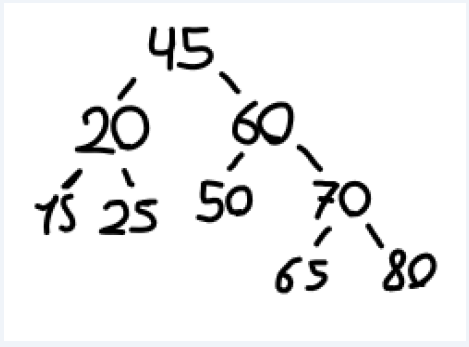
1. Construção e Remoção em Árvore Binária de Busca (BST)

a) Inserção dos valores 45, 20, 60, 15, 25, 50, 70, 65, 80:

1. **Insere 45:** 45 se torna a raiz.
2. **Insere 20:** 20 < 45, vai para a esquerda.
3. **Insere 60:** 60 > 45, vai para a direita.
4. **Insere 15:** 15 < 45 (esquerda), 15 < 20 (esquerda).
5. **Insere 25:** 25 < 45 (esquerda), 25 > 20 (direita).
6. **Insere 50:** 50 > 45 (direita), 50 < 60 (esquerda).
7. **Insere 70:** 70 > 45 (direita), 70 > 60 (direita).
8. **Insere 65:** 65 > 45 (direita), 65 > 60 (direita), 65 < 70 (esquerda).
9. **Insere 80:** 80 > 45 (direita), 80 > 60 (direita), 80 > 70 (direita).

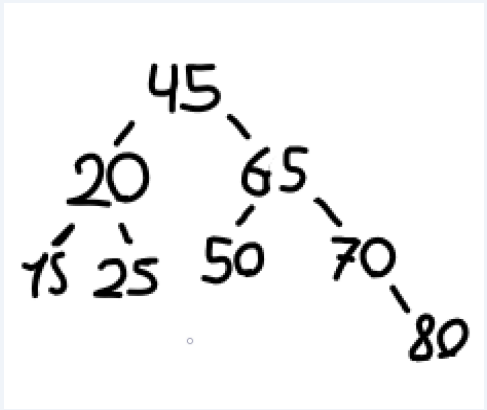
A **árvore resultante** é:



b) Remoção do nó 60:

- O nó 60 tem dois filhos (50 e 70). Para removê-lo, deve substituir pelo seu **sucessor em-ordem** (o menor nó da sub árvore direita) ou pelo seu **predecessor em-ordem** (o maior nó da sub árvore esquerda).
- Escolhi usar o sucessor: o menor nó na sub-árvore direita de 60 (que tem raiz 70) é o **65**.
- O valor 65 "sobe" para a posição do 60. O nó 65 é removido de sua posição original.

A **árvore após a remoção do 60** fica:

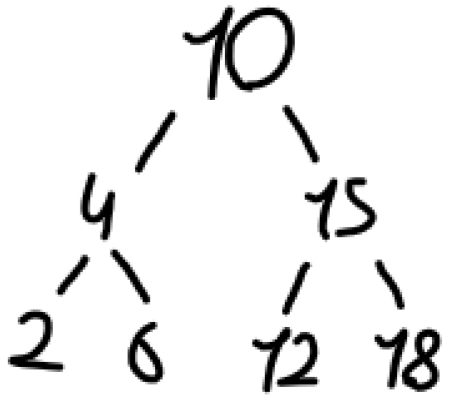


2. Percursos em Árvore

Construindo a árvore com a sequência de inserção **`10,4,15,2,6,12,18`**

1. **Insere 10:** 10 é a raiz.
2. **Insere 4:** 4 < 10, vai para a esquerda.
3. **Insere 15:** 15 > 10, vai para a direita.
4. **Insere 2:** 2 < 10 (esquerda), 2 < 4 (esquerda).
5. **Insere 6:** 6 < 10 (esquerda), 6 > 4 (direita).
6. **Insere 12:** 12 > 10 (direita), 12 < 15 (esquerda).
7. **Insere 18:** 18 > 10 (direita), 18 > 15 (direita).

