1. Implemente uma versão recursiva para a pesquisa Binária.

Código no arquivo ex01.c.

2. Dado um vetor ordenado de inteiros distintos v[1,...,n], você quer descobrir se existe um índice i tal que v[i] = i. Desenvolva um algoritmo do tipo dividir para conquistar que resolva este problema e cuja complexidade idade seja O(log n).

Código no arquivo ex02.c.

3. Implemente uma TAD Lista para o tipo Aluno (Matrícula e Nome), com as funcionalidades básicas de uma TAD, Implemente uma lista para a versão usando arranjos e outra para a versão usando ponteiros. Adicione a TAD as seguintes funcionalidades:

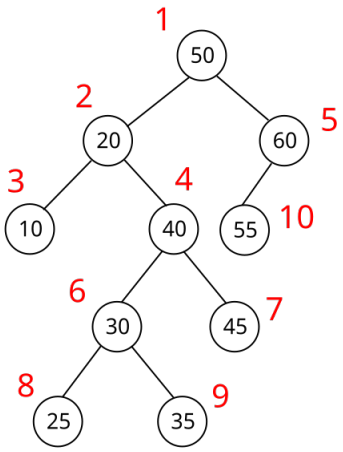
- **Uma função PesquisaSequencial, que recebe como parâmetro o nome de um aluno e retorna a sua matrícula.**
- **Uma função PesquisaBinaria, que recebe como parâmetro a matrícula de um aluno e retorne o seu nome.**

Código no diretório ex03. Comandos de compilação usados:

```
gcc main.c aluno.c pesqBinaria.c pesqSequencial.c lista_linear.c -o main
gcc main.c aluno.c pesqBinaria.c pesqSequencial.c lista_flexivel.c -o main
```

4. Insira os números abaixo na ordem que são apresentados numa árvore binária de busca. Mostre todos os passos.

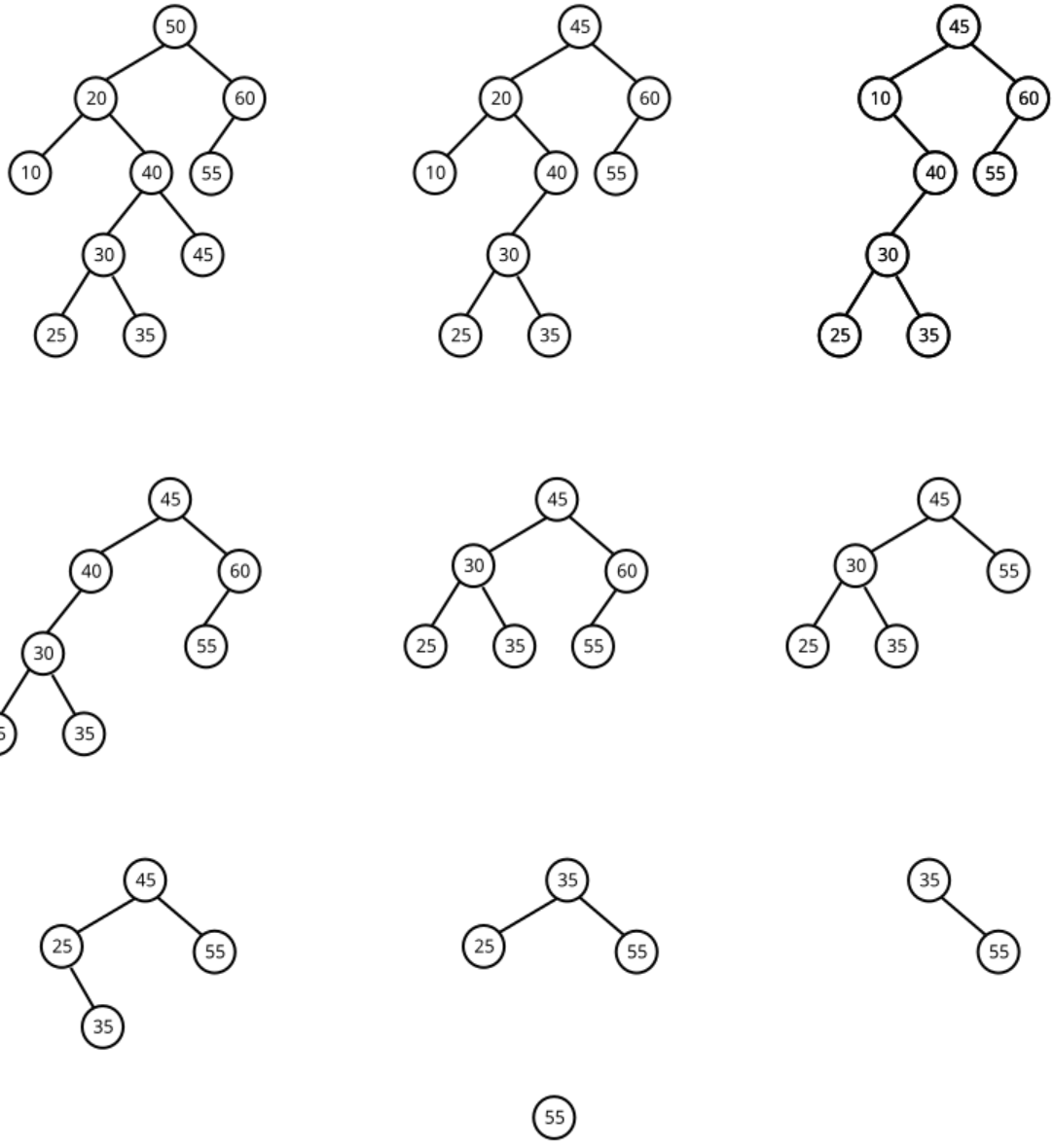
- **50; 20; 10; 40; 60; 30; 45; 25; 35; 55**



