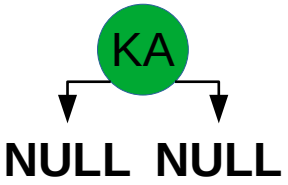
Questão 5 – ABB

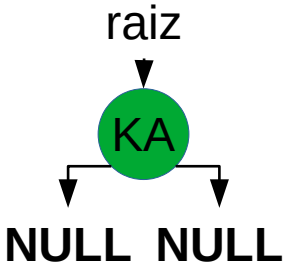
criarABB():
```
raiz ← NULL;
```

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;
  }
}
```

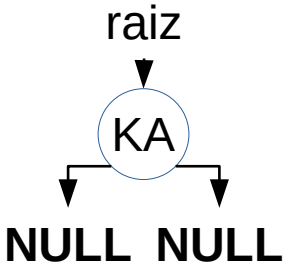
Inserir: KA JA XO CA TE UX PL MN FI LI MA

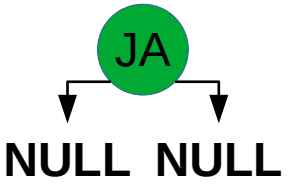


inserirIterativamente(umValor):

```
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

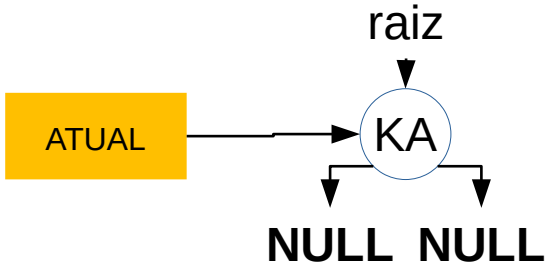


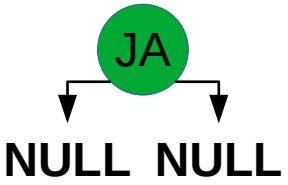


inserirIterativamente(umValor):

```
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;
  }
}
```

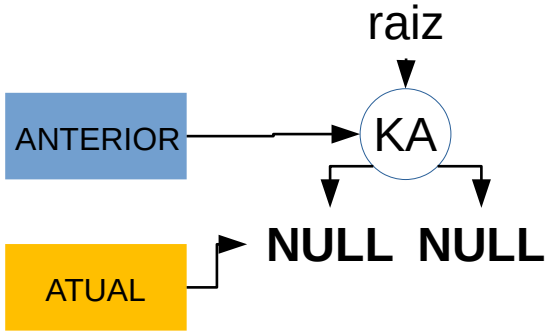
Inserir: KA JA XO CA TE UX PL MN FI LI MA

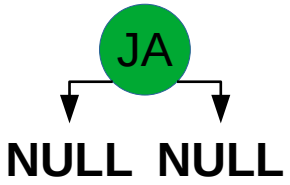





```
inserirIterativamente(umValor):
novo ← criar_noh(umValor);
se (raiz = NULL) {
  raiz ← novo;
}
senão {
  atual ← raiz;
  enquanto (atual ≠ NULL) {
    anterior ← atual;
    se (atual.valor > umValor) {
      atual ← atual.esquerdo;
    } senão {
      atual ← atual.direito;
    }
  }
novo.pai ← anterior;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

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

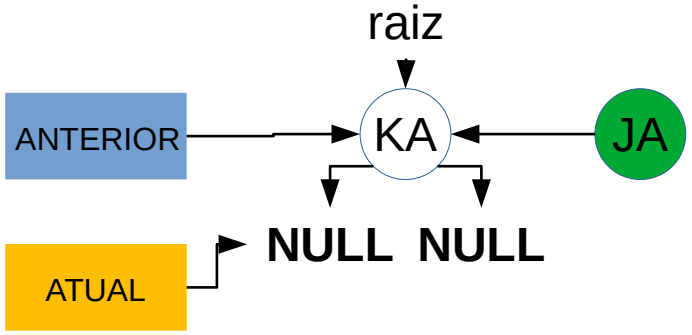




inserirIterativamente(umValor):

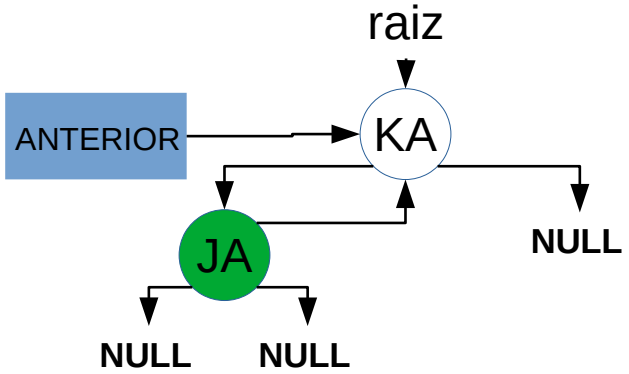
```
novo ← criar_noh(umValor);
se (raiz = NULL) {
  raiz ← novo;
}
senão {
  atual ← raiz;
  enquanto (atual ≠ NULL) {
    anterior ← atual;
    se (atual.valor > umValor) {
      atual ← atual.esquerdo;
    } senão {
      atual ← atual.direito;
    }
  }
novo.pai ← anterior;
  se (anterior.valor >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

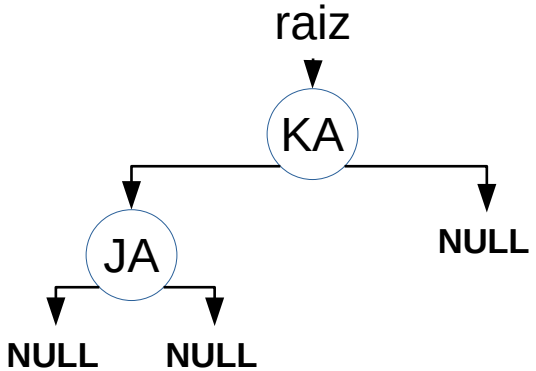
Inserir: KA JA XO CA TE UX PL MN FI LI MA

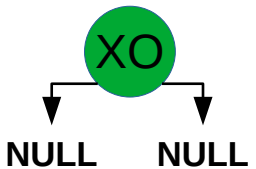


inserirIterativamente(umValor):

```
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;
  }
}
```

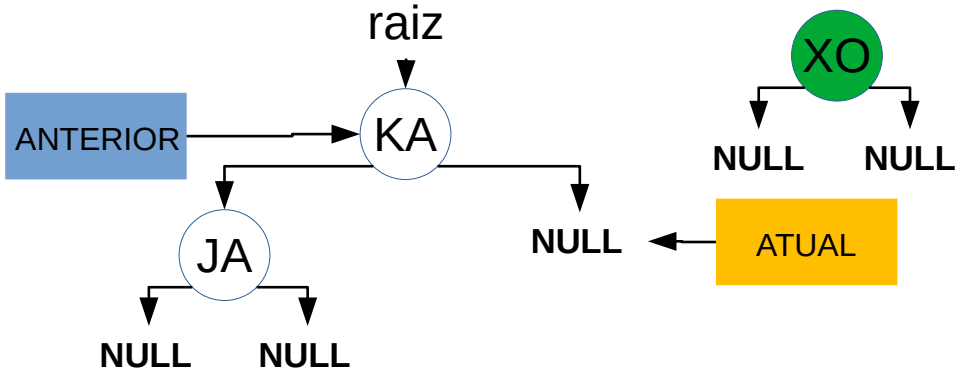
Inserir: KA JA XO CA TE UX PL MN FI LI MA





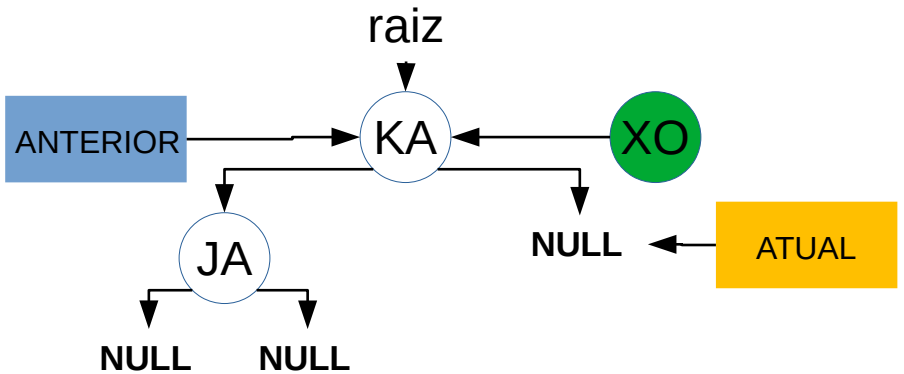

```
inserirIterativamente(umValor):
novo ← criar_noh(umValor);
se (raiz = NULL) {
  raiz ← novo;
}
senão {
  atual ← raiz;
  enquanto (atual ≠ NULL) {
    anterior ← atual;
    se (atual.valor > umValor) {
      atual ← atual.esquerdo;
    } senão {
      atual ← atual.direito;
    }
  }
novo.pai ← anterior;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

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



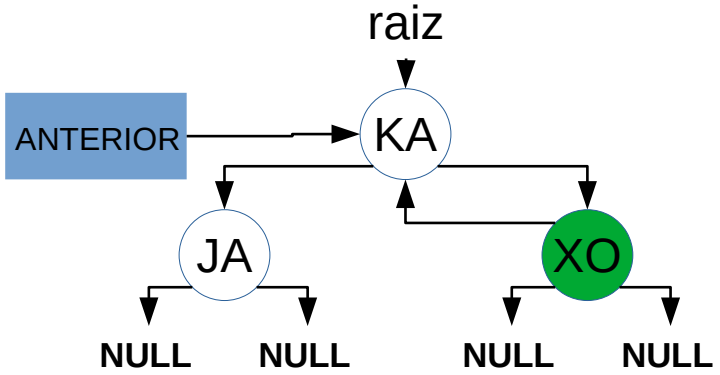

```
inserirIterativamente(umValor):
novo ← criar_noh(umValor);
se (raiz = NULL) {
  raiz ← novo;
}
senão {
  atual ← raiz;
  enquanto (atual ≠ NULL) {
    anterior ← atual;
    se (atual.valor > umValor) {
      atual ← atual.esquerdo;
    } senão {
      atual ← atual.direito;
    }
  }
novo.pai ← anterior;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

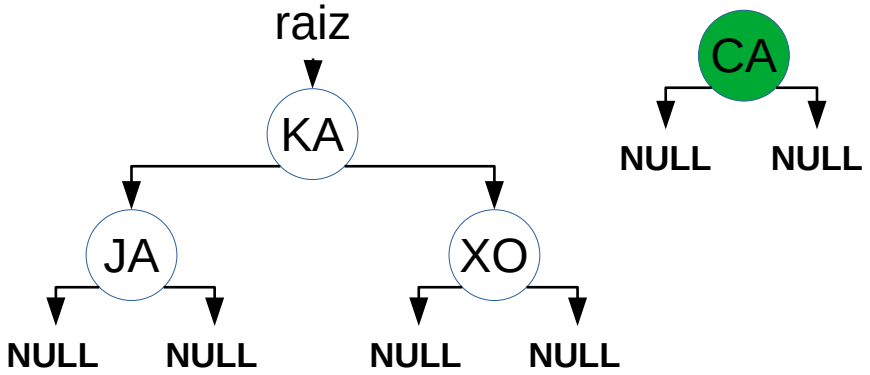
Inserir: KA JA XO CA TE UX PL MN FI LI MA



inserirIterativamente(umValor):

