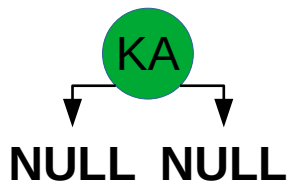
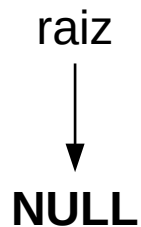


Questão 5 – ABB

criarABB():
raiz ← NULL;

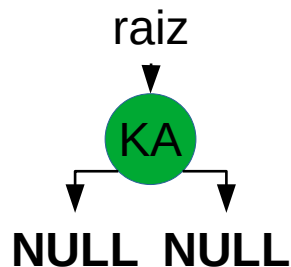
raiz
↓
NULL

Inserir: KA JA XO CA TE UX PL MN FI LI MA



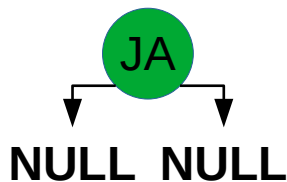
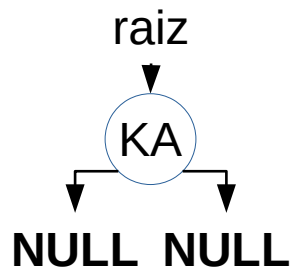
```
inserirIterativamente(umValor):  
novo ← criar_noh(umValor);  
se (raiz = NULL) {  
    raiz ← novo;  
}  
senão {  
    atual ← raiz;  
    enquanto (atual ≠ NULL) {  
        anterior ← atual;  
        se (atual.valor > umValor) {  
            atual ← atual.esquerdo;  
        } senão {  
            atual ← atual.direito;  
        }  
    }  
    novo.pai ← anterior;  
    se (anterior.valor > novo.valor) {  
        anterior.esquerdo ← novo;  
    } senão {  
        anterior.direito ← novo;  
    }  
}
```

Inserir: KA JA XO CA TE UX PL MN FI LI MA



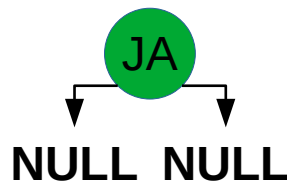
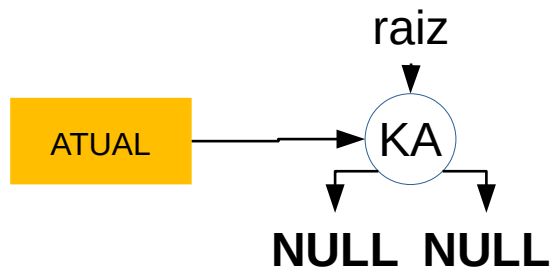
```
inserirIterativamente(umValor):  
novo ← criar_noh(umValor);  
se (raiz = NULL) {  
    raiz ← novo;  
}  
senão {  
    atual ← raiz;  
    enquanto (atual ≠ NULL) {  
        anterior ← atual;  
        se (atual.valor > umValor) {  
            atual ← atual.esquerdo;  
        } senão {  
            atual ← atual.direito;  
        }  
    }  
    novo.pai ← anterior;  
    se (anterior.valor > novo.valor) {  
        anterior.esquerdo ← novo;  
    } senão {  
        anterior.direito ← novo;  
    }  
}
```

Inserir: KA JA XO CA TE UX PL MN FI LI MA



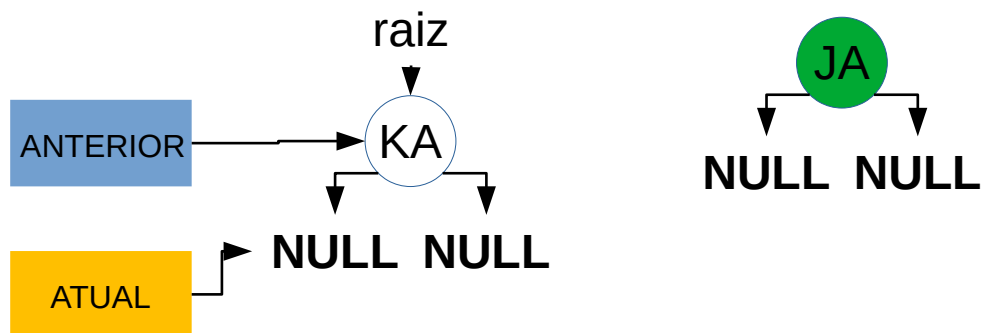
```
inserirIterativamente(umValor):  
novo ← criar_noh(umValor);  
se (raiz = NULL) {  
    raiz ← novo;  
}  
senão {  
    atual ← raiz;  
    enquanto (atual ≠ NULL) {  
        anterior ← atual;  
        se (atual.valor > umValor) {  
            atual ← atual.esquerdo;  
        } senão {  
            atual ← atual.direito;  
        }  
    }  
    novo.pai ← anterior;  
    se (anterior.valor > novo.valor) {  
        anterior.esquerdo ← novo;  
    } senão {  
        anterior.direito ← novo;  
    }  
}
```

Inserir: KA JA XO CA TE UX PL MN FI LI MA



```
inserirIterativamente(umValor):  
novo ← criar_noh(umValor);  
se (raiz = NULL) {  
    raiz ← novo;  
}  
senão {  
    atual ← raiz;  
    enquanto (atual ≠ NULL) {  
        anterior ← atual;  
        se (atual.valor > umValor) {  
            atual ← atual.esquerdo;  
        } senão {  
            atual ← atual.direito;  
        }  
    }  
    novo.pai ← anterior;  
    se (anterior.valor > novo.valor) {  
        anterior.esquerdo ← novo;  
    } senão {  
        anterior.direito ← novo;  
    }  
}
```

Inserir: KA JA XO CA TE UX PL MN FI LI MA

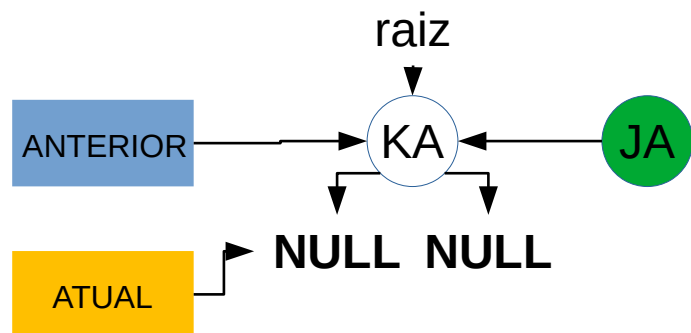


inserirIterativamente(umValor):

```

novo ← criar_noh(umValor);
se (raiz = NULL) {
    raiz ← novo;
}
senão {
    atual ← raiz;
    enquanto (atual ≠ NULL) {
        anterior ← atual;
        se (atual.valor > umValor) {
            atual ← atual.esquerdo;
        } senão {
            atual ← atual.direito;
        }
    }
    novo.pai ← anterior;
    se (anterior.valor > novo.valor) {
        anterior.esquerdo ← novo;
    } senão {
        anterior.direito ← novo;
    }
}
```

Inserir: KA JA XO CA TE UX PL MN FI LI MA

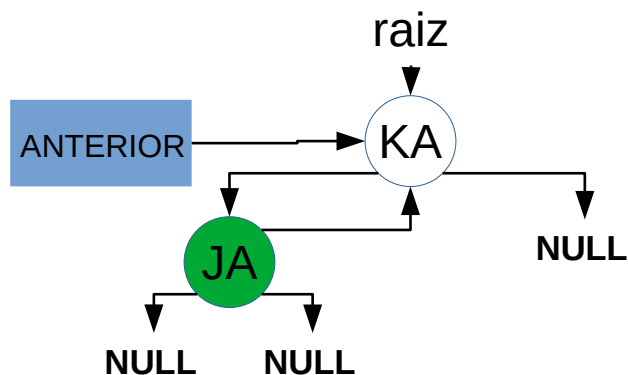


inserirIterativamente(umValor):

```

novo ← criar_noh(umValor);
se (raiz = NULL) {
    raiz ← novo;
}
senão {
    atual ← raiz;
    enquanto (atual ≠ NULL) {
        anterior ← atual;
        se (atual.valor > umValor) {
            atual ← atual.esquerdo;
        } senão {
            atual ← atual.direito;
        }
    }
    novo.pai ← anterior;
    se (anterior.valor > novo.valor) {
        anterior.esquerdo ← novo;
    } senão {
        anterior.direito ← novo;
    }
}
```

Inserir: KA JA XO CA TE UX PL MN FI LI MA

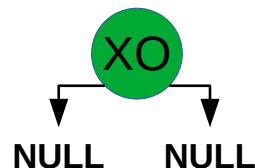
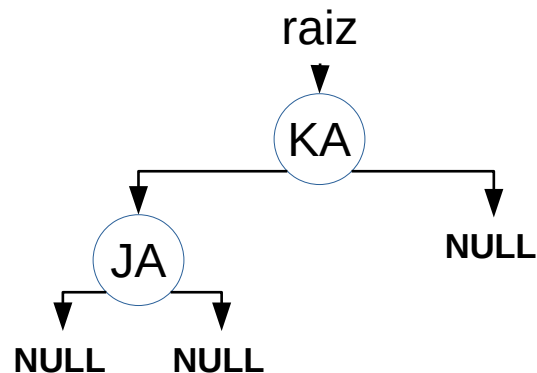


inserirIterativamente(umValor):

```

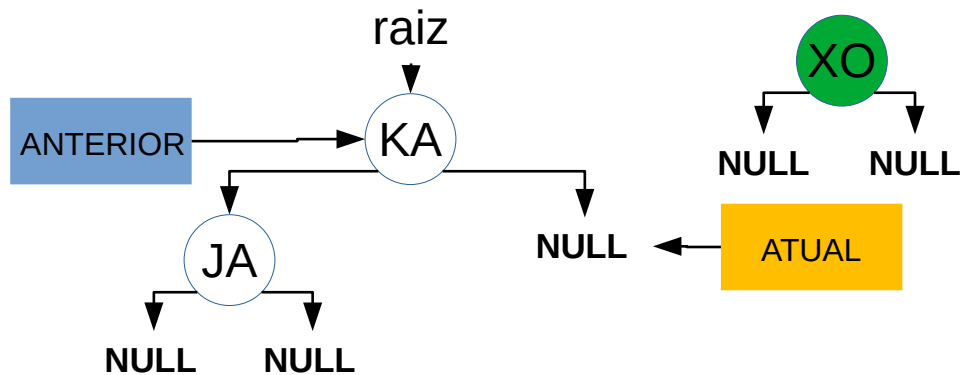
novo ← criar_noh(umValor);
se (raiz = NULL) {
    raiz ← novo;
}
senão {
    atual ← raiz;
    enquanto (atual ≠ NULL) {
        anterior ← atual;
        se (atual.valor > umValor) {
            atual ← atual.esquerdo;
        } senão {
            atual ← atual.direito;
        }
    }
    novo.pai ← anterior;
    se (anterior.valor > novo.valor) {
        anterior.esquerdo ← novo;
    } senão {
        anterior.direito ← novo;
    }
}
```


Inserir: KA JA XO CA TE UX PL MN FI LI MA



```
inserirIterativamente(umValor):  
novo ← criar_noh(umValor);  
se (raiz = NULL) {  
    raiz ← novo;  
}  
senão {  
    atual ← raiz;  
    enquanto (atual ≠ NULL) {  
        anterior ← atual;  
        se (atual.valor > umValor) {  
            atual ← atual.esquerdo;  
        } senão {  
            atual ← atual.direito;  
        }  
    }  
    novo.pai ← anterior;  
    se (anterior.valor > novo.valor) {  
        anterior.esquerdo ← novo;  
    } senão {  
        anterior.direito ← novo;  
    }  
}
```

Inserir: KA JA XO CA TE UX PL MN FI LI MA

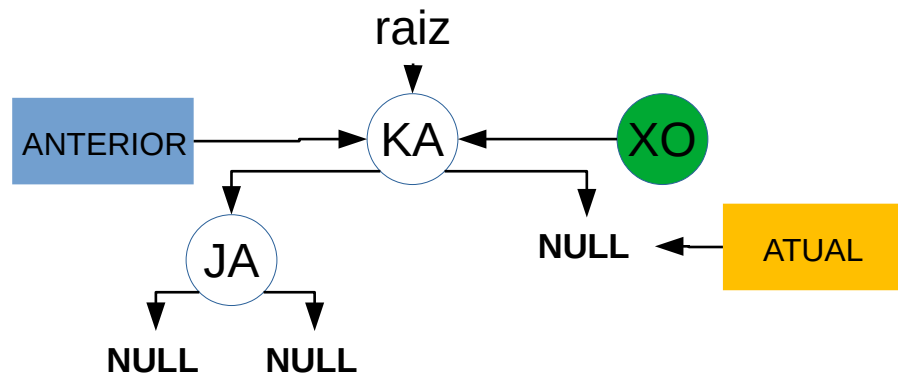


inserirIterativamente(umValor):

```

novo ← criar_noh(umValor);
se (raiz = NULL) {
    raiz ← novo;
}
senão {
    atual ← raiz;
    enquanto (atual ≠ NULL) {
        anterior ← atual;
        se (atual.valor > umValor) {
            atual ← atual.esquerdo;
        } senão {
            atual ← atual.direito;
        }
    }
    novo.pai ← anterior;
    se (anterior.valor > novo.valor) {
        anterior.esquerdo ← novo;
    } senão {
        anterior.direito ← novo;
    }
}
```

Inserir: KA JA XO CA TE UX PL MN FI LI MA

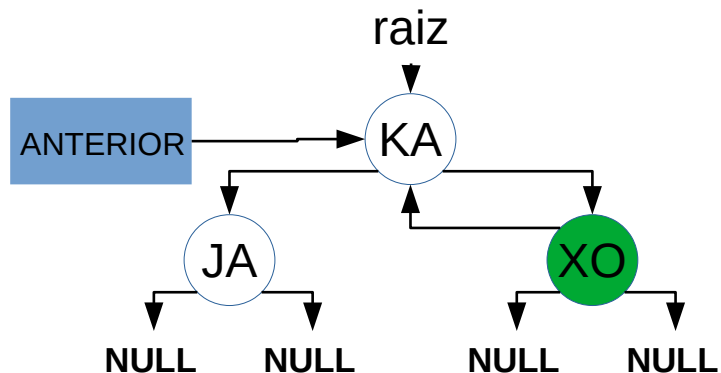


inserirIterativamente(umValor):

```

novo ← criar_noh(umValor);
se (raiz = NULL) {
    raiz ← novo;
}
senão {
    atual ← raiz;
    enquanto (atual ≠ NULL) {
        anterior ← atual;
        se (atual.valor > umValor) {
            atual ← atual.esquerdo;
        } senão {
            atual ← atual.direito;
        }
    }
    novo.pai ← anterior;
    se (anterior.valor > novo.valor) {
        anterior.esquerdo ← novo;
    } senão {
        anterior.direito ← novo;
    }
}
```

Inserir: KA JA XO CA TE UX PL MN FI LI MA

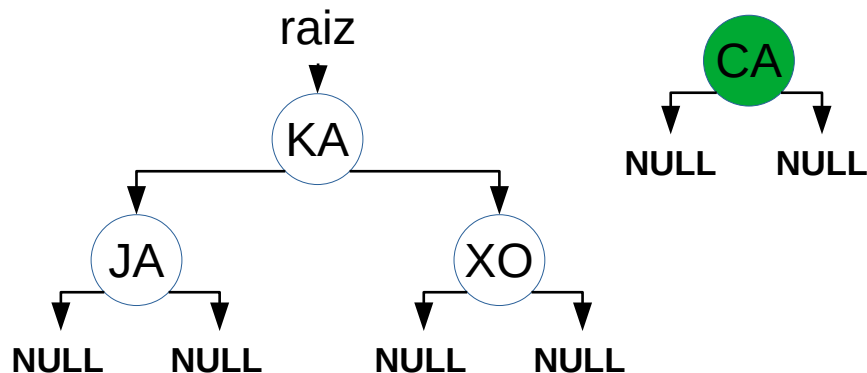


inserirIterativamente(umValor):

```

novo ← criar_noh(umValor);
se (raiz = NULL) {
    raiz ← novo;
}
senão {
    atual ← raiz;
    enquanto (atual ≠ NULL) {
        anterior ← atual;
        se (atual.valor > umValor) {
            atual ← atual.esquerdo;
        } senão {
            atual ← atual.direito;
        }
    }
    novo.pai ← anterior;
    se (anterior.valor > novo.valor) {
        anterior.esquerdo ← novo;
    } senão {
        anterior.direito ← novo;
    }
}
```

Inserir: KA JA XO **CA** TE UX PL MN FI LI MA

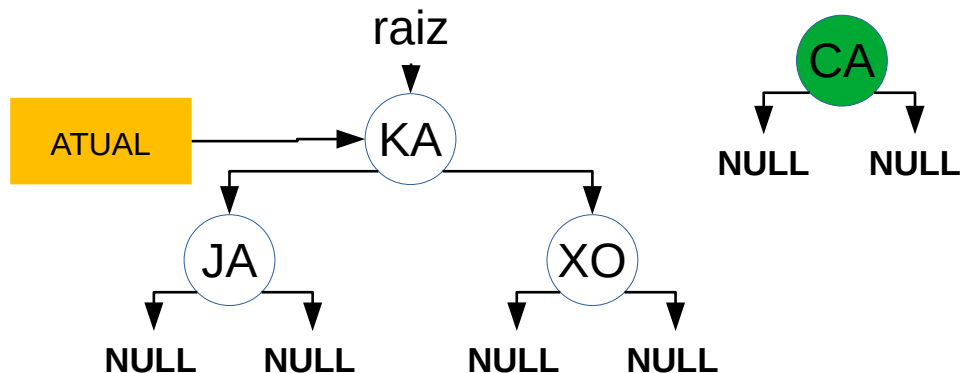


inserirIterativamente(umValor):

```

novo ← criar_noh(umValor);
se (raiz = NULL) {
    raiz ← novo;
}
senão {
    atual ← raiz;
    enquanto (atual ≠ NULL) {
        anterior ← atual;
        se (atual.valor > umValor) {
            atual ← atual.esquerdo;
        } senão {
            atual ← atual.direito;
        }
    }
    novo.pai ← anterior;
    se (anterior.valor > novo.valor) {
        anterior.esquerdo ← novo;
    } senão {
        anterior.direito ← novo;
    }
}
```

Inserir: KA JA XO CA TE UX PL MN FI LI MA

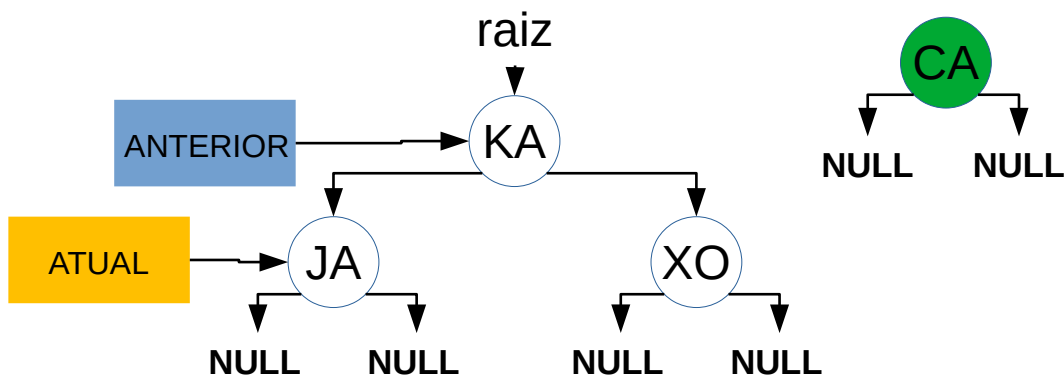


inserirIterativamente(umValor):

```

novo ← criar_noh(umValor);
se (raiz = NULL) {
    raiz ← novo;
}
senão {
    atual ← raiz;
    enquanto (atual ≠ NULL) {
        anterior ← atual;
        se (atual.valor > umValor) {
            atual ← atual.esquerdo;
        } senão {
            atual ← atual.direito;
        }
    }
    novo.pai ← anterior;
    se (anterior.valor > novo.valor) {
        anterior.esquerdo ← novo;
    } senão {
        anterior.direito ← novo;
    }
}
```

Inserir: KA JA XO **CA** TE UX PL MN FI LI MA

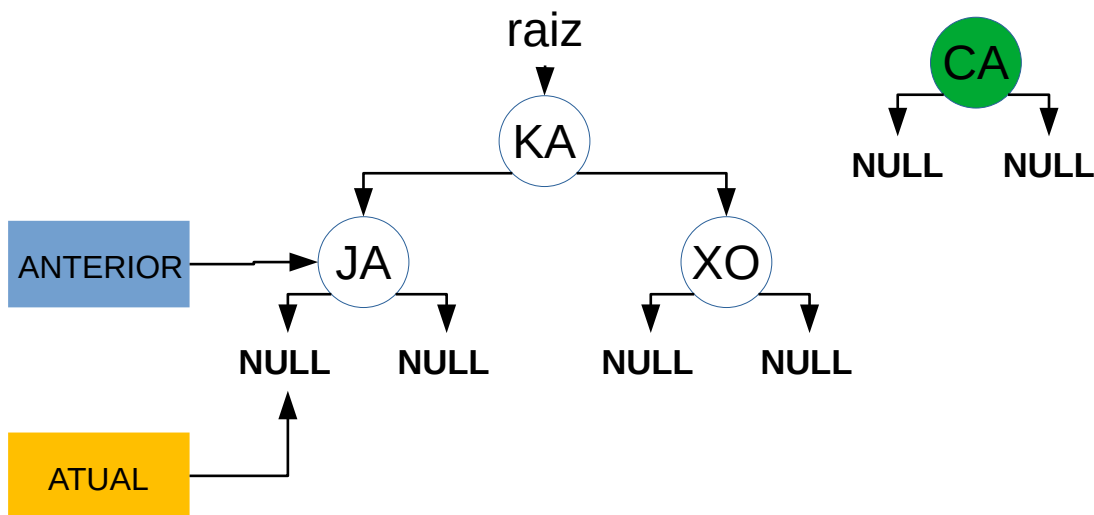


inserirIterativamente(umValor):

```

novo ← criar_noh(umValor);
se (raiz = NULL) {
    raiz ← novo;
}
senão {
    atual ← raiz;
    enquanto (atual ≠ NULL) {
        anterior ← atual;
        se (atual.valor > umValor) {
            atual ← atual.esquerdo;
        } senão {
            atual ← atual.direito;
        }
    }
    novo.pai ← anterior;
    se (anterior.valor > novo.valor) {
        anterior.esquerdo ← novo;
    } senão {
        anterior.direito ← novo;
    }
}
```

Inserir: KA JA XO **CA** TE UX PL MN FI LI MA

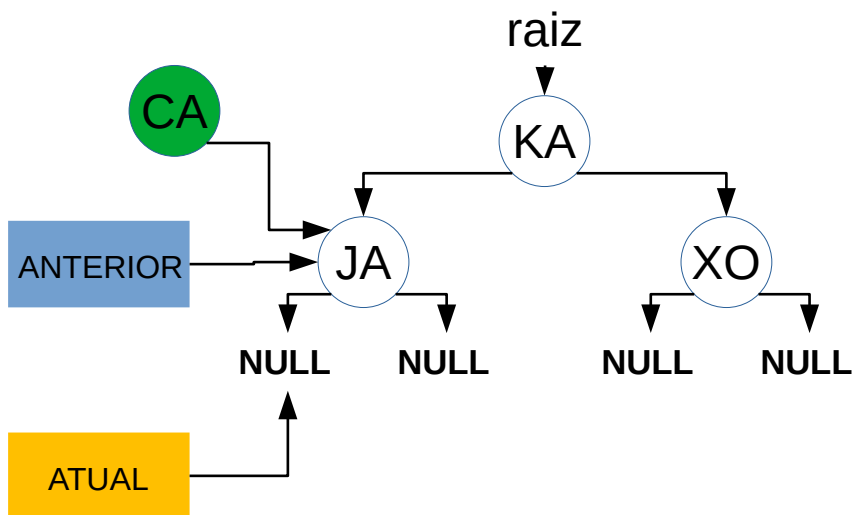


inserirIterativamente(umValor):

```

novo ← criar_noh(umValor);
se (raiz = NULL) {
    raiz ← novo;
}
senão {
    atual ← raiz;
    enquanto (atual ≠ NULL) {
        anterior ← atual;
        se (atual.valor > umValor) {
            atual ← atual.esquerdo;
        } senão {
            atual ← atual.direito;
        }
    }
    novo.pai ← anterior;
    se (anterior.valor > novo.valor) {
        anterior.esquerdo ← novo;
    } senão {
        anterior.direito ← novo;
    }
}
```


Inserir: KA JA XO **CA** TE UX PL MN FI LI MA

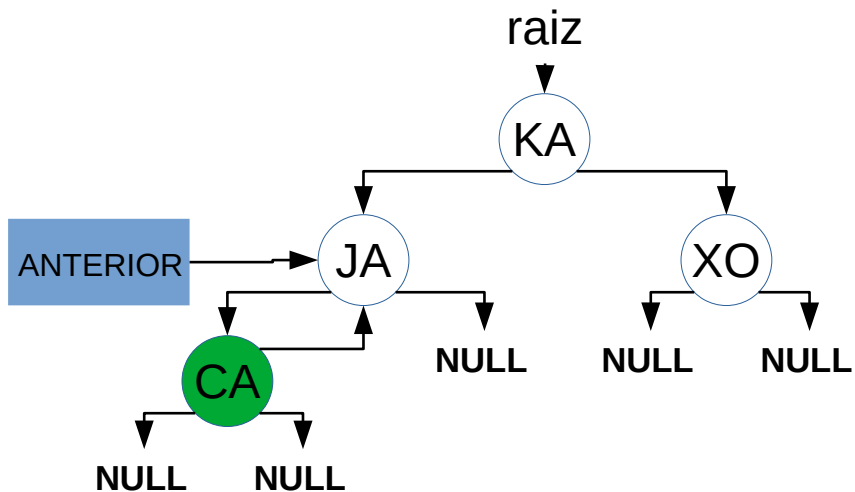


inserirIterativamente(umValor):

```

novo ← criar_noh(umValor);
se (raiz = NULL) {
    raiz ← novo;
}
senão {
    atual ← raiz;
    enquanto (atual ≠ NULL) {
        anterior ← atual;
        se (atual.valor > umValor) {
            atual ← atual.esquerdo;
        } senão {
            atual ← atual.direito;
        }
    }
    novo.pai ← anterior;
    se (anterior.valor > novo.valor) {
        anterior.esquerdo ← novo;
    } senão {
        anterior.direito ← novo;
    }
}
```

Inserir: KA JA XO **CA** TE UX PL MN FI LI MA

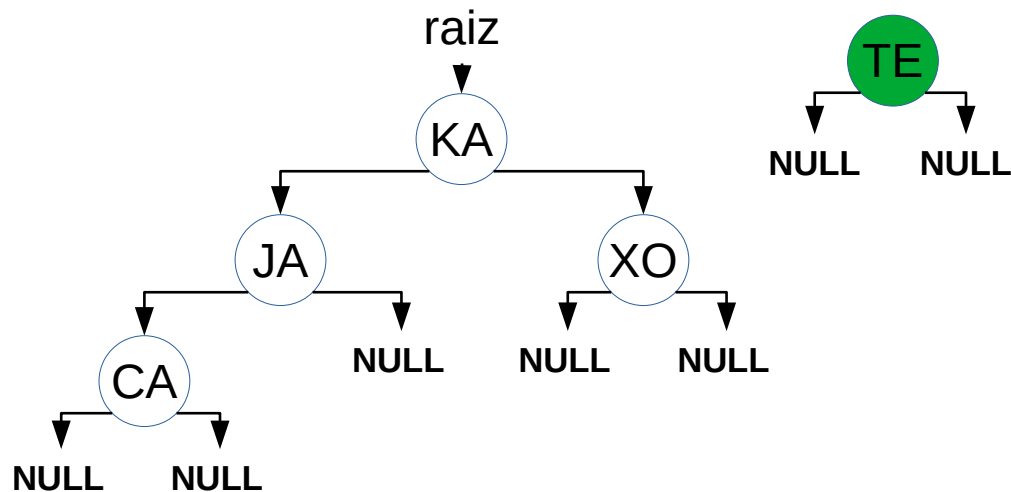


inserirIterativamente(umValor):

```

novo ← criar_noh(umValor);
se (raiz = NULL) {
    raiz ← novo;
}
senão {
    atual ← raiz;
    enquanto (atual ≠ NULL) {
        anterior ← atual;
        se (atual.valor > umValor) {
            atual ← atual.esquerdo;
        } senão {
            atual ← atual.direito;
        }
    }
    novo.pai ← anterior;
    se (anterior.valor > novo.valor) {
        anterior.esquerdo ← novo;
    } senão {
        anterior.direito ← novo;
    }
}
```

Inserir: KA JA XO CA **TE** UX PL MN FI LI MA

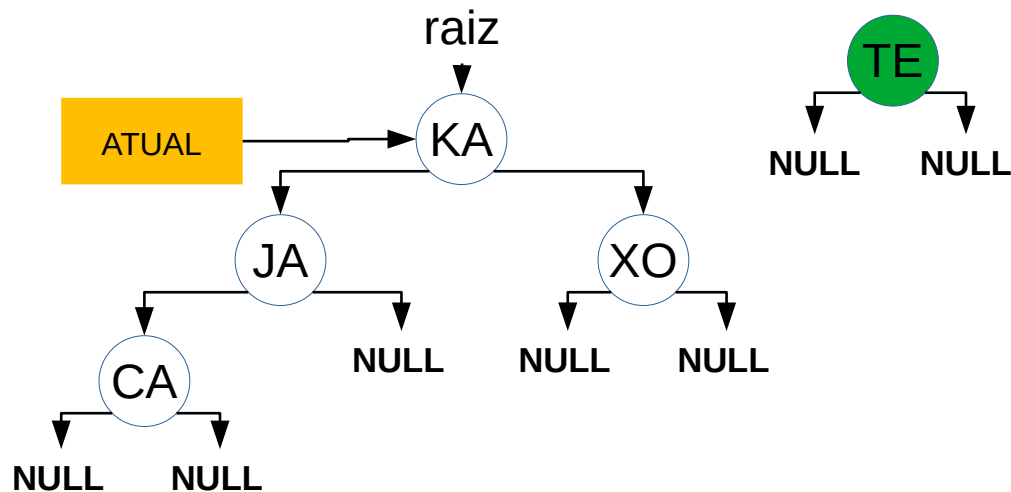


inserirIterativamente(umValor):

```

novo ← criar_noh(umValor);
se (raiz = NULL) {
    raiz ← novo;
}
senão {
    atual ← raiz;
    enquanto (atual ≠ NULL) {
        anterior ← atual;
        se (atual.valor > umValor) {
            atual ← atual.esquerdo;
        } senão {
            atual ← atual.direito;
        }
    }
    novo.pai ← anterior;
    se (anterior.valor > novo.valor) {
        anterior.esquerdo ← novo;
    } senão {
        anterior.direito ← novo;
    }
}
```

Inserir: KA JA XO CA **TE** UX PL MN FI LI MA

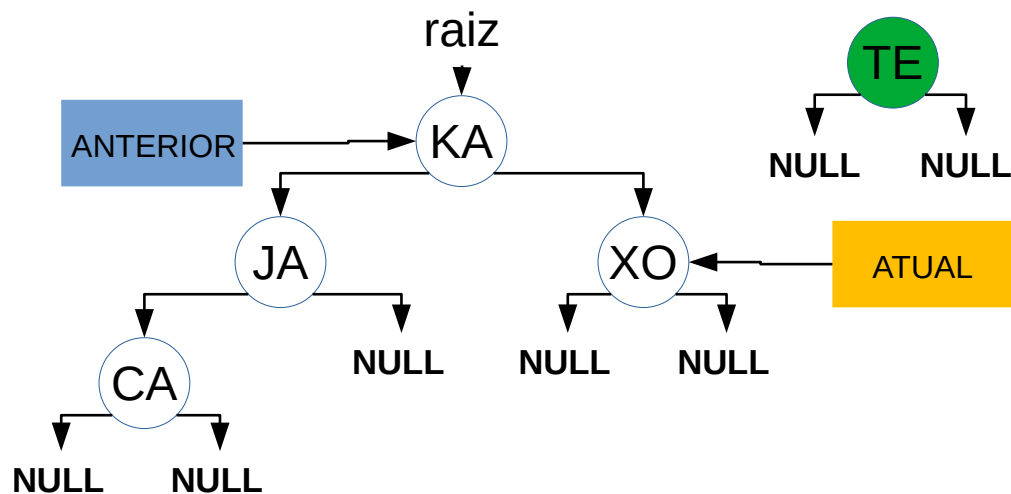


inserirIterativamente(umValor):

```

novo ← criar_noh(umValor);
se (raiz = NULL) {
    raiz ← novo;
}
senão {
    atual ← raiz;
    enquanto (atual ≠ NULL) {
        anterior ← atual;
        se (atual.valor > umValor) {
            atual ← atual.esquerdo;
        } senão {
            atual ← atual.direito;
        }
    }
    novo.pai ← anterior;
    se (anterior.valor > novo.valor) {
        anterior.esquerdo ← novo;
    } senão {
        anterior.direito ← novo;
    }
}
}
```