A numeração em vermelho indica a ordem de inserção.

5. A partir da árvore obtida no exercício anterior, remova os números abaixo na ordem que são apresentados. Mostre todos os passos.

- **50; 20; 10; 40; 60; 30; 45; 25; 35; 55**



6. Análise a árvore gerada na questão 4 e responda:

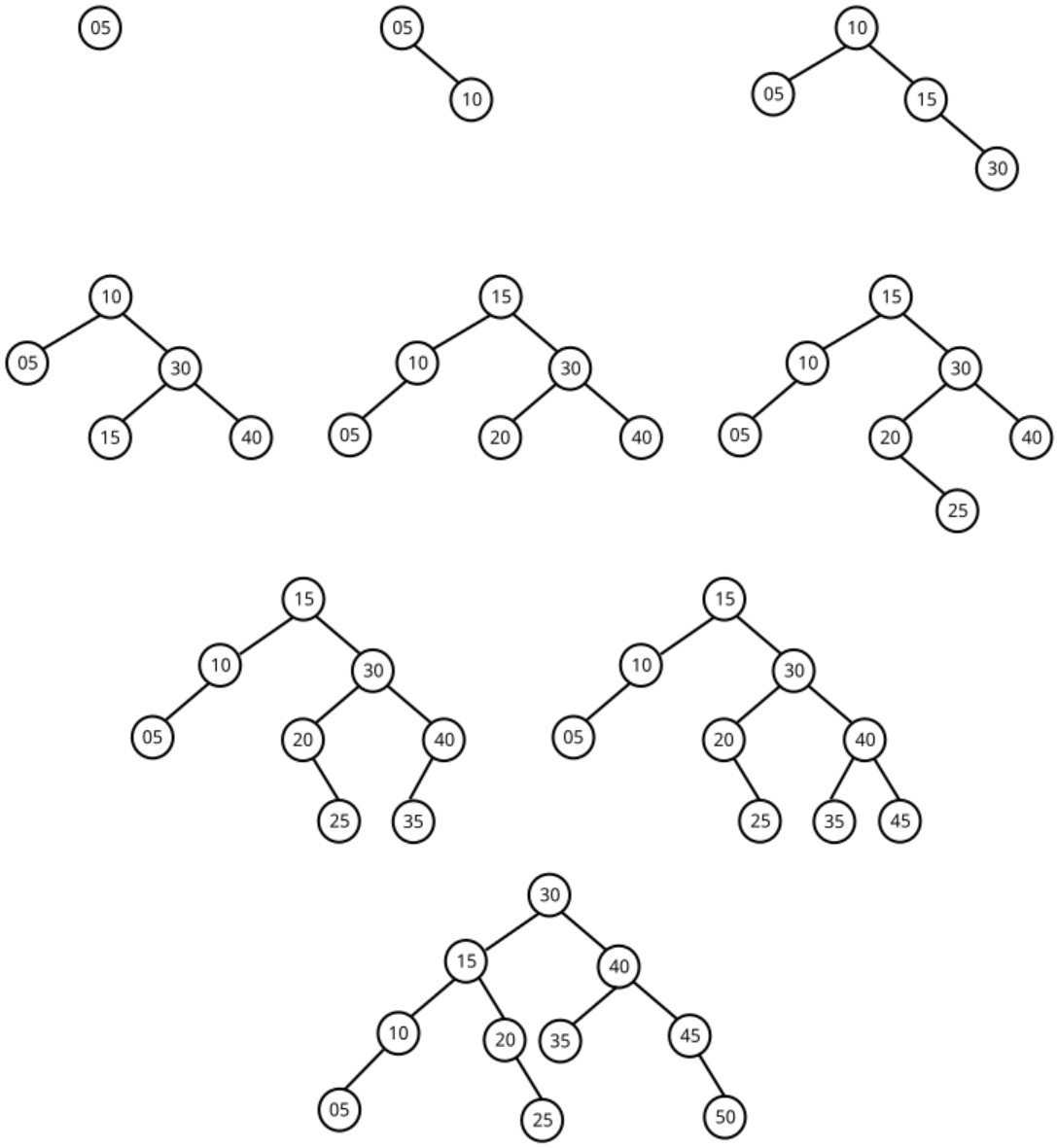
a) Qual o custo para se pesquisar o valor 25 na árvore? Explique.
O custo para pesquisar o valor 25 na árvore binária é de 4 verificações. Esse custo se deve a altura do nó contendo o elemento 25.