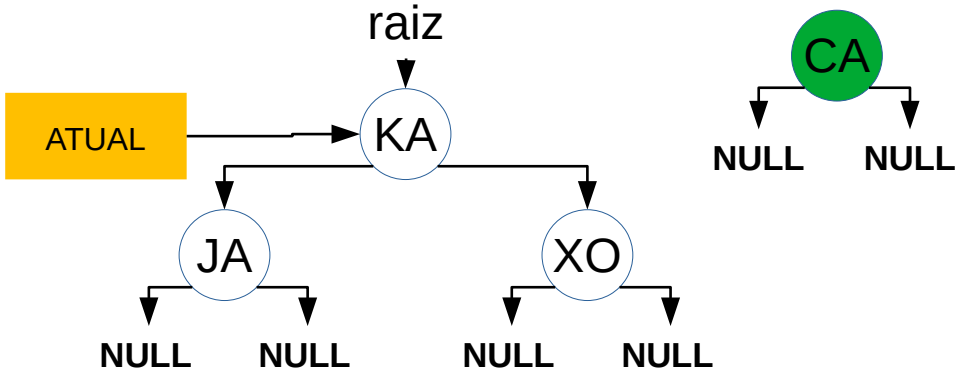
```
novo ← criar_noh(umValor);
se (raiz = NULL) {
  raiz ← novo;
}
senão {
  atual ← raiz;
  enquanto (atual ≠ NULL) {
    anterior ← atual;
    se (atual.valor > umValor) {
      atual ← atual.esquerdo;
    } senão {
      atual ← atual.direito;
    }
  }
novo.pai ← anterior;
  se (anterior.valor >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

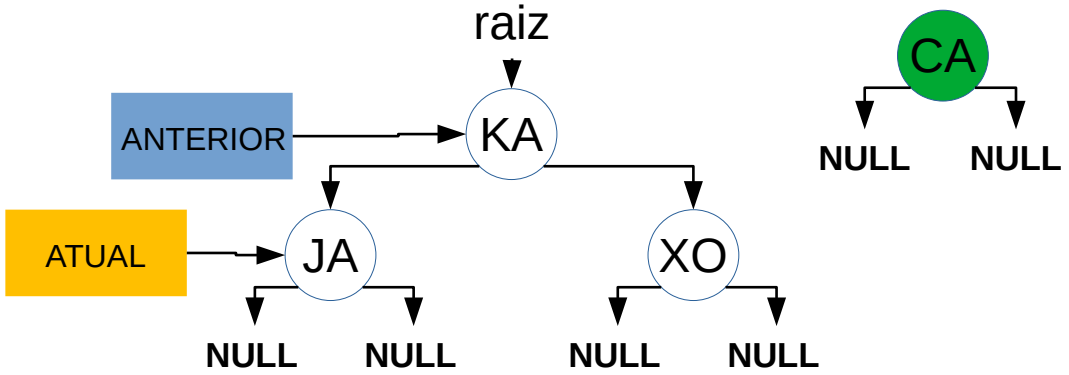
Inserir: KA JA XO CA TE UX PL MN FI LI MA



inserirIterativamente(umValor):

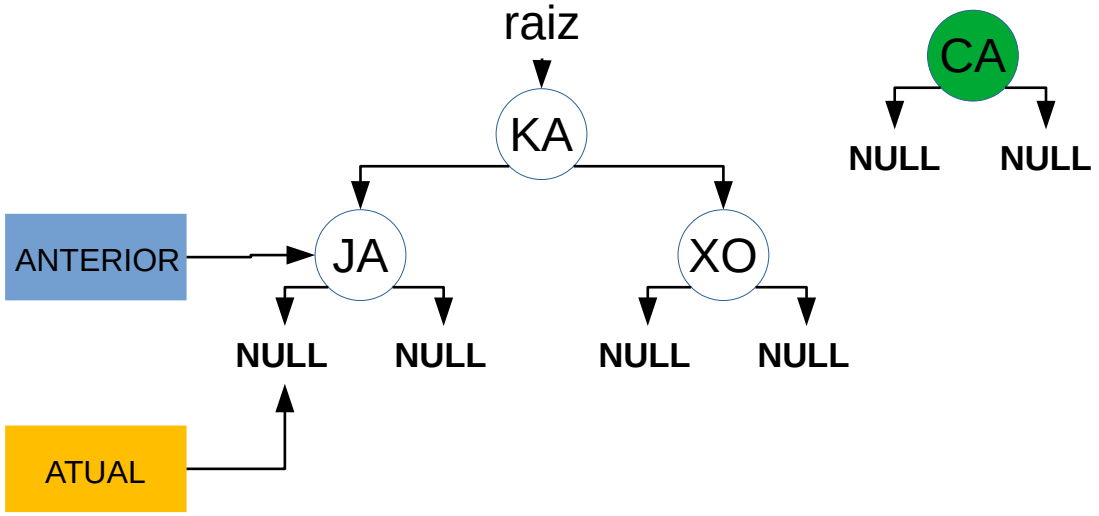
```
novo ← criar_noh(umValor);
se (raiz = NULL) {
  raiz ← novo;
}
senão {
  atual ← raiz;
  enquanto (atual ≠ NULL) {
    anterior ← atual;
    se (atual.valor > umValor) {
      atual ← atual.esquerdo;
    } senão {
      atual ← atual.direito;
    }
  }
novo.pai ← anterior;
  se (anterior.valor >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

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



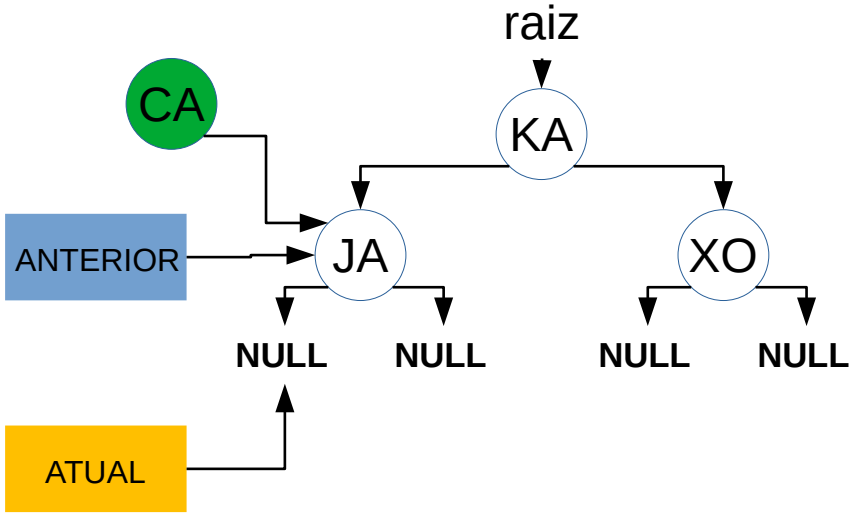

```
inserirIterativamente(umValor):
novo ← criar_noh(umValor);
se (raiz = NULL) {
  raiz ← novo;
}
senão {
  atual ← raiz;
  enquanto (atual ≠ NULL) {
    anterior ← atual;
    se (atual.valor > umValor) {
      atual ← atual.esquerdo;
    } senão {
      atual ← atual.direito;
    }
  }
novo.pai ← anterior;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

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



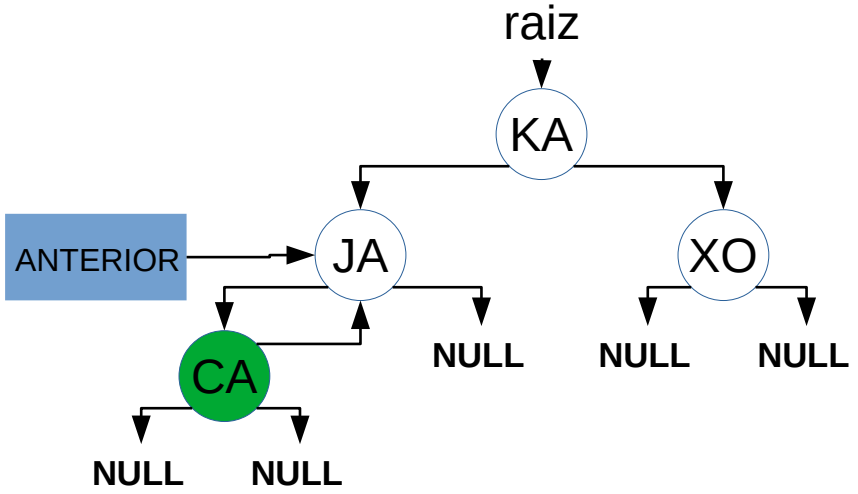

```
inserirIterativamente(umValor):
novo ← criar_noh(umValor);
se (raiz = NULL) {
  raiz ← novo;
}
senão {
  atual ← raiz;
  enquanto (atual ≠ NULL) {
    anterior ← atual;
    se (atual.valor > umValor) {
      atual ← atual.esquerdo;
    } senão {
      atual ← atual.direito;
    }
  }
novo.pai ← anterior;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

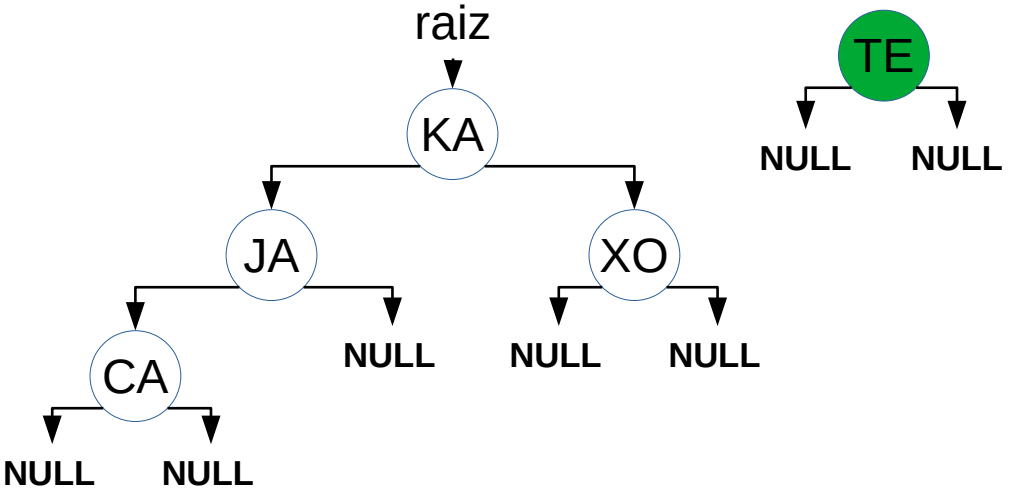
Inserir: KA JA XO CA TE UX PL MN FI LI MA



inserirIterativamente(umValor):