Inserir: KA JA XO CA **TE** UX PL MN FI LI MA

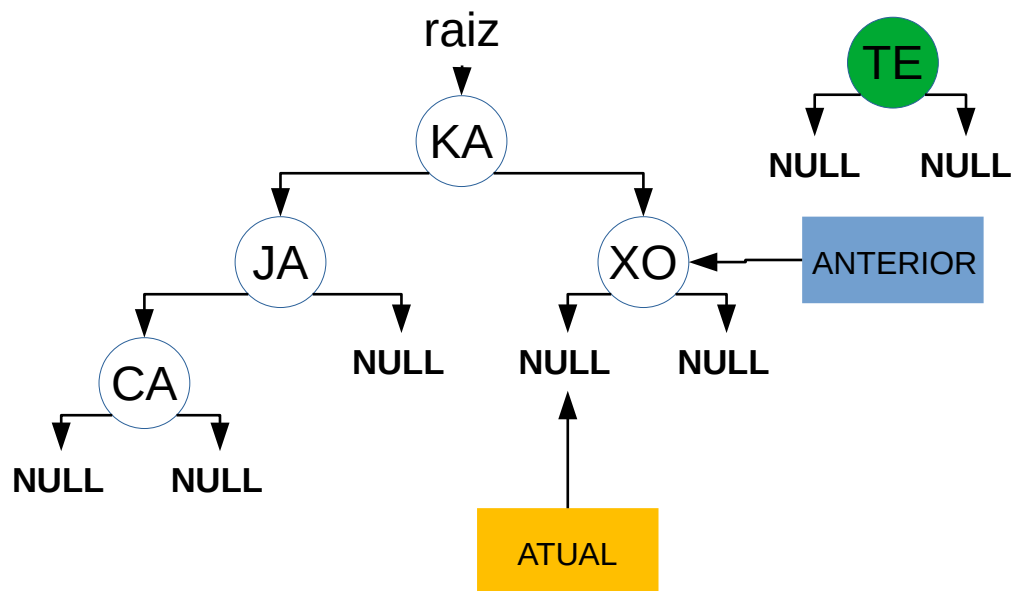


inserirIterativamente(umValor):

```

novo ← criar_noh(umValor);
se (raiz = NULL) {
    raiz ← novo;
}
senão {
    atual ← raiz;
    enquanto (atual ≠ NULL) {
        anterior ← atual;
        se (atual.valor > umValor) {
            atual ← atual.esquerdo;
        } senão {
            atual ← atual.direito;
        }
    }
    novo.pai ← anterior;
    se (anterior.valor > novo.valor) {
        anterior.esquerdo ← novo;
    } senão {
        anterior.direito ← novo;
    }
}
```

Inserir: KA JA XO CA **TE** UX PL MN FI LI MA

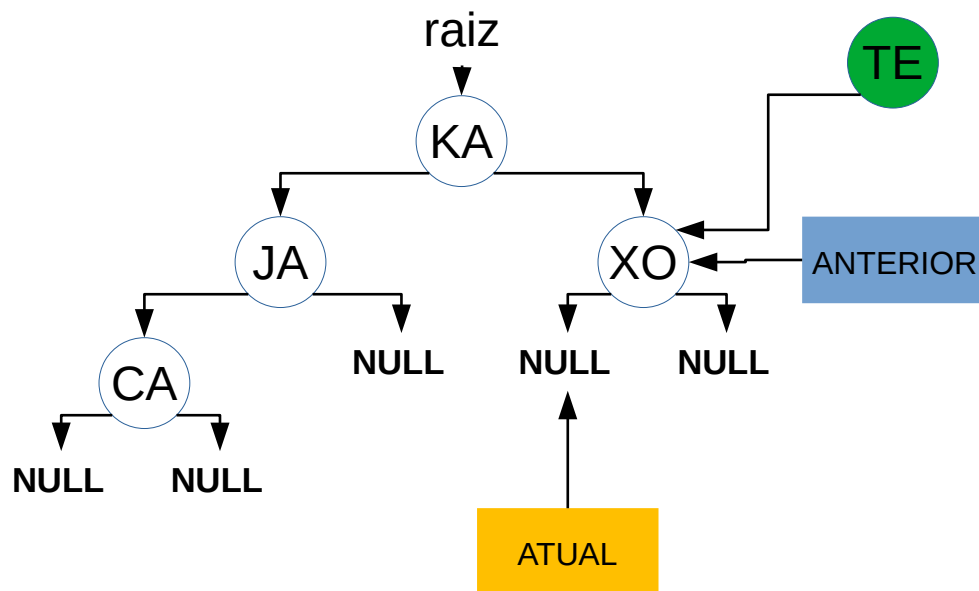


inserirIterativamente(umValor):

```

novo ← criar_noh(umValor);
se (raiz = NULL) {
    raiz ← novo;
}
senão {
    atual ← raiz;
    enquanto (atual ≠ NULL) {
        anterior ← atual;
        se (atual.valor > umValor) {
            atual ← atual.esquerdo;
        } senão {
            atual ← atual.direito;
        }
    }
    novo.pai ← anterior;
    se (anterior.valor > novo.valor) {
        anterior.esquerdo ← novo;
    } senão {
        anterior.direito ← novo;
    }
}
```

Inserir: KA JA XO CA **TE** UX PL MN FI LI MA

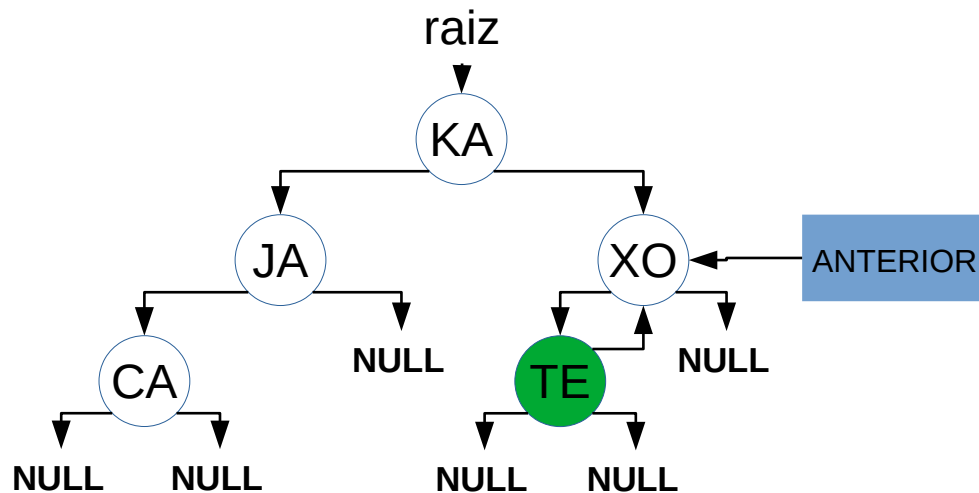


inserirIterativamente(umValor):

```

novo ← criar_noh(umValor);
se (raiz = NULL) {
    raiz ← novo;
}
senão {
    atual ← raiz;
    enquanto (atual ≠ NULL) {
        anterior ← atual;
        se (atual.valor > umValor) {
            atual ← atual.esquerdo;
        } senão {
            atual ← atual.direito;
        }
    }
    novo.pai ← anterior;
    se (anterior.valor > novo.valor) {
        anterior.esquerdo ← novo;
    } senão {
        anterior.direito ← novo;
    }
}
```

Inserir: KA JA XO CA **TE** UX PL MN FI LI MA

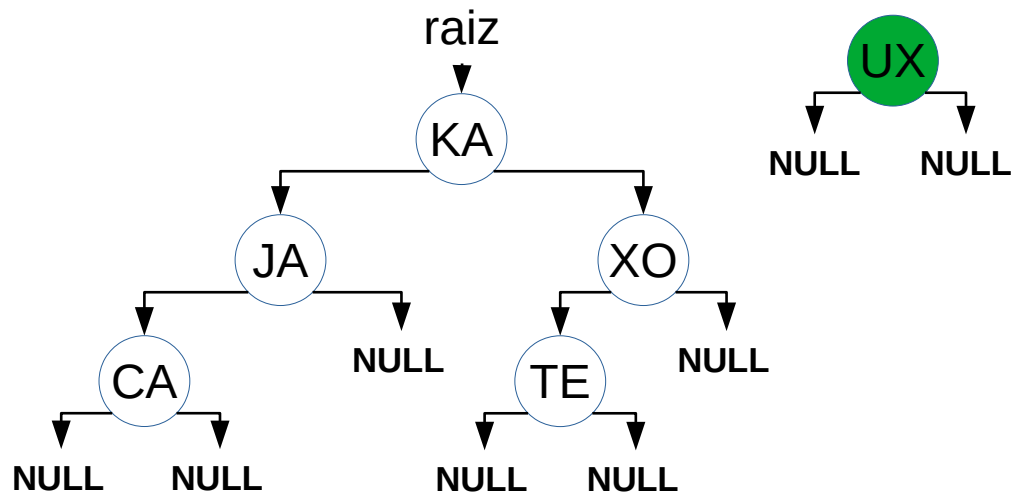


inserirIterativamente(umValor):

```

novo ← criar_noh(umValor);
se (raiz = NULL) {
    raiz ← novo;
}
senão {
    atual ← raiz;
    enquanto (atual ≠ NULL) {
        anterior ← atual;
        se (atual.valor > umValor) {
            atual ← atual.esquerdo;
        } senão {
            atual ← atual.direito;
        }
    }
    novo.pai ← anterior;
    se (anterior.valor > novo.valor) {
        anterior.esquerdo ← novo;
    } senão {
        anterior.direito ← novo;
    }
}
```


Inserir: KA JA XO CA TE UX PL MN FI LI MA

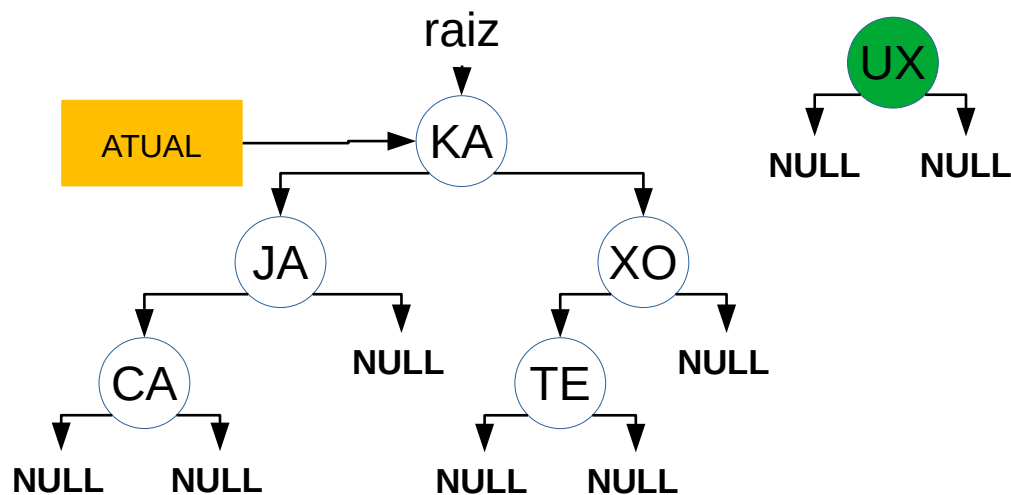


inserirIterativamente(umValor):

```

novo ← criar_noh(umValor);
se (raiz = NULL) {
    raiz ← novo;
}
senão {
    atual ← raiz;
    enquanto (atual ≠ NULL) {
        anterior ← atual;
        se (atual.valor > umValor) {
            atual ← atual.esquerdo;
        } senão {
            atual ← atual.direito;
        }
    }
    novo.pai ← anterior;
    se (anterior.valor > novo.valor) {
        anterior.esquerdo ← novo;
    } senão {
        anterior.direito ← novo;
    }
}
```

Inserir: KA JA XO CA TE UX PL MN FI LI MA

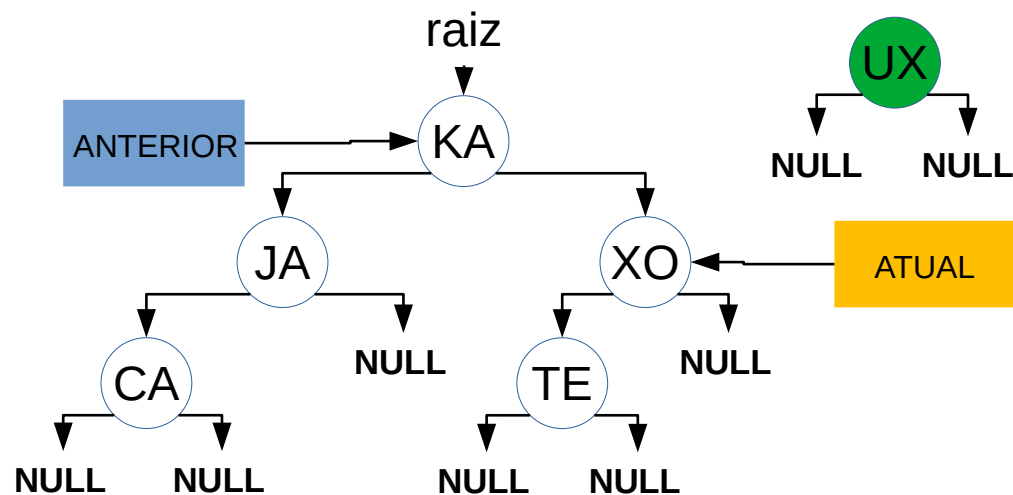


inserirIterativamente(umValor):

```

novo ← criar_noh(umValor);
se (raiz = NULL) {
    raiz ← novo;
}
senão {
    atual ← raiz;
    enquanto (atual ≠ NULL) {
        anterior ← atual;
        se (atual.valor > umValor) {
            atual ← atual.esquerdo;
        } senão {
            atual ← atual.direito;
        }
    }
    novo.pai ← anterior;
    se (anterior.valor > novo.valor) {
        anterior.esquerdo ← novo;
    } senão {
        anterior.direito ← novo;
    }
}
```

Inserir: KA JA XO CA TE UX PL MN FI LI MA

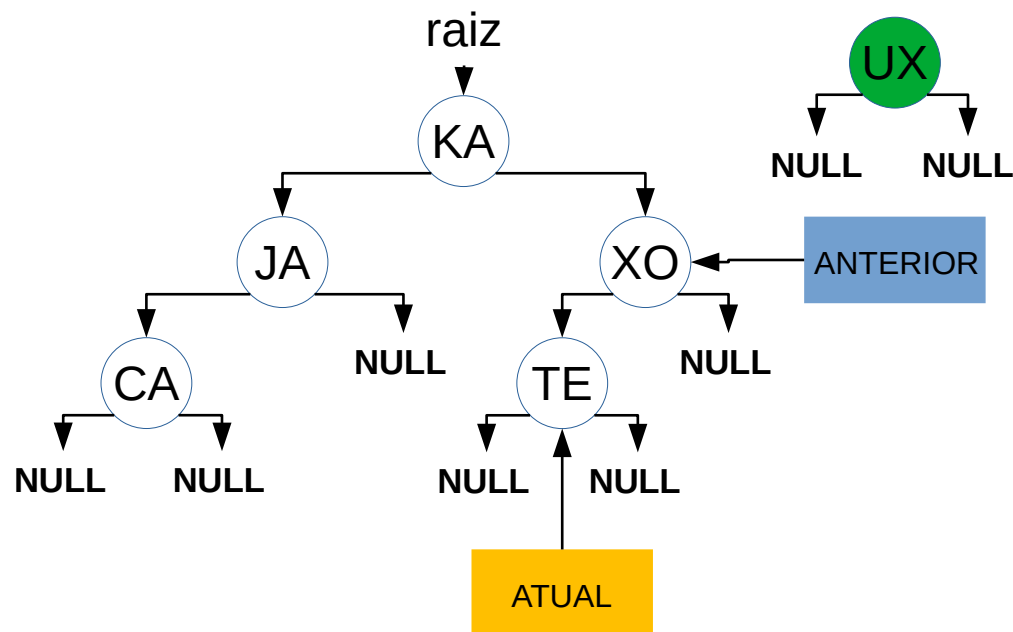


inserirIterativamente(umValor):

```

novo ← criar_noh(umValor);
se (raiz = NULL) {
    raiz ← novo;
}
senão {
    atual ← raiz;
    enquanto (atual ≠ NULL) {
        anterior ← atual;
        se (atual.valor > umValor) {
            atual ← atual.esquerdo;
        } senão {
            atual ← atual.direito;
        }
    }
    novo.pai ← anterior;
    se (anterior.valor > novo.valor) {
        anterior.esquerdo ← novo;
    } senão {
        anterior.direito ← novo;
    }
}
```

Inserir: KA JA XO CA TE UX PL MN FI LI MA

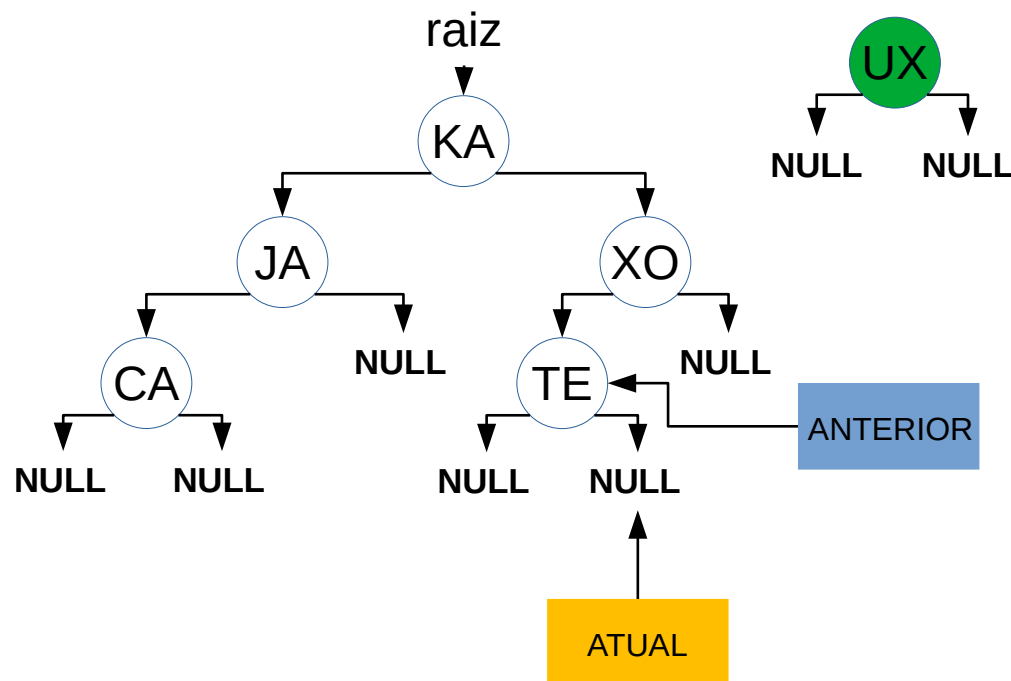


inserirIterativamente(umValor):

```

novo ← criar_noh(umValor);
se (raiz = NULL) {
    raiz ← novo;
}
senão {
    atual ← raiz;
    enquanto (atual ≠ NULL) {
        anterior ← atual;
        se (atual.valor > umValor) {
            atual ← atual.esquerdo;
        } senão {
            atual ← atual.direito;
        }
    }
    novo.pai ← anterior;
    se (anterior.valor > novo.valor) {
        anterior.esquerdo ← novo;
    } senão {
        anterior.direito ← novo;
    }
}
```

Inserir: KA JA XO CA TE UX PL MN FI LI MA

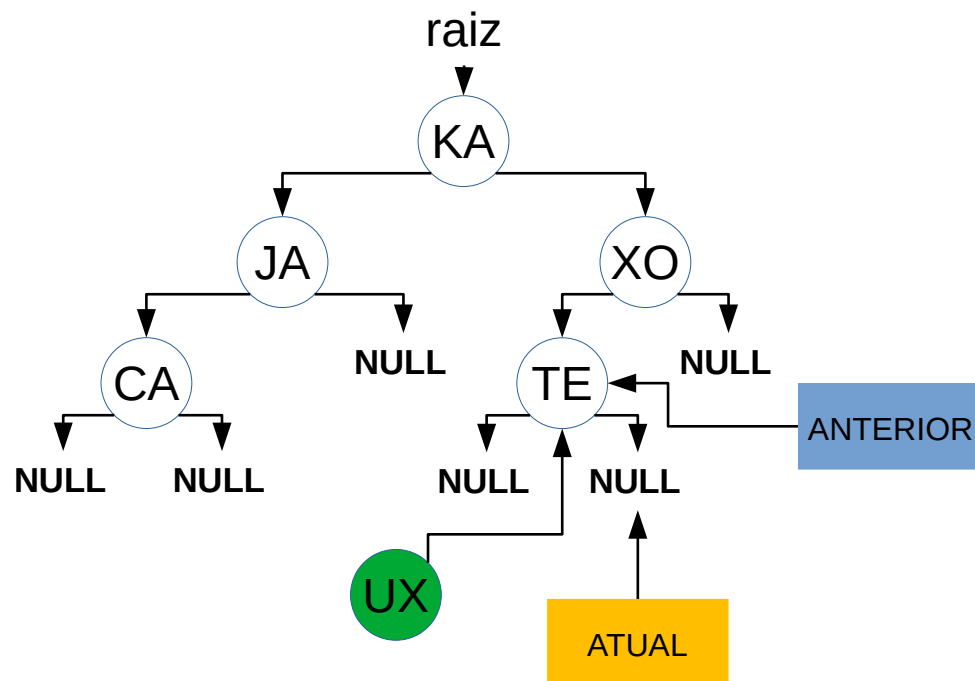


inserirIterativamente(umValor):

```

novo ← criar_noh(umValor);
se (raiz = NULL) {
    raiz ← novo;
}
senão {
    atual ← raiz;
    enquanto (atual ≠ NULL) {
        anterior ← atual;
        se (atual.valor > umValor) {
            atual ← atual.esquerdo;
        } senão {
            atual ← atual.direito;
        }
    }
    novo.pai ← anterior;
    se (anterior.valor > novo.valor) {
        anterior.esquerdo ← novo;
    } senão {
        anterior.direito ← novo;
    }
}
```

Inserir: KA JA XO CA TE UX PL MN FI LI MA

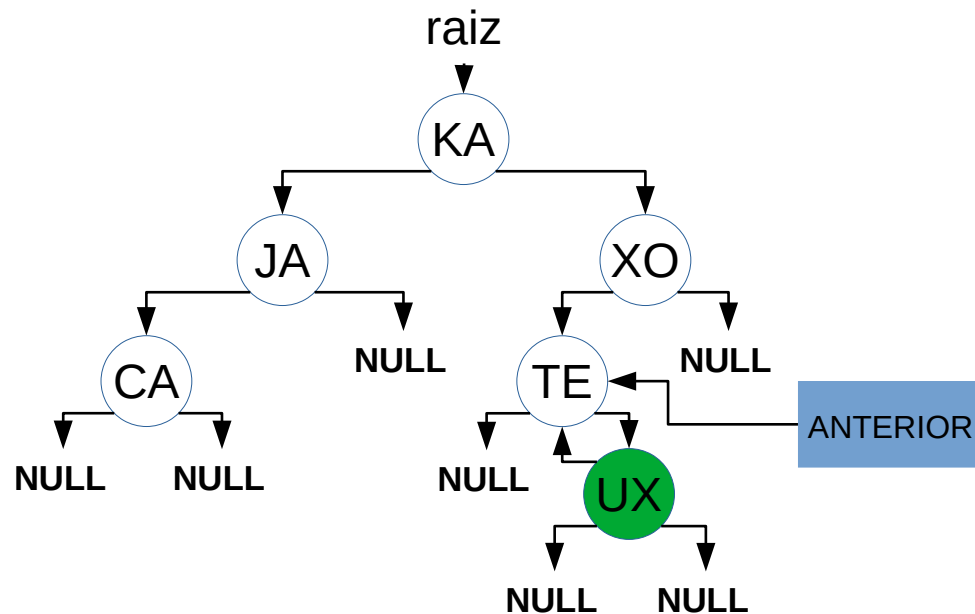


inserirIterativamente(umValor):

```

novo ← criar_noh(umValor);
se (raiz = NULL) {
    raiz ← novo;
}
senão {
    atual ← raiz;
    enquanto (atual ≠ NULL) {
        anterior ← atual;
        se (atual.valor > umValor) {
            atual ← atual.esquerdo;
        } senão {
            atual ← atual.direito;
        }
    }
    novo.pai ← anterior;
    se (anterior.valor > novo.valor) {
        anterior.esquerdo ← novo;
    } senão {
        anterior.direito ← novo;
    }
}
```

Inserir: KA JA XO CA TE UX PL MN FI LI MA

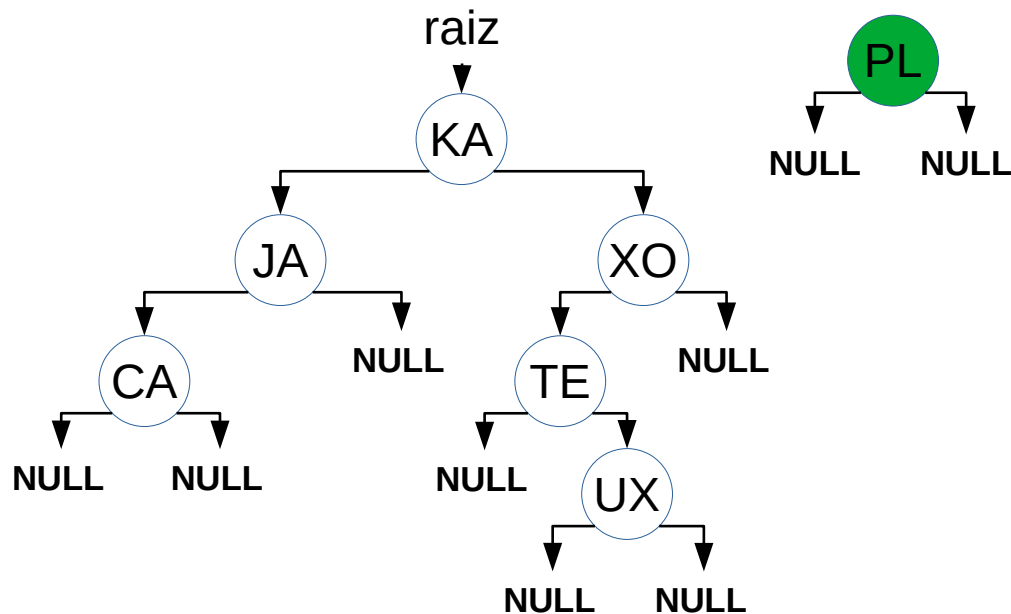


inserirIterativamente(umValor):

```

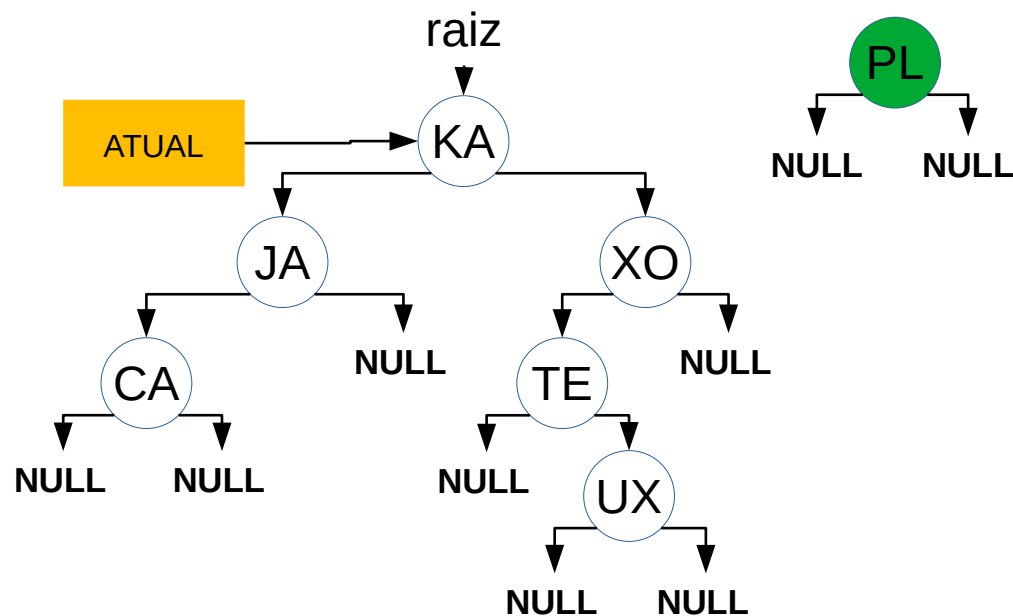
novo ← criar_noh(umValor);
se (raiz = NULL) {
    raiz ← novo;
}
senão {
    atual ← raiz;
    enquanto (atual ≠ NULL) {
        anterior ← atual;
        se (atual.valor > umValor) {
            atual ← atual.esquerdo;
        } senão {
            atual ← atual.direito;
        }
    }
    novo.pai ← anterior;
    se (anterior.valor > novo.valor) {
        anterior.esquerdo ← novo;
    } senão {
        anterior.direito ← novo;
    }
}
```

Inserir: KA JA XO CA TE UX **PL** MN FI LI MA



```
inserirIterativamente(umValor):  
novo ← criar_noh(umValor);  
se (raiz = NULL) {  
    raiz ← novo;  
}  
senão {  
    atual ← raiz;  
    enquanto (atual ≠ NULL) {  
        anterior ← atual;  
        se (atual.valor > umValor) {  
            atual ← atual.esquerdo;  
        } senão {  
            atual ← atual.direito;  
        }  
    }  
    novo.pai ← anterior;  
    se (anterior.valor > novo.valor) {  
        anterior.esquerdo ← novo;  
    } senão {  
        anterior.direito ← novo;  
    }  
}
```


Inserir: KA JA XO CA TE UX **PL** MN FI LI MA

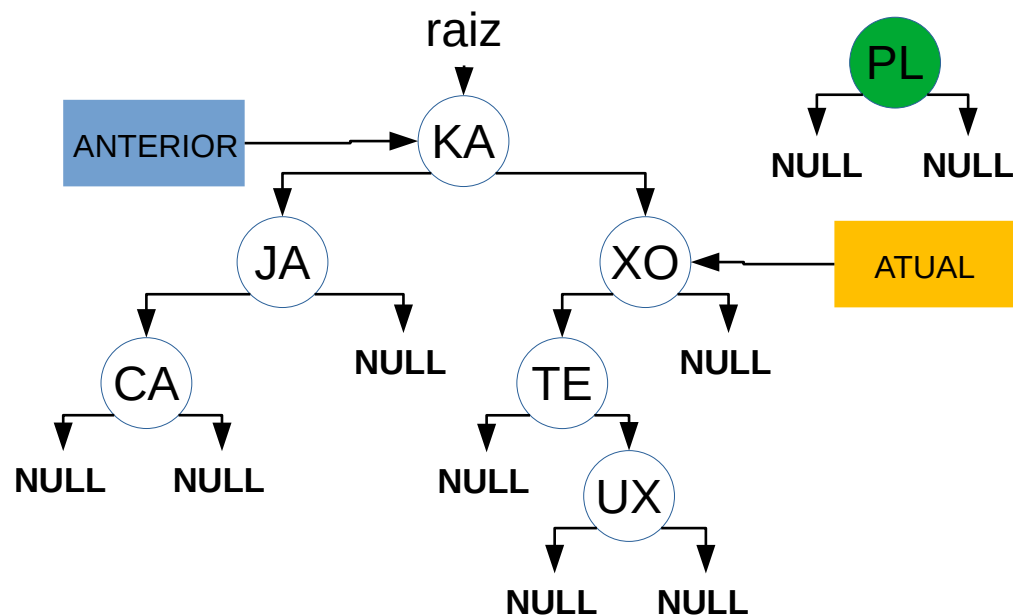


inserirIterativamente(umValor):

```

novo ← criar_noh(umValor);
se (raiz = NULL) {
    raiz ← novo;
}
senão {
    atual ← raiz;
    enquanto (atual ≠ NULL) {
        anterior ← atual;
        se (atual.valor > umValor) {
            atual ← atual.esquerdo;
        } senão {
            atual ← atual.direito;
        }
    }
    novo.pai ← anterior;
    se (anterior.valor > novo.valor) {
        anterior.esquerdo ← novo;
    } senão {
        anterior.direito ← novo;
    }
}
```

Inserir: KA JA XO CA TE UX **PL** MN FI LI MA

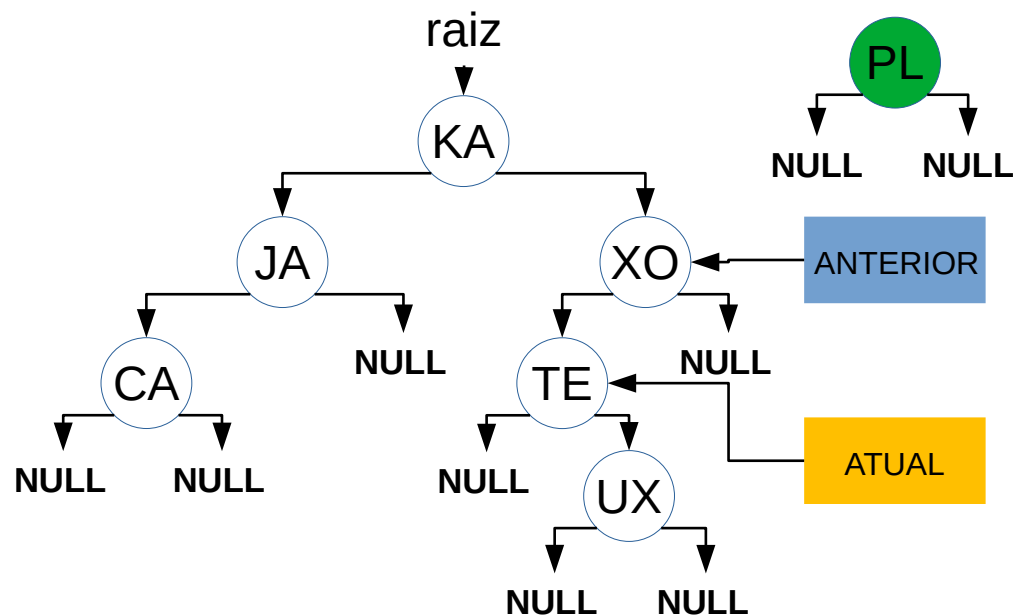


inserirIterativamente(umValor):

```

novo ← criar_noh(umValor);
se (raiz = NULL) {
    raiz ← novo;
}
senão {
    atual ← raiz;
    enquanto (atual ≠ NULL) {
        anterior ← atual;
        se (atual.valor > umValor) {
            atual ← atual.esquerdo;
        } senão {
            atual ← atual.direito;
        }
    }
    novo.pai ← anterior;
    se (anterior.valor > novo.valor) {
        anterior.esquerdo ← novo;
    } senão {
        anterior.direito ← novo;
    }
}
```

