

PESQUISA

Prof. Muriel Mazzetto
Estrutura de Dados

Pesquisa

2

- Buscar por uma chave presente na lista de dados.
- Podem ser dados numéricos ou caracteres, desde que exista um critério de comparação.

Pesquisa

3

- Buscar por uma chave presente na lista de dados.
- Podem ser dados numéricos ou caracteres, desde que exista um critério de comparação.
- Depende da estrutura que está sendo utilizada:
 - Tabelas Hash necessitam de uma função de hash e tratamento de colisão.
 - Árvores Binárias de Busca simulam uma busca binária.
 - Listas Encadeadas utilizam apenas busca sequencial.
 - Listas em vetores possibilitam a busca binária, se estiverem ordenados.

Pesquisa

4

- Listas encadeadas:
 - ▣ Ganham na velocidade de inserção e remoção.
 - ▣ Perdem na pesquisa.
- Vetores:
 - ▣ Ganham na pesquisa (quando ordenados).
 - ▣ Perdem na inserção e remoção.

Pesquisa Sequencial

5

- Consiste em percorrer toda a estrutura, comparando a chave com cada elemento até que seja encontrada.
- Não necessita a ordenação da lista.

Pesquisa Sequencial

6

- Consiste em percorrer toda a estrutura, comparando a chave com cada elemento até que seja encontrada.
- Não necessita a ordenação da lista.

```
int BuscaSequencial(int* vetor, int tam, int chave)
{
    int i;
    for(i = 0; i < tam; i++)
    {
        if(vetor[i] == chave)
        {
            return i; //retornar índice ou ponteiro
        }
    }
    return -1; //não encontrou índice;
}
```

Pesquisa Sequencial

7

- Pior caso:

- Quando a chave pesquisada está na última posição ou quando ela não existe na lista.
- $O(n)$: necessário comparar todos os n elementos.

- Melhor caso:

- Quando a chave pesquisada é a primeira na lista.

Pesquisa Binária

8

- ❑ **Os dados devem estar ordenados.**
- ❑ Trabalha com os índices do vetor.
- ❑ Similar ao processo de uma busca em ABB.

Pesquisa Binária

9

- ❑ **Os dados devem estar ordenados.**
- ❑ Trabalha com os índices do vetor.
- ❑ Similar ao processo de uma busca em ABB.
- ❑ Exemplo:
 - ❑ Buscar o valor 8.

0	1	2	3	4	5	6	7	8	9
2	4	8	10	11	13	15	16	20	25

Pesquisa Binária

10

- ❑ **Os dados devem estar ordenados.**
- ❑ Trabalha com os índices do vetor.
- ❑ Similar ao processo de uma busca em ABB.
- ❑ Exemplo:
 - ❑ Buscar o valor 8.

0	1	2	3	4	5	6	7	8	9
2	4	8	10	11	13	15	16	20	25

- ❑ 1: Encontrar índice do meio e comparar com a chave.

Pesquisa Binária

11

- ❑ **Os dados devem estar ordenados.**
- ❑ Trabalha com os índices do vetor.
- ❑ Similar ao processo de uma busca em ABB.
- ❑ Exemplo:
 - ❑ Buscar o valor 8.

0	1	2	3	4	5	6	7	8	9
2	4	8	10	11	13	15	16	20	25

- ❑ 1: Encontrar índice do meio e comparar com a chave.
- ❑ 2: Se for menor, buscar meio da parte esquerda.

Pesquisa Binária

12

- ❑ **Os dados devem estar ordenados.**
- ❑ Trabalha com os índices do vetor.
- ❑ Similar ao processo de uma busca em ABB.
- ❑ Exemplo:
 - ❑ Buscar o valor 8.

0	1	2	3	4	5	6	7	8	9
2	4	8	10	11	13	15	16	20	25

- ❑ 1: Encontrar índice do meio e comparar com a chave.

Pesquisa Binária

13

- ❑ **Os dados devem estar ordenados.**
- ❑ Trabalha com os índices do vetor.
- ❑ Similar ao processo de uma busca em ABB.
- ❑ Exemplo:
 - ❑ Buscar o valor 8.

0	1	2	3	4	5	6	7	8	9
2	4	8	10	11	13	15	16	20	25

- ❑ 1: Encontrar índice do meio e comparar com a chave.
- ❑ 2: Se for maior, buscar meio da parte direita.

Pesquisa Binária

14

- ❑ **Os dados devem estar ordenados.**
- ❑ Trabalha com os índices do vetor.
- ❑ Similar ao processo de uma busca em ABB.
- ❑ Exemplo:
 - ❑ Buscar o valor 8.

0	1	2	3	4	5	6	7	8	9
2	4	8	10	11	13	15	16	20	25

- ❑ 1: Encontrar índice do meio e comparar com a chave.

Pesquisa Binária

15

- ❑ **Os dados devem estar ordenados.**
- ❑ Trabalha com os índices do vetor.
- ❑ Similar ao processo de uma busca em ABB.
- ❑ Exemplo:
 - ❑ Buscar o valor 8.

0	1	2	3	4	5	6	7	8	9
2	4	8	10	11	13	15	16	20	25

- ❑ 1: Encontrar índice do meio e comparar com a chave.
- ❑ 2: Se for igual, retornar índice ou ponteiro do dado.

Pesquisa Binária

16

□ Implementação com iteração.

```
int BuscaBinariaI(int* vetor, int tam, int chave)
{
    int esquerda = 0;
    int direita = tam-1;
    int meio;
    while(esquerda <= direita)
    {
        meio = (esquerda + direita)/2;
        if(vetor[meio] == chave) return meio;
        if(vetor[meio] > chave) direita = meio - 1;
        else esquerda = meio + 1;
    }
    return -1;
}
```


Pesquisa Binária

17

□ Implementação com recursão.

```
int BuscaBinariaR(int*Vetor, int chave, int esq, int dir)
{
    if(esq > dir) return -1; //não encontrou índice

    int meio = (esq + dir)/2;
    if(Vetor[meio] == chave) return meio;
    if(Vetor[meio] < chave) return BuscaBinariaR(Vetor, chave, meio + 1, dir);
    else return BuscaBinariaR(Vetor, chave, esq, meio - 1);
}
```

Pesquisa Binária

18

□ Implementação com recursão.

```
int BuscaBinariaR(int*Vetor, int chave, int esq, int dir)
{
    if(esq > dir) return -1; //não encontrou índice

    int meio = (esq + dir)/2;
    if(Vetor[meio] == chave) return meio;
    if(Vetor[meio] < chave) return BuscaBinariaR(Vetor, chave, meio + 1, dir);
    else return BuscaBinariaR(Vetor, chave, esq, meio - 1);
}
```

Curiosos: pesquisem por bsearch da biblioteca padrão do C (stdlib.h).

Pesquisa Binária

19

❑ Pior caso:

- ❑ Quando a chave pesquisada não existe na lista.
- ❑ **$O(\log n)$** : necessário realizar todas as divisões.
 - Equivalente a percorrer uma árvore binária balanceada até chegar em uma folha.

❑ Melhor caso:

- ❑ Quando a chave pesquisada é a primeira na lista.