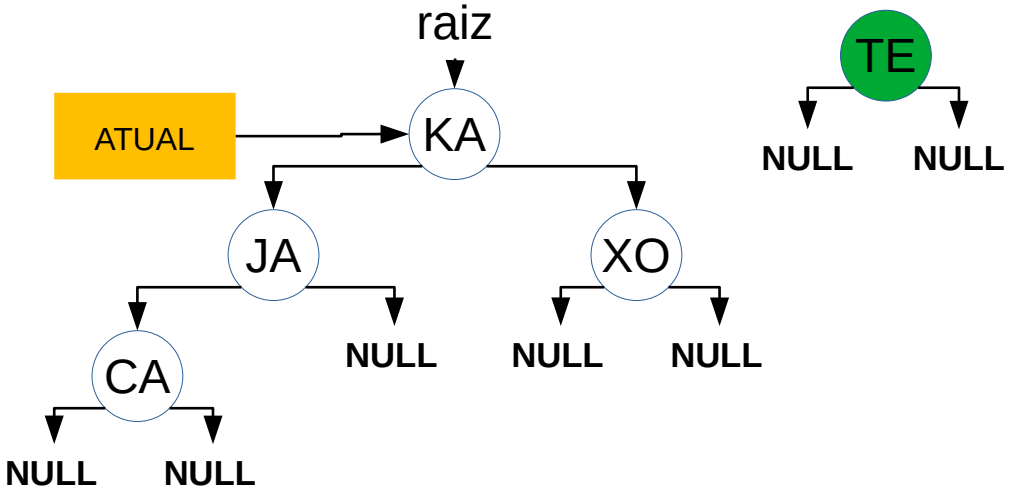
```
novo ← criar_noh(umValor);
se (raiz = NULL) {
  raiz ← novo;
}
senão {
  atual ← raiz;
  enquanto (atual ≠ NULL) {
    anterior ← atual;
    se (atual.valor > umValor) {
      atual ← atual.esquerdo;
    } senão {
      atual ← atual.direito;
    }
  }
novo.pai ← anterior;
  se (anterior.valor >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

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



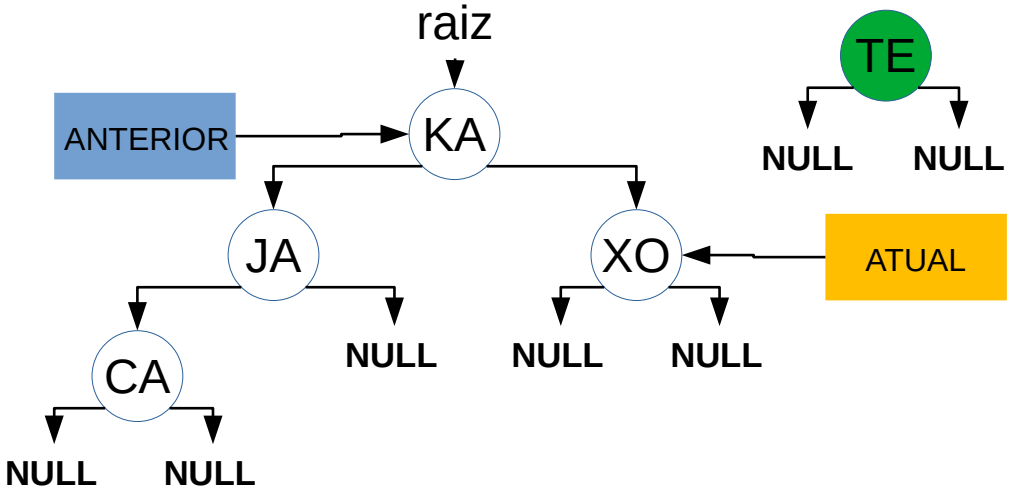

```
inserirIterativamente(umValor):
novo ← criar_noh(umValor);
se (raiz = NULL) {
  raiz ← novo;
}
senão {
  atual ← raiz;
  enquanto (atual ≠ NULL) {
    anterior ← atual;
    se (atual.valor > umValor) {
      atual ← atual.esquerdo;
    } senão {
      atual ← atual.direito;
    }
  }
novo.pai ← anterior;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

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



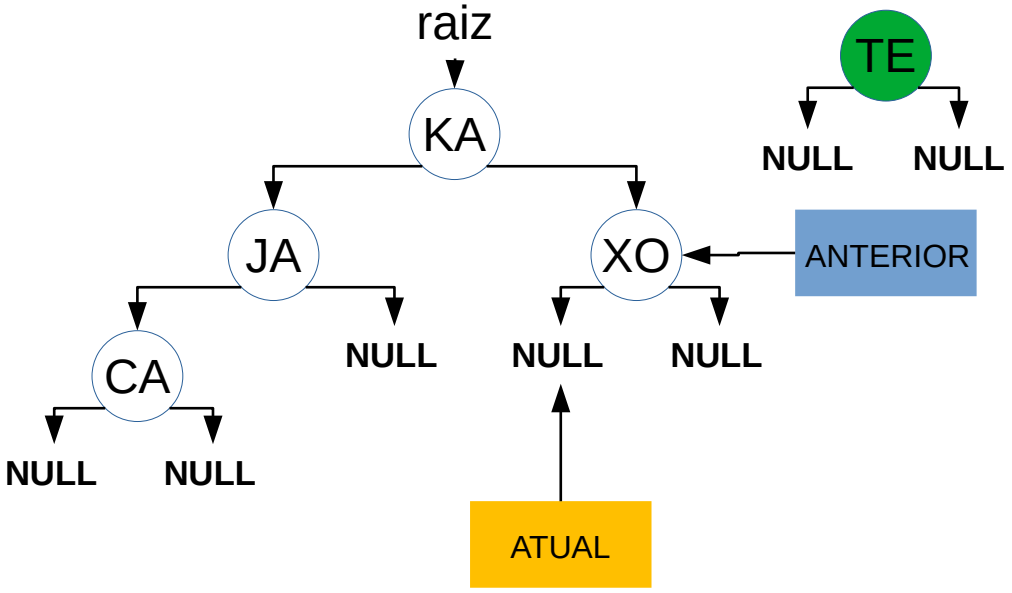

```
inserirIterativamente(umValor):
novo ← criar_noh(umValor);
se (raiz = NULL) {
  raiz ← novo;
}
senão {
  atual ← raiz;
  enquanto (atual ≠ NULL) {
    anterior ← atual;
    se (atual.valor > umValor) {
      atual ← atual.esquerdo;
    } senão {
      atual ← atual.direito;
    }
  }
novo.pai ← anterior;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

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



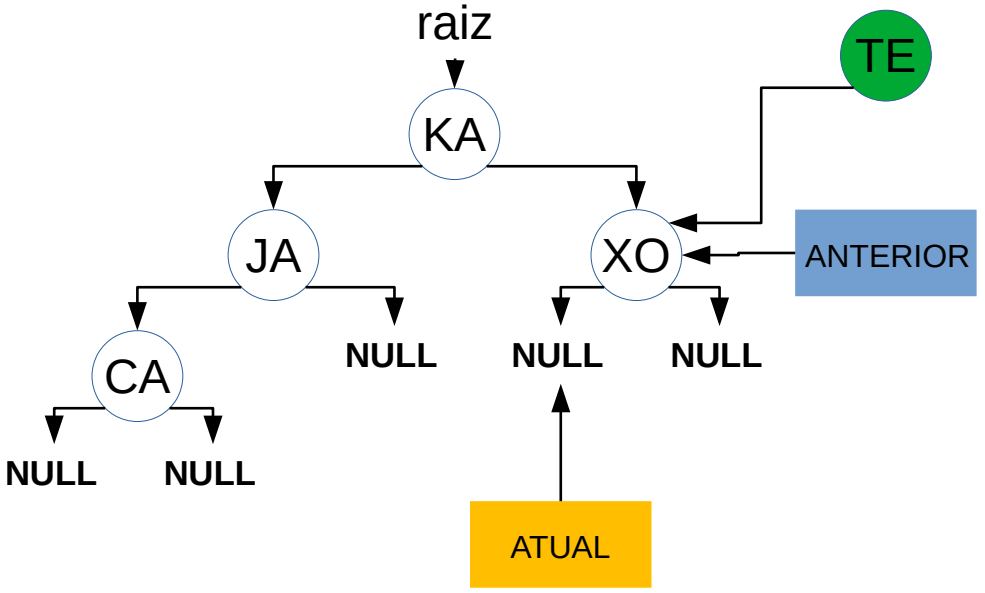

```
inserirIterativamente(umValor):
novo ← criar_noh(umValor);
se (raiz = NULL) {
  raiz ← novo;
}
senão {
  atual ← raiz;
  enquanto (atual ≠ NULL) {
    anterior ← atual;
    se (atual.valor > umValor) {
      atual ← atual.esquerdo;
    } senão {
      atual ← atual.direito;
    }
  }
novo.pai ← anterior;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