Inserir: KA JA XO CA TE UX **PL** MN FI LI MA

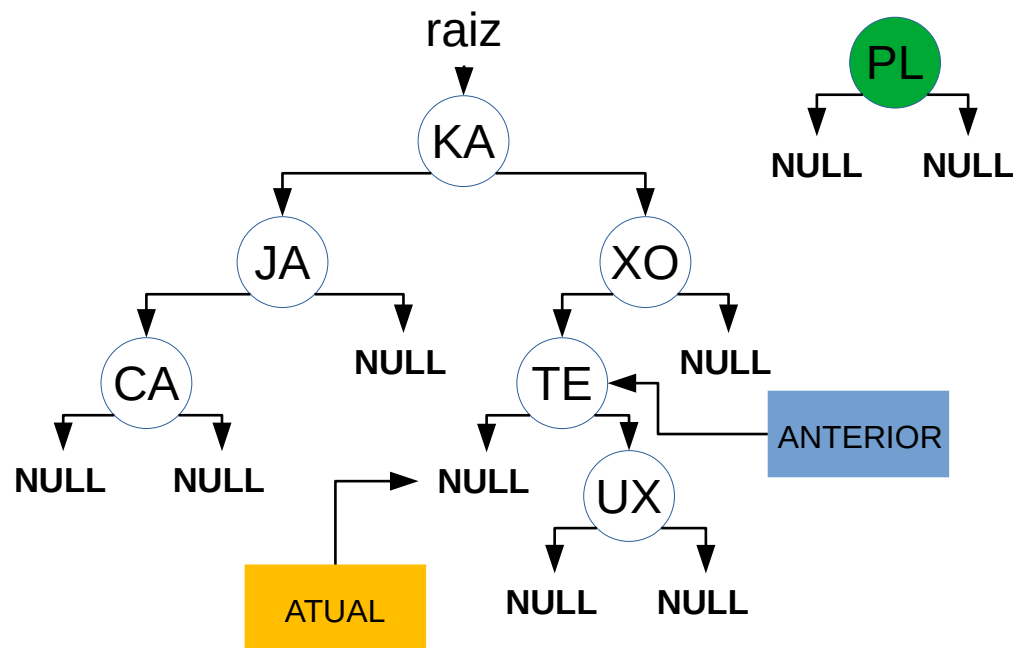


inserirIterativamente(umValor):

```

novo ← criar_noh(umValor);
se (raiz = NULL) {
    raiz ← novo;
}
senão {
    atual ← raiz;
    enquanto (atual ≠ NULL) {
        anterior ← atual;
        se (atual.valor > umValor) {
            atual ← atual.esquerdo;
        } senão {
            atual ← atual.direito;
        }
    }
    novo.pai ← anterior;
    se (anterior.valor > novo.valor) {
        anterior.esquerdo ← novo;
    } senão {
        anterior.direito ← novo;
    }
}
```

Inserir: KA JA XO CA TE UX **PL** MN FI LI MA

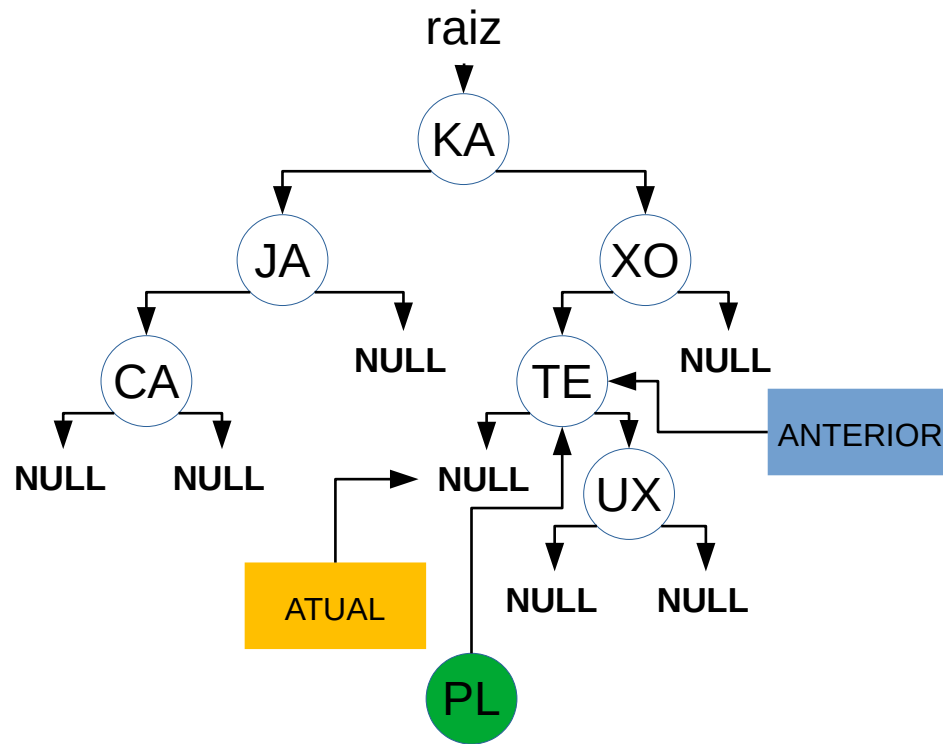


inserirIterativamente(umValor):

```

novo ← criar_noh(umValor);
se (raiz = NULL) {
    raiz ← novo;
}
senão {
    atual ← raiz;
    enquanto (atual ≠ NULL) {
        anterior ← atual;
        se (atual.valor > umValor) {
            atual ← atual.esquerdo;
        } senão {
            atual ← atual.direito;
        }
    }
    novo.pai ← anterior;
    se (anterior.valor > novo.valor) {
        anterior.esquerdo ← novo;
    } senão {
        anterior.direito ← novo;
    }
}
```

Inserir: KA JA XO CA TE UX **PL** MN FI LI MA

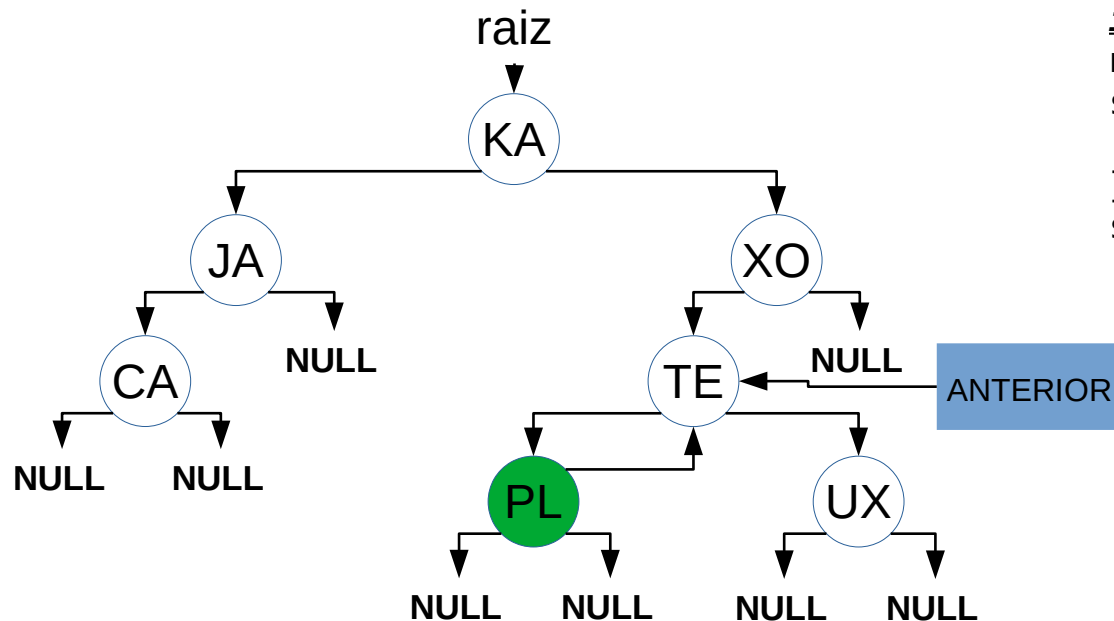


inserirIterativamente(umValor):

```

novo ← criar_noh(umValor);
se (raiz = NULL) {
    raiz ← novo;
}
senão {
    atual ← raiz;
    enquanto (atual ≠ NULL) {
        anterior ← atual;
        se (atual.valor > umValor) {
            atual ← atual.esquerdo;
        } senão {
            atual ← atual.direito;
        }
    }
    novo.pai ← anterior;
    se (anterior.valor > novo.valor) {
        anterior.esquerdo ← novo;
    } senão {
        anterior.direito ← novo;
    }
}
```

Inserir: KA JA XO CA TE UX **PL** MN FI LI MA

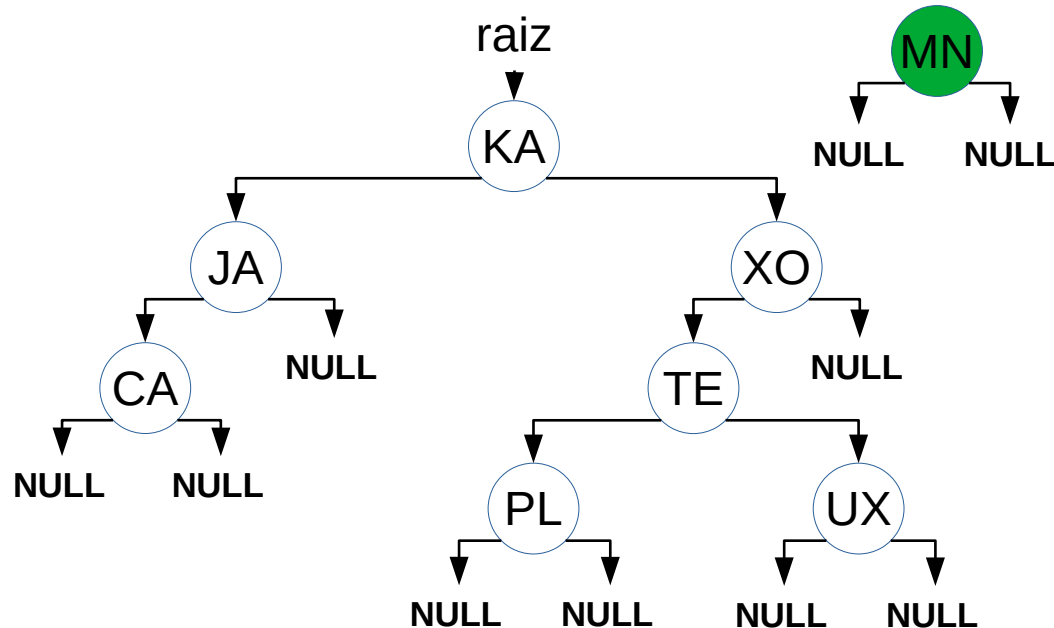


inserirIterativamente(umValor):

```

novo ← criar_noh(umValor);
se (raiz = NULL) {
    raiz ← novo;
}
senão {
    atual ← raiz;
    enquanto (atual ≠ NULL) {
        anterior ← atual;
        se (atual.valor > umValor) {
            atual ← atual.esquerdo;
        } senão {
            atual ← atual.direito;
        }
    }
    novo.pai ← anterior;
    se (anterior.valor > novo.valor) {
        anterior.esquerdo ← novo;
    } senão {
        anterior.direito ← novo;
    }
}
```

Inserir: KA JA XO CA TE UX PL MN FI LI MA

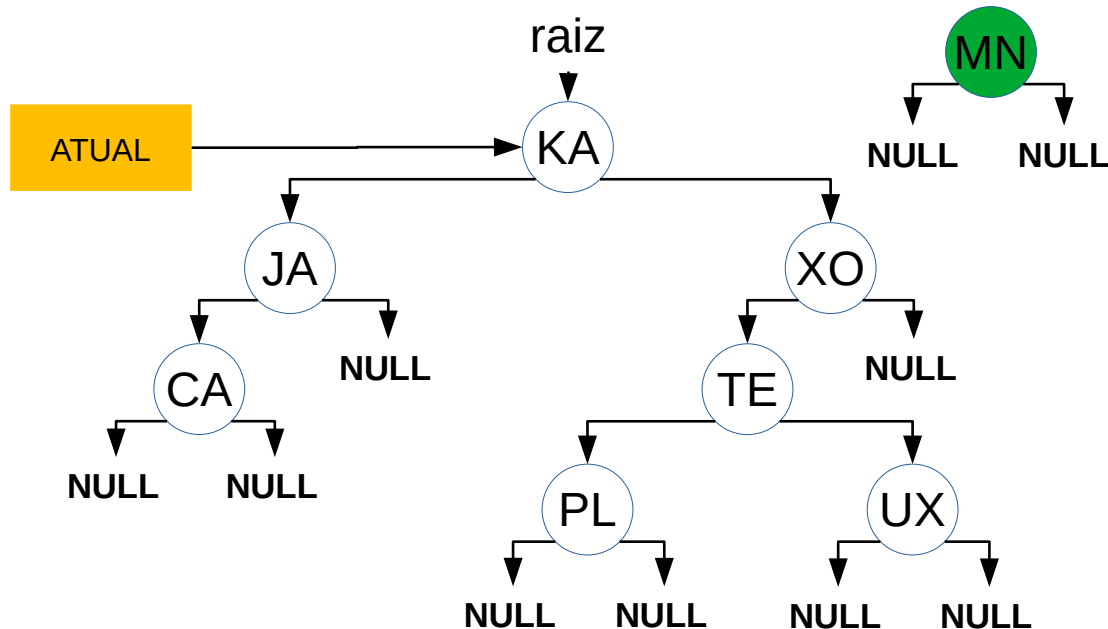


inserirIterativamente(umValor):

```

novo ← criar_noh(umValor);
se (raiz = NULL) {
    raiz ← novo;
}
senão {
    atual ← raiz;
    enquanto (atual ≠ NULL) {
        anterior ← atual;
        se (atual.valor > umValor) {
            atual ← atual.esquerdo;
        } senão {
            atual ← atual.direito;
        }
    }
    novo.pai ← anterior;
    se (anterior.valor > novo.valor) {
        anterior.esquerdo ← novo;
    } senão {
        anterior.direito ← novo;
    }
}
```

Inserir: KA JA XO CA TE UX PL MN FI LI MA

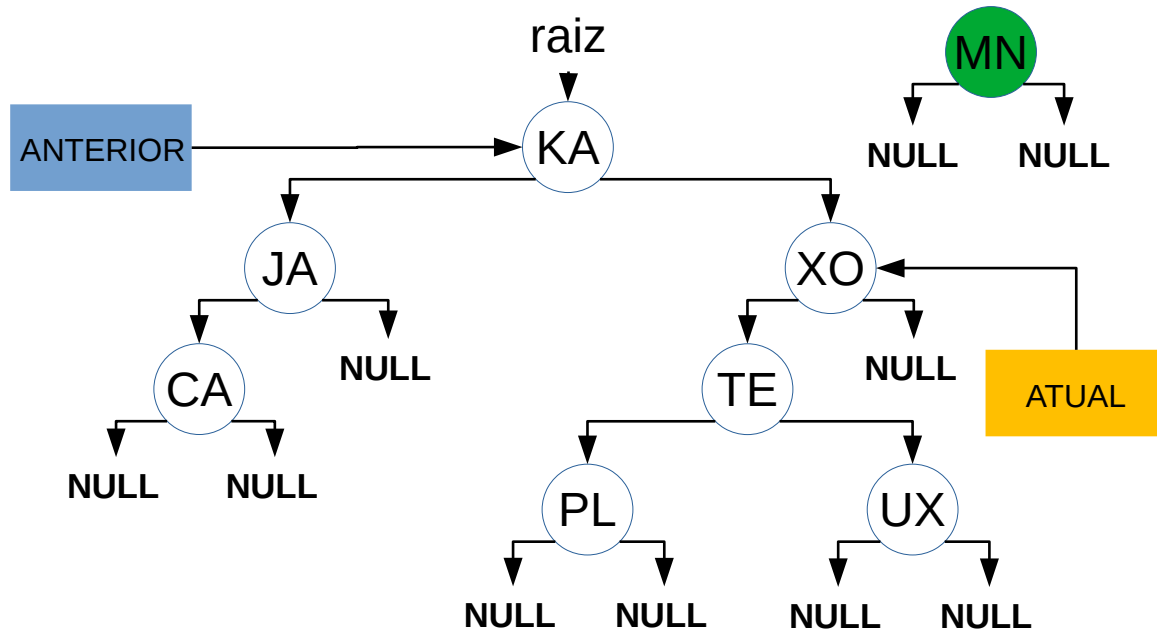


inserirIterativamente(umValor):

```

novo ← criar_noh(umValor);
se (raiz = NULL) {
    raiz ← novo;
}
senão {
    atual ← raiz;
    enquanto (atual ≠ NULL) {
        anterior ← atual;
        se (atual.valor > umValor) {
            atual ← atual.esquerdo;
        } senão {
            atual ← atual.direito;
        }
    }
    novo.pai ← anterior;
    se (anterior.valor > novo.valor) {
        anterior.esquerdo ← novo;
    } senão {
        anterior.direito ← novo;
    }
}
```


Inserir: KA JA XO CA TE UX PL MN FI LI MA

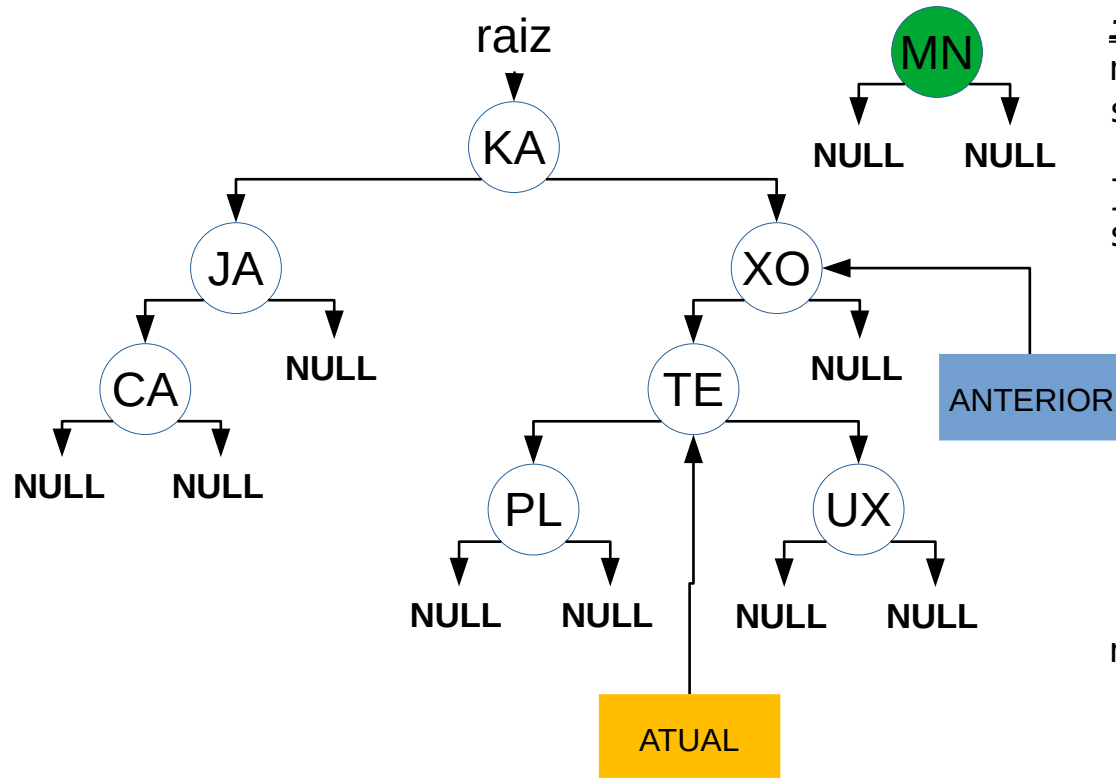


inserirIterativamente(umValor):

```

novo ← criar_noh(umValor);
se (raiz = NULL) {
    raiz ← novo;
}
senão {
    atual ← raiz;
    enquanto (atual ≠ NULL) {
        anterior ← atual;
        se (atual.valor > umValor) {
            atual ← atual.esquerdo;
        } senão {
            atual ← atual.direito;
        }
    }
    novo.pai ← anterior;
    se (anterior.valor > novo.valor) {
        anterior.esquerdo ← novo;
    } senão {
        anterior.direito ← novo;
    }
}
```

Inserir: KA JA XO CA TE UX PL MN FI LI MA

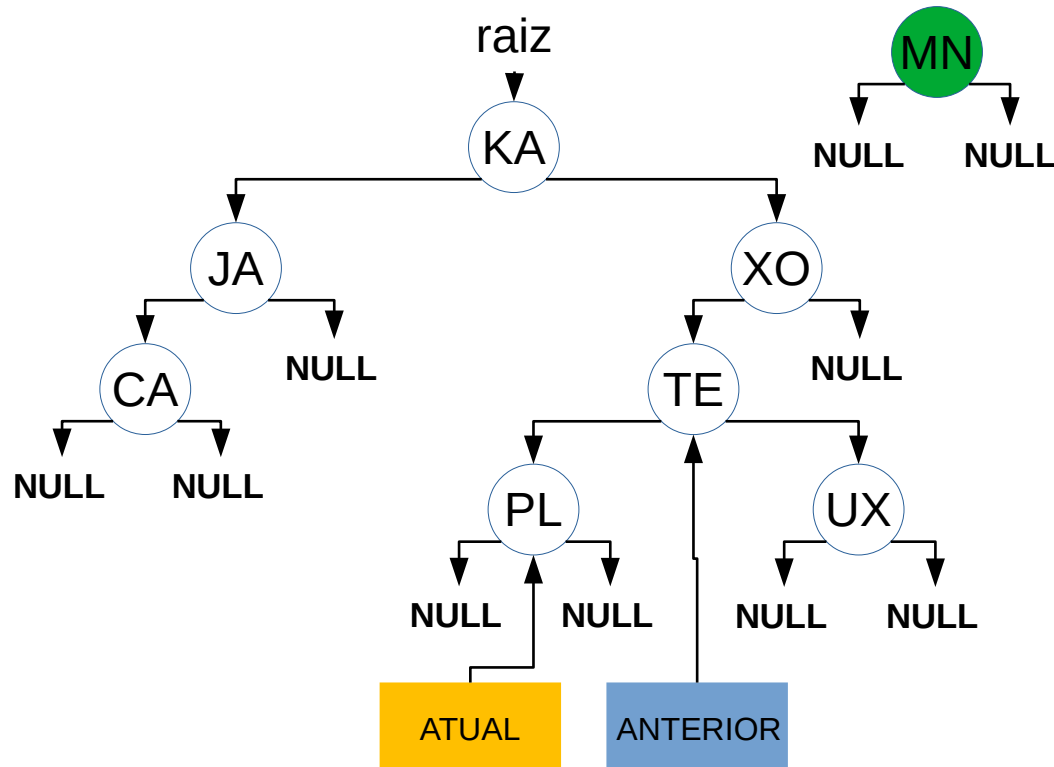


inserirIterativamente(umValor):

```

novo ← criar_noh(umValor);
se (raiz = NULL) {
    raiz ← novo;
}
senão {
    atual ← raiz;
    enquanto (atual ≠ NULL) {
        anterior ← atual;
        se (atual.valor > umValor) {
            atual ← atual.esquerdo;
        } senão {
            atual ← atual.direito;
        }
    }
    novo.pai ← anterior;
    se (anterior.valor > novo.valor) {
        anterior.esquerdo ← novo;
    } senão {
        anterior.direito ← novo;
    }
}
```

Inserir: KA JA XO CA TE UX PL MN FI LI MA

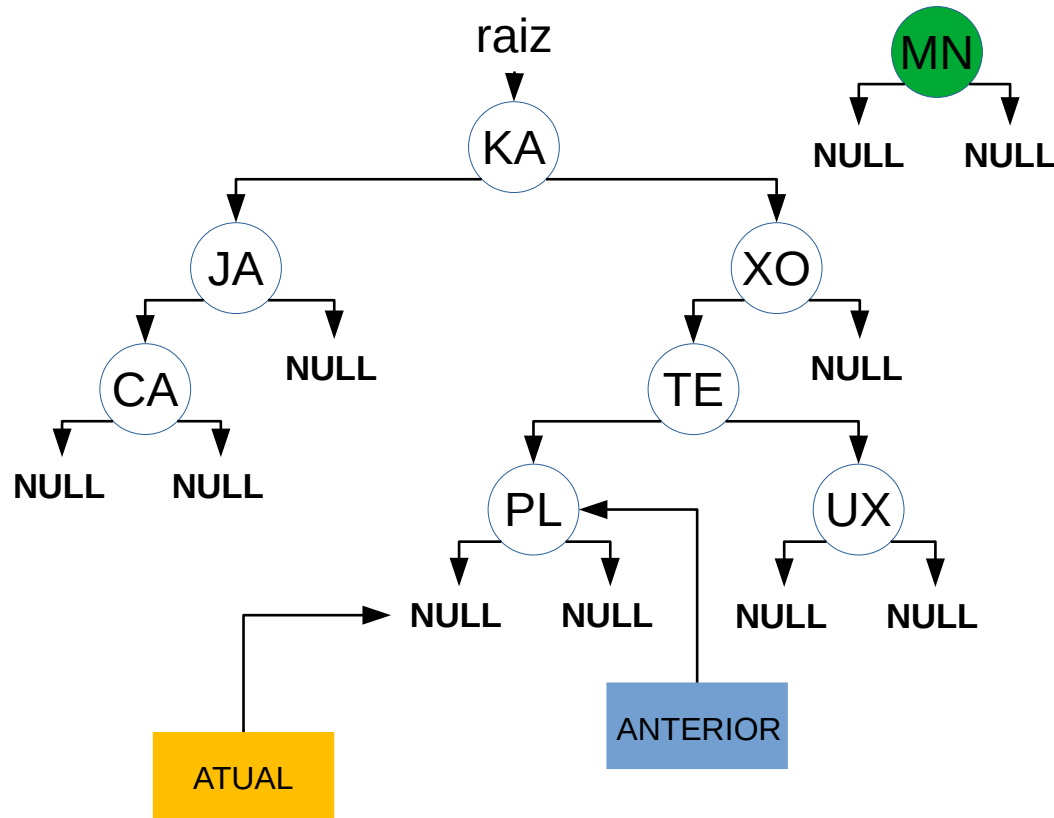


inserirIterativamente(umValor):

```

novo ← criar_noh(umValor);
se (raiz = NULL) {
    raiz ← novo;
}
senão {
    atual ← raiz;
    enquanto (atual ≠ NULL) {
        anterior ← atual;
        se (atual.valor > umValor) {
            atual ← atual.esquerdo;
        } senão {
            atual ← atual.direito;
        }
    }
    novo.pai ← anterior;
    se (anterior.valor > novo.valor) {
        anterior.esquerdo ← novo;
    } senão {
        anterior.direito ← novo;
    }
}
```

Inserir: KA JA XO CA TE UX PL MN FI LI MA

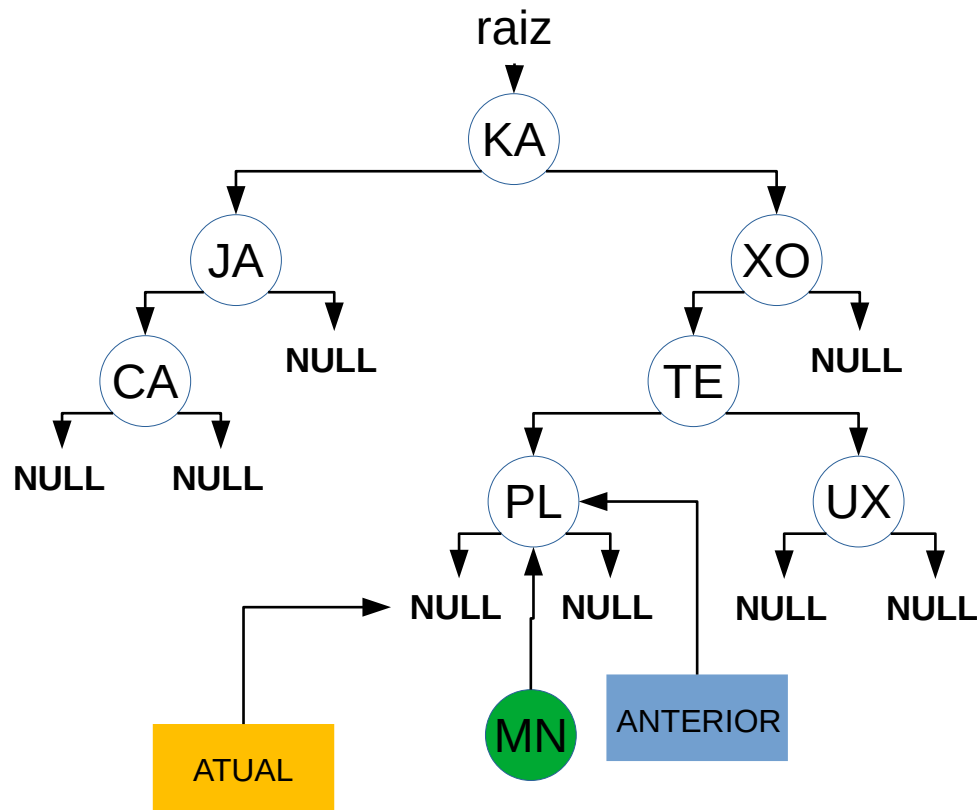


inserirIterativamente(umValor):

```

novo ← criar_noh(umValor);
se (raiz = NULL) {
    raiz ← novo;
}
senão {
    atual ← raiz;
    enquanto (atual ≠ NULL) {
        anterior ← atual;
        se (atual.valor > umValor) {
            atual ← atual.esquerdo;
        } senão {
            atual ← atual.direito;
        }
    }
    novo.pai ← anterior;
    se (anterior.valor > novo.valor) {
        anterior.esquerdo ← novo;
    } senão {
        anterior.direito ← novo;
    }
}
```

