1500
3	143	Ana Bueno	1500
4	200	Caio Gusmao	4000
5	239	Bianca Amarilo	3000
6	254	Arnaldo Souza	4300
7	305	Marisa Clara	5000
8	403	Bruno Simao	4500
9	410	Guilherme Santos	2000
10	502	Tatiana Andrade	2500

Repete procedimento na metade do arquivo correspondente (se chave menor, na metade de cima, se chave maior, na metade de baixo)

início = meio + 1

BUSCA BINÁRIA POR FUNCIONÁRIO 305

	Código	Nome	Salário
1	102	Joao Silva	1000
2	123	Carlos Albuquerque	1500
3	143	Ana Bueno	1500
4	200	Caio Gusmao	4000
5	239	Bianca Amarilo	3000
6	254	Arnaldo Souza	4300
7	305	Marisa Clara	5000
8	403	Bruno Simao	4500
9	410	Guilherme Santos	2000
10	502	Tatiana Andrade	2500



Lê registro do meio e compara chave buscada com a chave do registro lido

início = 6

fim = 10

meio = trunc((inicio + fim)/2) = 8

305 < 403

BUSCA BINÁRIA POR FUNCIONÁRIO 305

	Código	Nome	Salário
1	102	Joao Silva	1000
2	123	Carlos Albuquerque	1500
3	143	Ana Bueno	1500
4	200	Caio Gusmao	4000
5	239	Bianca Amarilo	3000
6	254	Arnaldo Souza	4300
7	305	Marisa Clara	5000
8	403	Bruno Simao	4500
9	410	Guilherme Santos	2000
10	502	Tatiana Andrade	2500

Repete procedimento na metade do arquivo correspondente (se chave menor, na metade de cima, se chave maior, na metade de baixo)

fim = meio - 1

BUSCA BINÁRIA POR FUNCIONÁRIO 305

	Código	Nome	Salário
1	102	Joao Silva	1000
2	123	Carlos Albuquerque	1500
3	143	Ana Bueno	1500
4	200	Caio Gusmao	4000
5	239	Bianca Amarilo	3000
6	254	Arnaldo Souza	4300
7	305	Marisa Clara	5000
8	403	Bruno Simao	4500
9	410	Guilherme Santos	2000
10	502	Tatiana Andrade	2500

Lê registro do meio e compara chave buscada com a chave do registro lido

início = 6
fim = 7
meio = trunc((inicio + fim)/2) = 6
305 > 254

BUSCA BINÁRIA POR FUNCIONÁRIO 305

	Código	Nome	Salário
1	102	Joao Silva	1000
2	123	Carlos Albuquerque	1500
3	143	Ana Bueno	1500
4	200	Caio Gusmao	4000
5	239	Bianca Amarilo	3000
6	254	Arnaldo Souza	4300
7	305	Marisa Clara	5000
8	403	Bruno Simao	4500
9	410	Guilherme Santos	2000
10	502	Tatiana Andrade	2500

Repete procedimento na metade do arquivo correspondente (se chave menor, na metade de cima, se chave maior, na metade de baixo)

inicio = meio + 1

BUSCA BINÁRIA POR FUNCIONÁRIO 305

	Código	Nome	Salário
1	102	Joao Silva	1000
2	123	Carlos Albuquerque	1500
3	143	Ana Bueno	1500
4	200	Caio Gusmao	4000
5	239	Bianca Amarilo	3000
6	254	Arnaldo Souza	4300
7	305	Marisa Clara	5000
8	403	Bruno Simao	4500
9	410	Guilherme Santos	2000
10	502	Tatiana Andrade	2500

Lê registro do meio e compara chave buscada com a chave do registro lido

início = 7
fim = 7
meio = trunc((inicio + fim)/2) = 7
305 = 305

BUSCA BINÁRIA – DETALHES

Exige que se saiba o **endereço** de um determinado registro, para que seja possível fazer o **seek** no arquivo para aquele endereço

