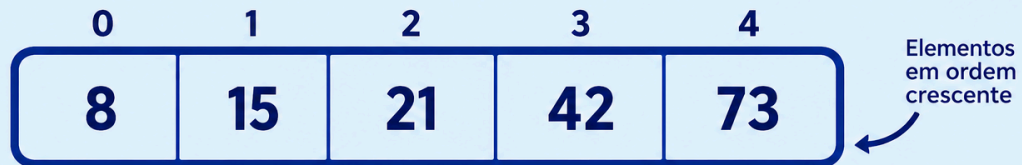


TÓPICO 05: VETORES ORDENADOS

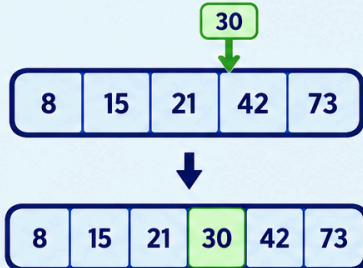
Estrutura de Dados - Professor Ramon Venson - SATC 2026.2

Vetor Ordenado



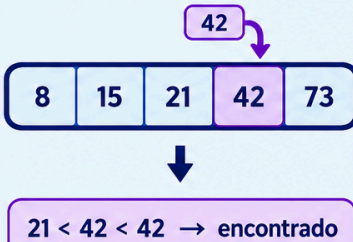
Inserir

Encontra a posição correta e insere o valor, mantendo a ordenação.



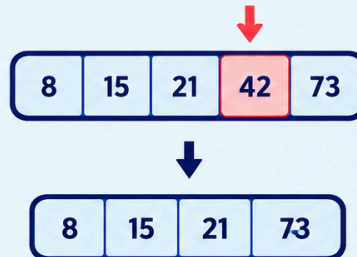
Pesquisar

Como o vetor está ordenado, podemos parar a busca assim que encontrar o valor ou um valor maior.



Remover

Encontra o valor, remove e mantém a ordenação, deslocando os elementos.



Vetores Ordenados

Estrutura onde os elementos são armazenados em **ordem crescente** ou **decrescente**, o que permite uma **pesquisa mais eficiente**.

Características

- **Tamanho fixo:** O tamanho do vetor é definido no momento da criação.
- **Acesso direto:** É possível acessar qualquer elemento diretamente pelo seu índice.
- **Ordenado:** Os elementos seguem uma ordem específica, o que permite uma pesquisa mais eficiente.
- **Inserção e remoção:** Inserir ou remover elementos pode ser mais complexo, pois é necessário manter a ordem dos elementos.

Inicialização de um Vetor Ordenado

```
classe VetorOrdenado:  
  capacidade: inteiro  
  ultima_posicao: inteiro  
  valores: array  
  
  construtor(capacidade):  
    self.capacidade <- capacidade  
    self.ultima_posicao <- -1  
    self.valores <- novo array de inteiro com tamanho capacidade
```

Imprimindo os Elementos do Vetor

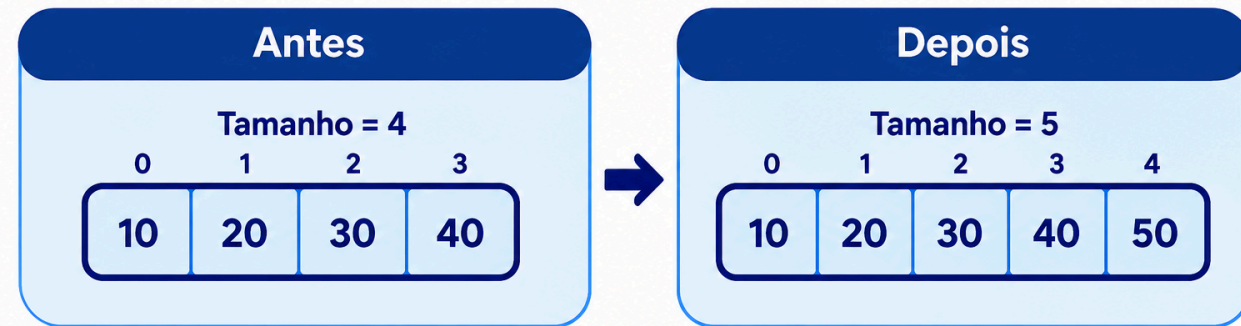
Para imprimir todos os valores de um vetor ordenado, vamos percorrer cada um dos elementos do vetor até a posição final.

```
imprimir():  
    para cada i no intervalo de 0 até ultima_posicao faça:  
        escreva(valores[i])
```

Inserção de Elementos

Para inserir um elemento em um vetor ordenado, é necessário **encontrar a posição correta** para manter a ordem dos elementos.

Vetor Ordenado Inserir



O que acontece

O valor é inserido mantendo o vetor ordenado: o elemento 50 é colocado na posição correta.