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



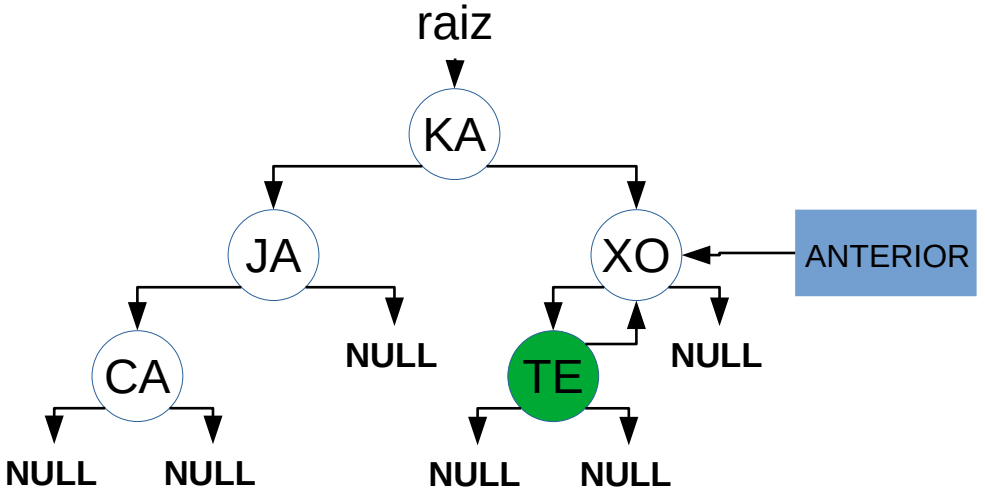

```
inserirIterativamente(umValor):
novo ← criar_noh(umValor);
se (raiz = NULL) {
  raiz ← novo;
}
senão {
  atual ← raiz;
  enquanto (atual ≠ NULL) {
    anterior ← atual;
    se (atual.valor > umValor) {
      atual ← atual.esquerdo;
    } senão {
      atual ← atual.direito;
    }
  }
novo.pai ← anterior;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

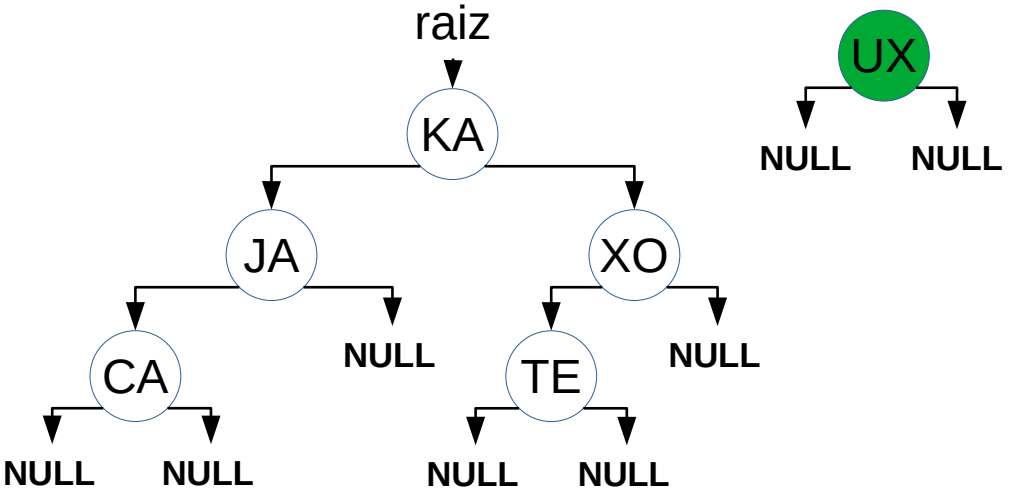
Inserir: KA JA XO CA TE UX PL MN FI LI MA



inserirIterativamente(umValor):

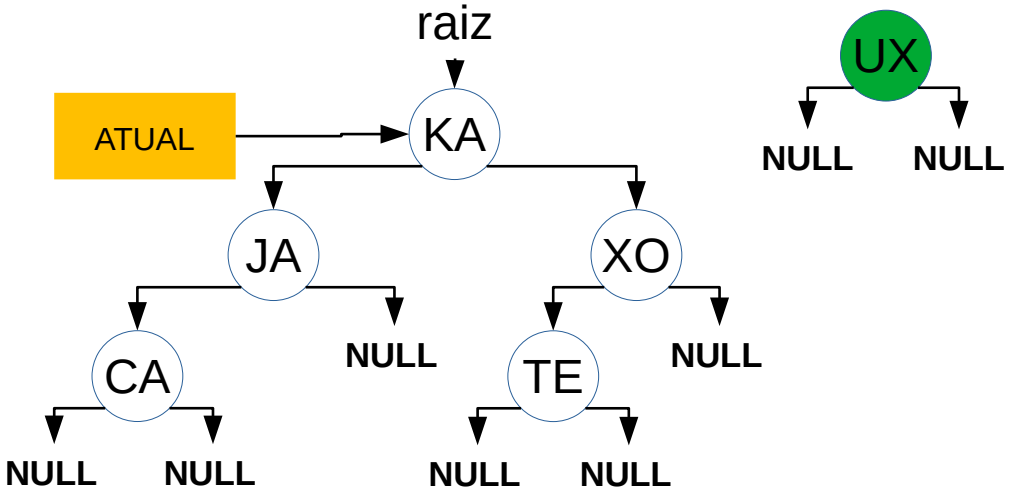
```
novo ← criar_noh(umValor);
se (raiz = NULL) {
  raiz ← novo;
}
senão {
  atual ← raiz;
  enquanto (atual ≠ NULL) {
    anterior ← atual;
    se (atual.valor > umValor) {
      atual ← atual.esquerdo;
    } senão {
      atual ← atual.direito;
    }
  }
novo.pai ← anterior;
  se (anterior.valor >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

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



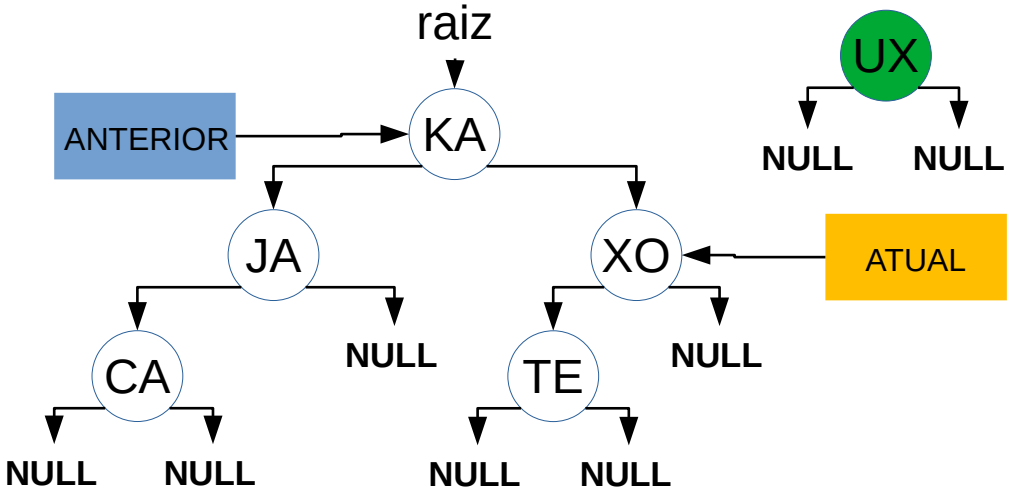

```
inserirIterativamente(umValor):
novo ← criar_noh(umValor);
se (raiz = NULL) {
  raiz ← novo;
}
senão {
  atual ← raiz;
  enquanto (atual ≠ NULL) {
    anterior ← atual;
    se (atual.valor > umValor) {
      atual ← atual.esquerdo;
    } senão {
      atual ← atual.direito;
    }
  }
novo.pai ← anterior;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

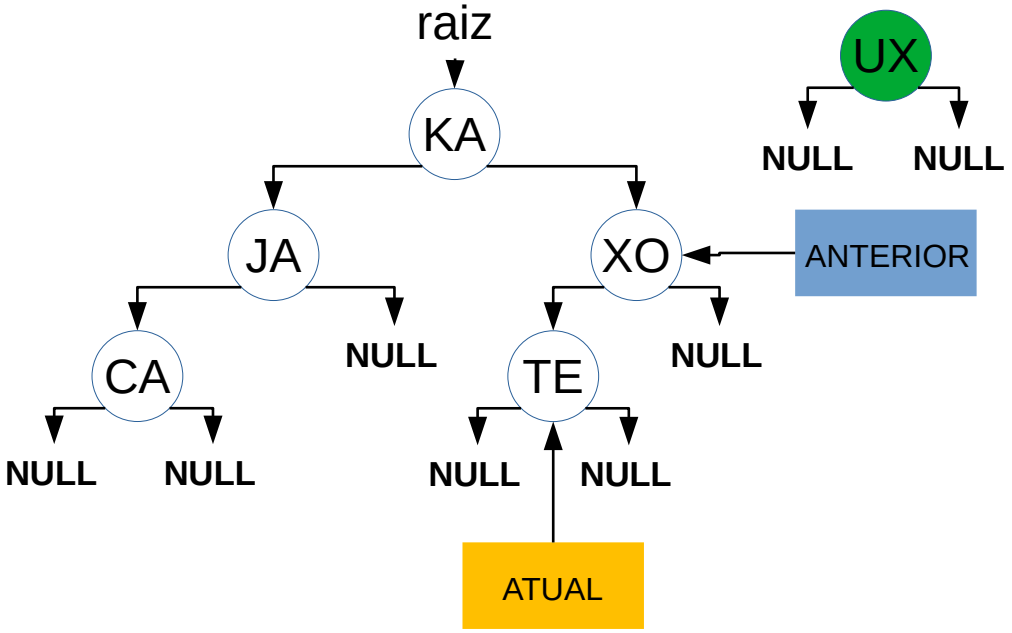
Inserir: KA JA XO CA TE UX PL MN FI LI MA



inserirIterativamente(umValor):

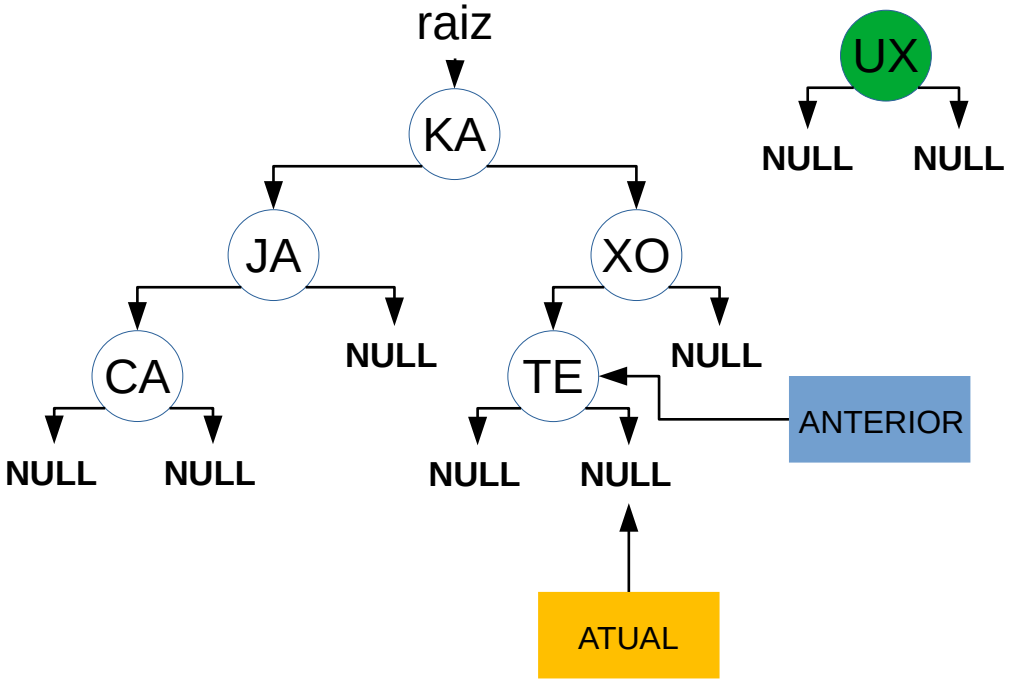
```
novo ← criar_noh(umValor);
se (raiz = NULL) {
  raiz ← novo;
}
senão {
  atual ← raiz;
  enquanto (atual ≠ NULL) {
    anterior ← atual;
    se (atual.valor > umValor) {
      atual ← atual.esquerdo;
    } senão {
      atual ← atual.direito;
    }
  }
novo.pai ← anterior;
  se (anterior.valor >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