Inserir: KA JA XO CA TE UX PL MN FI LI MA

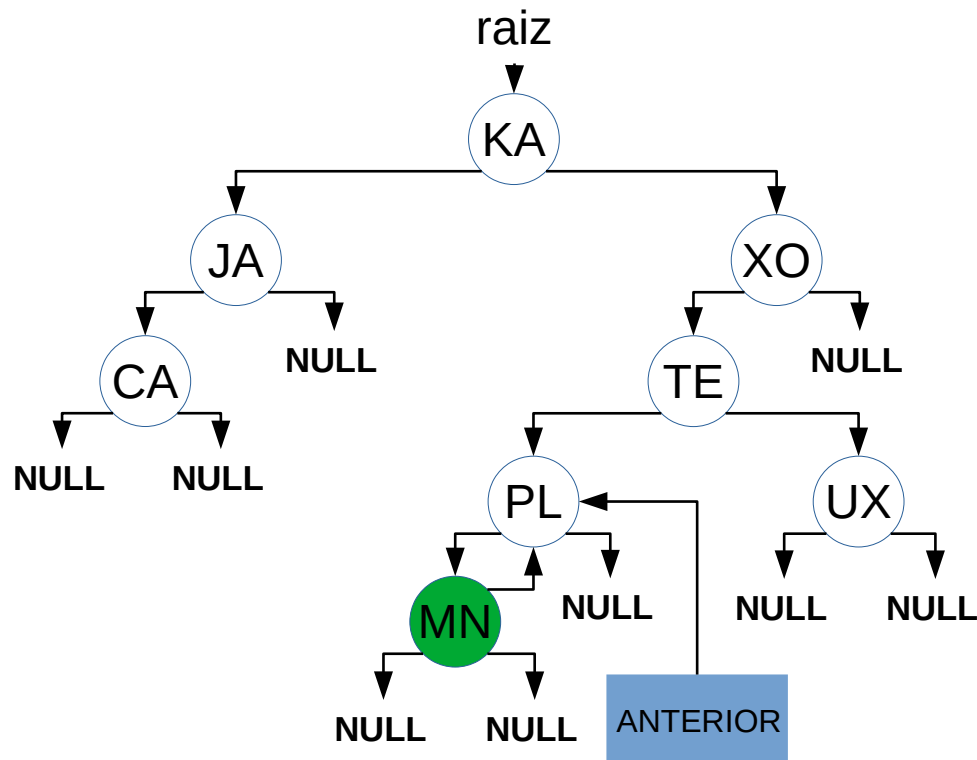


inserirIterativamente(umValor):

```

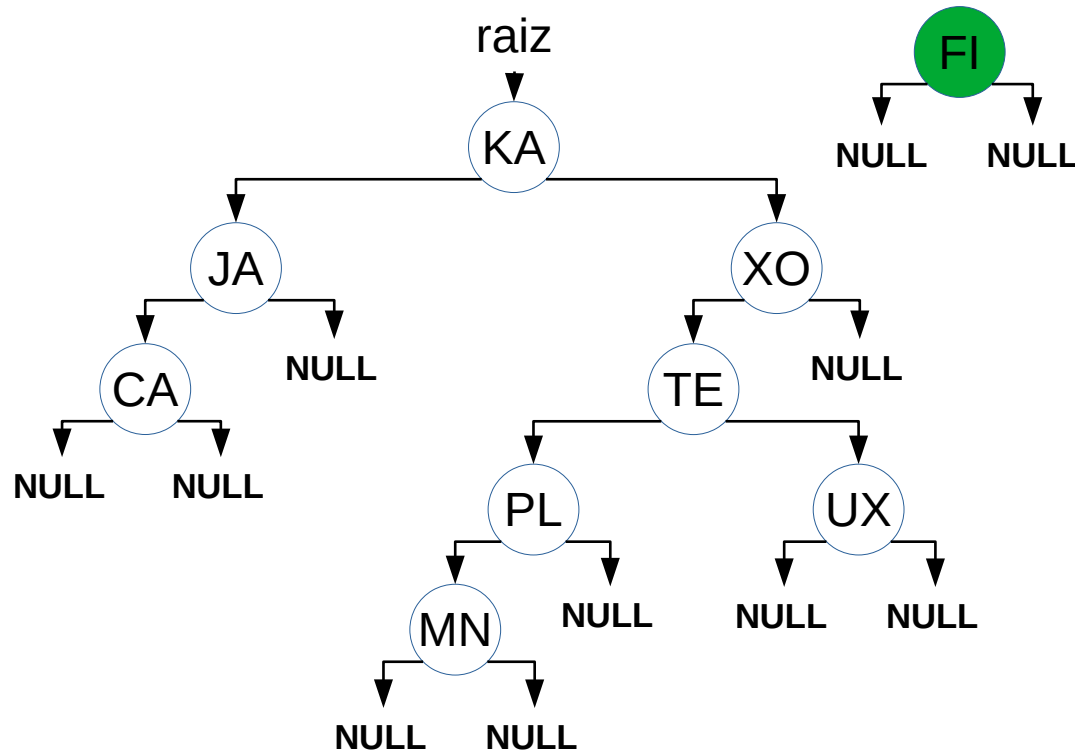
novo ← criar_noh(umValor);
se (raiz = NULL) {
    raiz ← novo;
}
senão {
    atual ← raiz;
    enquanto (atual ≠ NULL) {
        anterior ← atual;
        se (atual.valor > umValor) {
            atual ← atual.esquerdo;
        } senão {
            atual ← atual.direito;
        }
    }
    novo.pai ← anterior;
    se (anterior.valor > novo.valor) {
        anterior.esquerdo ← novo;
    } senão {
        anterior.direito ← novo;
    }
}
```

Inserir: KA JA XO CA TE UX PL MN FI LI MA



```
inserirIterativamente(umValor):
novo ← criar_noh(umValor);
se (raiz = NULL) {
    raiz ← novo;
}
senão {
    atual ← raiz;
    enquanto (atual ≠ NULL) {
        anterior ← atual;
        se (atual.valor > umValor) {
            atual ← atual.esquerdo;
        } senão {
            atual ← atual.direito;
        }
    }
    novo.pai ← anterior;
    se (anterior.valor > novo.valor) {
        anterior.esquerdo ← novo;
    } senão {
        anterior.direito ← novo;
    }
}
```

Inserir: KA JA XO CA TE UX PL MN **FI** LI MA

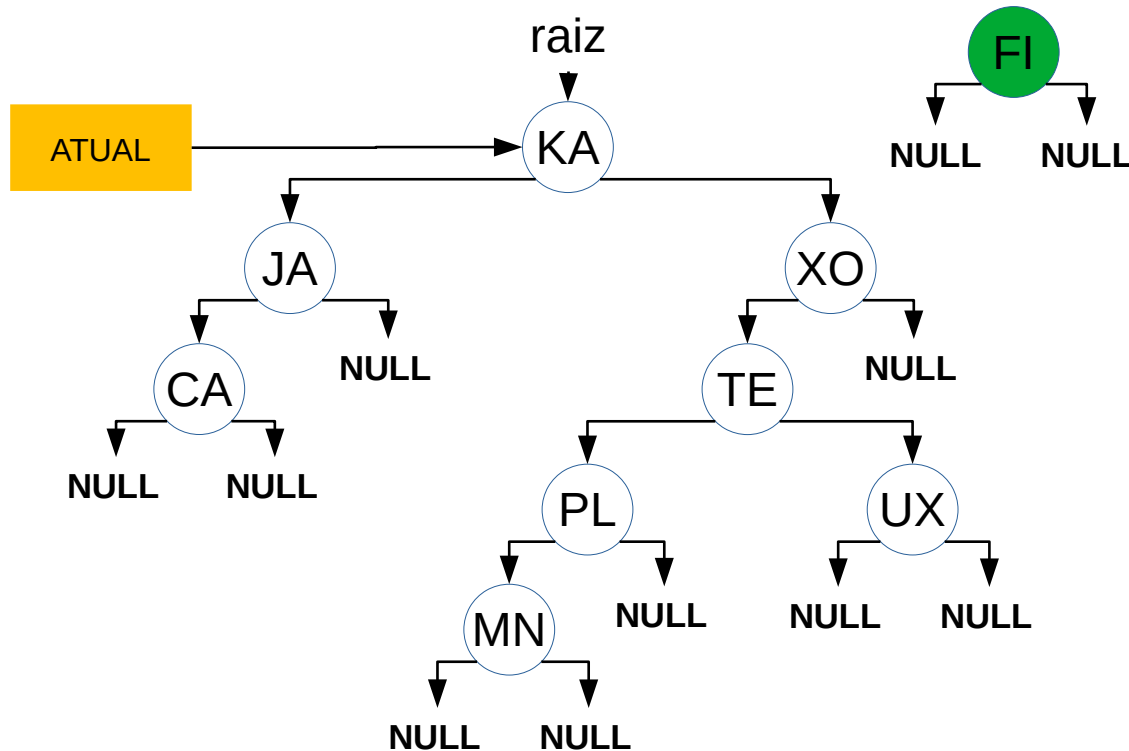


inserirIterativamente(umValor):

```

novo ← criar_noh(umValor);
se (raiz = NULL) {
    raiz ← novo;
}
senão {
    atual ← raiz;
    enquanto (atual ≠ NULL) {
        anterior ← atual;
        se (atual.valor > umValor) {
            atual ← atual.esquerdo;
        } senão {
            atual ← atual.direito;
        }
    }
    novo.pai ← anterior;
    se (anterior.valor > novo.valor) {
        anterior.esquerdo ← novo;
    } senão {
        anterior.direito ← novo;
    }
}
```

Inserir: KA JA XO CA TE UX PL MN **FI** LI MA

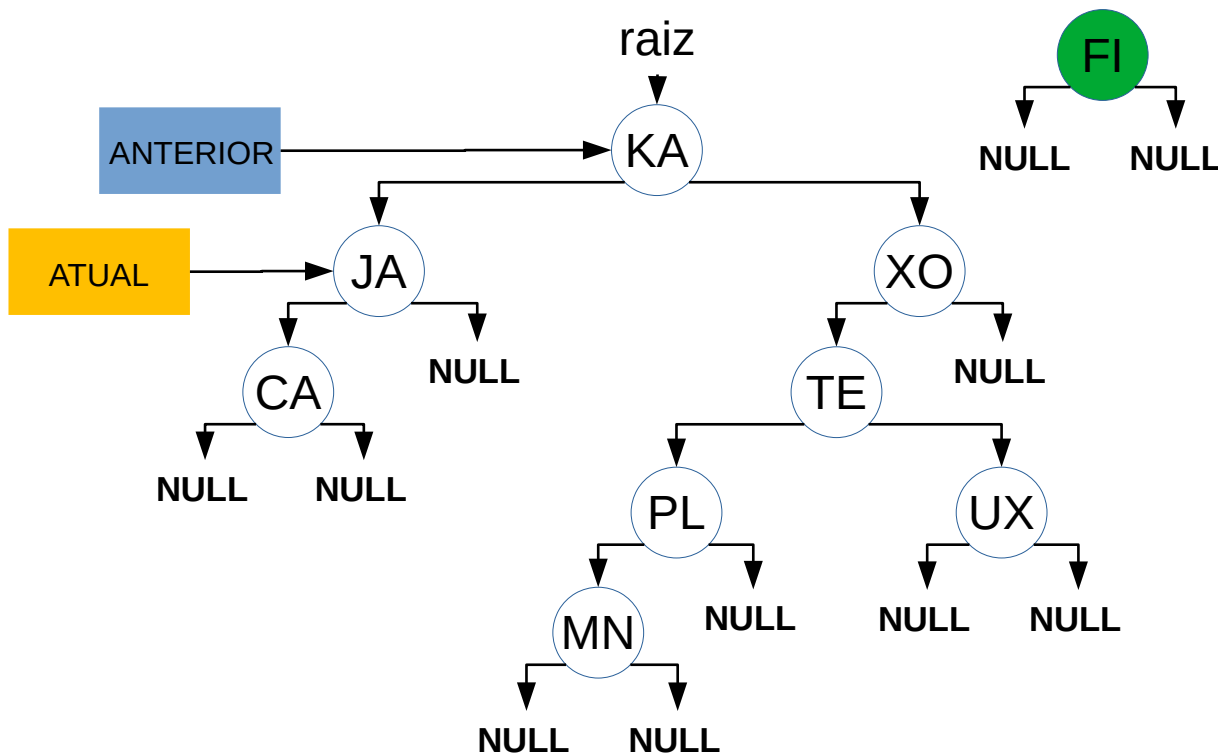


inserirIterativamente(umValor):

```

novo ← criar_noh(umValor);
se (raiz = NULL) {
    raiz ← novo;
}
senão {
    atual ← raiz;
    enquanto (atual ≠ NULL) {
        anterior ← atual;
        se (atual.valor > umValor) {
            atual ← atual.esquerdo;
        } senão {
            atual ← atual.direito;
        }
    }
    novo.pai ← anterior;
    se (anterior.valor > novo.valor) {
        anterior.esquerdo ← novo;
    } senão {
        anterior.direito ← novo;
    }
}
```


Inserir: KA JA XO CA TE UX PL MN **FI** LI MA

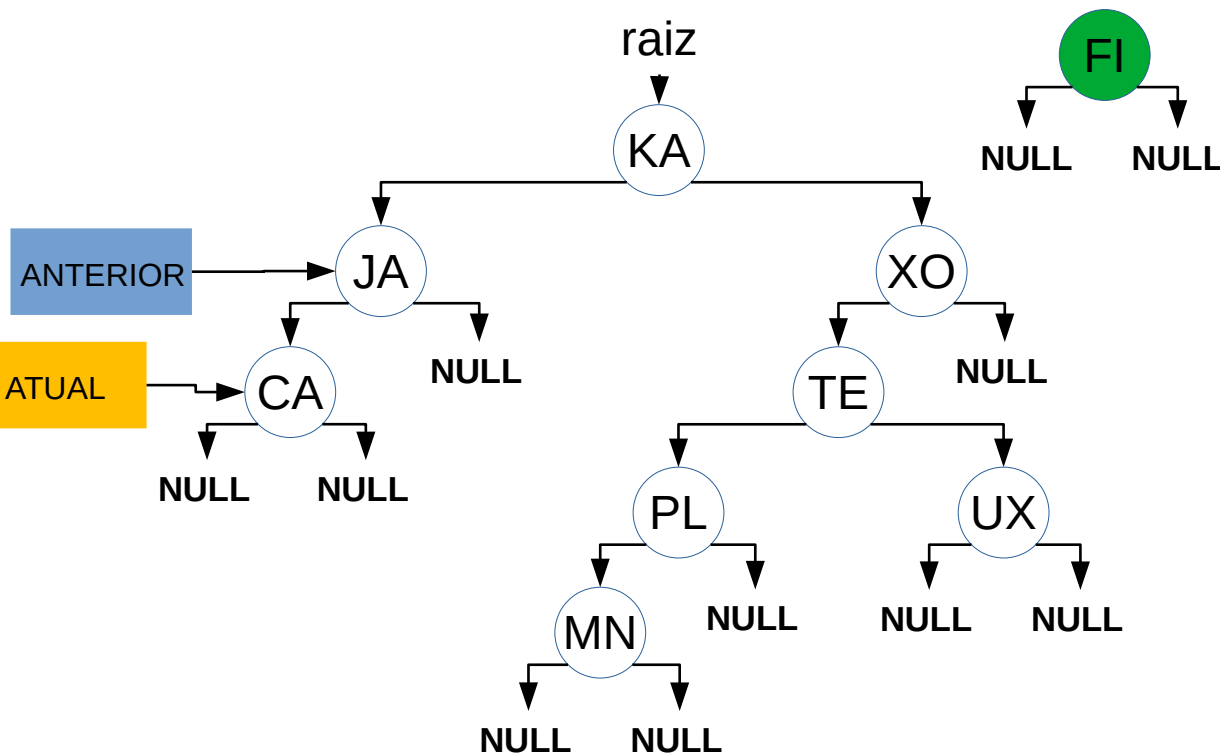


inserirIterativamente(umValor):

```

novo ← criar_noh(umValor);
se (raiz = NULL) {
    raiz ← novo;
}
senão {
    atual ← raiz;
    enquanto (atual ≠ NULL) {
        anterior ← atual;
        se (atual.valor > umValor) {
            atual ← atual.esquerdo;
        } senão {
            atual ← atual.direito;
        }
    }
    novo.pai ← anterior;
    se (anterior.valor > novo.valor) {
        anterior.esquerdo ← novo;
    } senão {
        anterior.direito ← novo;
    }
}
```

Inserir: KA JA XO CA TE UX PL MN **FI** LI MA

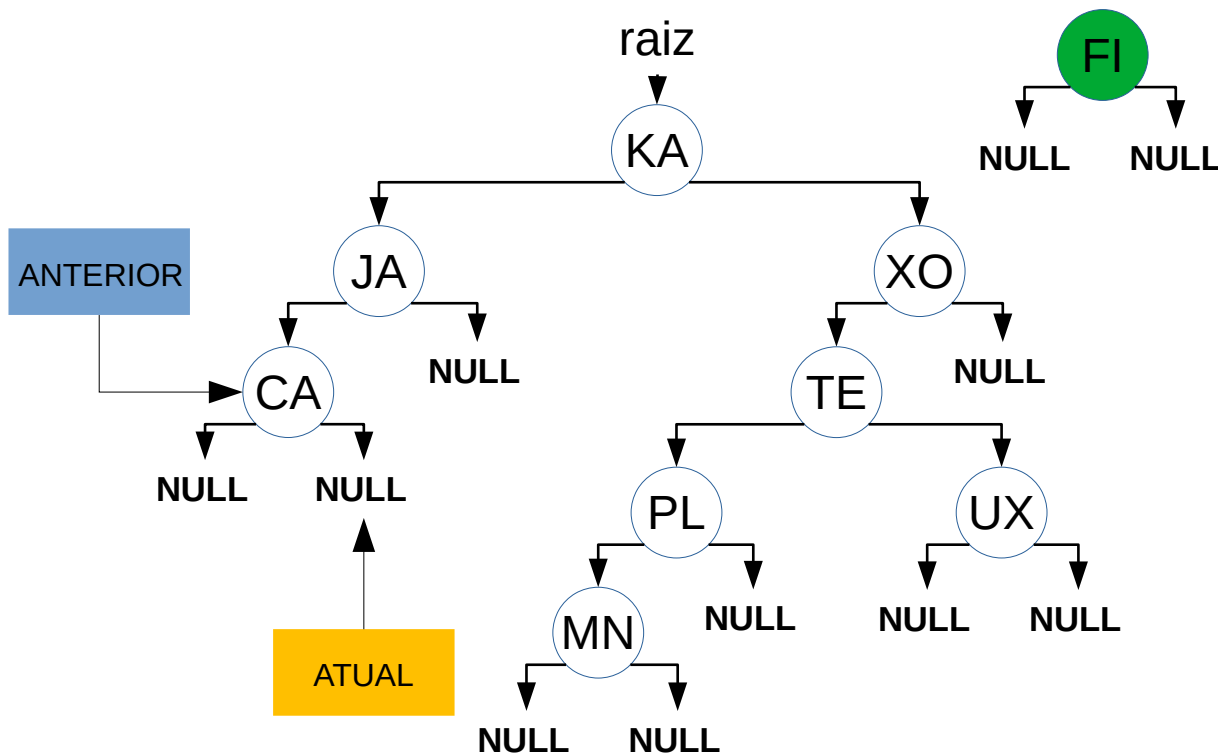


inserirIterativamente(umValor):

```

novo ← criar_noh(umValor);
se (raiz = NULL) {
    raiz ← novo;
}
senão {
    atual ← raiz;
    enquanto (atual ≠ NULL) {
        anterior ← atual;
        se (atual.valor > umValor) {
            atual ← atual.esquerdo;
        } senão {
            atual ← atual.direito;
        }
    }
    novo.pai ← anterior;
    se (anterior.valor > novo.valor) {
        anterior.esquerdo ← novo;
    } senão {
        anterior.direito ← novo;
    }
}
```

Inserir: KA JA XO CA TE UX PL MN **FI** LI MA

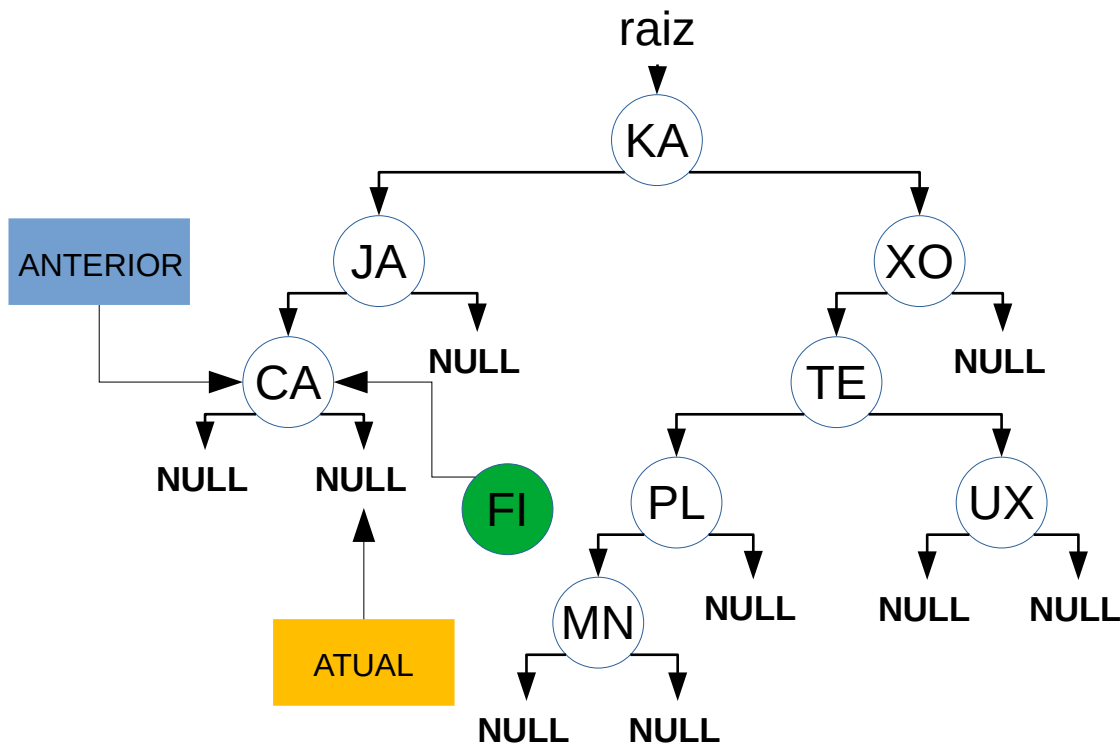


inserirIterativamente(umValor):

```

novo ← criar_noh(umValor);
se (raiz = NULL) {
    raiz ← novo;
}
senão {
    atual ← raiz;
    enquanto (atual ≠ NULL) {
        anterior ← atual;
        se (atual.valor > umValor) {
            atual ← atual.esquerdo;
        } senão {
            atual ← atual.direito;
        }
    }
    novo.pai ← anterior;
    se (anterior.valor > novo.valor) {
        anterior.esquerdo ← novo;
    } senão {
        anterior.direito ← novo;
    }
}
```

Inserir: KA JA XO CA TE UX PL MN **FI** LI MA

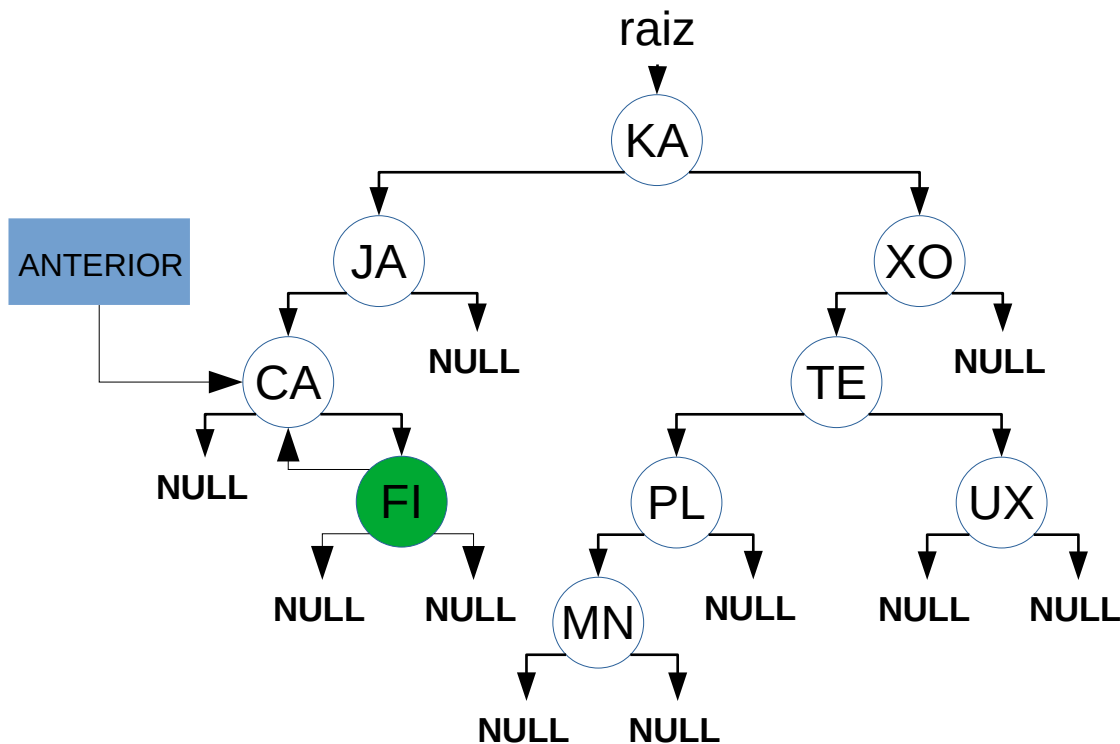


inserirIterativamente(umValor):

```

novo ← criar_noh(umValor);
se (raiz = NULL) {
    raiz ← novo;
}
senão {
    atual ← raiz;
    enquanto (atual ≠ NULL) {
        anterior ← atual;
        se (atual.valor > umValor) {
            atual ← atual.esquerdo;
        } senão {
            atual ← atual.direito;
        }
    }
    novo.pai ← anterior;
    se (anterior.valor > novo.valor) {
        anterior.esquerdo ← novo;
    } senão {
        anterior.direito ← novo;
    }
}
```

Inserir: KA JA XO CA TE UX PL MN **FI** LI MA

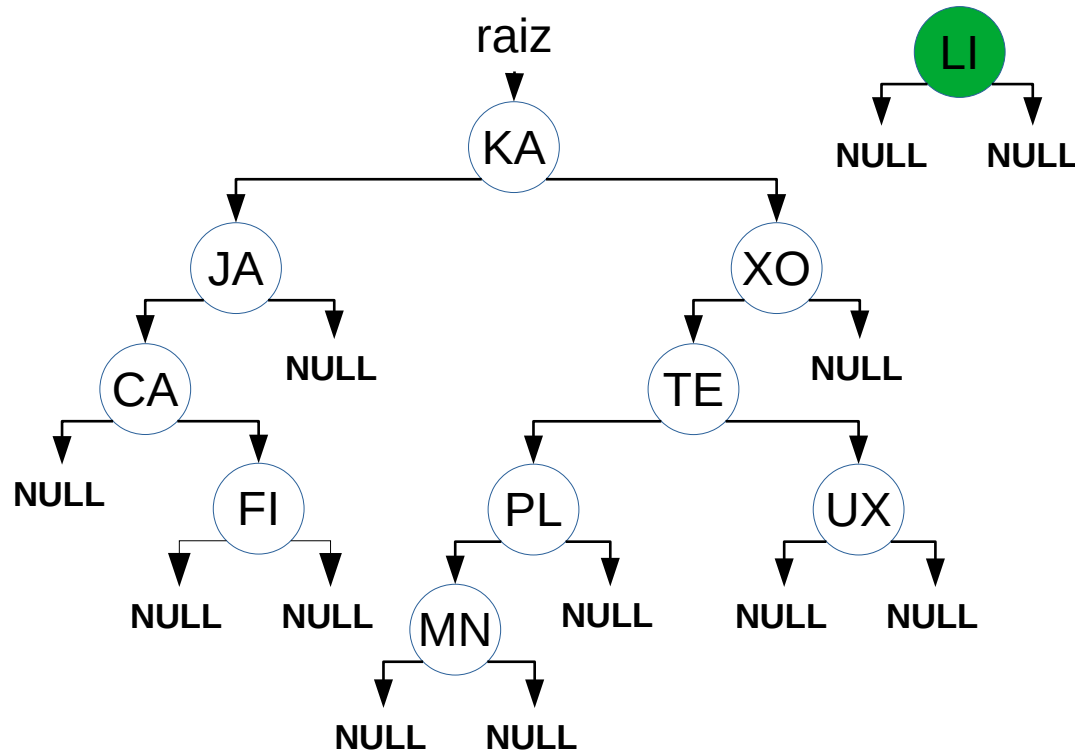


inserirIterativamente(umValor):

```

novo ← criar_noh(umValor);
se (raiz = NULL) {
    raiz ← novo;
}
senão {
    atual ← raiz;
    enquanto (atual ≠ NULL) {
        anterior ← atual;
        se (atual.valor > umValor) {
            atual ← atual.esquerdo;
        } senão {
            atual ← atual.direito;
        }
    }
    novo.pai ← anterior;
    se (anterior.valor > novo.valor) {
        anterior.esquerdo ← novo;
    } senão {
        anterior.direito ← novo;
    }
}
```

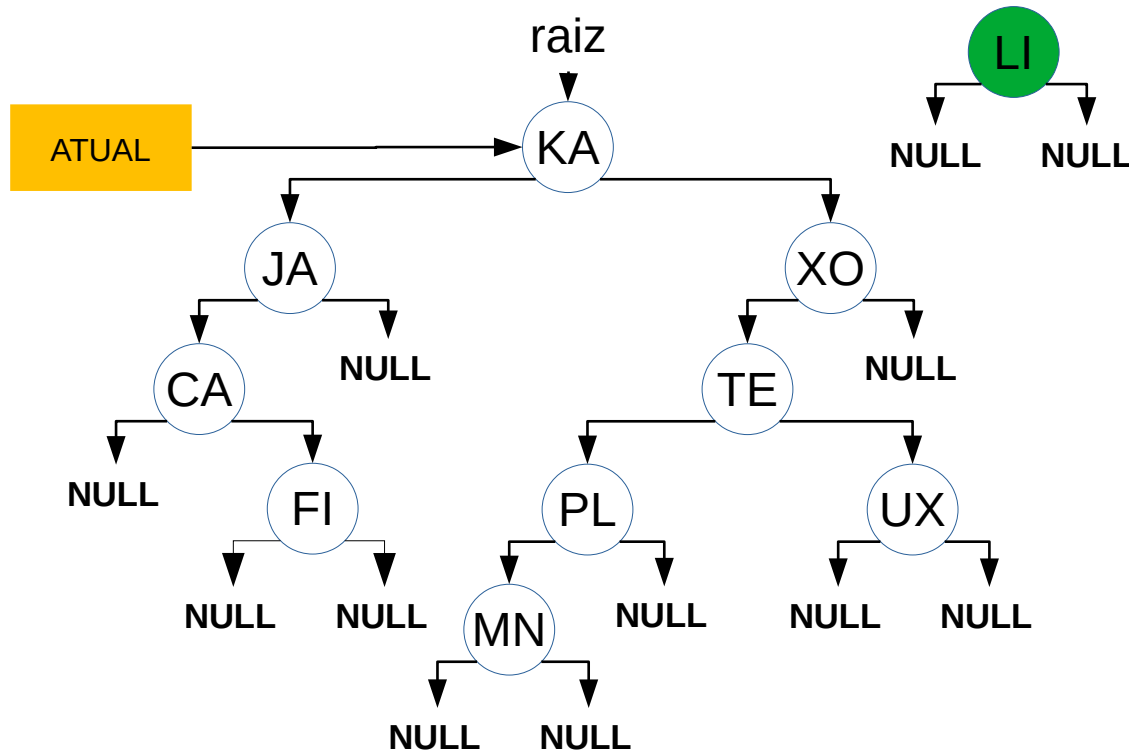
Inserir: KA JA XO CA TE UX PL MN FI **LI** MA



inserirIterativamente(umValor):

```
    novo ← criar_noh(umValor);  
    se (raiz = NULL) {  
        raiz ← novo;  
    }  
    senão {  
        atual ← raiz;  
        enquanto (atual ≠ NULL) {  
            anterior ← atual;  
            se (atual.valor > umValor) {  
                atual ← atual.esquerdo;  
            } senão {  
                atual ← atual.direito;  
            }  
        }  
        novo.pai ← anterior;  
        se (anterior.valor > novo.valor) {  
            anterior.esquerdo ← novo;  
        } senão {  
            anterior.direito ← novo;  
        }  
    }
```

Inserir: KA JA XO CA TE UX PL MN FI **LI** MA

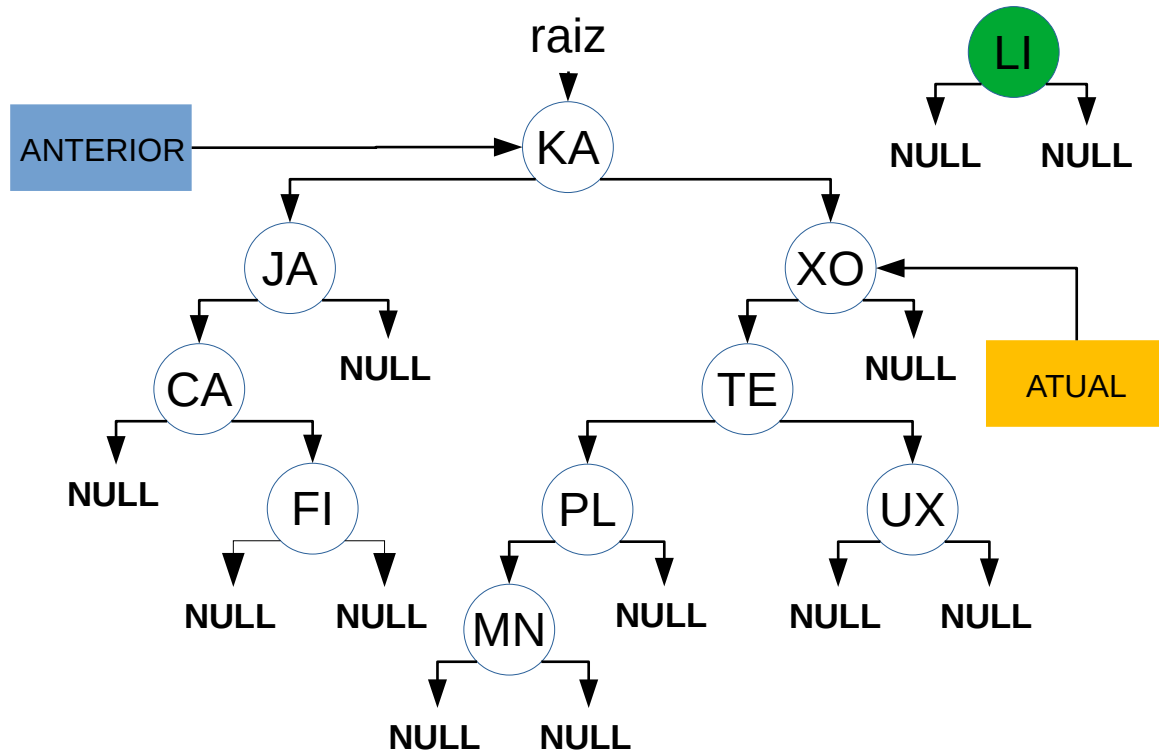


inserirIterativamente(umValor):

```

novo ← criar_noh(umValor);
se (raiz = NULL) {
    raiz ← novo;
}
senão {
    atual ← raiz;
    enquanto (atual ≠ NULL) {
        anterior ← atual;
        se (atual.valor > umValor) {
            atual ← atual.esquerdo;
        } senão {
            atual ← atual.direito;
        }
    }
    novo.pai ← anterior;
    se (anterior.valor > novo.valor) {
        anterior.esquerdo ← novo;
    } senão {
        anterior.direito ← novo;
    }
}
```

