## MÉTODOS DE BUSCA

São métodos utilizados para realização de buscas de uma determinada informação dentro de uma estrutura como um vetor, por exemplo.

Imagine o vetor a seguir:

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 46 | 39 | 32 | 15 | 88 | 3 | 7 | 24 | 6 | 31 |

Qual a melhor forma para verificar se o número 7 está presente neste vetor? Uma das formas seria percorrer o vetor da esquerda para a direita até achar o valor desejado.

Esta operação poderia ser descrita nos seguintes passos:

1. Percorrer o vetor da primeira até a última posição.
2. Para cada elemento, verificar se ele é igual ao valor a ser encontrado.
3. Se for igual, retornar *true*.
4. Caso contrário, continuar procurando.
5. Se não encontrar o elemento, retornar *false*.

Este método, também conhecido como Busca Sequencial, é o método mais simples de busca, onde se busca um determinado valor percorrendo uma lista de elementos do início até o final.

A Figura 1 mostra a definição de uma função que recebe um vetor como parâmetro, o seu tamanho e um determinado número a ser procurado. Caso o número seja encontrado, a função retorna *true*, se não, retorna *false*.

```
int buscaSequencial (int *vetor, int tamanho, int numero) {
      for (int i = 0; i < tamanho; i++) {
            if (vetor[i] == numero)
                  return true;
      }
      return false;
}
```

Figura 1 - Algoritmo de Busca Sequencial

E se os valores do vetor estivessem ordenados? Haveria uma maneira mais eficiente de efetuar a busca? No exemplo anterior, caso o elemento não seja encontrado, o algoritmo percorre o vetor até o final.

Se o vetor estivesse na ordem crescente, poderíamos parar de percorrê-lo quando o valor buscado fosse menor que o valor armazenado na posição corrente do vetor.

A Figura 2 mostra a definição de uma função que realiza a Busca Sequencial de forma mais eficiente.

```
int buscaSequencial (int *vetor, int tamanho, int numero) {
      for (int i = 0; i < tamanho; i++) {
            if (vetor[i] == numero) {
                  return true;
            } else if (vetor[i] > numero) {
                  break;
            }
      }
      return false;
}
```

Figura 2 - Algoritmo de Busca Sequencial

Existe um outro tipo de busca, chamada Busca Binária, que só pode ser utilizada se os elementos da estrutura de dados também estiverem ordenados.

Este método de busca verifica se o elemento central do vetor é maior, menor ou o próprio elemento a ser buscado. Se for maior, apenas a parte superior do vetor é considerada para fazer uma nova comparação, se for menor, apenas a parte inferior do vetor será considerada e se for igual, a função retornará o valor lógico *true*.

A Figura 3 mostra a definição de uma função que realiza a Busca Binária dentro de um vetor.

```
int buscaBinaria (int *vetor, int tamanho, int numero) {
      int inicio = 0;
      int fim = tamanho - 1;
      int centro;
      while (inicio <= fim) {
            centro = inicio + ((fim - inicio) / 2);
            if (vetor[centro] == n) {
                  return true;
            } else if (vetor[centro] > n) {
                  fim = centro - 1;
            } else {
                  inicio = centro + 1;
            }
      }
      return false;
}
```

Figura 3 - Algoritmo de Busca Binária

Se comparada com a busca sequencial, a busca binária se prova mais eficiente, desde que todos os elementos do vetor estejam devidamente ordenados.

**REFERÊNCIAS**

ASCENCIO, A. F. G.; CAMPOS, E. A. V. Fundamentos da Programação de Computadores. 3. ed. São Paulo: Pearson, 2012.

FORBELLONE, A. L. V.; EBERSPÄCHER, H. F. Lógica de Programação: A Construção de Algoritmos e Estruturas de Dados. 3. ed. São Paulo: Pearson, 2005.

C++ Pointers, W3Schools, 2020. Disponível em <https://www.w3schools.com/cpp/cpp_pointers.asp>. Acesso em: 17 de novembro de 2020.