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



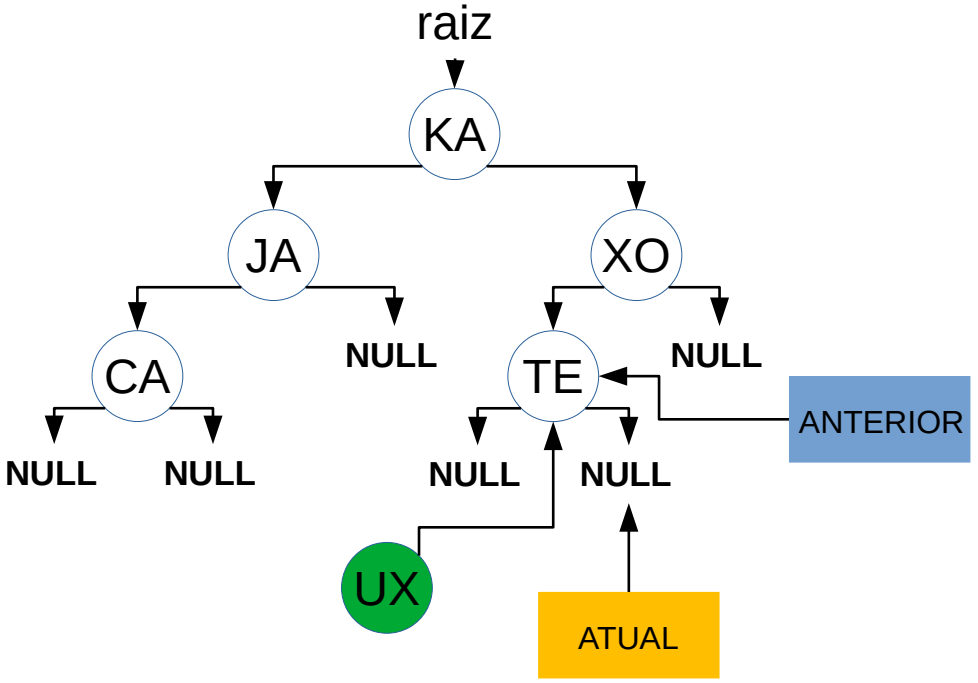

```
inserirIterativamente(umValor):
novo ← criar_noh(umValor);
se (raiz = NULL) {
  raiz ← novo;
}
senão {
  atual ← raiz;
  enquanto (atual ≠ NULL) {
    anterior ← atual;
    se (atual.valor > umValor) {
      atual ← atual.esquerdo;
    } senão {
      atual ← atual.direito;
    }
  }
novo.pai ← anterior;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

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



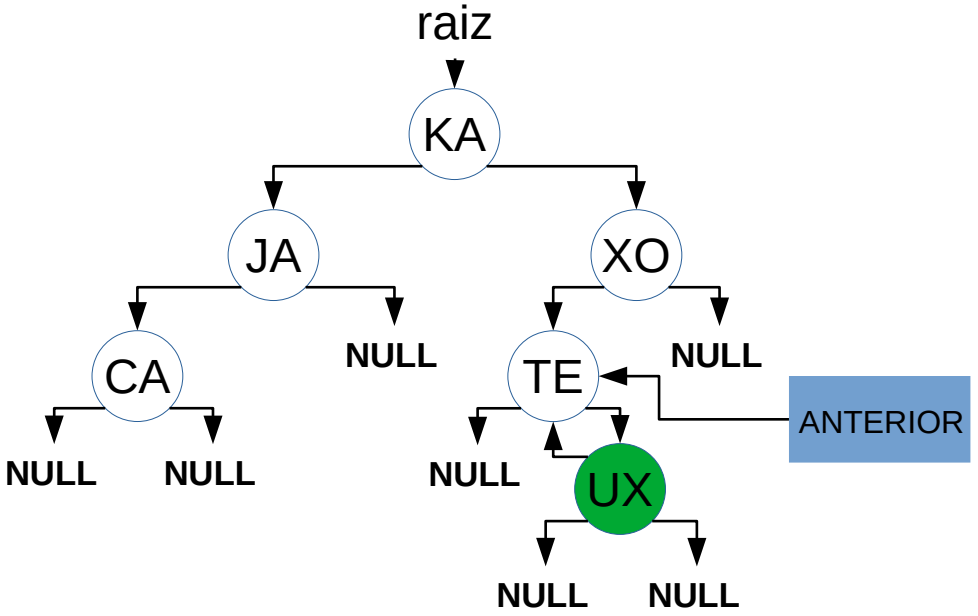

```
inserirIterativamente(umValor):
novo ← criar_noh(umValor);
se (raiz = NULL) {
  raiz ← novo;
}
senão {
  atual ← raiz;
  enquanto (atual ≠ NULL) {
    anterior ← atual;
    se (atual.valor > umValor) {
      atual ← atual.esquerdo;
    } senão {
      atual ← atual.direito;
    }
  }
novo.pai ← anterior;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

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



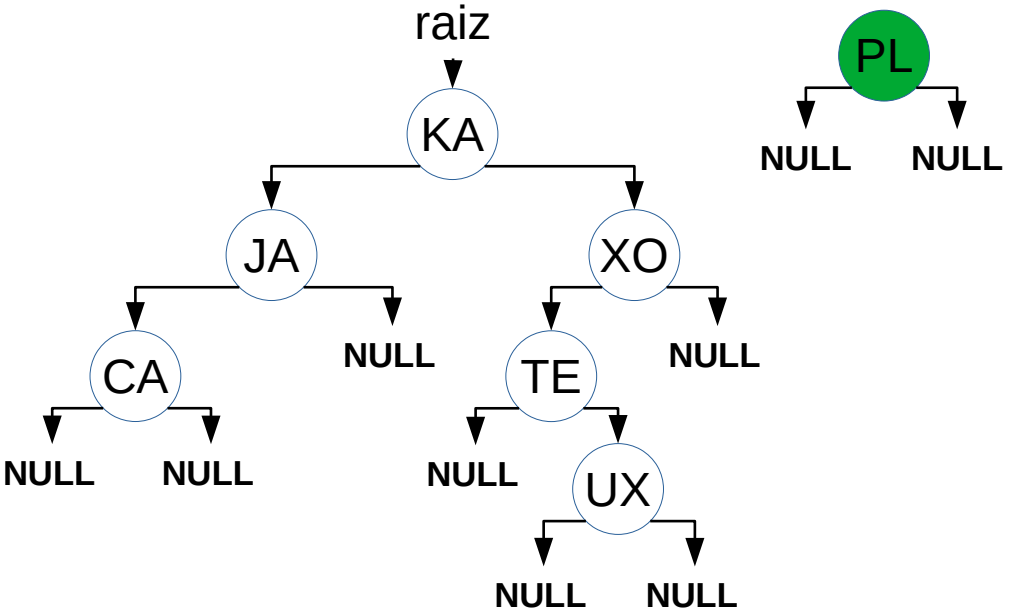

```
inserirIterativamente(umValor):
novo ← criar_noh(umValor);
se (raiz = NULL) {
  raiz ← novo;
}
senão {
  atual ← raiz;
  enquanto (atual ≠ NULL) {
    anterior ← atual;
    se (atual.valor > umValor) {
      atual ← atual.esquerdo;
    } senão {
      atual ← atual.direito;
    }
  }
novo.pai ← anterior;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

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



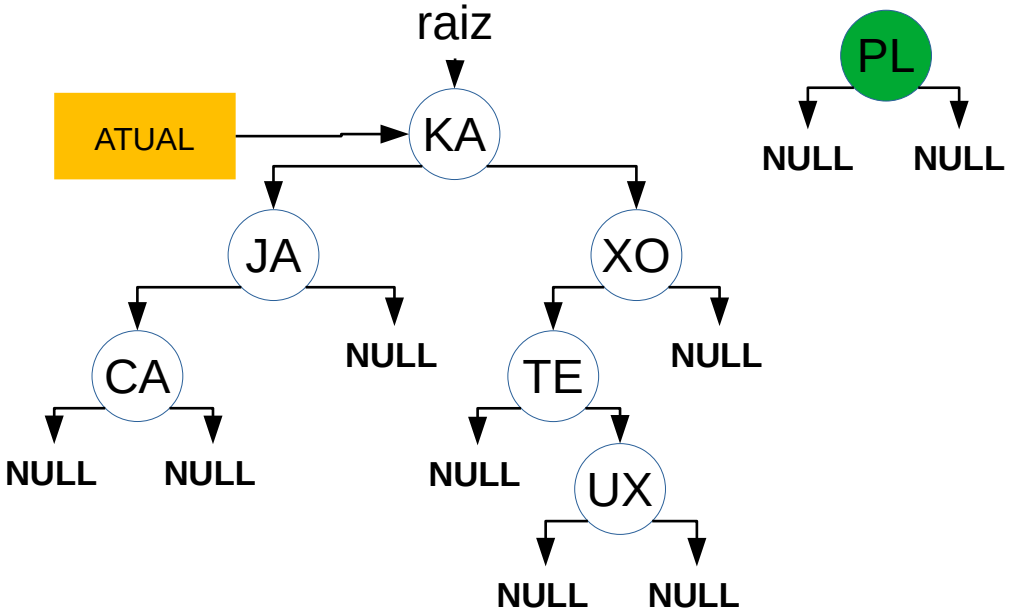

```
inserirIterativamente(umValor):
novo ← criar_noh(umValor);
se (raiz = NULL) {
  raiz ← novo;
}
senão {
  atual ← raiz;
  enquanto (atual ≠ NULL) {
    anterior ← atual;
    se (atual.valor > umValor) {
      atual ← atual.esquerdo;
    } senão {
      atual ← atual.direito;
    }
  }
novo.pai ← anterior;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