Inserir: KA JA XO CA TE UX PL MN FI **LI** MA

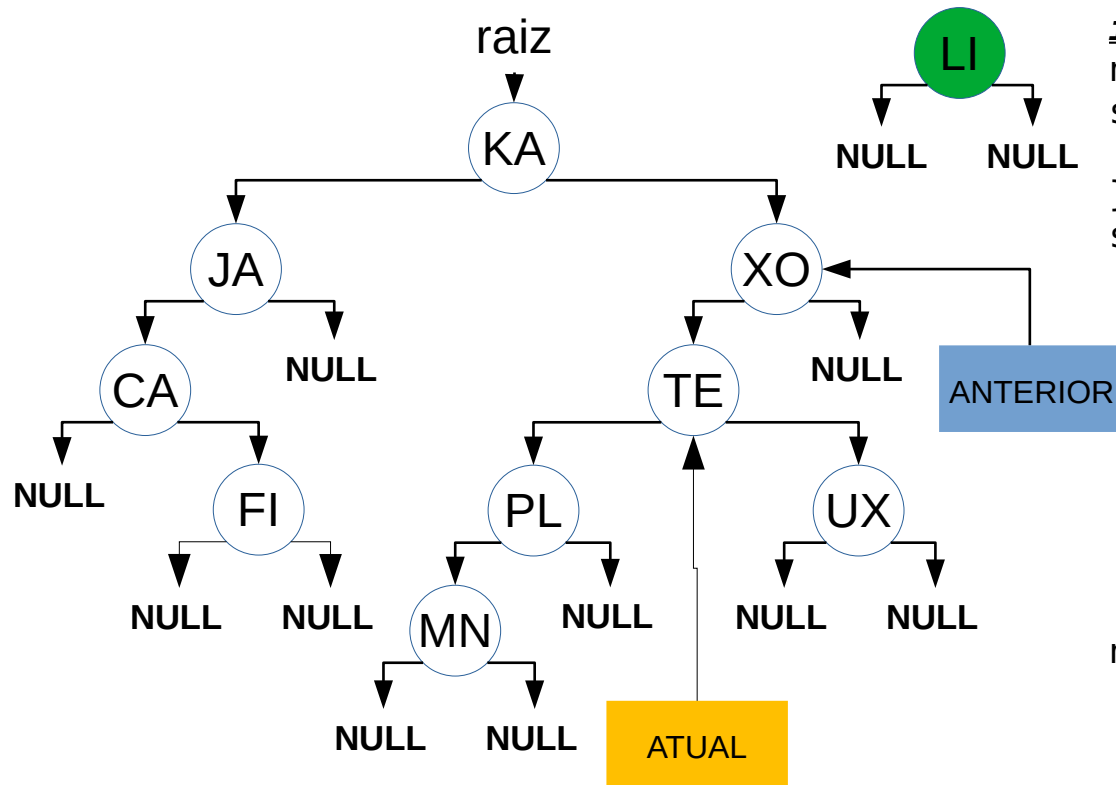


inserirIterativamente(umValor):

```

novo ← criar_noh(umValor);
se (raiz = NULL) {
    raiz ← novo;
}
senão {
    atual ← raiz;
    enquanto (atual ≠ NULL) {
        anterior ← atual;
        se (atual.valor > umValor) {
            atual ← atual.esquerdo;
        } senão {
            atual ← atual.direito;
        }
    }
    novo.pai ← anterior;
    se (anterior.valor > novo.valor) {
        anterior.esquerdo ← novo;
    } senão {
        anterior.direito ← novo;
    }
}
```


Inserir: KA JA XO CA TE UX PL MN FI **LI** MA

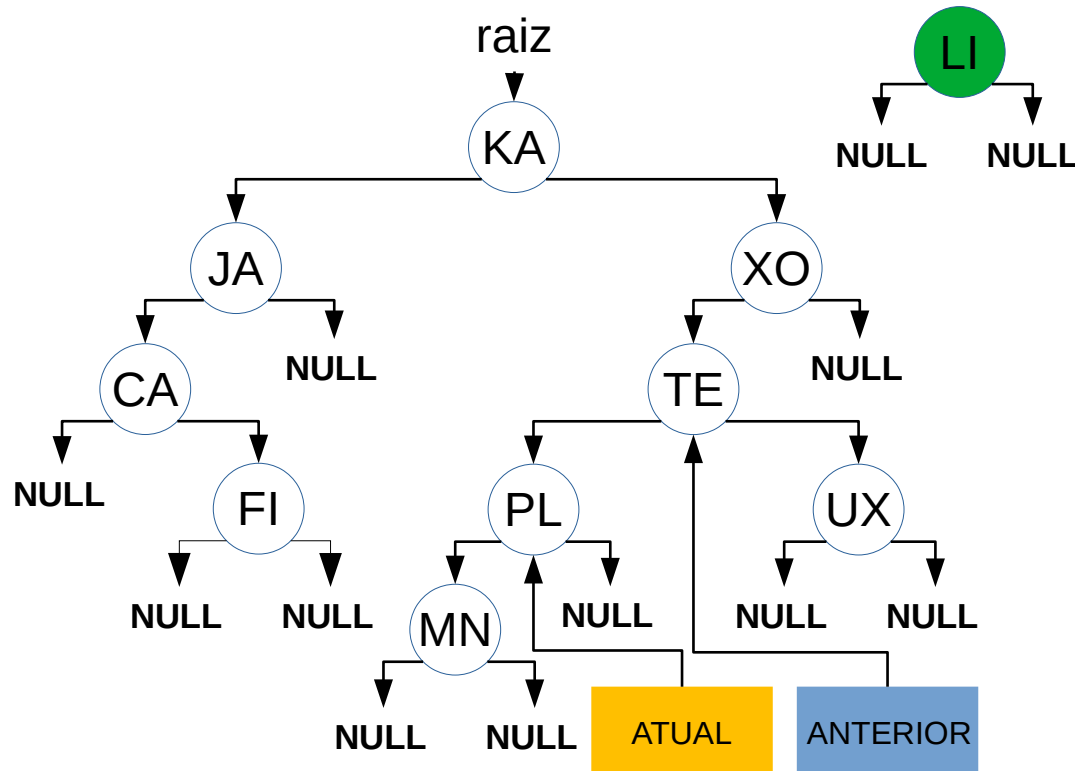


inserirIterativamente(umValor):

```

novo ← criar_noh(umValor);
se (raiz = NULL) {
    raiz ← novo;
}
senão {
    atual ← raiz;
    enquanto (atual ≠ NULL) {
        anterior ← atual;
        se (atual.valor > umValor) {
            atual ← atual.esquerdo;
        } senão {
            atual ← atual.direito;
        }
    }
    novo.pai ← anterior;
    se (anterior.valor > novo.valor) {
        anterior.esquerdo ← novo;
    } senão {
        anterior.direito ← novo;
    }
}
```

Inserir: KA JA XO CA TE UX PL MN FI **LI** MA

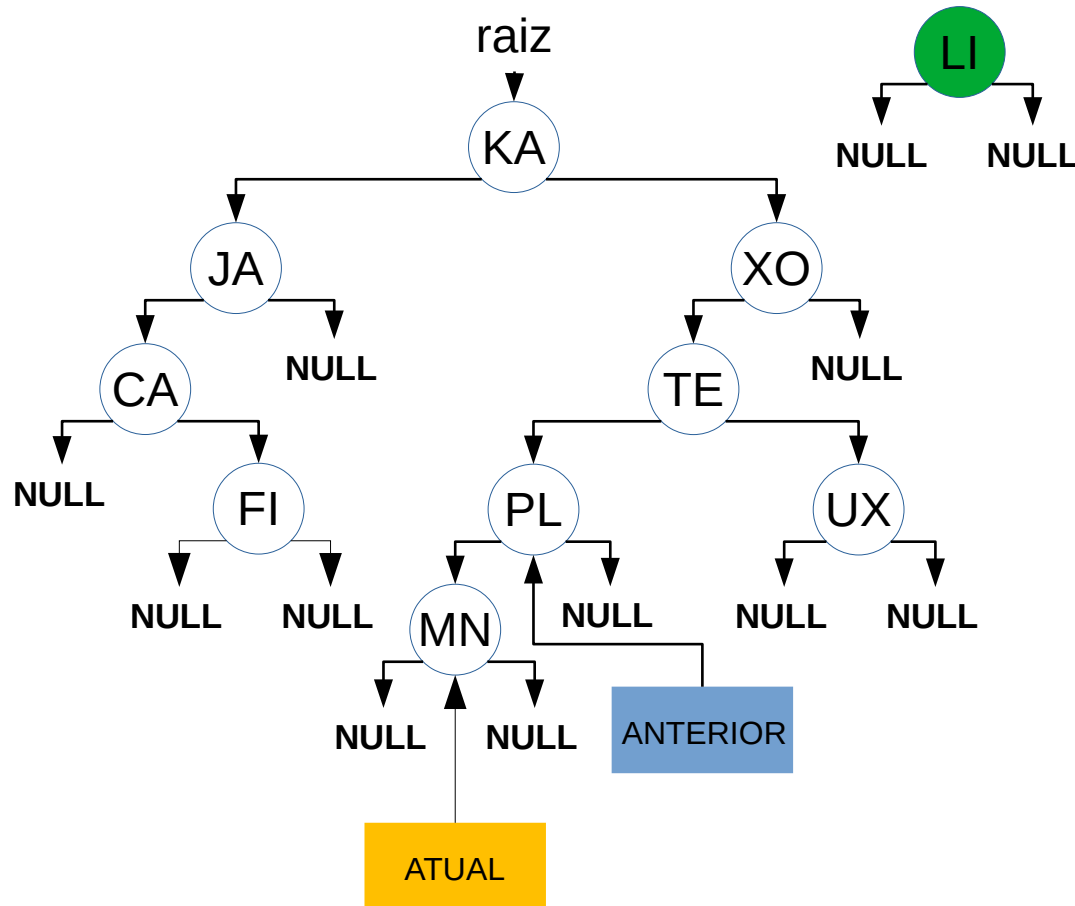


inserirIterativamente(umValor):

```

novo ← criar_noh(umValor);
se (raiz = NULL) {
    raiz ← novo;
}
senão {
    atual ← raiz;
    enquanto (atual ≠ NULL) {
        anterior ← atual;
        se (atual.valor > umValor) {
            atual ← atual.esquerdo;
        } senão {
            atual ← atual.direito;
        }
    }
    novo.pai ← anterior;
    se (anterior.valor > novo.valor) {
        anterior.esquerdo ← novo;
    } senão {
        anterior.direito ← novo;
    }
}
```

Inserir: KA JA XO CA TE UX PL MN FI **LI** MA

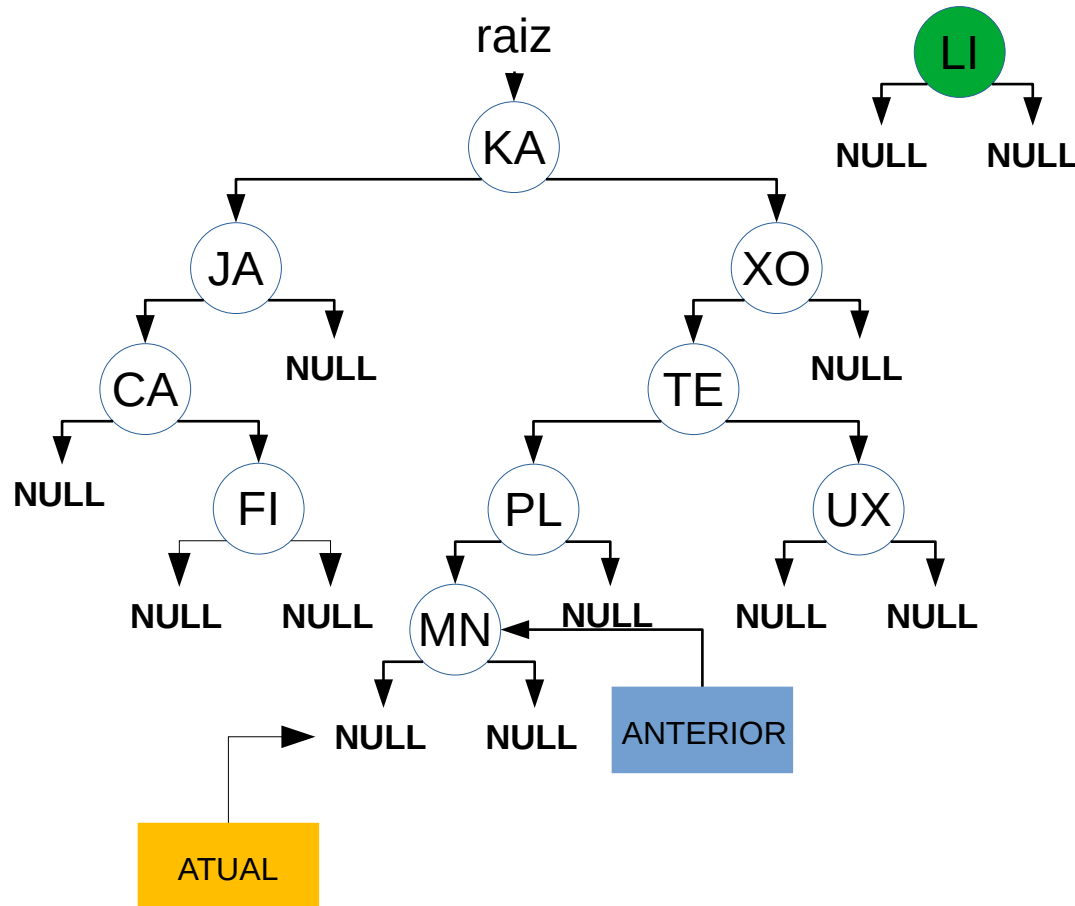


inserirIterativamente(umValor):

```

novo ← criar_noh(umValor);
se (raiz = NULL) {
    raiz ← novo;
}
senão {
    atual ← raiz;
    enquanto (atual ≠ NULL) {
        anterior ← atual;
        se (atual.valor > umValor) {
            atual ← atual.esquerdo;
        } senão {
            atual ← atual.direito;
        }
    }
    novo.pai ← anterior;
    se (anterior.valor > novo.valor) {
        anterior.esquerdo ← novo;
    } senão {
        anterior.direito ← novo;
    }
}
```

Inserir: KA JA XO CA TE UX PL MN FI **LI** MA

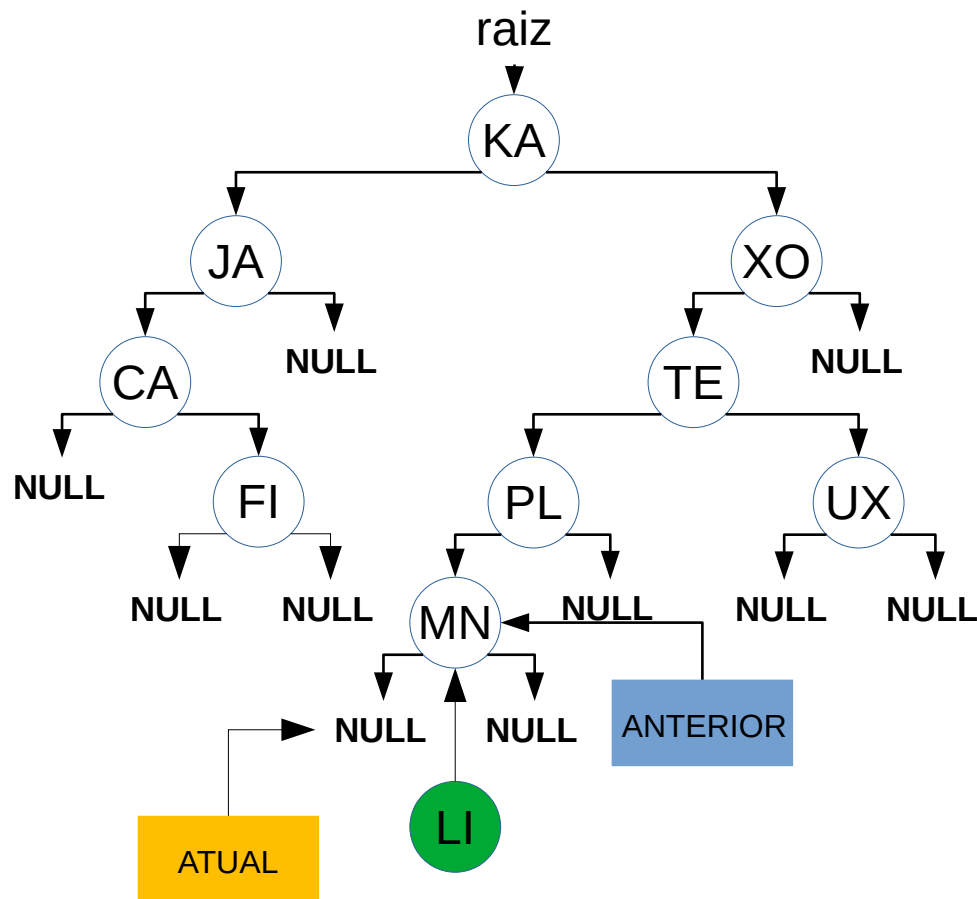


inserirIterativamente(umValor):

```

novo ← criar_noh(umValor);
se (raiz = NULL) {
    raiz ← novo;
}
senão {
    atual ← raiz;
    enquanto (atual ≠ NULL) {
        anterior ← atual;
        se (atual.valor > umValor) {
            atual ← atual.esquerdo;
        } senão {
            atual ← atual.direito;
        }
    }
    novo.pai ← anterior;
    se (anterior.valor > novo.valor) {
        anterior.esquerdo ← novo;
    } senão {
        anterior.direito ← novo;
    }
}
```

Inserir: KA JA XO CA TE UX PL MN FI **LI** MA

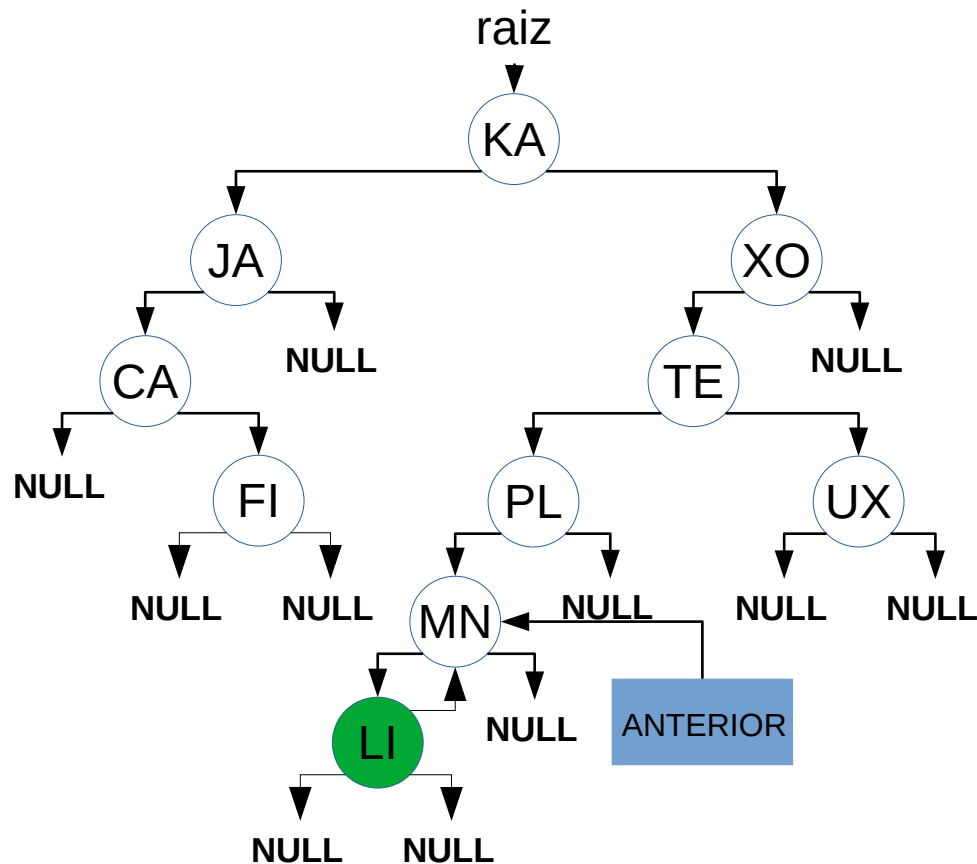


inserirIterativamente(umValor):

```

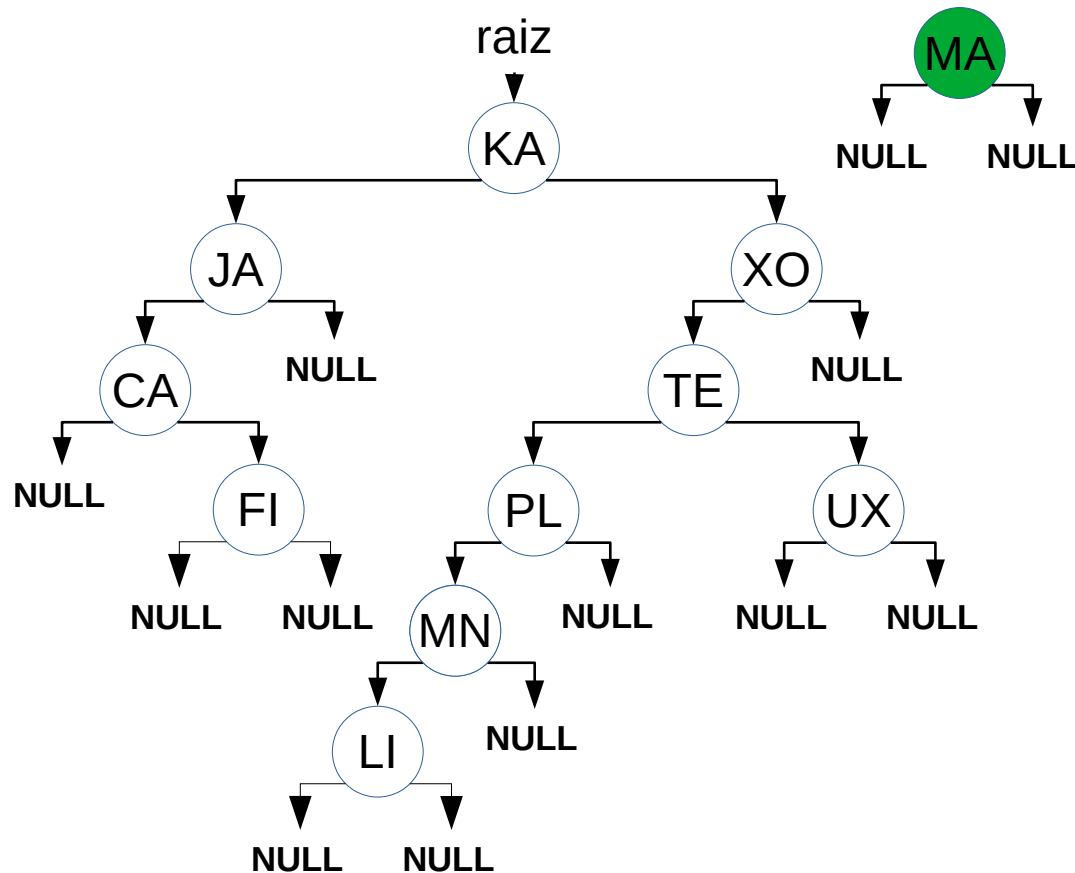
novo ← criar_noh(umValor);
se (raiz = NULL) {
    raiz ← novo;
}
senão {
    atual ← raiz;
    enquanto (atual ≠ NULL) {
        anterior ← atual;
        se (atual.valor > umValor) {
            atual ← atual.esquerdo;
        } senão {
            atual ← atual.direito;
        }
    }
    novo.pai ← anterior;
    se (anterior.valor > novo.valor) {
        anterior.esquerdo ← novo;
    } senão {
        anterior.direito ← novo;
    }
}
```

Inserir: KA JA XO CA TE UX PL MN FI **LI** MA



```
inserirIterativamente(umValor):
novo ← criar_noh(umValor);
se (raiz = NULL) {
    raiz ← novo;
}
senão {
    atual ← raiz;
    enquanto (atual ≠ NULL) {
        anterior ← atual;
        se (atual.valor > umValor) {
            atual ← atual.esquerdo;
        } senão {
            atual ← atual.direito;
        }
    }
    novo.pai ← anterior;
    se (anterior.valor > novo.valor) {
        anterior.esquerdo ← novo;
    } senão {
        anterior.direito ← novo;
    }
}
```

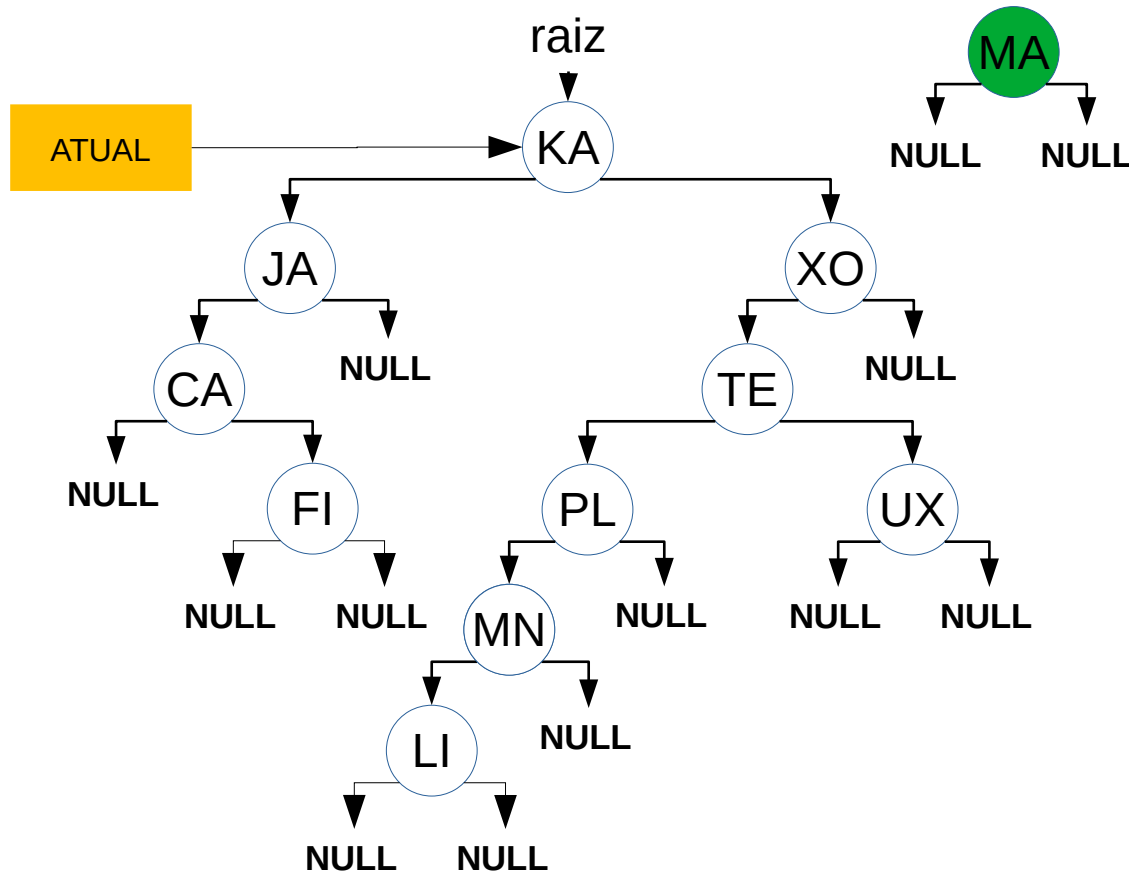
Inserir: KA JA XO CA TE UX PL MN FI LI MA



inserirIterativamente(umValor):

```
    novo ← criar_noh(umValor);
    se (raiz = NULL) {
        raiz ← novo;
    }
    senão {
        atual ← raiz;
        enquanto (atual ≠ NULL) {
            anterior ← atual;
            se (atual.valor > umValor) {
                atual ← atual.esquerdo;
            } senão {
                atual ← atual.direito;
            }
        }
        novo.pai ← anterior;
        se (anterior.valor > novo.valor) {
            anterior.esquerdo ← novo;
        } senão {
            anterior.direito ← novo;
        }
    }
```

Inserir: KA JA XO CA TE UX PL MN FI LI MA

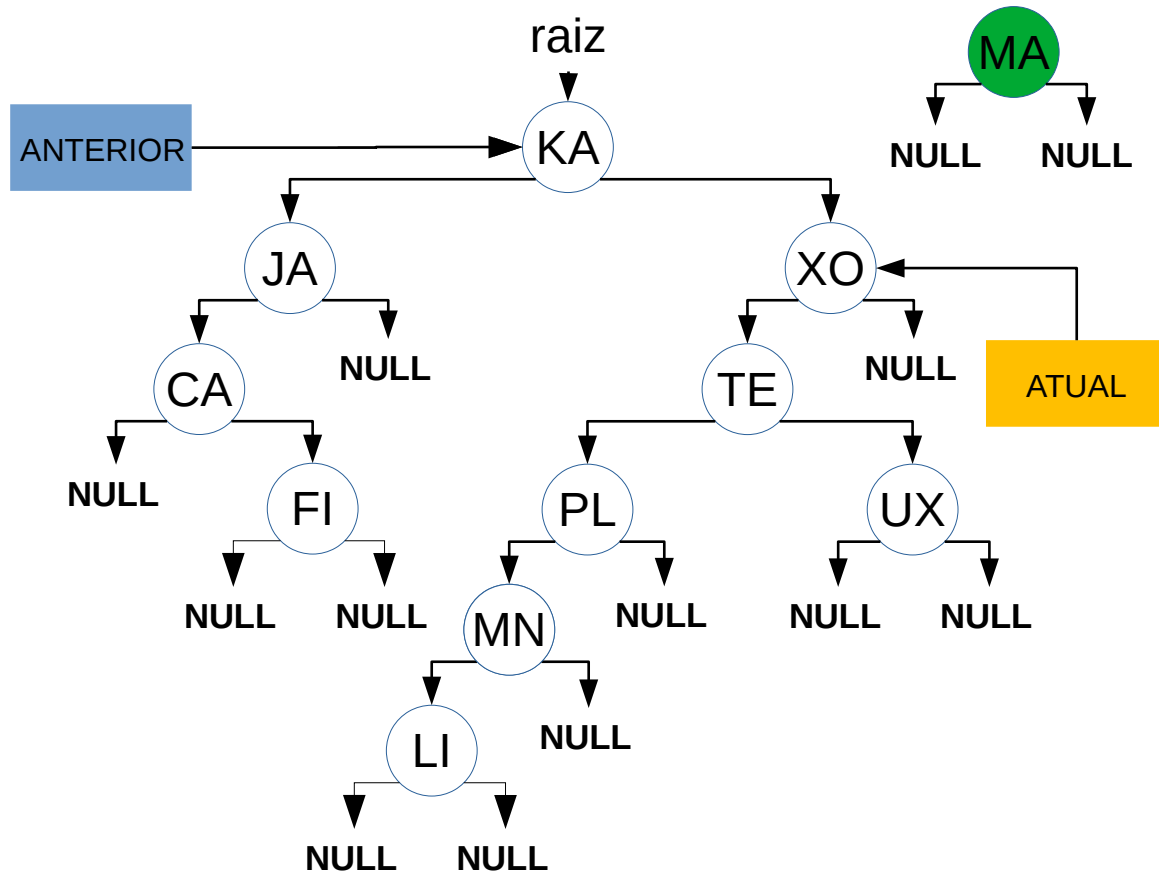


inserirIterativamente(umValor):

```

novo ← criar_noh(umValor);
se (raiz = NULL) {
    raiz ← novo;
}
senão {
    atual ← raiz;
    enquanto (atual ≠ NULL) {
        anterior ← atual;
        se (atual.valor > umValor) {
            atual ← atual.esquerdo;
        } senão {
            atual ← atual.direito;
        }
    }
    novo.pai ← anterior;
    se (anterior.valor > novo.valor) {
        anterior.esquerdo ← novo;
    } senão {
        anterior.direito ← novo;
    }
}
```


