

GSI011 – Estrutura de Dados 2

Prof^a: Christiane Regina Soares Brasil

GSI011 – Estrutura de Dados 2 – Aula 4

- Busca
 - Objetivo: procurar um elemento (chave) que pode ser:
 - Valor armazenado no vetor ou matriz, por exemplo.
 - Campo de uma **struct**

- Tipos de Busca

- As buscas dependem de como os dados estão armazenados:
 - Array, lista, árvore
 - Ordenados
 - Duplicados
- Métodos de busca:
 - Sequencial ou linear
 - Sequencial Ordenada
 - Binária

GSI011 – Estrutura de Dados 2 – Aula 4

- Busca Sequencial ou Linear

```
int buscaLinear(int* v, int n, int x)
{
    int i;
    for(i=0; i<n; i++)
    {
        if(x==v[i]) return i;
    }
    return -1;
}
```

Vantagem:

- Simplicidade

Desvantagem:

- Dados duplicados

Exemplo:

0	1	2	3	4	5
12	2	45	36	2	56

- Busca Sequencial ou Linear

Análise de Complexidade:

- $O(1)$, melhor caso: elemento encontrado na primeira posição.
- $O(n)$, pior caso: elemento encontrado na última posição (ou não encontrado).
- $O(n)$: caso médio.

A red, cloud-like shape with a blue outline, containing the text 'Alto custo computacional'.

Alto custo computacional

GSI011 – Estrutura de Dados 2 – Aula 4

- Busca Sequencial Ordenada

```
int buscaOrdenada(int* v, int n, int x)
{
    int i = 0;
    while(i < n && v[i] < x) i++;

    if(x == v[i]) return i;
    else return -1;
}
```

Vantagem:

- Dados duplicados

Desvantagem:

- Custo de ordenação

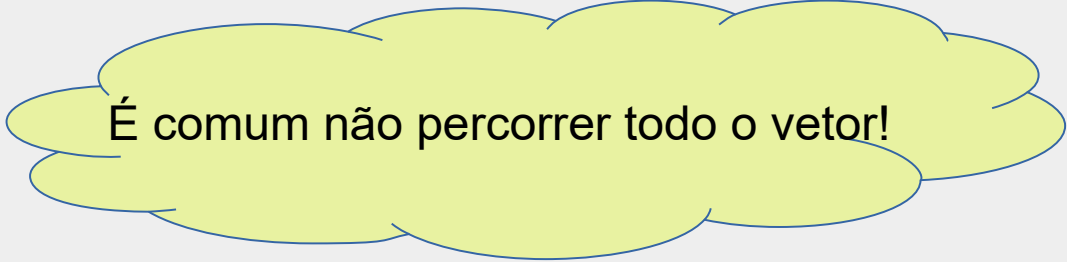
Exemplo:

0	1	2	3	4	5
2	2	12	36	45	56

- Busca Sequencial Ordenada

Análise de Complexidade:

- $O(1)$, melhor caso: elemento encontrado na primeira posição.
- $O(n)$, pior caso: elemento encontrado na última posição (ou não encontrado).
- $O(n)$: caso médio.



É comum não percorrer todo o vetor!

- Busca Binária

- Muito mais eficiente que a Busca Sequencial Ordenada
- Assume-se vetor ordenado
- Aplica-se a estratégia de “Divisão e Conquista”.

GSI011 – Estrutura de Dados 2 – Aula 4

- Busca Binária

Exemplo:

0	1	2	3	4	5
2	3	12	36	45	56

x=56	
meio = 2	meio = 5
if(56<12)?	if(56<56)?
inicio = 2 + 1	
inicio = 3	if(56>56)?
meio = 4	
If(56<45)?	return 5;
inicio = 5	

```
int buscaBinaria(int* v, int n, int x)
{
    int inicio, fim, meio;
    inicio = 0;
    fim = n-1;
    while(inicio<=fim)
    {
        meio = (inicio + fim)/2;
        if(x<v[meio]) fim = meio - 1;//metade esquerda
        else
        {
            if(x>v[meio]) inicio = meio + 1;//metade direita
            else return meio;
        }
    } return -1;
}
```

- Busca Binária

Análise de Complexidade:

- $O(1)$, melhor caso: elemento encontrado no meio.
- $O(\log_2 n)$, pior caso: elemento encontrado no último teste ou não existe.
- $O(\log_2 n)$: caso médio.

- Busca Binária

Comparando métodos, vamos supor vetor de 1000 elementos.

No pior caso:

- A busca sequencial terá 1000 comparações
- A busca binária terá aproximadamente 10 comparações.