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



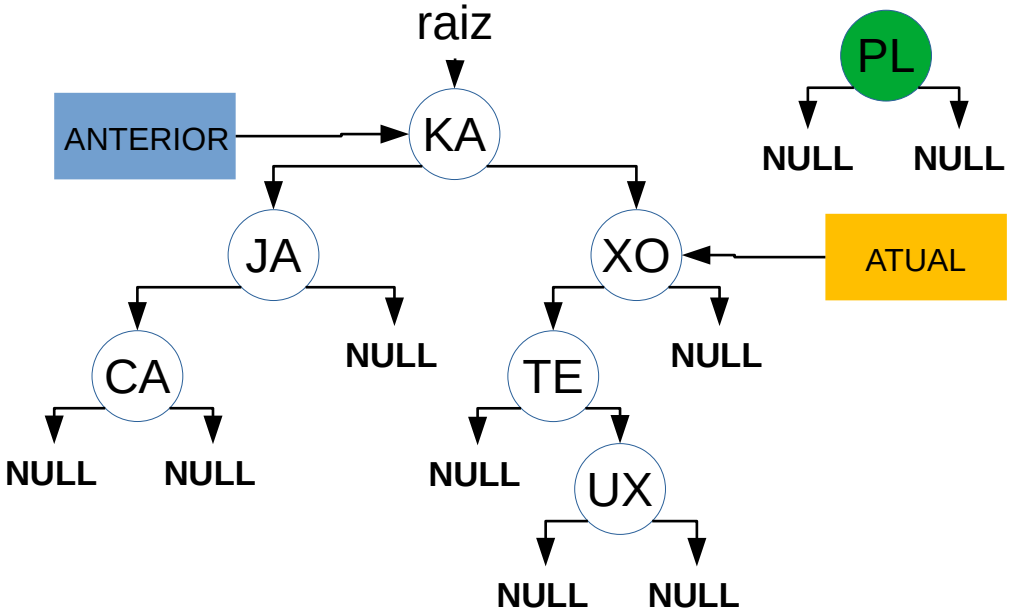

```
inserirIterativamente(umValor):
novo ← criar_noh(umValor);
se (raiz = NULL) {
  raiz ← novo;
}
senão {
  atual ← raiz;
  enquanto (atual ≠ NULL) {
    anterior ← atual;
    se (atual.valor > umValor) {
      atual ← atual.esquerdo;
    } senão {
      atual ← atual.direito;
    }
  }
novo.pai ← anterior;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

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



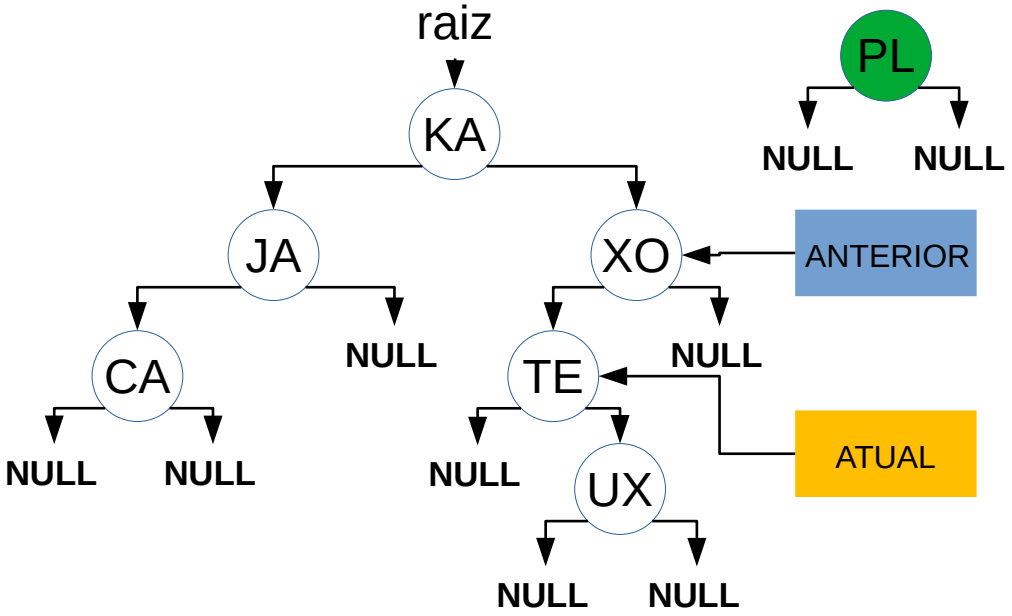

```
inserirIterativamente(umValor):
novo ← criar_noh(umValor);
se (raiz = NULL) {
  raiz ← novo;
}
senão {
  atual ← raiz;
  enquanto (atual ≠ NULL) {
    anterior ← atual;
    se (atual.valor > umValor) {
      atual ← atual.esquerdo;
    } senão {
      atual ← atual.direito;
    }
  }
novo.pai ← anterior;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

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



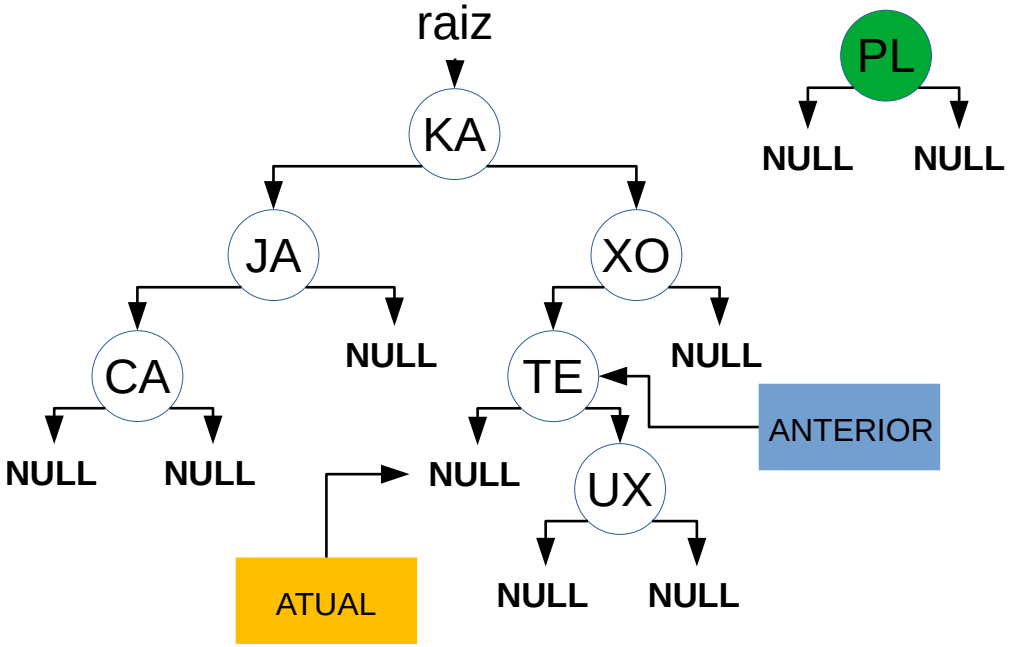

```
inserirIterativamente(umValor):
novo ← criar_noh(umValor);
se (raiz = NULL) {
  raiz ← novo;
}
senão {
  atual ← raiz;
  enquanto (atual ≠ NULL) {
    anterior ← atual;
    se (atual.valor > umValor) {
      atual ← atual.esquerdo;
    } senão {
      atual ← atual.direito;
    }
  }
novo.pai ← anterior;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

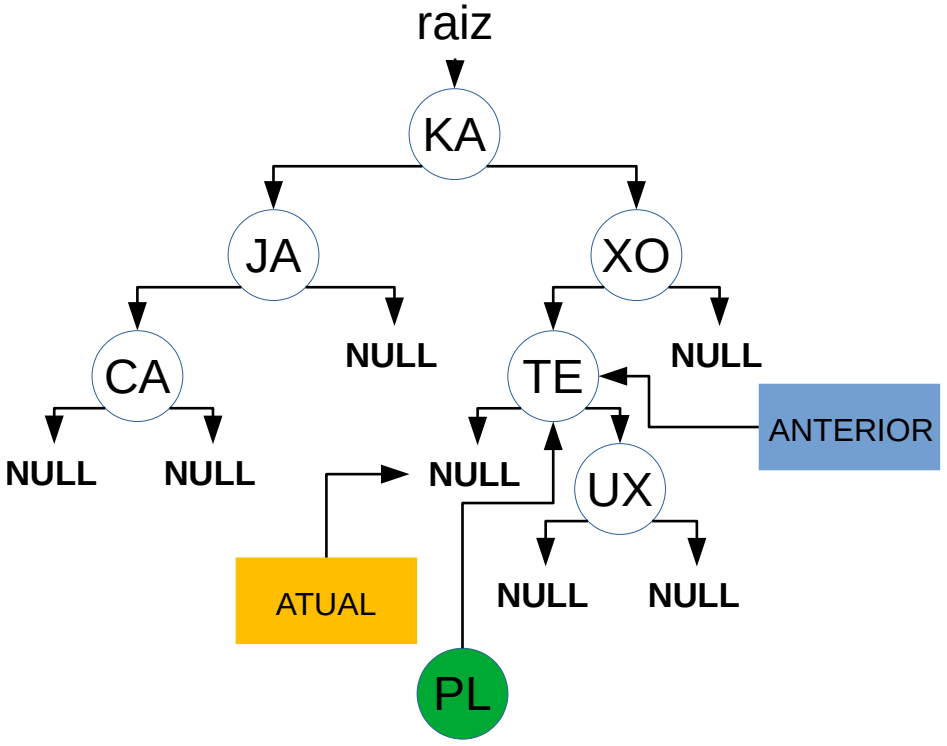
Inserir: KA JA XO CA TE UX PL MN FI LI MA



inserirIterativamente(umValor):

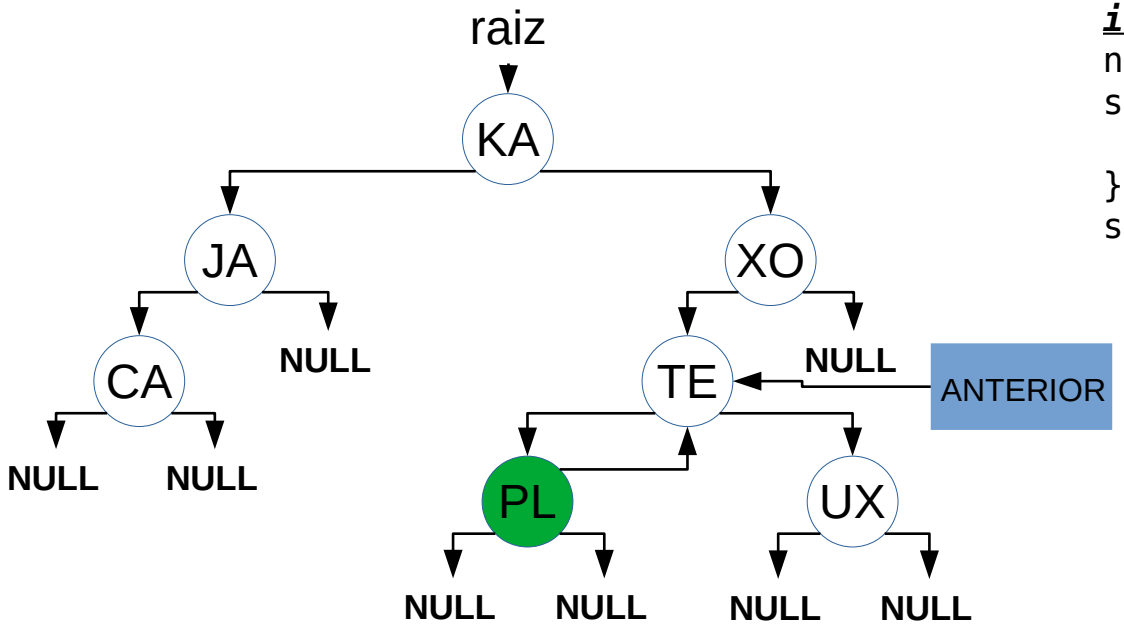
```
novo ← criar_noh(umValor);
se (raiz = NULL) {
  raiz ← novo;
}
senão {
  atual ← raiz;
  enquanto (atual ≠ NULL) {
    anterior ← atual;
    se (atual.valor > umValor) {
      atual ← atual.esquerdo;
    } senão {
      atual ← atual.direito;
    }
  }
novo.pai ← anterior;
  se (anterior.valor >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