Inserir: KA JA XO CA TE UX PL MN FI LI MA

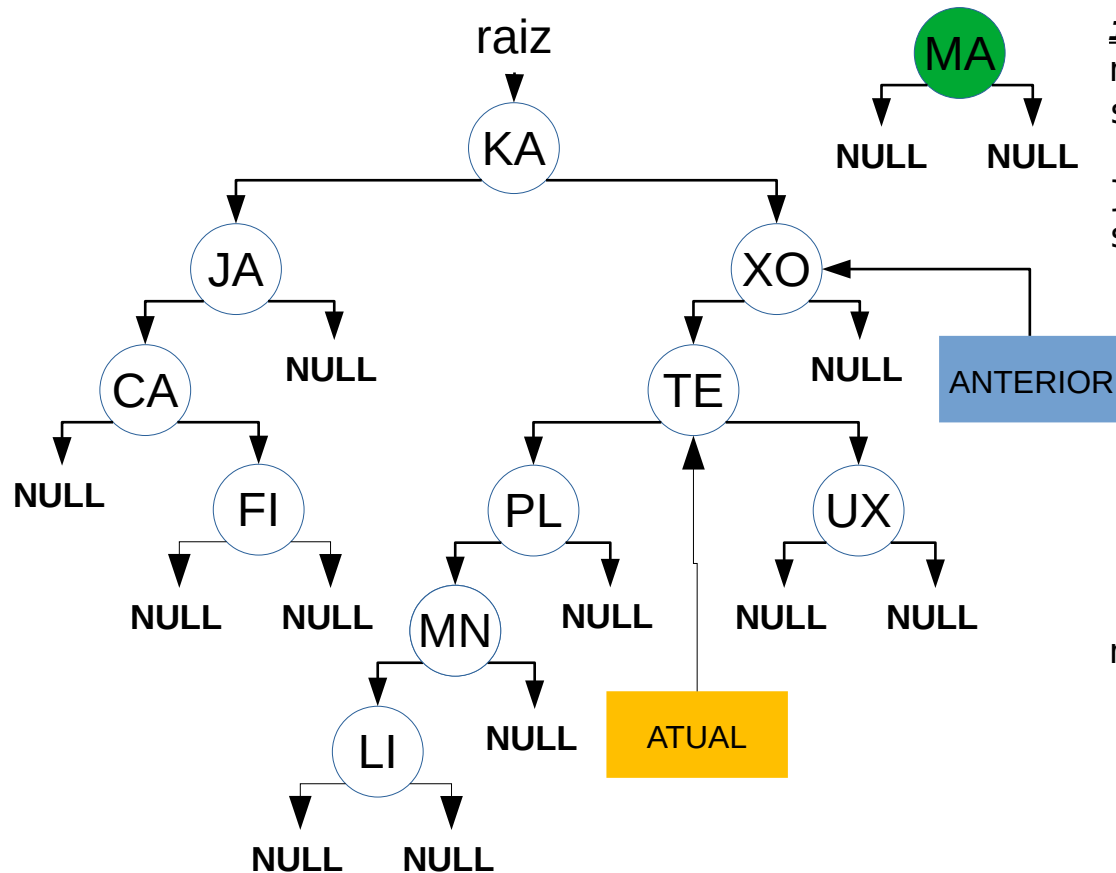


inserirIterativamente(umValor):

```

novo ← criar_noh(umValor);
se (raiz = NULL) {
    raiz ← novo;
}
senão {
    atual ← raiz;
    enquanto (atual ≠ NULL) {
        anterior ← atual;
        se (atual.valor > umValor) {
            atual ← atual.esquerdo;
        } senão {
            atual ← atual.direito;
        }
    }
    novo.pai ← anterior;
    se (anterior.valor > novo.valor) {
        anterior.esquerdo ← novo;
    } senão {
        anterior.direito ← novo;
    }
}
```

Inserir: KA JA XO CA TE UX PL MN FI LI MA

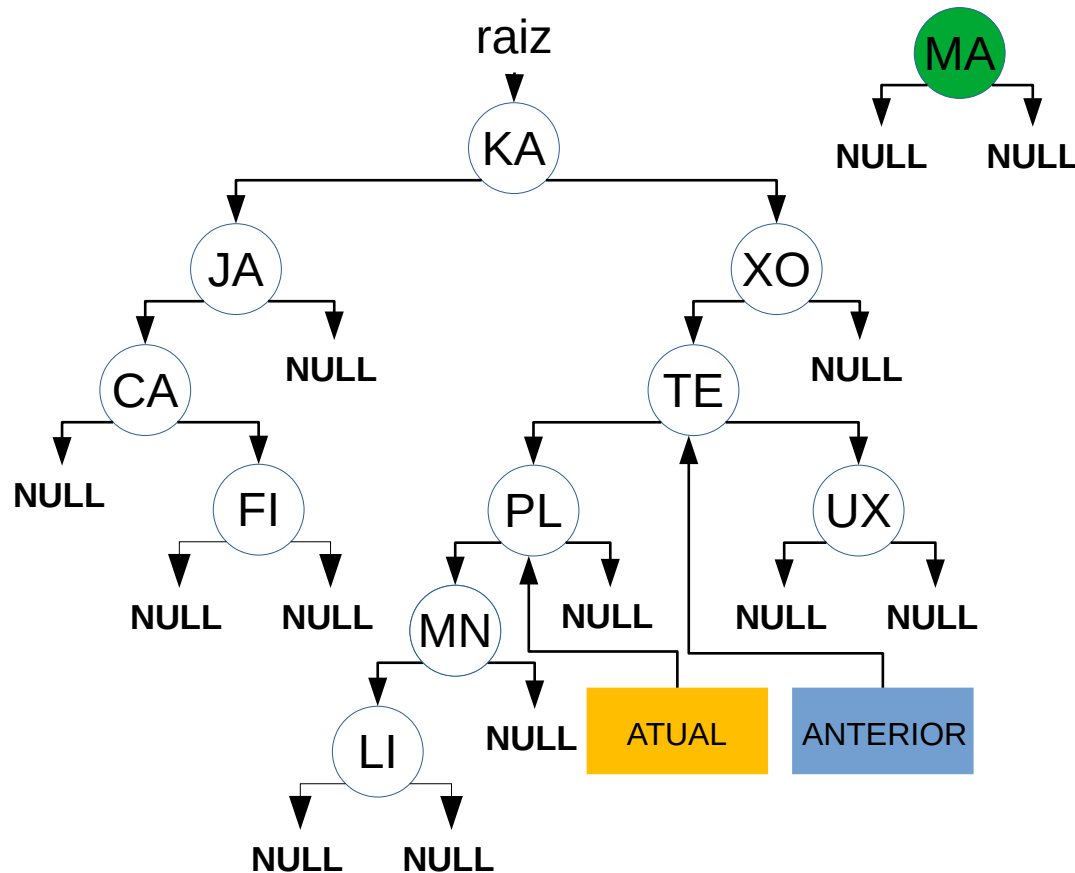


inserirIterativamente(umValor):

```

novo ← criar_noh(umValor);
se (raiz = NULL) {
    raiz ← novo;
}
senão {
    atual ← raiz;
    enquanto (atual ≠ NULL) {
        anterior ← atual;
        se (atual.valor > umValor) {
            atual ← atual.esquerdo;
        } senão {
            atual ← atual.direito;
        }
    }
    novo.pai ← anterior;
    se (anterior.valor > novo.valor) {
        anterior.esquerdo ← novo;
    } senão {
        anterior.direito ← novo;
    }
}
```

Inserir: KA JA XO CA TE UX PL MN FI LI MA

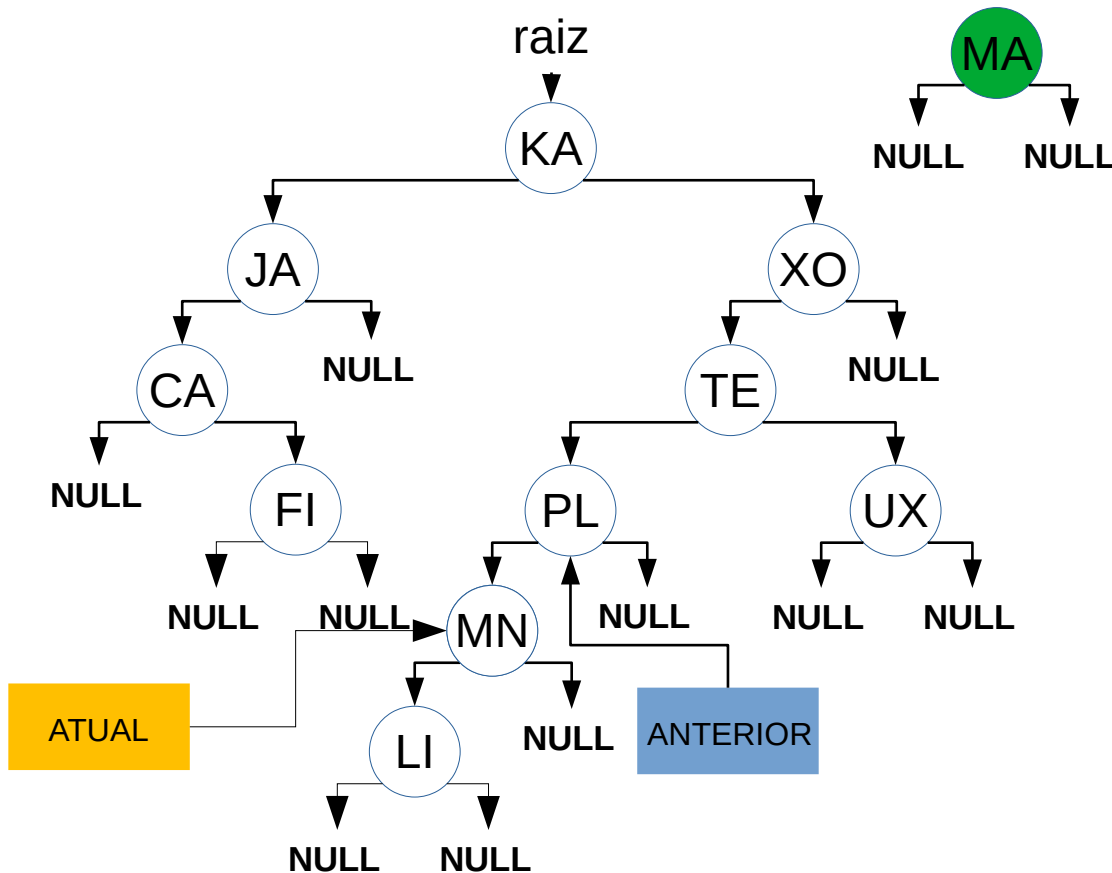


inserirIterativamente(umValor):

```

novo ← criar_noh(umValor);
se (raiz = NULL) {
    raiz ← novo;
}
senão {
    atual ← raiz;
    enquanto (atual ≠ NULL) {
        anterior ← atual;
        se (atual.valor > umValor) {
            atual ← atual.esquerdo;
        } senão {
            atual ← atual.direito;
        }
    }
    novo.pai ← anterior;
    se (anterior.valor > novo.valor) {
        anterior.esquerdo ← novo;
    } senão {
        anterior.direito ← novo;
    }
}
```

Inserir: KA JA XO CA TE UX PL MN FI LI MA



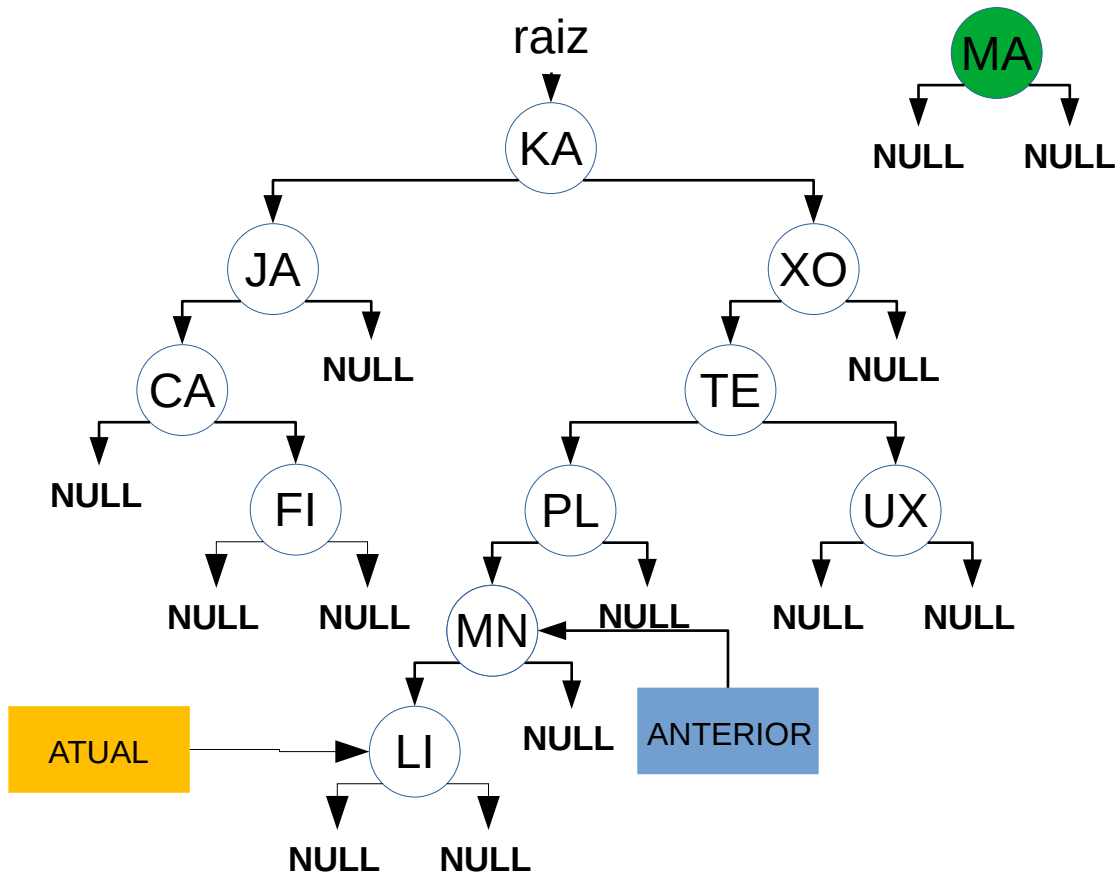
inserirIterativamente(umValor):

```

novo ← criar_noh(umValor);
se (raiz = NULL) {
    raiz ← novo;
}
senão {
    atual ← raiz;
    enquanto (atual ≠ NULL) {
        anterior ← atual;
        se (atual.valor > umValor) {
            atual ← atual.esquerdo;
        } senão {
            atual ← atual.direito;
        }
    }
    novo.pai ← anterior;
    se (anterior.valor > novo.valor) {
        anterior.esquerdo ← novo;
    } senão {
        anterior.direito ← novo;
    }
}

```

Inserir: KA JA XO CA TE UX PL MN FI LI MA

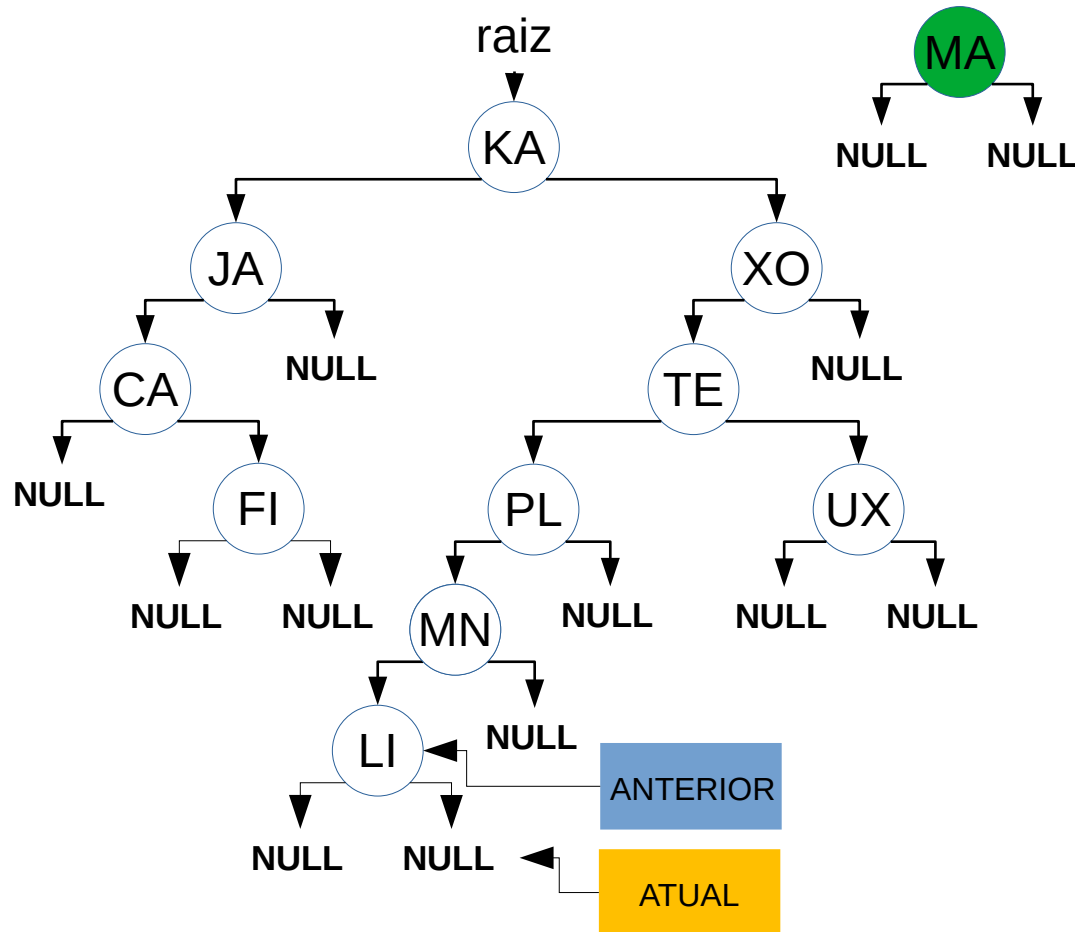


inserirIterativamente(umValor):

```

novo ← criar_noh(umValor);
se (raiz = NULL) {
    raiz ← novo;
}
senão {
    atual ← raiz;
    enquanto (atual ≠ NULL) {
        anterior ← atual;
        se (atual.valor > umValor) {
            atual ← atual.esquerdo;
        } senão {
            atual ← atual.direito;
        }
    }
    novo.pai ← anterior;
    se (anterior.valor > novo.valor) {
        anterior.esquerdo ← novo;
    } senão {
        anterior.direito ← novo;
    }
}
```

Inserir: KA JA XO CA TE UX PL MN FI LI MA

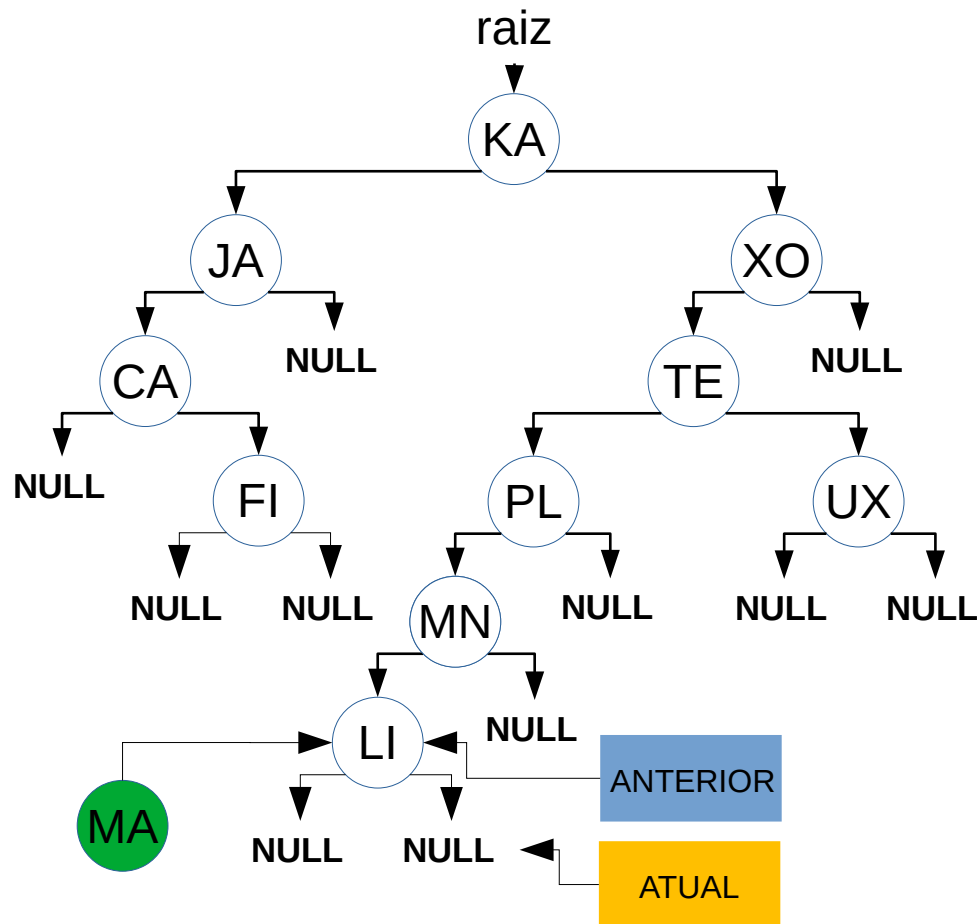


inserirIterativamente(umValor):

```

novo ← criar_noh(umValor);
se (raiz = NULL) {
    raiz ← novo;
}
senão {
    atual ← raiz;
    enquanto (atual ≠ NULL) {
        anterior ← atual;
        se (atual.valor > umValor) {
            atual ← atual.esquerdo;
        } senão {
            atual ← atual.direito;
        }
    }
    novo.pai ← anterior;
    se (anterior.valor > novo.valor) {
        anterior.esquerdo ← novo;
    } senão {
        anterior.direito ← novo;
    }
}
```

Inserir: KA JA XO CA TE UX PL MN FI LI MA

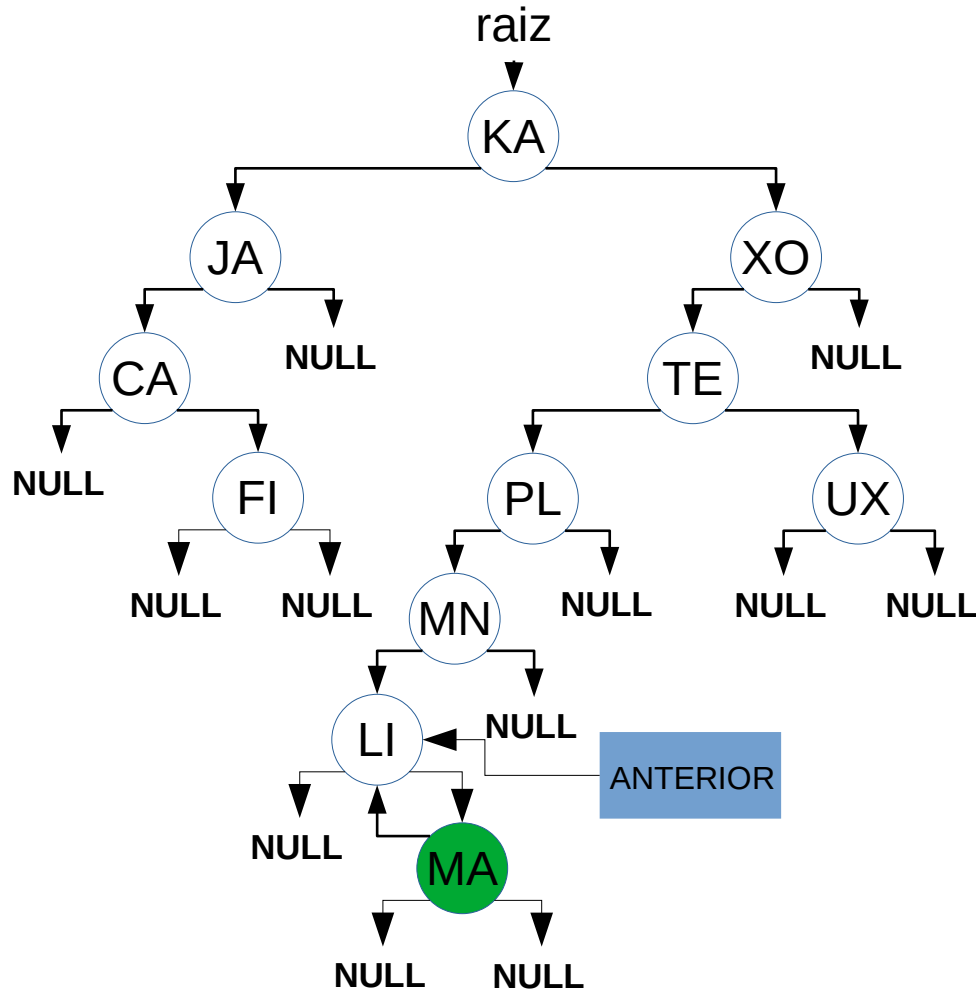


inserirIterativamente(umValor):

```

novo ← criar_noh(umValor);
se (raiz = NULL) {
    raiz ← novo;
}
senão {
    atual ← raiz;
    enquanto (atual ≠ NULL) {
        anterior ← atual;
        se (atual.valor > umValor) {
            atual ← atual.esquerdo;
        } senão {
            atual ← atual.direito;
        }
    }
    novo.pai ← anterior;
    se (anterior.valor > novo.valor) {
        anterior.esquerdo ← novo;
    } senão {
        anterior.direito ← novo;
    }
}
```

Inserir: KA JA XO CA TE UX PL MN FI LI MA



inserirIterativamente(umValor):

```

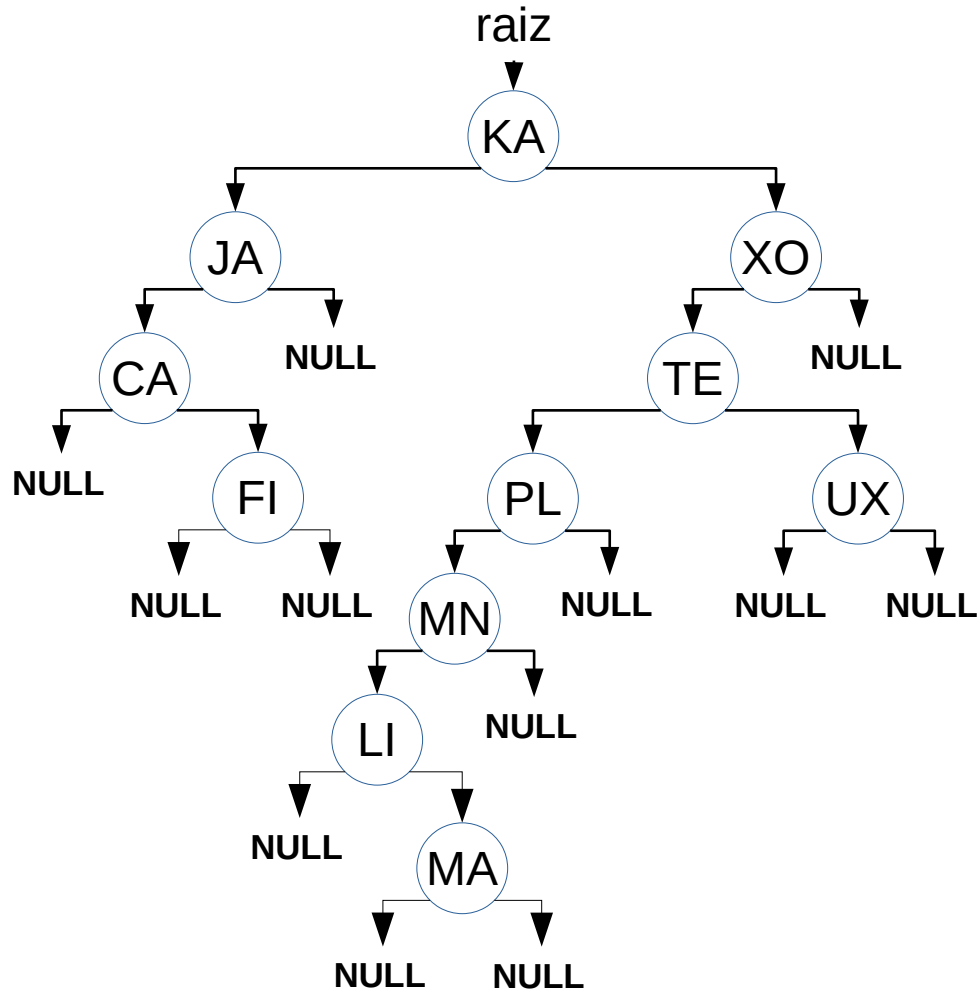
novo ← criar_noh(umValor);
se (raiz = NULL) {
    raiz ← novo;
}
senão {
    atual ← raiz;
    enquanto (atual ≠ NULL) {
        anterior ← atual;
        se (atual.valor > umValor) {
            atual ← atual.esquerdo;
        } senão {
            atual ← atual.direito;
        }
    }
    novo.pai ← anterior;
    se (anterior.valor > novo.valor) {
        anterior.esquerdo ← novo;
    } senão {
        anterior.direito ← novo;
    }
}
```


Remover



remove(umValor):

nohRemover ← buscaAux(umValor);

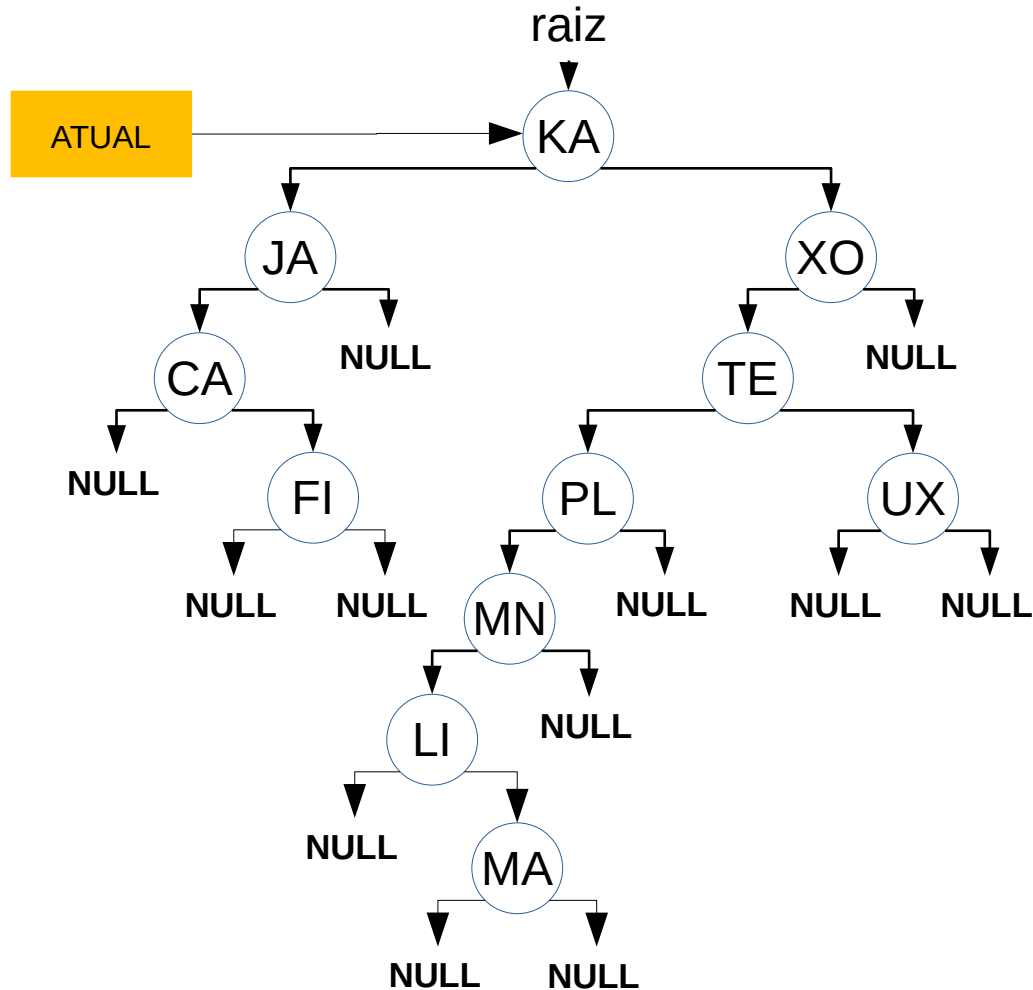


Remover

PL

CA

TE



buscaAux(umValor):

atual ← raiz;

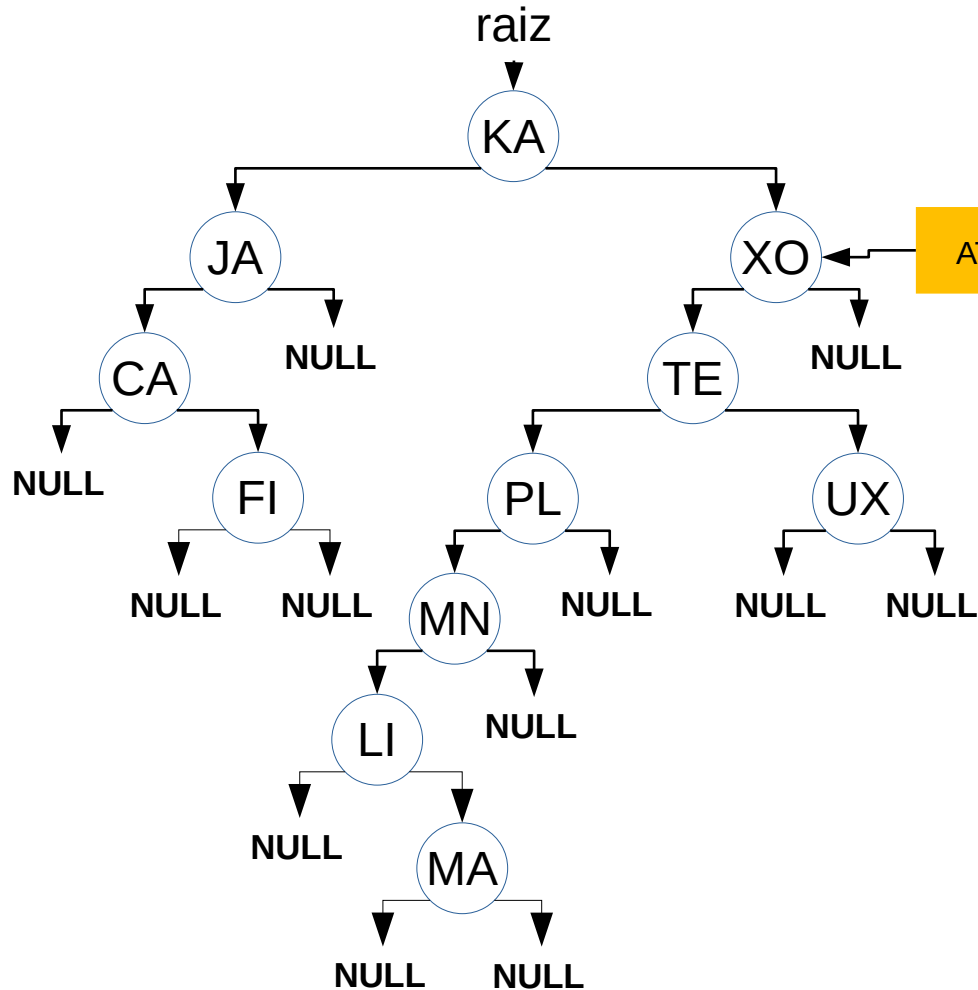
```
enquanto (atual ≠ NULL) {  
    se (atual.valor = umValor) {  
        retorna atual;  
    } senão se (atual.valor > umValor) {  
        atual ← atual->esquerdo;  
    } senão {  
        atual ← atual.direito;  
    }  
}  
retorna atual;
```