b) Qual o custo para se pesquisar o valor 25 se usássemos uma estrutura de pesquisa sequencial. Explique.
O custo para pesquisar o valor 25 em pesquisa sequencial é de 8 verificações, levando em consideração a inserção dos elementos na ordem descrita acima. Esse custo se deve ao fato de ser necessário verificar todos os elementos anteriores ao 25 durante a pesquisa.

c) Qual o custo para se pesquisar o valor 25 se usássemos uma estrutura de pesquisa binária. Explique.
O custo para pesquisar o valor 25 em pesquisa binária é de 3 verificações. Esse custo se dá porque 25 é será o elemento do meio após três verificações do algoritmo dividir e conquistar.

7. Implemente uma estrutura de pesquisa usando a Tabela Hashing, usando as seguintes definições:
(a) Tratamento de colisão usando Endereçamento Aberto, onde os registros em conflito são armazenados dentro da própria tabela, executando uma busca sequencial.
(b) Tratamento de colisão usando Endereçamento Separado, onde os registros em conflito são armazenados dentro de uma lista encadeada.
Exercício não foi feito.

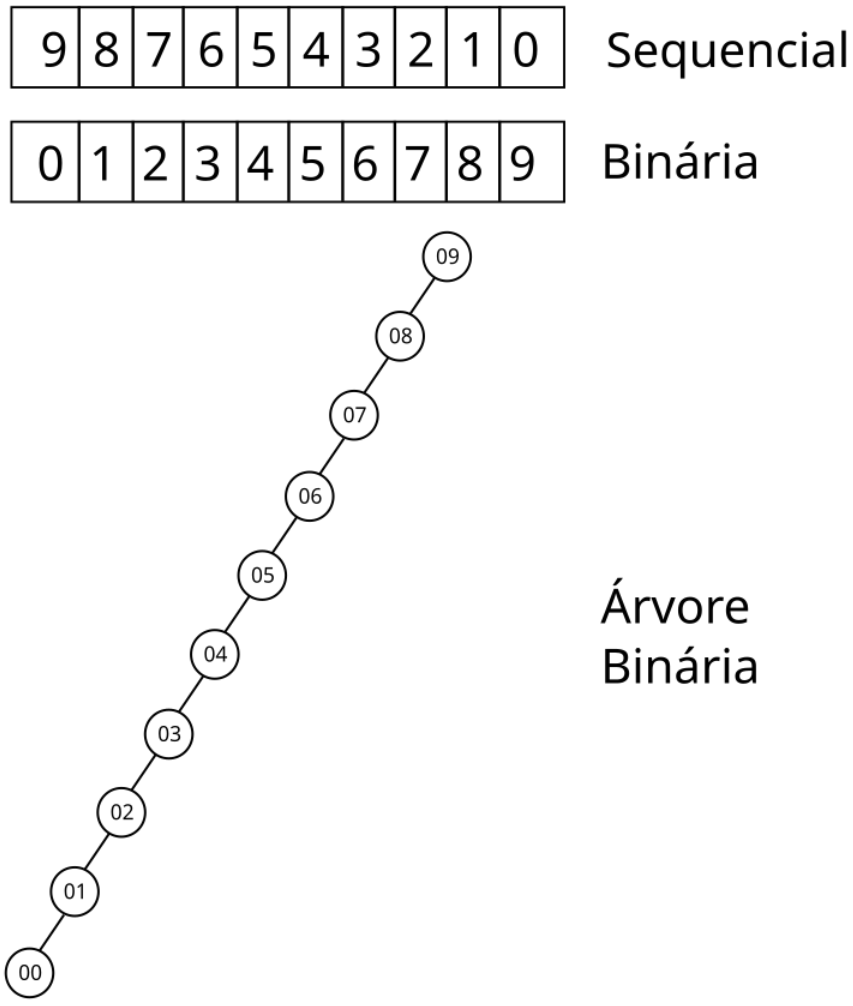
8. Crie uma Árvore Binária AVL com a inserção dos seguintes itens, mostrando a construção da árvore passo a passo:
5 - 10 - 15 - 30 - 40 - 20 - 25 - 35 - 45 - 50