Pseudocódigo para Inserção

```
inserir(valor):  
    se ultima_posicao == capacidade - 1 então  
        retorne "Vetor cheio"  
    senao:  
        posicao <- 0  
        para cada i no intervalo de 0 até ultima_posicao faça:  
            se valor > valores[i] então  
                posicao <- i + 1  
        valores[posicao] <- valor  
        ultima_posicao <- ultima_posicao + 1
```

Nesse pseudocódigo, o algoritmo percorre o vetor até encontrar a posição correta (que chamamos de pesquisa linear) para inserir o novo valor, deslocando os elementos subsequentes para a direita.



Vetor Ordenado

Pesquisar Linear e Binário

Pesquisa Linear

Percorre o vetor do início ao fim, comparando cada elemento.

Buscar: 30



Pesquisa Binária

Divide o vetor ao meio a cada passo, eliminando metade das possibilidades.

Buscar: 30



1º passo:
compara com
o elemento
do meio (30)



Encontrado!
30 está no vetor.



A pesquisa binária é mais eficiente em vetores ordenados, pois reduz pela metade o número de comparações a cada passo.

Pesquisa de Elementos

Em um vetor ordenado, a pesquisa pode ser feita de duas formas:

- **Busca linear:** Percorre o vetor do início ao fim até encontrar o elemento desejado.
- **Busca binária:** Divide o vetor em duas partes e compara o elemento com o valor do meio, repetindo até encontrar o elemento ou determinar que não está presente.

Pseudocódigo para Pesquisa Linear

```
pesquisar(valor):  
    para cada i no intervalo de 0 até ultima_posicao faça:  
        se valores[i] = valor então  
            retorne i  
    retorne -1
```

- **Melhor caso:** O elemento a ser pesquisado está na primeira posição.
- **Pior caso:** O elemento a ser pesquisado está na última posição ou não está presente no vetor.

Pseudocódigo para Pesquisa Binária

```
pesquisar_binaria(valor):  
    inicio <- 0  
    fim <- ultima_posicao  
    enquanto inicio <= fim faça:  
        meio <- (inicio + fim) / 2  
        se valores[meio] = valor então  
            retorne meio  
        senao se valores[meio] < valor então  
            inicio <- meio + 1  
        senao:  
            fim <- meio - 1  
    retorne -1
```

- **Melhor caso:** O elemento a ser pesquisado está na posição do meio.
- **Pior caso:** O elemento a ser pesquisado não está presente no vetor.

Comparação entre Pesquisa Linear e Binária

Tamanho do Vetor	Pesquisa Linear $O(n)$	Pesquisa Binária $O(\log n)$
10	5 iterações	3 iterações
100	50 iterações	7 iterações
1000	500 iterações	10 iterações
10.000	5.000 iterações	14 iterações
100.000	50.000 iterações	17 iterações
1.000.000	500.000 iterações	20 iterações

O **Caso Médio** é a média de tempo para encontrar um elemento.

Remoção de Elementos

Para remover um elemento, o elemento a ser removido é removido e os elementos restantes são deslocados para preencher o espaço vazio.

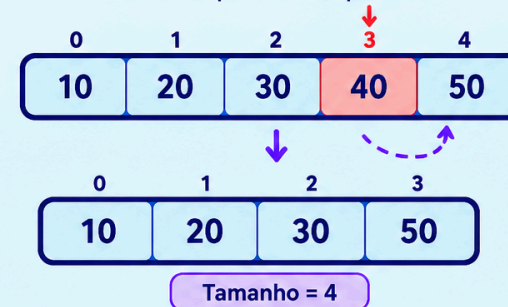
Utiliza a função de pesquisa para localizar o elemento a ser removido.



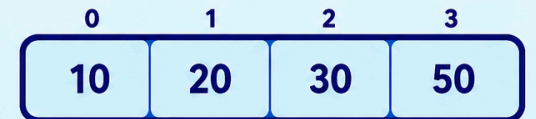
Vetor Ordenado Remove

Como Remover

- 1 Localizar o elemento na posição correta (vetor ordenado).
- 2 Remover o elemento, deslocando os demais para a esquerda.



Após Remover



Em um vetor ordenado, ao remover um elemento, os demais são deslocados para a esquerda para manter a ordem crescente.

Pseudocódigo para Remoção

```
excluir(valor):  
    posicao <- pesquisar(valor)  
    se posicao = -1 então  
        retorne -1  
    senao:  
        para cada i no intervalo de posicao até ultima_posicao faça:  
            vetor[i] <- vetor[i + 1]  
        ultima_posicao <- ultima_posicao - 1
```

Ordenado vs Não Ordenado

Característica	Vetor Ordenado	Vetor Não Ordenado
Estrutura	Linear	Linear
Inserção	Complexa (manter ordem)	Simples (adiciona no final)
Pesquisa	Mais Rápida (binária)	Mais Lenta (linear)
Remoção	Realizada com deslocamento	Realizada com deslocamento