Remover

PL

CA

TE



buscaAux(umValor):

atual ← raiz;

enquanto (atual ≠ NULL) {

se (atual.valor = umValor) {
retorna atual;

} senão se (atual.valor > umValor) {
atual ← atual->esquerdo;

} senão {
atual ← atual.direito;

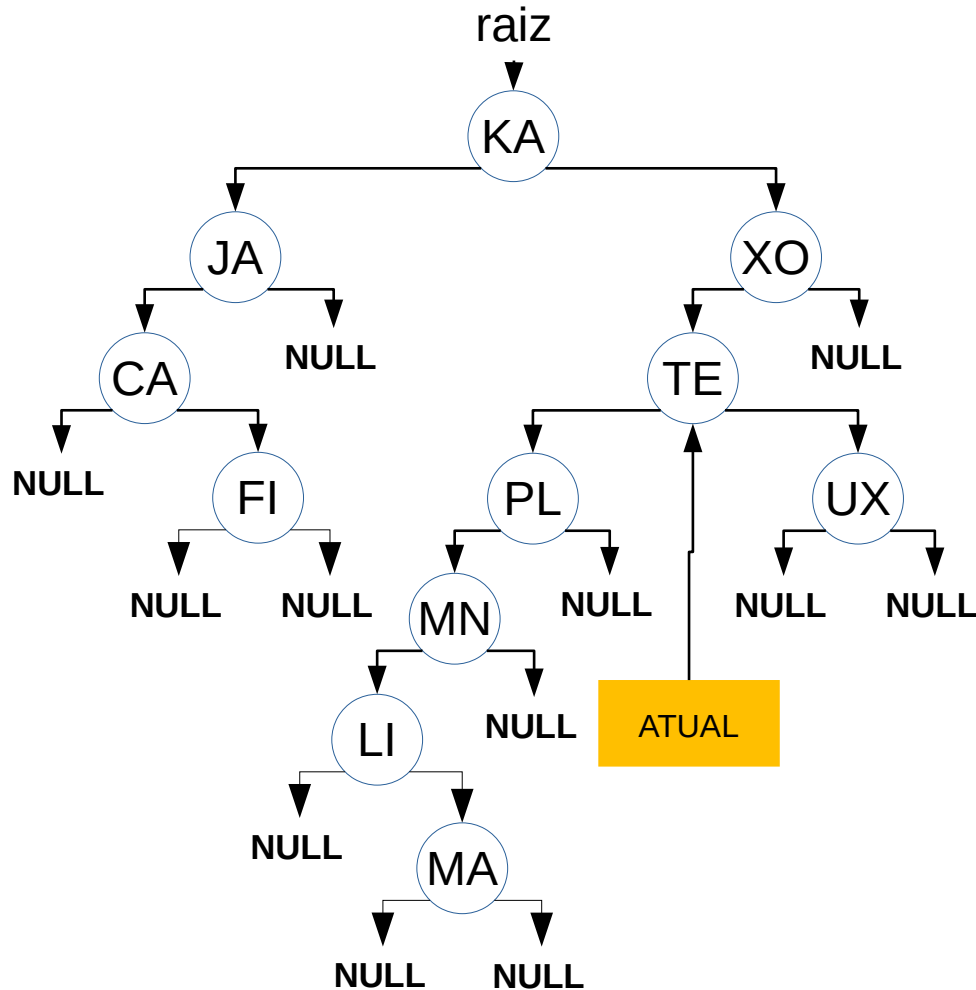
}
retorna atual;

Remover

PL

CA

TE



buscaAux(umValor):

atual ← raiz;

enquanto (atual ≠ NULL) {

se (atual.valor = umValor) {
retorna atual;

} senão se (atual.valor > umValor) {
atual ← atual->esquerdo;

} senão {
atual ← atual.direito;
}

}

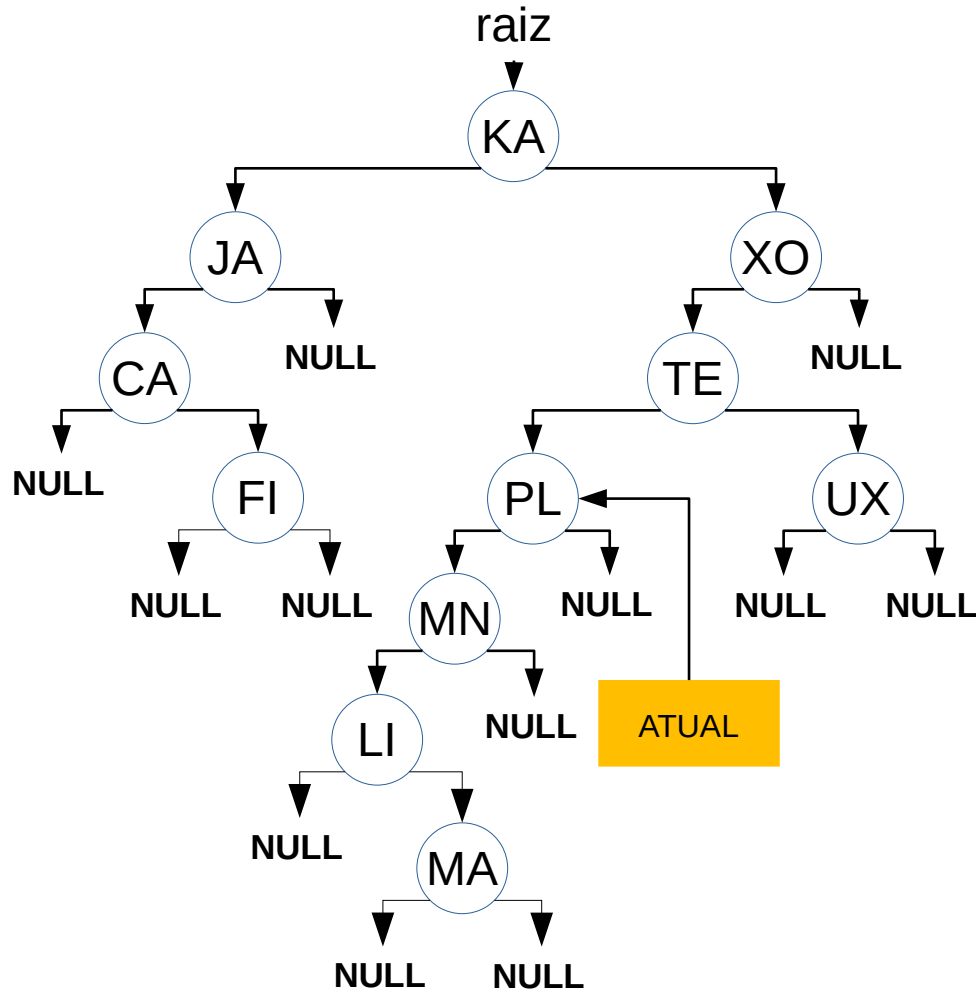
retorna atual;

Remover

PL

CA

TE



buscaAux(umValor):

atual ← raiz;

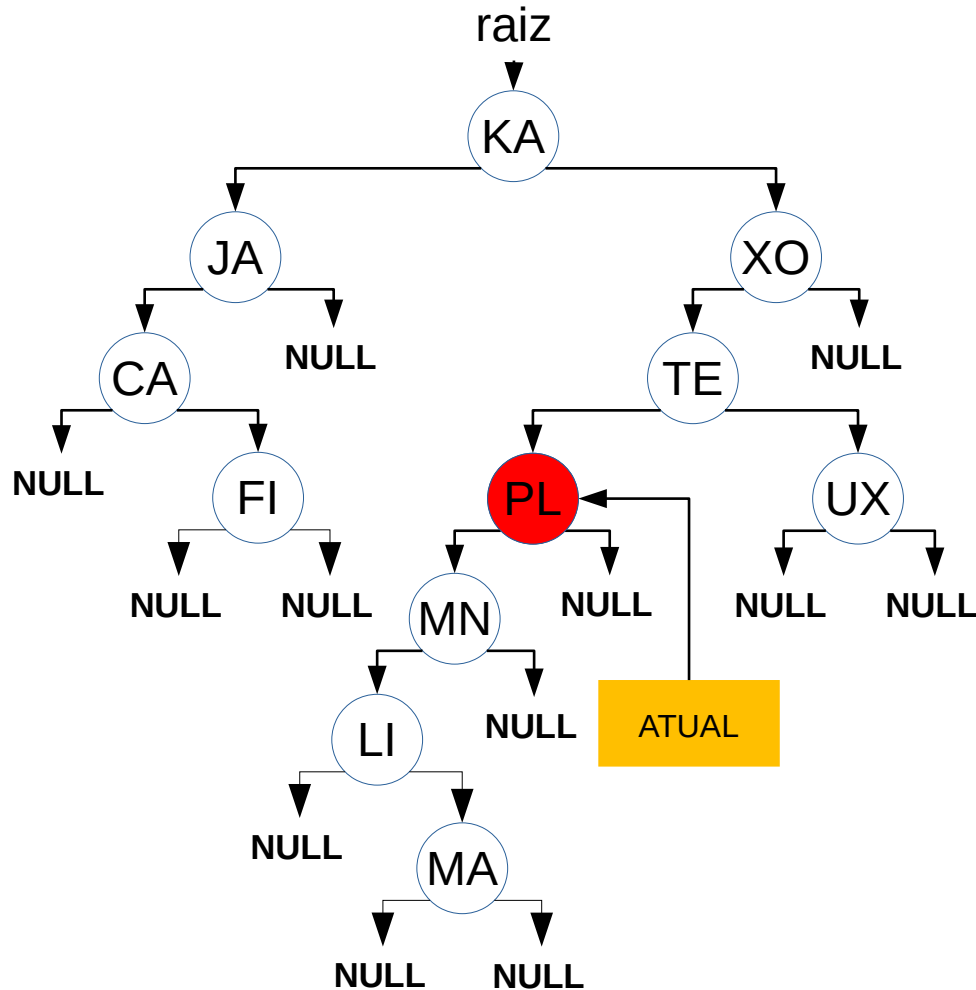
```
enquanto (atual ≠ NULL) {  
    se (atual.valor = umValor) {  
        retorna atual;  
    } senão se (atual.valor > umValor) {  
        atual ← atual->esquerdo;  
    } senão {  
        atual ← atual->direito;  
    }  
}  
retorna atual;
```

Remover

PL

CA

TE



buscaAux(umValor):

atual ← raiz;

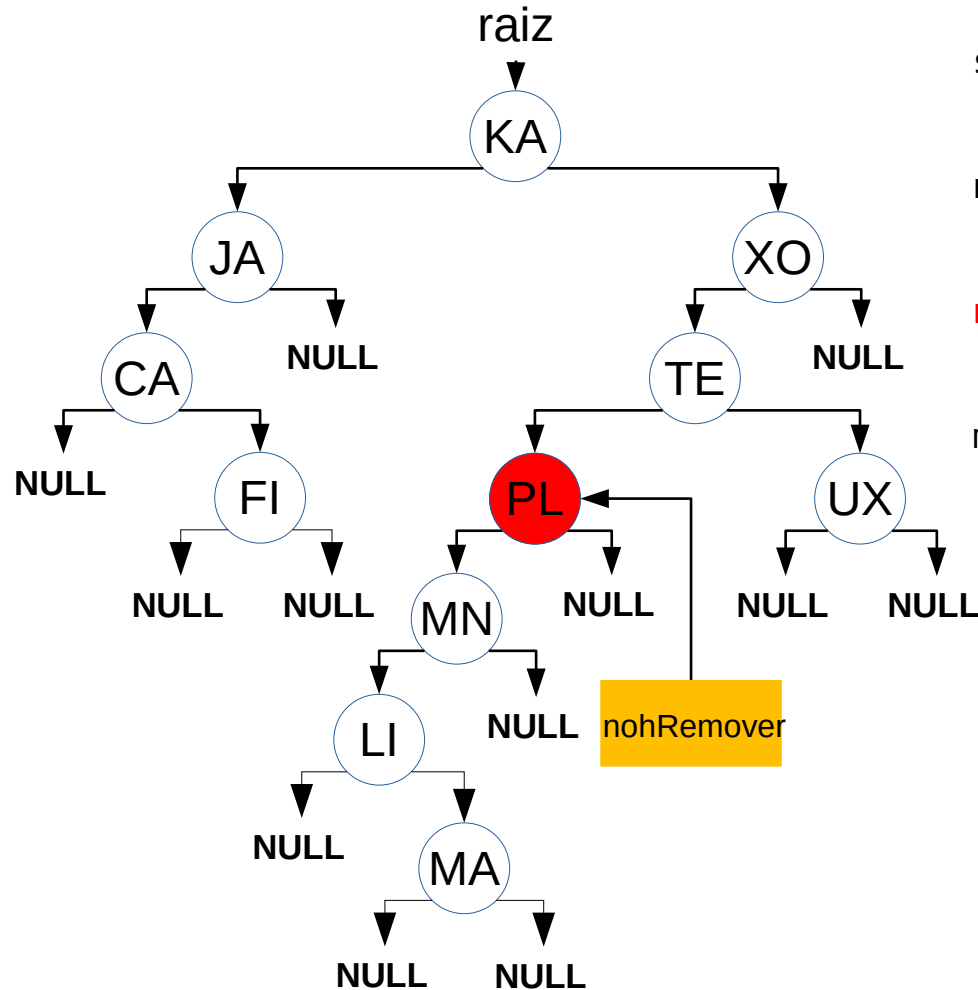
```
enquanto (atual ≠ NULL) {  
    se (atual.valor = umValor) {  
        retorna atual;  
    } senão se (atual.valor > umValor) {  
        atual ← atual->esquerdo;  
    } senão {  
        atual ← atual.direito;  
    }  
}  
retorna atual;
```

Remover

PL

CA

TE



remove(umValor):

nohRemover ← buscaAux(umValor);

se (nohRemover = NULL)

 geraErro("Nó não encontrado!");

senão {

 se (nohRemover.esquerdo = NULL) {

 transplanta(nohRemover,

nohRemover.direito);

 } senão se (nohRemover.direito = NULL) {

 transplanta(nohRemover,

nohRemover.esquerdo);

 } senão {

 sucessor ←

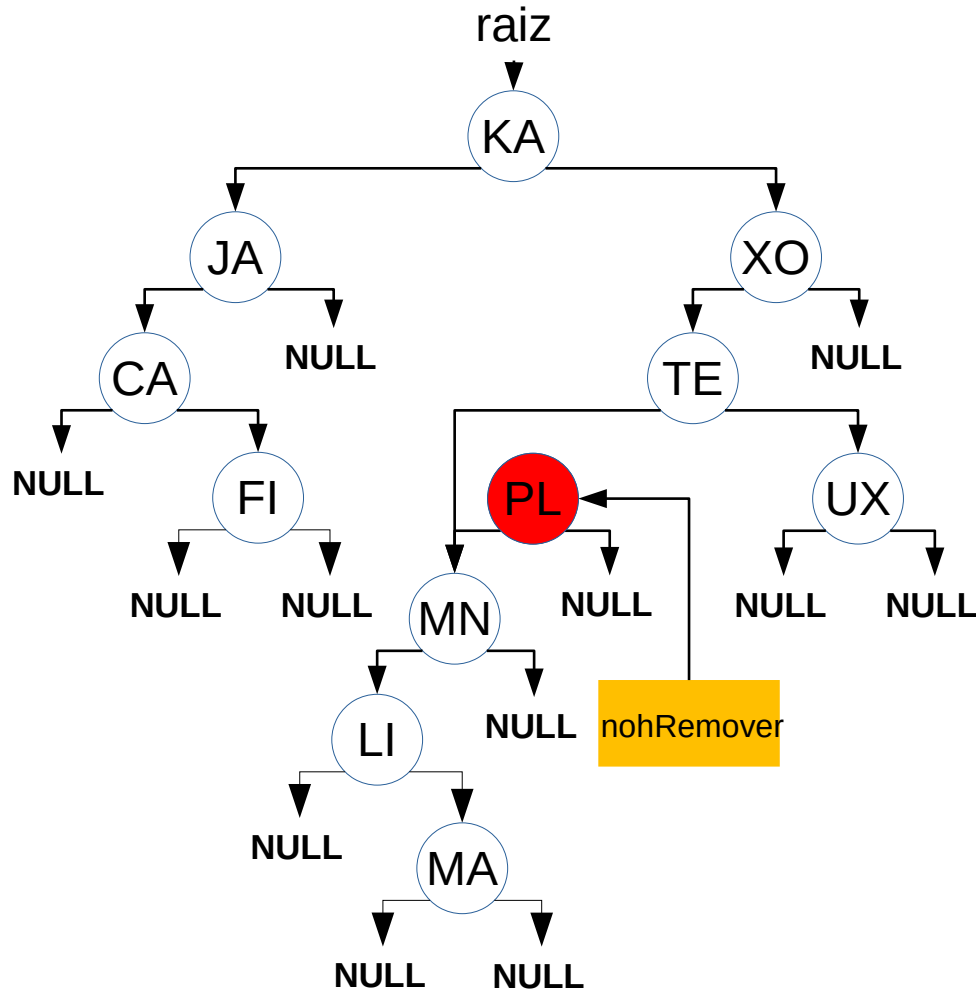
minimoAux(nohRemover.direito);

Remover

PL

CA

TE



transplanta(antigo, novo):

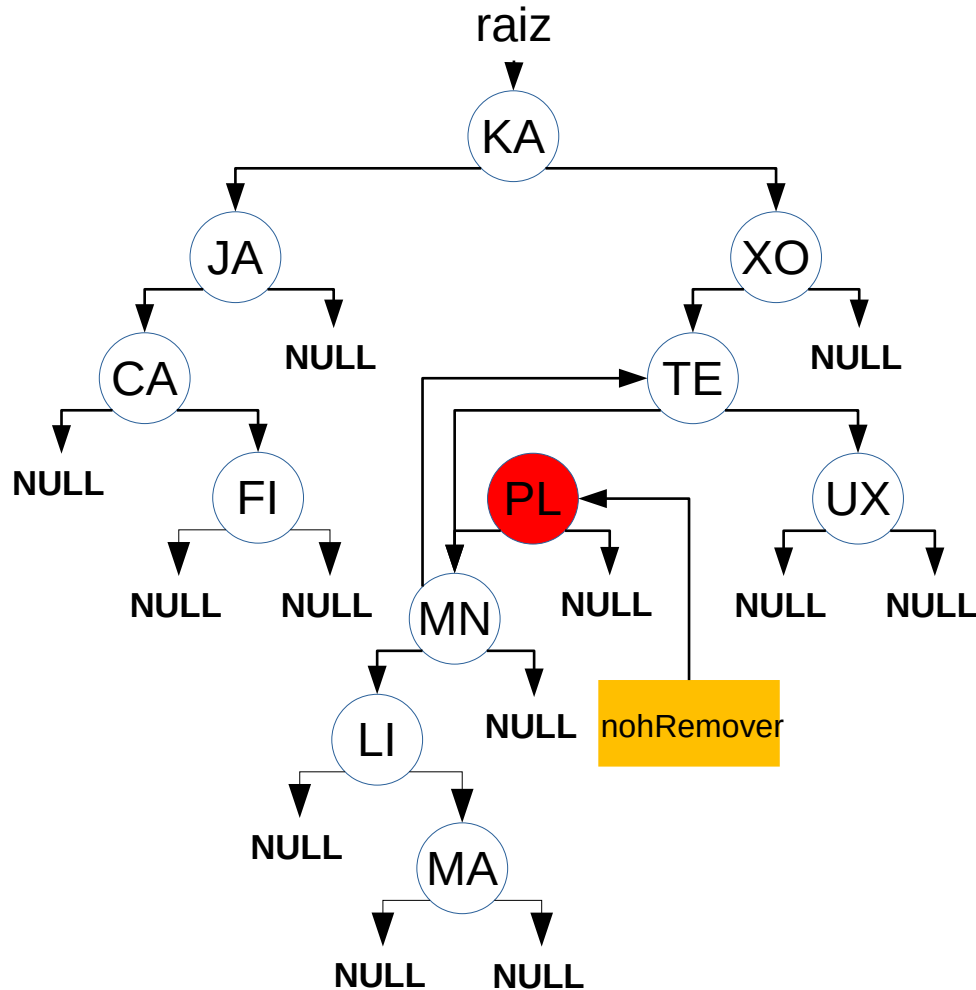
```
se (raiz = antigo) {  
    raiz ← novo;  
}  
senão se (antigo = antigo.pai.esquerdo) {  
    antigo.pai.esquerdo ← novo;  
}  
senão { // antigo = antigo.pai.direito  
    antigo.pai.direito ← novo;  
}  
se (novo ≠ NULO) {  
    novo.pai ← antigo.pai;  
}
```


Remover

PL

CA

TE



transplanta(antigo, novo):

```
se (raiz = antigo) {  
    raiz ← novo;  
}  
senão se (antigo = antigo.pai.esquerdo) {  
    antigo.pai.esquerdo ← novo;  
}  
senão { // antigo = antigo.pai.direito  
    antigo.pai.direito ← novo;  
}  
se (novo ≠ NULO) {  
    novo.pai ← antigo.pai;  
}
```

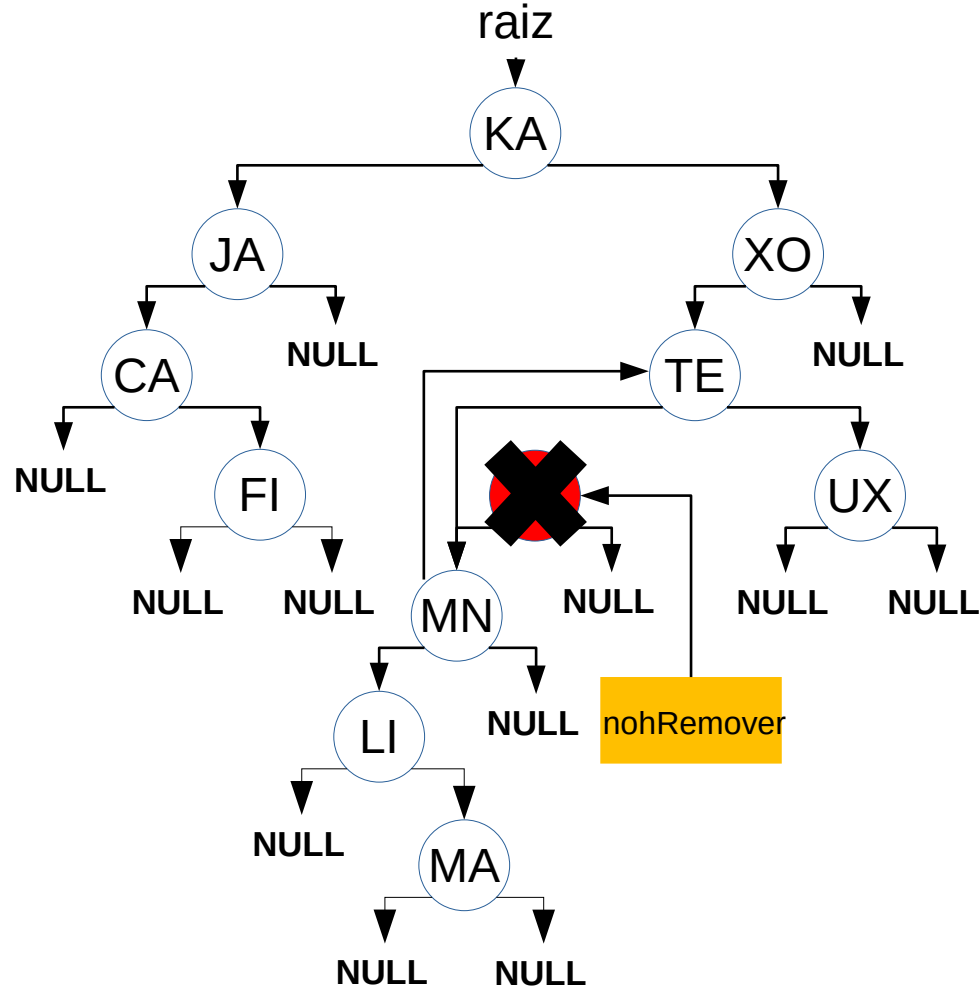
Remover

PL

CA

TE

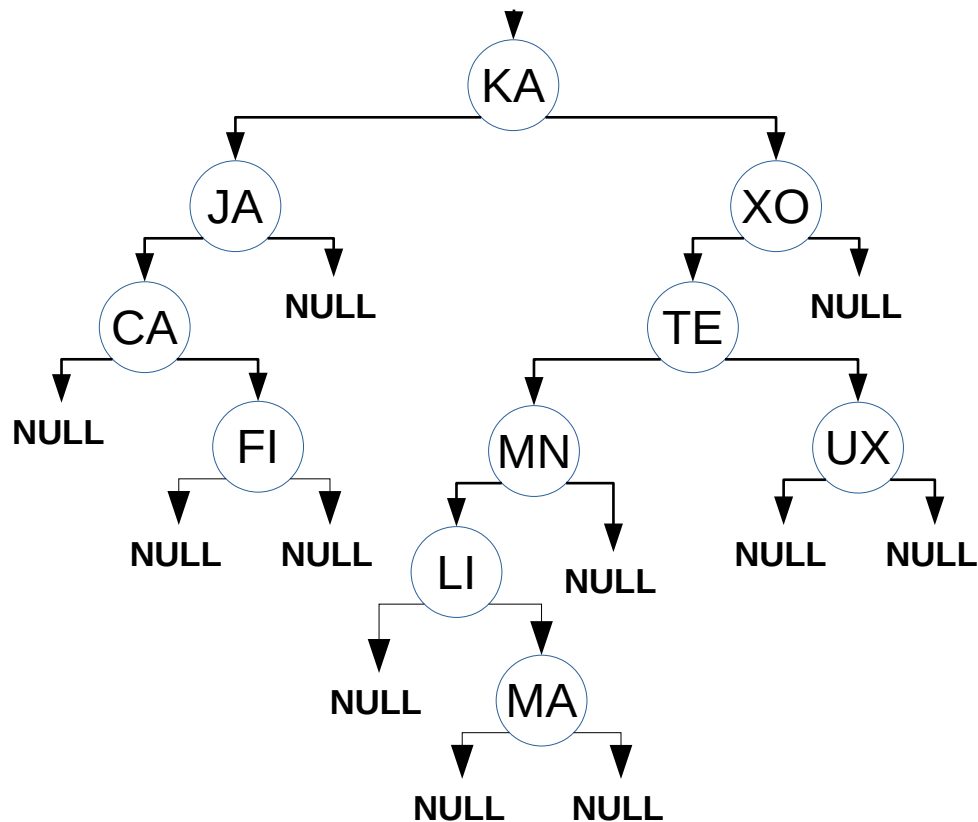
remove(umValor):
apagar(nohRemover);



Remover



raiz



remove(umValor):

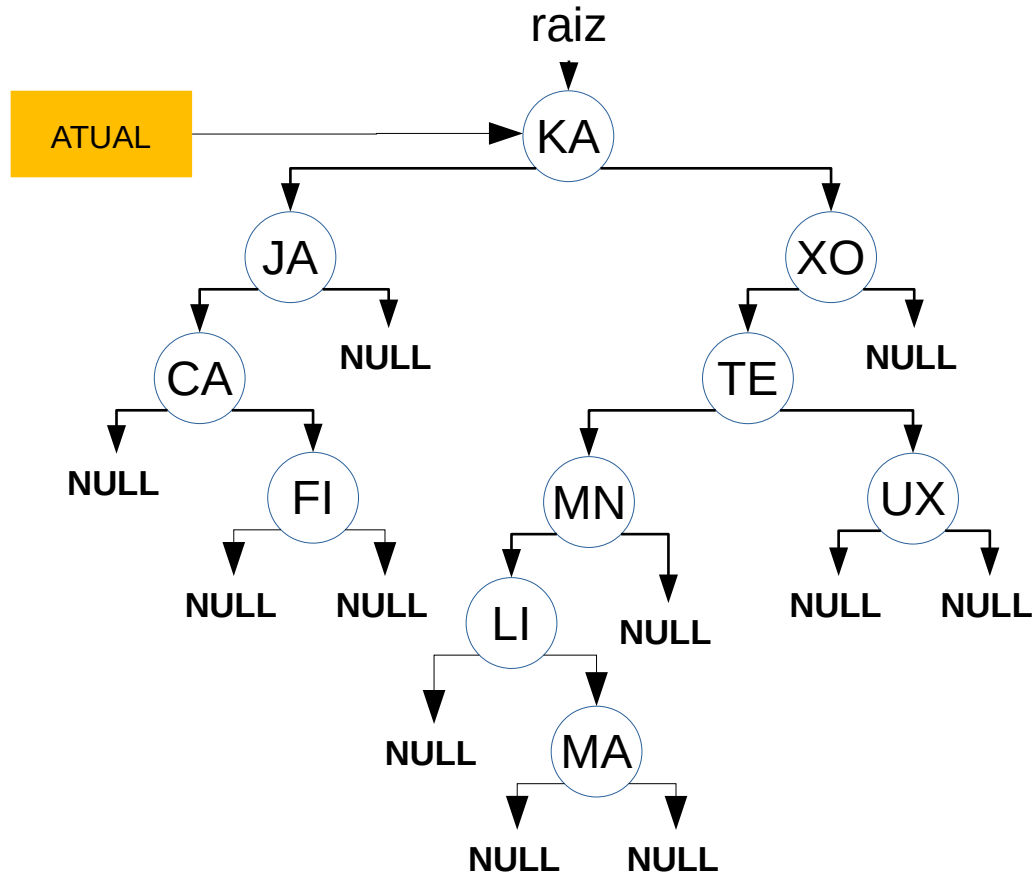
nohRemover ← buscaAux(umValor);

Remover

PL

CA

TE



buscaAux(umValor):

atual \leftarrow raiz;

```
enquanto (atual  $\neq$  NULL) {  
    se (atual.valor = umValor) {  
        retorna atual;  
    } senão se (atual.valor > umValor) {  
        atual  $\leftarrow$  atual->esquerdo;  
    } senão {  
        atual  $\leftarrow$  atual.direito;  
    }  
}  
retorna atual;
```

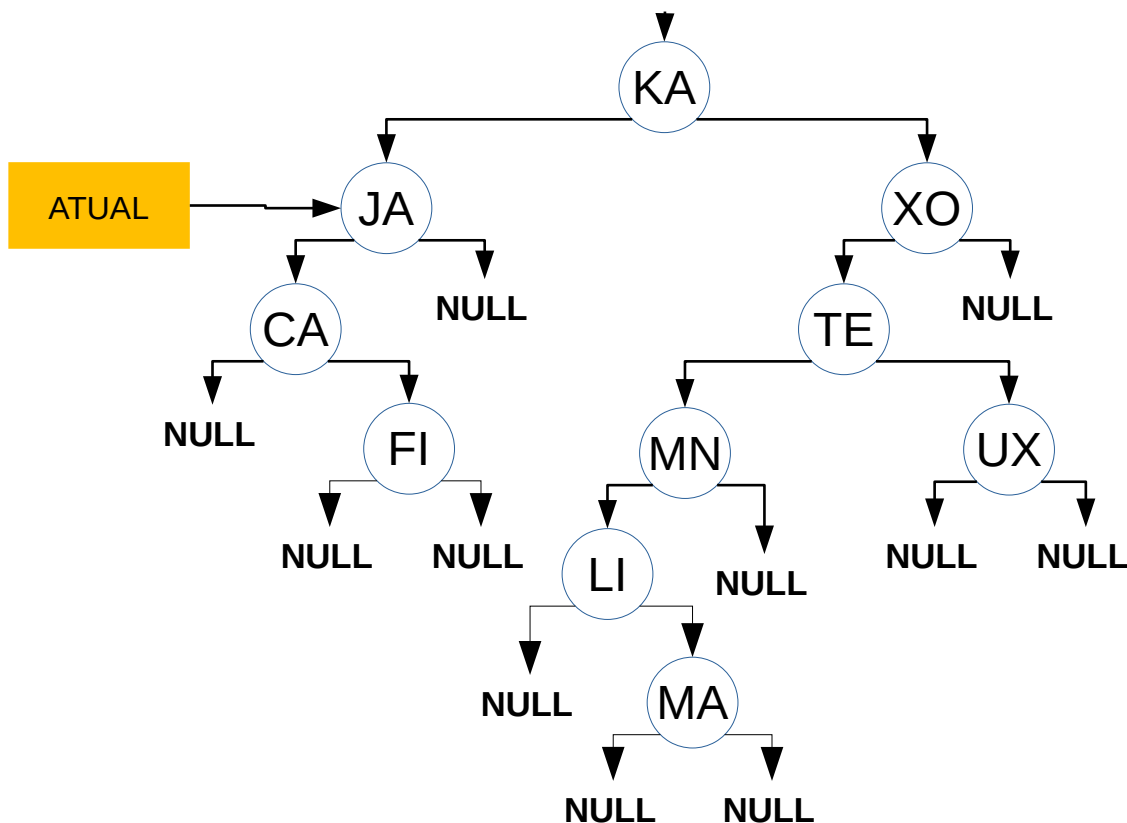
Remover

PL

CA

TE

raiz



buscaAux(umValor):

atual ← raiz;

```
enquanto (atual ≠ NULL) {  
    se (atual.valor = umValor) {  
        retorna atual;  
    } senão se (atual.valor > umValor) {  
        atual ← atual->esquerdo;  
    } senão {  
        atual ← atual.direito;  
    }  
}  
retorna atual;
```