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



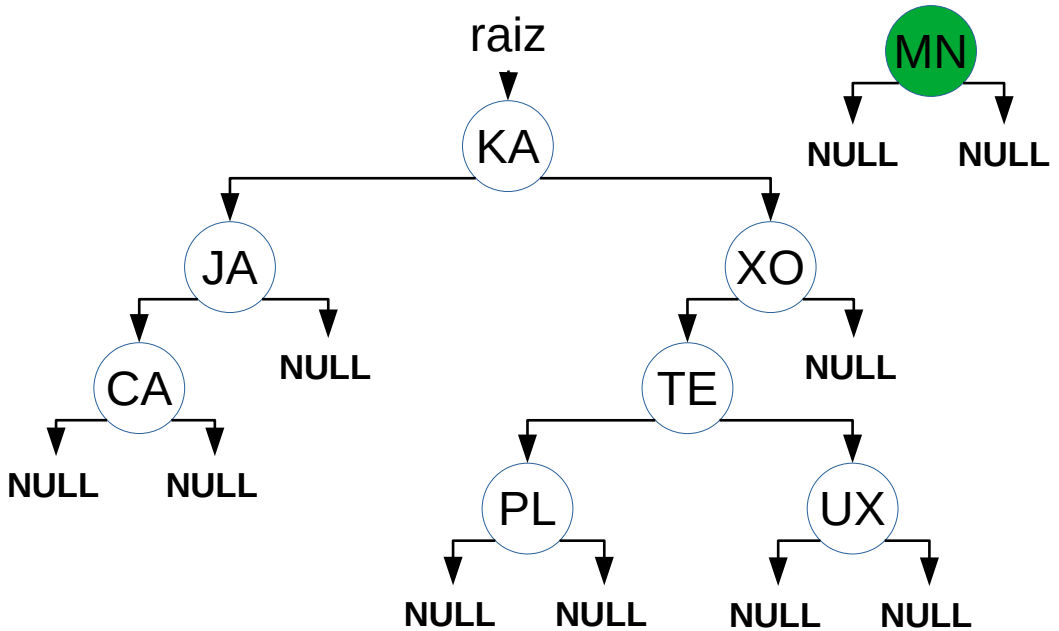

```
inserirIterativamente(umValor):
novo ← criar_noh(umValor);
se (raiz = NULL) {
  raiz ← novo;
}
senão {
  atual ← raiz;
  enquanto (atual ≠ NULL) {
    anterior ← atual;
    se (atual.valor > umValor) {
      atual ← atual.esquerdo;
    } senão {
      atual ← atual.direito;
    }
  }
novo.pai ← anterior;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

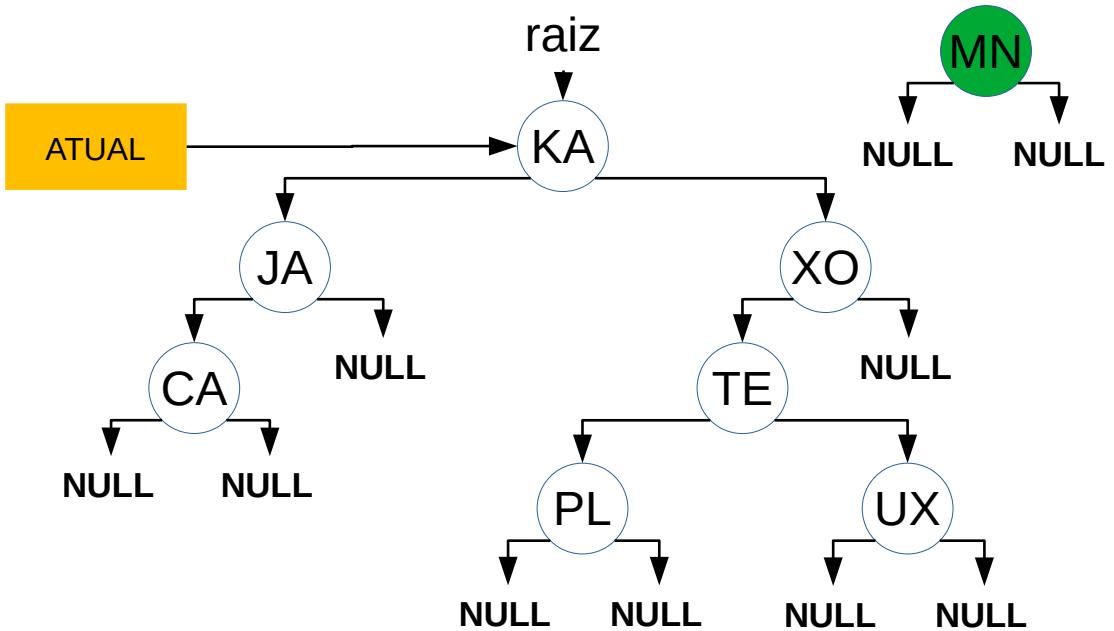
Inserir: KA JA XO CA TE UX PL MN FI LI MA



inserirIterativamente(umValor):

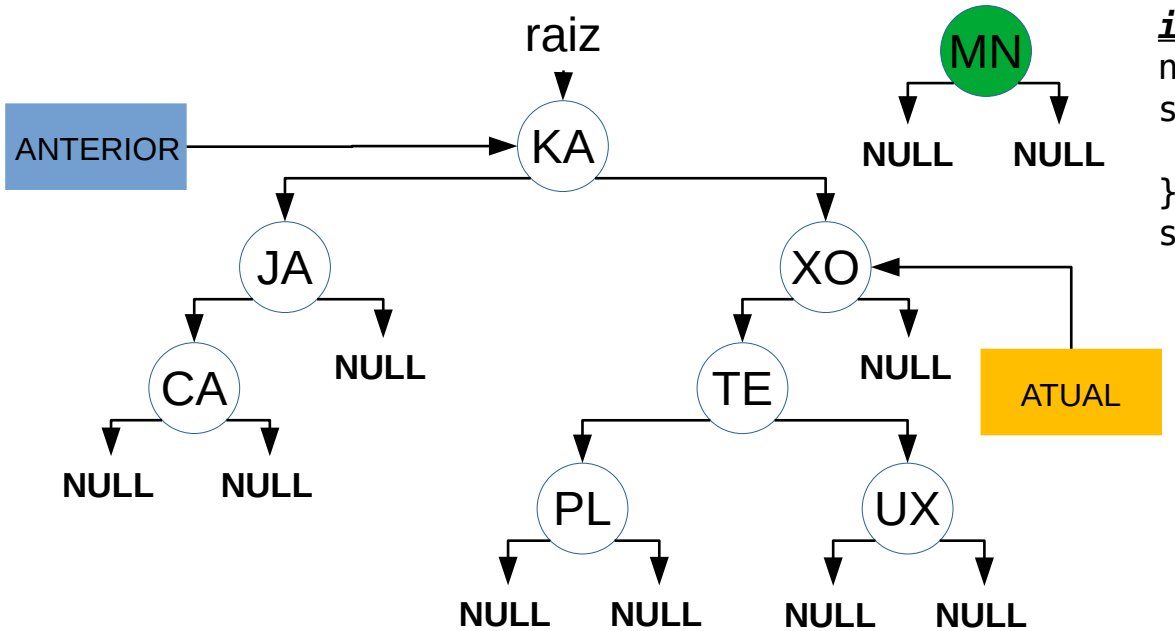
```
novo ← criar_noh(umValor);
se (raiz = NULL) {
  raiz ← novo;
}
senão {
  atual ← raiz;
  enquanto (atual ≠ NULL) {
    anterior ← atual;
    se (atual.valor > umValor) {
      atual ← atual.esquerdo;
    } senão {
      atual ← atual.direito;
    }
  }
novo.pai ← anterior;
  se (anterior.valor >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

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



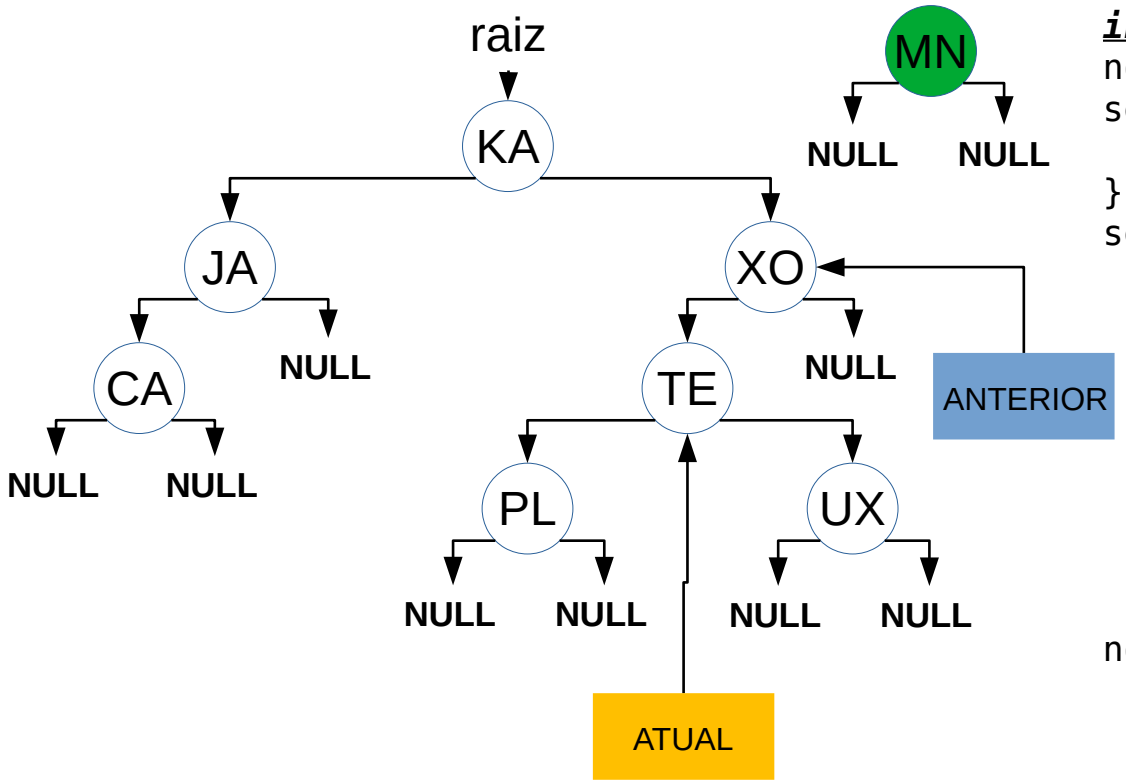

```
inserirIterativamente(umValor):
novo ← criar_noh(umValor);
se (raiz = NULL) {
  raiz ← novo;
}
senão {
  atual ← raiz;
  enquanto (atual ≠ NULL) {
    anterior ← atual;
    se (atual.valor > umValor) {
      atual ← atual.esquerdo;
    } senão {
      atual ← atual.direito;
    }
  }
novo.pai ← anterior;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

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



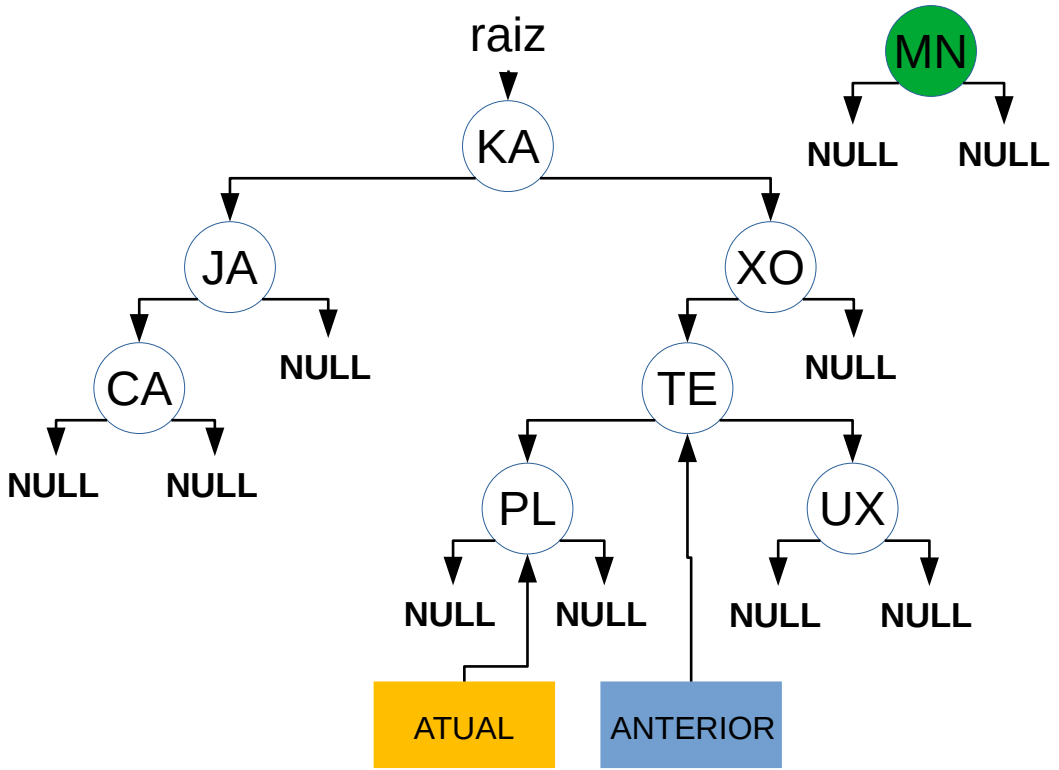