- Usar cálculo de endereço visto anteriormente

Exige que se saiba quantos registros o arquivo possui

- Usar função **tamanho_arquivo** vista anteriormente

COMPARAÇÃO

Na busca sequencial, para esse exemplo, foram lidos 7 registros até encontrar o funcionário desejado

Na busca binária, foram lidos apenas 4 registros

Assumindo que o arquivo tem **n** registros:

- Complexidade da busca sequencial: **O(n)**
- Complexidade da busca binária: **O(log n)**

EXERCÍCIO

Dado um arquivo de funcionários, ordenado, implementar uma função que faz busca binária no arquivo

```
 /* cod é a chave buscada
  * *arq é o ponteiro para o arquivo
  * tam é o número de registros do arquivo
  */
TFunc *busca_binaria(int cod, FILE *arq, int tam)
```

ORDENAÇÃO

ORDENAÇÃO

Busca binária exige que arquivo esteja ordenado

Como ordenar um arquivo?

MÉTODOS DE ORDENAÇÃO DE ARQUIVOS

Vários métodos podem ser aplicados

Possível solução:

- métodos de ordenação em memória

1. Ler arquivo e armazenar os dados num array em memória
2. Ordenar o array
3. Gravar novo arquivo com os dados ordenados

CONVENÇÃO

Os algoritmos que veremos assumem que todas as chaves do arquivo estão num vetor A

Na prática isso será algo como A[i].chave

ORDENAÇÃO POR INSERÇÃO

Insertion Sort

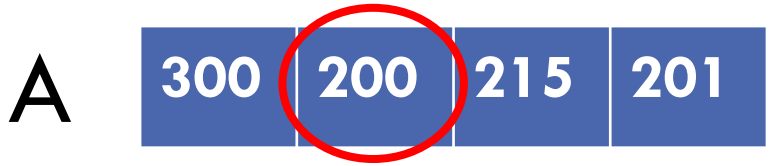
Nome do método se deve ao fato de que no i-ésimo passo ele insere o i-ésimo elemento A[i] na posição correta entre A[1], A[2], …, A[i-1] **que já foram previamente ordenados**

FUNCIONAMENTO DO INSERTION SORT

1. Assume que o primeiro valor já está ordenado
2. Pega o próximo valor, compara com os anteriores até descobrir em que posição ele deveria estar
3. Abre espaço no vetor para encaixar o valor na posição correta
4. Encaixa o valor na posição correta
5. Se vetor ainda não terminou, volta para o passo 2

FUNCIONAMENTO DO INSERTION SORT

1. Assume que o primeiro valor já está ordenado
2. **Pega o próximo valor, compara com os anteriores até descobrir em que posição ele deveria estar**
3. Abre espaço no vetor para encaixar o valor na posição correta
4. Encaixa o valor na posição correta
5. Se vetor ainda não terminou, volta para o passo 2



FUNCIONAMENTO DO INSERTION SORT

1. Assume que o primeiro valor já está ordenado
2. Pega o próximo valor, compara com os anteriores até descobrir em que posição ele deveria estar
3. **Abre espaço no vetor para encaixar o valor na posição correta**
4. Encaixa o valor na posição correta
5. Se vetor ainda não terminou, volta para o passo 2

A

FUNCIONAMENTO DO INSERTION SORT

1. Assume que o primeiro valor já está ordenado
2. Pega o próximo valor, compara com os anteriores até descobrir em que posição ele deveria estar
3. Abre espaço no vetor para encaixar o valor na posição correta
4. **Encaixa o valor na posição correta**
5. Se vetor ainda não terminou, volta para o passo 2

A

FUNCIONAMENTO DO INSERTION SORT

1. Assume que o primeiro valor já está ordenado
2. **Pega o próximo valor, compara com os anteriores até descobrir em que posição ele deveria estar**
3. Abre espaço no vetor para encaixar o valor na posição correta
4. Encaixa o valor na posição correta
5. Se vetor ainda não terminou, volta para o passo 2