Remover

PL

CA

TE

raiz

KA

JA

XO

ATUAL

CA

FI

TE

MN

UX

LI

MA

buscaAux(umValor):

atual ← raiz;

enquanto (atual ≠ NULL) {

se (atual.valor = umValor) {
retorna atual;

} senão se (atual.valor > umValor) {
atual ← atual->esquerdo;

} senão {
atual ← atual.direito;

}

retorna atual;

Remover



raiz

KA

JA

XO

ATUAL

CA

FI

MN

UX

LI

MA

NULL

NULL

NULL

NULL

NULL

NULL

NULL

NULL

NULL

NULL

buscaAux(umValor):

atual ← raiz;

enquanto (atual ≠ NULL) {

se (atual.valor = umValor) {

retorna atual;

} senão se (atual.valor > umValor) {

atual ← atual->esquerdo;

} senão {

atual ← atual.direito;

}

}

retorna atual;

Remover



raiz

KA

JA

XO

CA

TE

FI

MN

UX

LI

MA

nohRemover

NULL

NULL

NULL

NULL

NULL

NULL

NULL

NULL

NULL

NULL

NULL

remove(umValor):

nohRemover ← buscaAux(umValor);

se (nohRemover = NULL)

geraErro("Nó não encontrado!");

senão {

se (nohRemover.esquerdo = NULL) {

transplanta(nohRemover,

nohRemover.direito);

} senão se (nohRemover.direito = NULL) {

transplanta(nohRemover,

nohRemover.esquerdo);

} senão {

sucessor ←

minimoAux(nohRemover.direito);

Remover

PL

CA

TE

raiz

KA

JA

XO

antigo

CA

NULL

NULL

FI

NULL

NULL

MN

LI

NULL

NULL

MA

NULL

NULL

NULL

TE

UX

NULL

NULL

novo

transplanta(antigo, novo):

```
se (raiz = antigo) {
```

```
    raiz ← novo;
```

```
} senão se (antigo = antigo.pai.esquerdo) {
```

```
    antigo.pai.esquerdo ← novo;
```

```
} senão { // antigo = antigo.pai.direito
```

```
    antigo.pai.direito ← novo;
```

```
}
```

```
se (novo ≠ NULO) {
```

```
    novo.pai ← antigo.pai;
```

```
}
```

Remover



raiz

KA

JA

XO

antigo

CA

FI

MN

UX

novo

NULL

NULL

LI

NULL

NULL

NULL

NULL

MA

NULL

NULL

transplanta(antigo, novo):

se (raiz = antigo) {

 raiz ← novo;

} senão se (antigo = antigo.pai.esquerdo) {

 antigo.pai.esquerdo ← novo;

} senão { // antigo = antigo.pai.direito

 antigo.pai.direito ← novo;

}

se (novo ≠ NULO) {

 novo.pai ← antigo.pai;

}

Remover

PL

CA

TE

raiz

KA

JA

XO

antigo

CA

FI

TE

MN

UX

novo

NULL

NULL

LI

NULL

NULL

NULL

NULL

MA

NULL

NULL

transplanta(antigo, novo):

```
se (raiz = antigo) {  
    raiz ← novo;  
}  
senão se (antigo = antigo.pai.esquerdo) {  
    antigo.pai.esquerdo ← novo;  
}  
senão { // antigo = antigo.pai.direito  
    antigo.pai.direito ← novo;  
}  
}  
se (novo ≠ NULO) {  
    novo.pai ← antigo.pai;  
}
```

Remover

PL

CA

TE

remove(umValor):
apagar(nohRemover);

raiz

KA

JA

XO

TE

FI

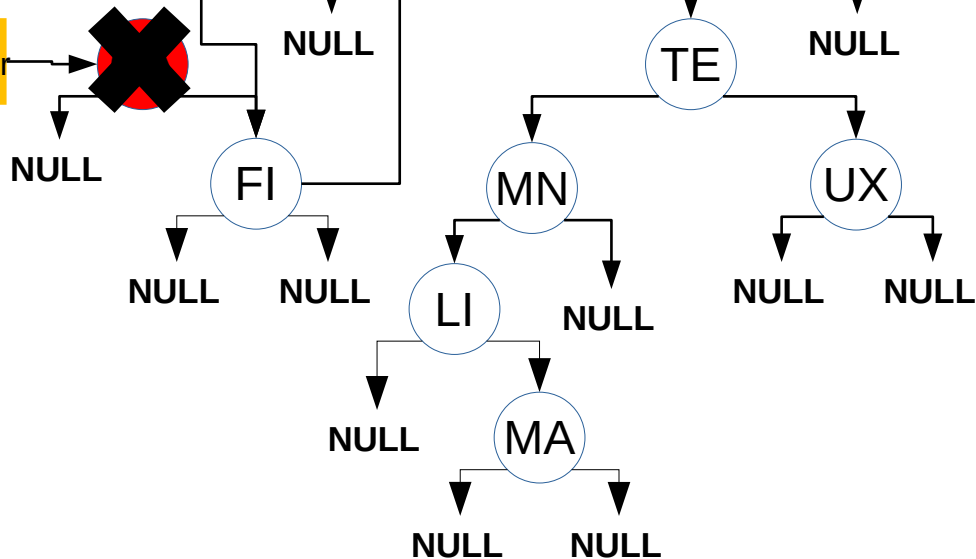
MN

UX

LI

MA

nohRemover

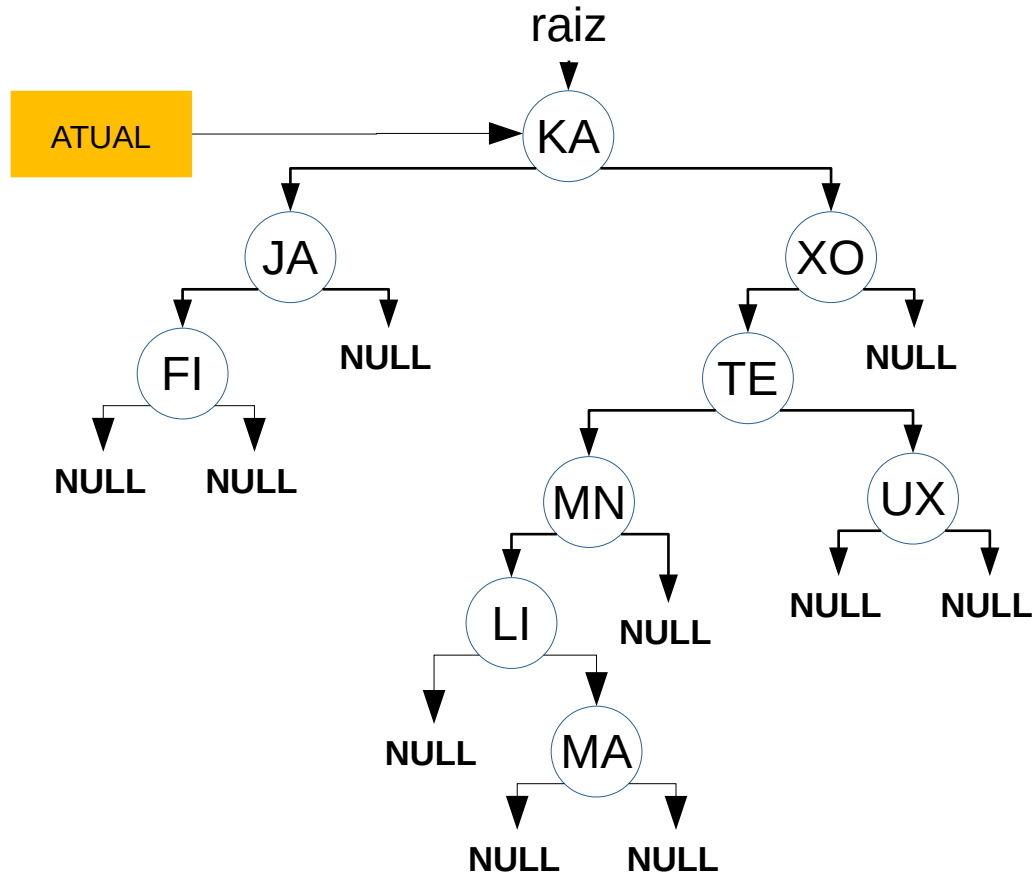


Remover

PL

CA

TE



buscaAux(umValor):

atual ← raiz;

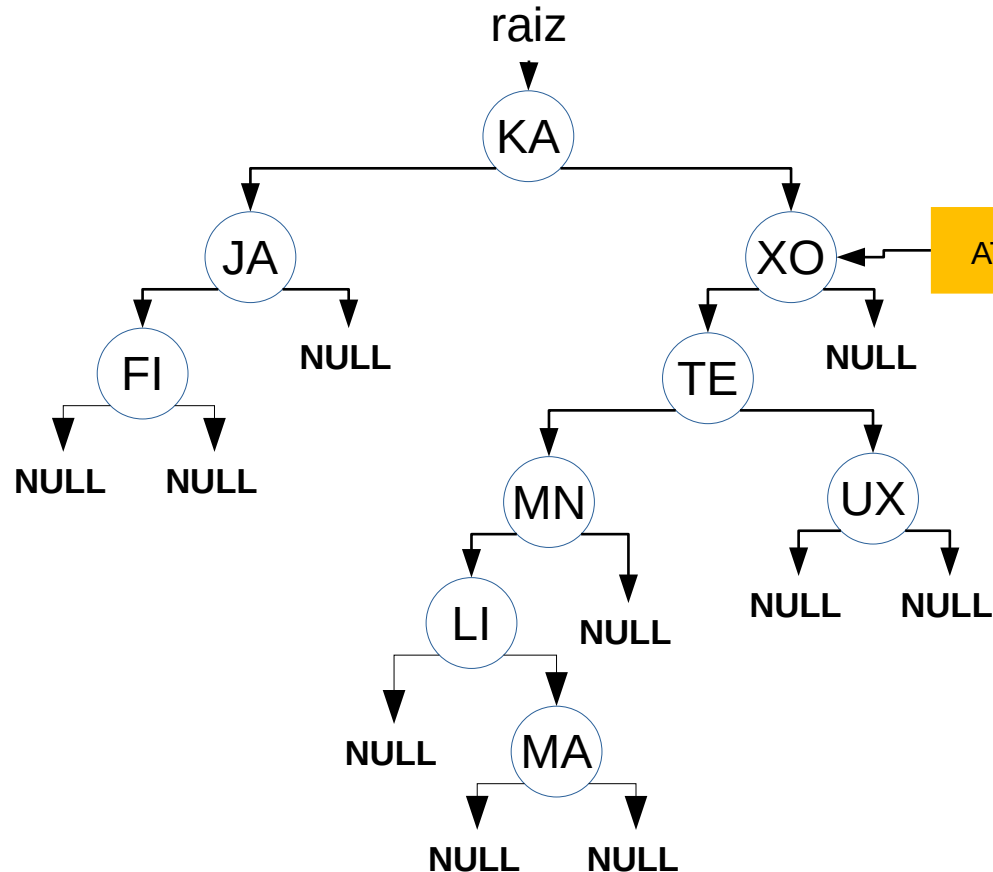
```
enquanto (atual ≠ NULL) {  
    se (atual.valor = umValor) {  
        retorna atual;  
    } senão se (atual.valor > umValor) {  
        atual ← atual->esquerdo;  
    } senão {  
        atual ← atual.direito;  
    }  
}  
retorna atual;
```

Remover

PL

CA

TE



buscaAux(umValor):

atual ← raiz;

enquanto (atual ≠ NULL) {

se (atual.valor = umValor) {

retorna atual;

} senão se (atual.valor > umValor) {

atual ← atual->esquerdo;

} senão {

atual ← atual.direito;

}

}

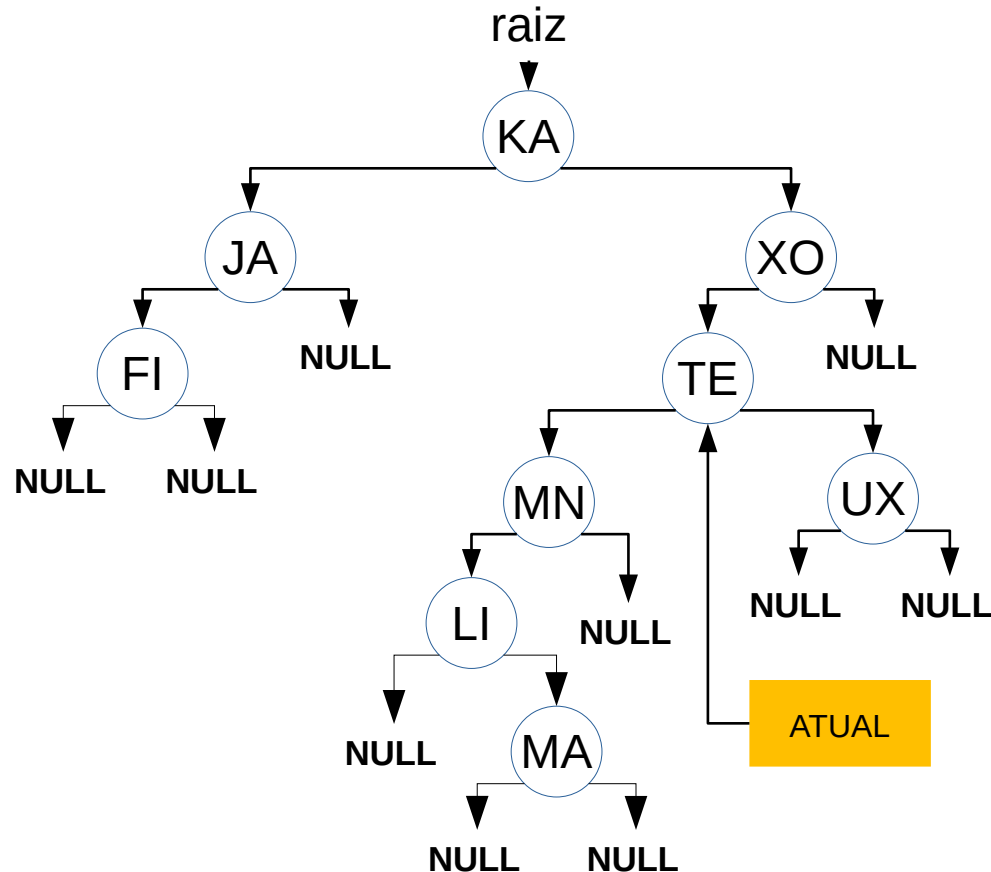
retorna atual;

Remover

PL

CA

TE



buscaAux(umValor):

atual ← raiz;

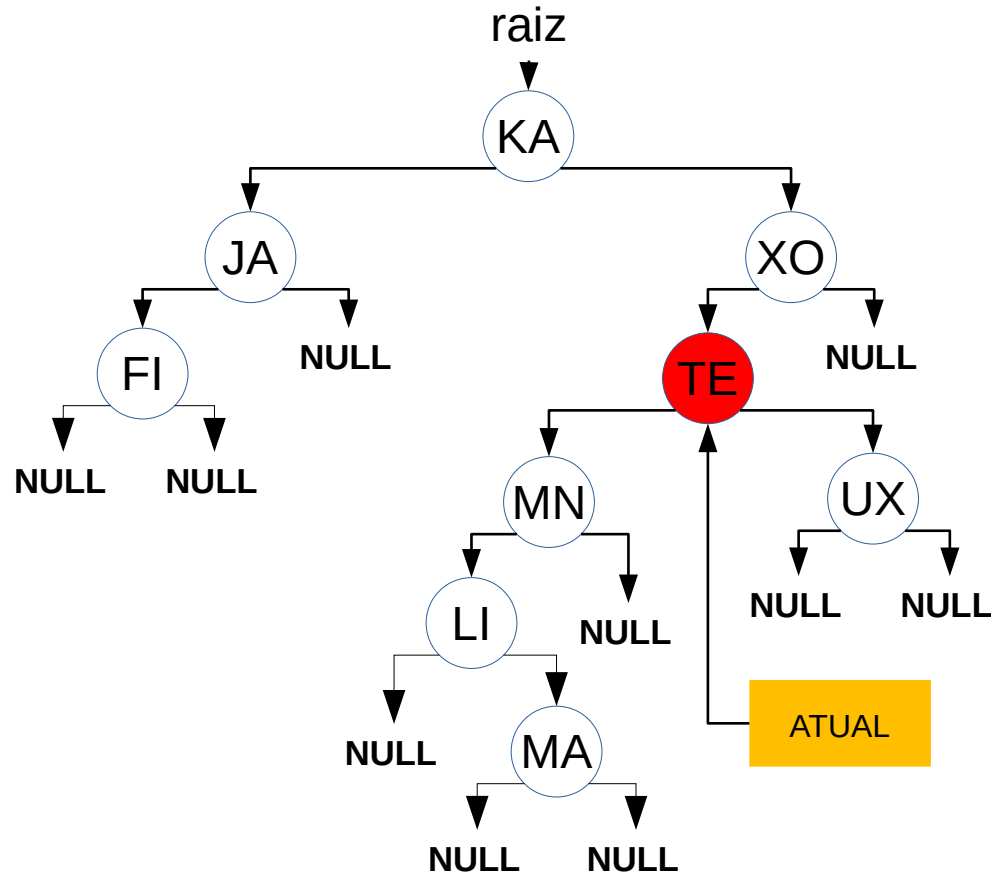
```
enquanto (atual ≠ NULL) {  
    se (atual.valor = umValor) {  
        retorna atual;  
    } senão se (atual.valor > umValor) {  
        atual ← atual->esquerdo;  
    } senão {  
        atual ← atual.direito;  
    }  
}  
retorna atual;
```

Remover

PL

CA

TE



buscaAux(umValor):

atual ← raiz;

```
enquanto (atual ≠ NULL) {  
    se (atual.valor = umValor) {  
        retorna atual;  
    } senão se (atual.valor > umValor) {  
        atual ← atual->esquerdo;  
    } senão {  
        atual ← atual.direito;  
    }  
}
```

retorna atual;

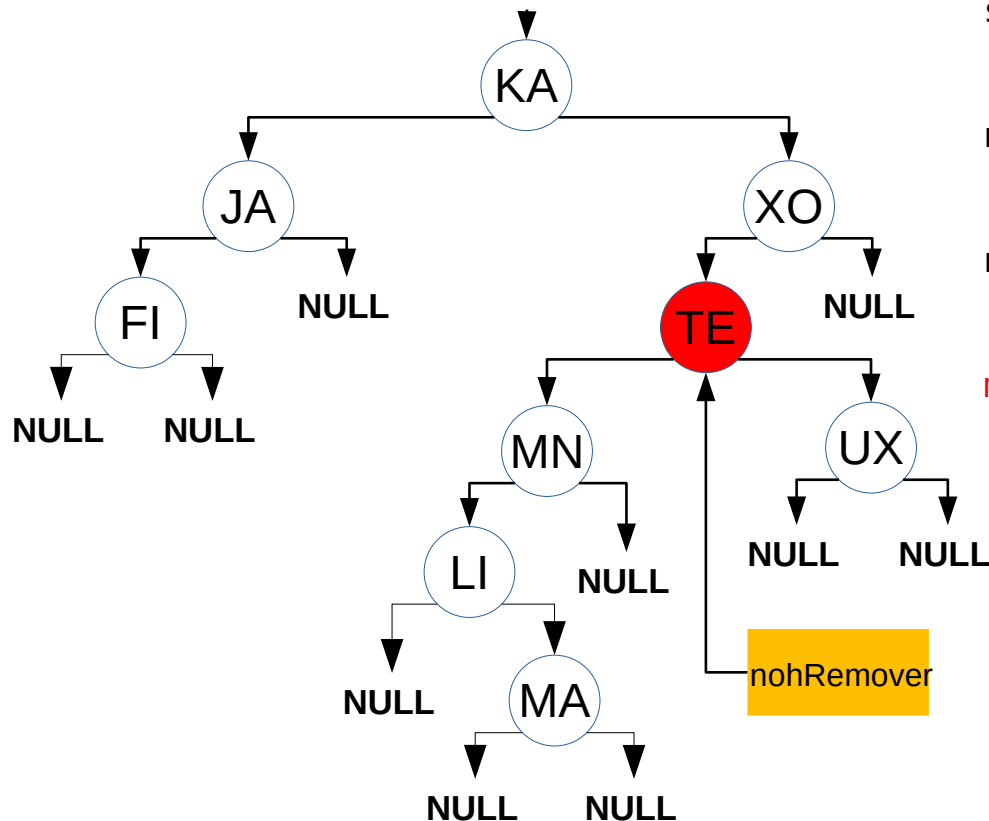
Remover

PL

CA

TE

raiz



remove(umValor):

nohRemover ← buscaAux(umValor);

se (nohRemover = NULL)

 geraErro("Nó não encontrado!");

senão {

 se (nohRemover.esquerdo = NULL) {

 transplanta(nohRemover,

nohRemover.direito);

 } senão se (nohRemover.direito = NULL) {

 transplanta(nohRemover,

nohRemover.esquerdo);

 } **senão {**

sucessor ←

minimoAux(nohRemover.direito);

Remover

PL

CA

TE

raiz

KA

JA

XO

FI

TE

MN

UX

LI

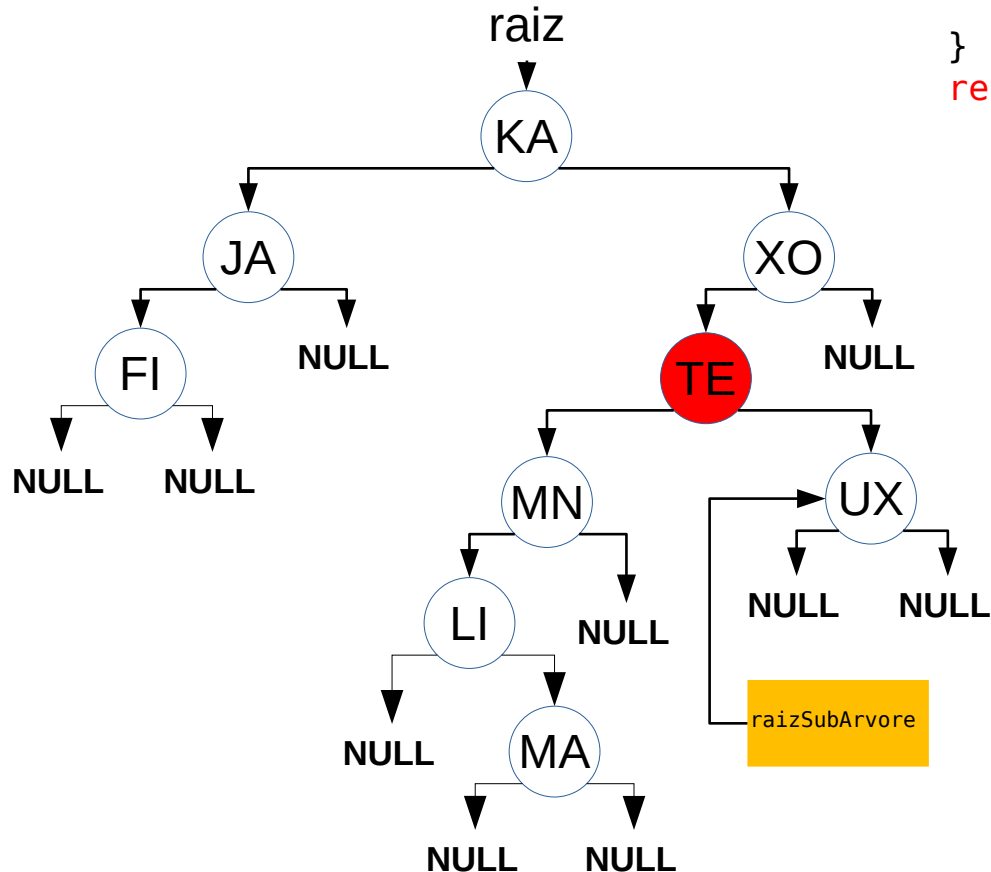
MA

raizSubArvore

minimoAux(raizSubArvore):

```
enquanto (raizSubArvore.esquerdo ≠ NULO) {  
    raizSubArvore ← raizSubArvore.esquerdo;  
}
```

retornar raizSubArvore;

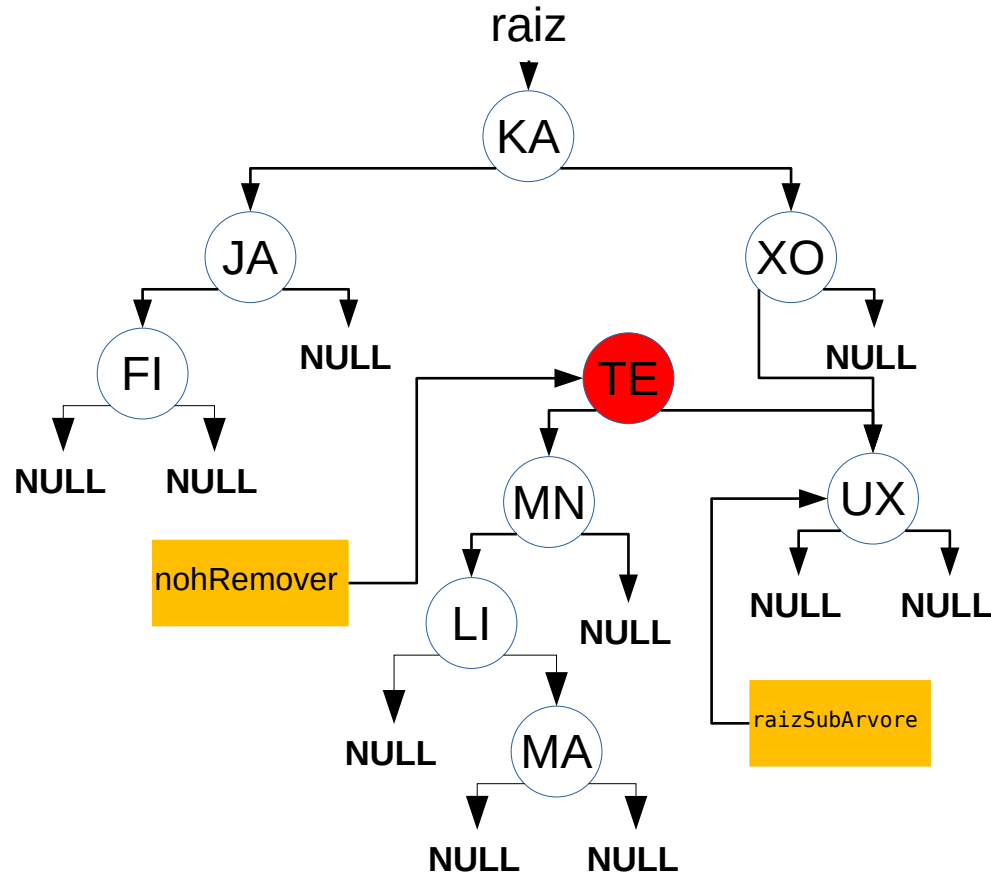


Remover

PL

CA

TE



transplanta(antigo, novo):

```
se (raiz = antigo) {  
    raiz ← novo;  
}  
senão se (antigo = antigo.pai.esquerdo) {  
    antigo.pai.esquerdo ← novo;  
}  
senão { // antigo = antigo.pai.direito  
    antigo.pai.direito ← novo;  
}  
se (novo ≠ NULO) {  
    novo.pai ← antigo.pai;  
}
```

Remover

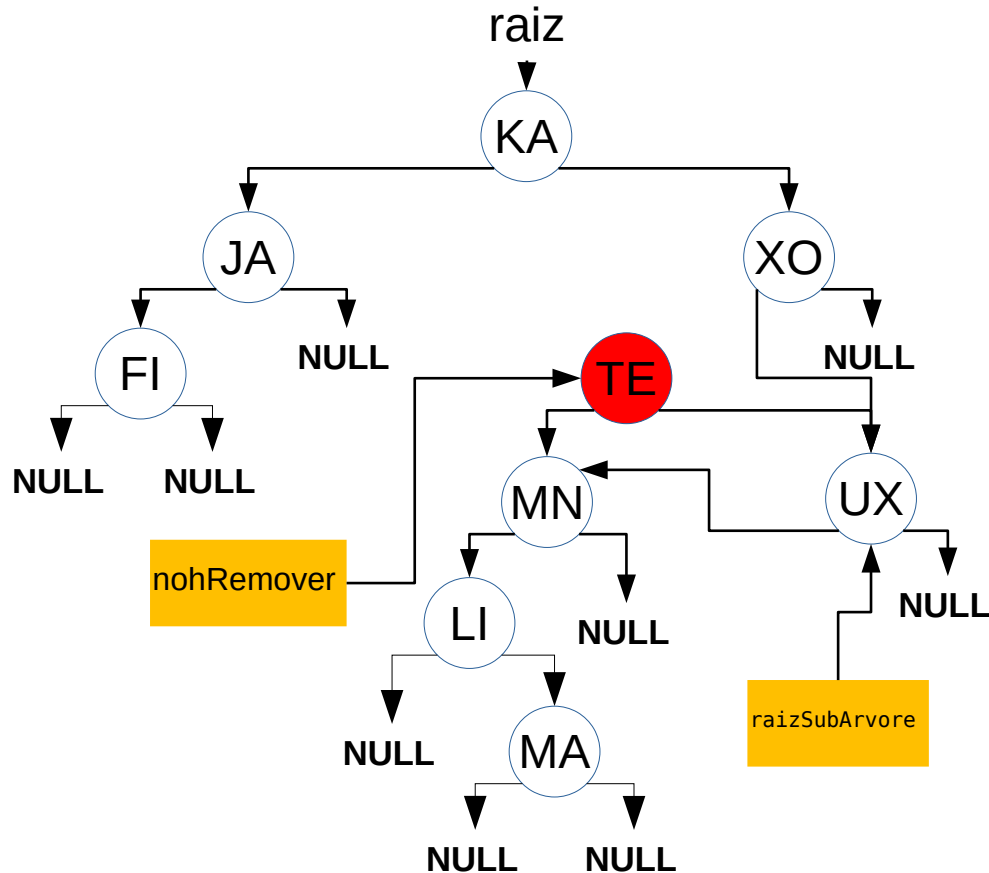
PL

CA

TE

```
...sucessor.esquerdo ← nohRemover.esquerdo;
```

```
Sucessor.esquerdo.pai ← sucessor;
```



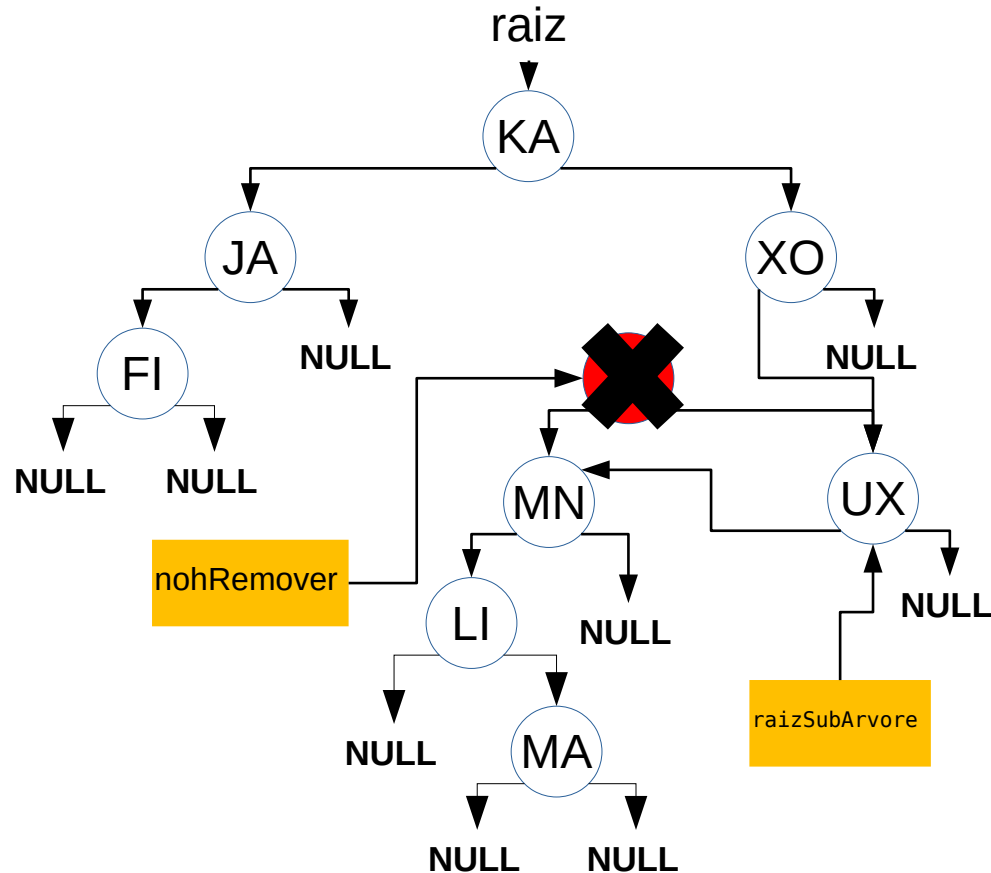
Remover

PL

CA

TE

remove(unValor):
apagar(nohRemover);



Resultado:

remove(umValor):
apagar(nohRemover);