A estrutura final da árvore é:



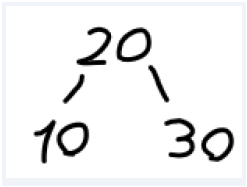
Os três percursos nesta árvore:

- **Pré-ordem (Raiz, Esquerda, Direita):** 10, 4, 2, 6, 15, 12, 18
- **Em-ordem (Esquerda, Raiz, Direita):** 2, 4, 6, 10, 12, 15, 18
- **Pós-ordem (Esquerda, Direita, Raiz):** 2, 6, 4, 12, 18, 15, 10

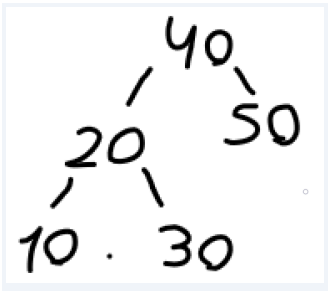
3. Rotações em Árvore AVL

Inserção de 10, 20, 30, 40, 50, 25:

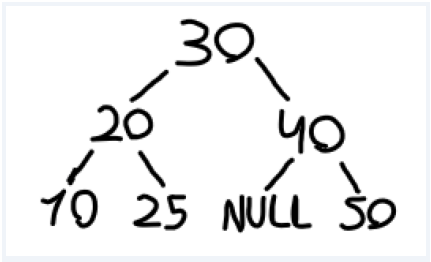
1. **Insere 10, 20:** OK.
2. **Insere 30:** O nó 10 fica desbalanceado (FB = -2). Caso **Direita-Direita (RR)**.
   - **Rotação Simples à Esquerda** em 10.
   - **Resultado:**



3. **Insere 40:** O nó 20 fica desbalanceado (FB = -2).
4. **Insere 50:** O nó 30 fica desbalanceado (FB = -2). Caso **Direita-Direita (RR)**.
   - **Rotação Simples à Esquerda** em 30.
   - **Resultado (sub árvore):** 40 com filhos 30 e 50. A árvore completa fica desbalanceada em 20 (FB=-2).
   - **Rotação Simples à Esquerda** em 20.
   - **Resultado:**



5. **Insere 25:** O nó 30 fica desbalanceado (FB = 1), mas a árvore como um todo não. O nó 20 fica desbalanceado. O caso é **Direita-Esquerda (RL)** no nó 20.
   - **Rotação à Direita** no filho à direita (30). O 25 sobe.
   - **Rotação à Esquerda** na raiz do desbalanceamento (20). O 25 sobe.
   - **Correção:** O desbalanceamento ocorre no nó 40. A inserção do 25 faz o nó 30 ter altura 1. A subárvore esquerda (raiz 20) tem altura 2, a direita (raiz 50) tem altura 0. FB(40)=2. O caso é **Esquerda-Direita (LR)**.
   1. **Rotação à Esquerda** no filho à esquerda (20). O nó 30 sobe.
   2. **Rotação à Direita** na raiz do desbalanceamento (40). O nó 30 sobe.
   - **Resultado Final:**



4. Grafo: Matriz e Percursos

a) Matriz de Adjacência:

```
   A B C D E F
A [0,1,1,0,0,0]
B [1,0,0,1,1,0]
C [1,0,0,0,0,1]
D [0,1,0,0,0,0]
E [0,1,0,0,0,1]
F [0,0,1,0,1,0]
```

b) Busca em Largura (BFS) iniciando em A:

- **Fila:** [A]
- **Visita:** A
- **Tira A, adiciona B, C:** Fila [B, C]
- **Visita:** B, C
- **Tira B, adiciona D, E:** Fila [C, D, E]
- **Visita:** D, E
- **Tira C, adiciona F:** Fila [D, E, F]
- **Visita:** F
- **Tira D, E, F:** Fila Vazia. Fim.
- **Ordem de Percurso BFS:** **A, B, C, D, E, F**

c) Busca em Profundidade (DFS) iniciando em A:

- **Visita A.**
- **Vai para B** (primeiro vizinho em ordem alfabética). **Visita B.**
- **Vai para D** (primeiro vizinho de B não visitado). **Visita D.**
- D não tem mais vizinhos não visitados. **Volta para B.**
- **Vai para E** (próximo vizinho de B). **Visita E.**
- **Vai para F** (primeiro vizinho de E). **Visita F.**
- **Vai para C** (primeiro vizinho de F). **Visita C.**
- C já tem todos os vizinhos visitados. **Volta para F, E, B, A.** Fim.
- **Ordem de Percurso DFS:** **A, B, D, E, F, C**