A

FUNCIONAMENTO DO INSERTION SORT

1. Assume que o primeiro valor já está ordenado
2. Pega o próximo valor, compara com os anteriores até descobrir em que posição ele deveria estar
3. **Abre espaço no vetor para encaixar o valor na posição correta**
4. Encaixa o valor na posição correta
5. Se vetor ainda não terminou, volta para o passo 2

A

FUNCIONAMENTO DO INSERTION SORT

1. Assume que o primeiro valor já está ordenado
2. Pega o próximo valor, compara com os anteriores até descobrir em que posição ele deveria estar
3. Abre espaço no vetor para encaixar o valor na posição correta
4. **Encaixa o valor na posição correta**
5. Se vetor ainda não terminou, volta para o passo 2

A

FUNCIONAMENTO DO INSERTION SORT

1. Assume que o primeiro valor já está ordenado
2. **Pega o próximo valor, compara com os anteriores até descobrir em que posição ele deveria estar**
3. Abre espaço no vetor para encaixar o valor na posição correta
4. Encaixa o valor na posição correta
5. Se vetor ainda não terminou, volta para o passo 2

FUNCIONAMENTO DO INSERTION SORT

1. Assume que o primeiro valor já está ordenado
2. Pega o próximo valor, compara com os anteriores até descobrir em que posição ele deveria estar
3. **Abre espaço no vetor para encaixar o valor na posição correta**
4. Encaixa o valor na posição correta
5. Se vetor ainda não terminou, volta para o passo 2

FUNCIONAMENTO DO INSERTION SORT

1. Assume que o primeiro valor já está ordenado
2. Pega o próximo valor, compara com os anteriores até descobrir em que posição ele deveria estar
3. Abre espaço no vetor para encaixar o valor na posição correta
4. **Encaixa o valor na posição correta**
5. Se vetor ainda não terminou, volta para o passo 2

ALGORITMO INSERTION SORT

```
procedure insertionSort(A: array, size: int) {
//ENTRADA: A: array com as chaves (posições vão de [0, tam-1]);
//         size: tamanho do array
//SAÍDA: A: array ordenado
for j := 1 to size-1 do {
   key := A[j];
   i := j - 1;
   while (i >= 0) and (A[i] > key) do {
      A[i+1] := A[i];
      i := i -1;
   }
   A[i+1] := key
}
```

INSERTION SORT PARA ORDENAR FUNCIONÁRIOS USANDO VETOR EM MEMÓRIA

Ver código no site da disciplina

COMPLEXIDADE DO INSERTION SORT

Complexidade pior caso $O(n^2)$

Complexidade caso médio $O(n^2)$

Complexidade melhor caso $O(n)$

Complexidade de espaço $O(n)$

OUTROS ALGORITMOS DE ORDENAÇÃO

Selection Sort

Buble Sort

Quick Sort

...

ORDENAÇÃO DE ARQUIVOS

Qualquer algoritmo de ordenação pode ser usado para ordenar arquivos, desde que:

- os **registros caibam todos na memória** de uma só vez

Alternativa que gasta menos memória:

- Fazer a ordenação direto no arquivo, **sem usar um vetor auxiliar**
- Só pode ser feito para **arquivos binários**, pois é necessário usar **fseek** para se deslocar de um registro a outro

INSERTION SORT DIRETO NO ARQUIVO

Ver código no site da disciplina

EXERCÍCIO

Implementar uma função que faz o Selection Sort direto no arquivo, sem usar um vetor auxiliar

```
void selection_sort_disco(FILE *arq, int tam)
```

Lembrete: o Selecion Sort percorre o arquivo, procurando o menor elemento. Quando encontra, coloca-o na posição 1 (troca o elemento da posição 1 por ele). Depois, procura o próximo menor. Quando encontra, coloca-o na posição 2 (troca o elemento menor pelo da posição 2), e assim por diante.

REFERÊNCIA

Schildt, H. C Completo e Total. Ed. McGraw-Hill