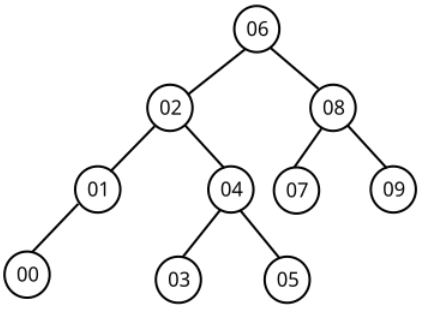


9. Considere que você tem os seguintes itens a serem armazenados no sistema:
9 - 8 - 7 - 6 - 5 - 4 - 3 - 2 - 1 - 0

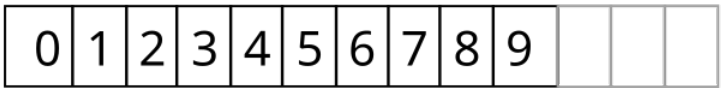
Faça a modelagem para armazenar essas informações, na ordem em que aparecem, nas seguintes estruturas de dados:

- **Pesquisa Sequencial**
- **Pesquisa Binária**
- **Árvore Binária**
- **Árvore Binária Balanceada AVL**
- **Tabela Hash**





AVL



Tab. Hash
(mod 10)

Usando a modelagem das estrutura de pesquisa, calcule o tempo de acesso para pesquisar os seguintes valores, em cada uma das estruturas:

(a) 9

Sequencial: 1 Verificação

Binária: 4 Verificações

Árvore Binária: 1 Verificação

AVL: 2 Verificações

Tab. Hash: 1 Verificação

(b) 7

Sequencial: 3 Verificações

Binária: 2 Verificações

Árvore Binária: 2 Verificações

AVL: 2 Verificações

Tab. Hash: 1 Verificação

(c) 5

Sequencial: 5 Verificações

Binária: 3 Verificações

Árvore Binária: 4 Verificações

AVL: 3 Verificações

Tab. Hash: 1 Verificação

(d) 3

Sequencial: 7 Verificações

Binária: 4 Verificações

Árvore Binária: 6 Verificações

AVL: 3 Verificações

Tab. Hash: 1 Verificação

(e) 1

Sequencial: 9 Verificações

Binária: 2 Verificações

Árvore Binária: 8 Verificações

AVL: 2 Verificações

Tab. Hash: 1 Verificação