```
inserirIterativamente(umValor):
novo ← criar_noh(umValor);
se (raiz = NULL) {
  raiz ← novo;
}
senão {
  atual ← raiz;
  enquanto (atual ≠ NULL) {
    anterior ← atual;
    se (atual.valor > umValor) {
      atual ← atual.esquerdo;
    } senão {
      atual ← atual.direito;
    }
  }
novo.pai ← anterior;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

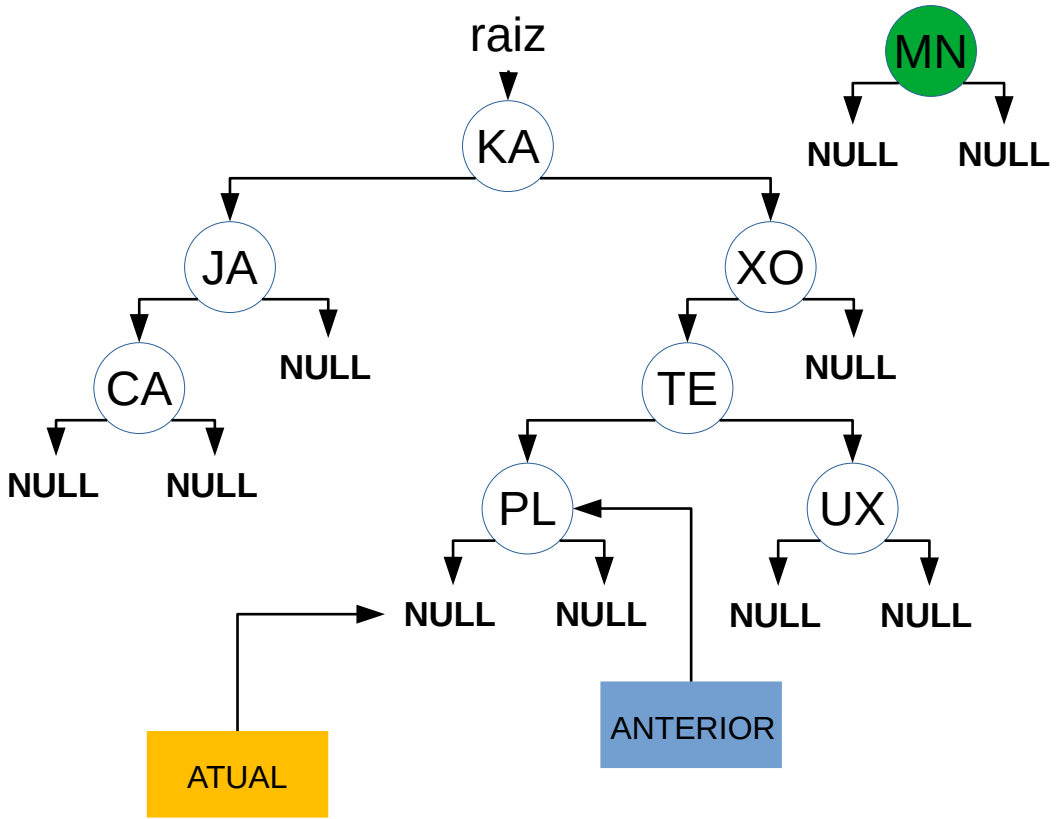
Inserir: KA JA XO CA TE UX PL MN FI LI MA



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;
  }
}
```

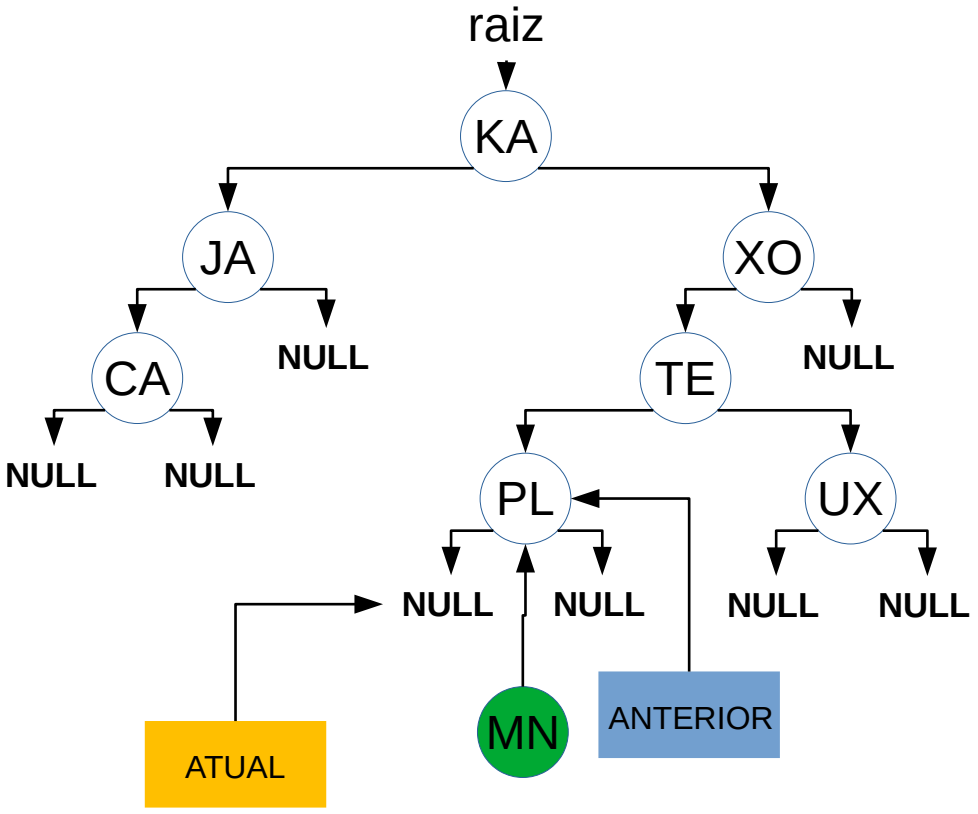
Inserir: KA JA XO CA TE UX PL MN FI LI MA



inserirIterativamente(umValor):

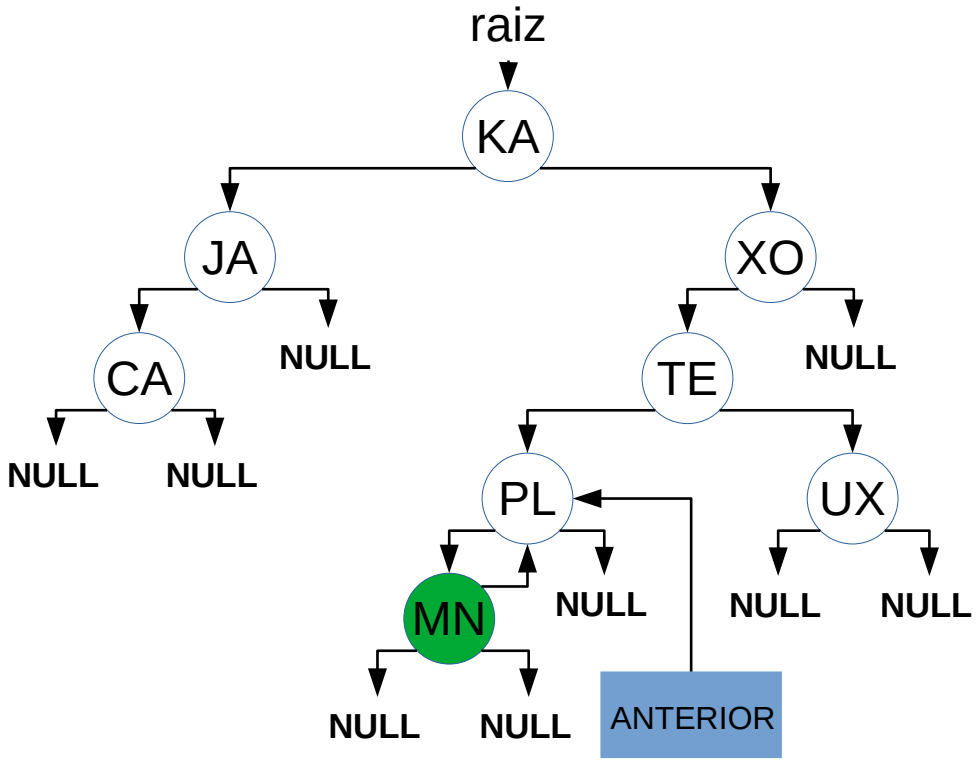
```
novo ← criar_noh(umValor);
se (raiz = NULL) {
  raiz ← novo;
}
senão {
  atual ← raiz;
  enquanto (atual ≠ NULL) {
    anterior ← atual;
    se (atual.valor > umValor) {
      atual ← atual.esquerdo;
    } senão {
      atual ← atual.direito;
    }
  }
novo.pai ← anterior;
  se (anterior.valor >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

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



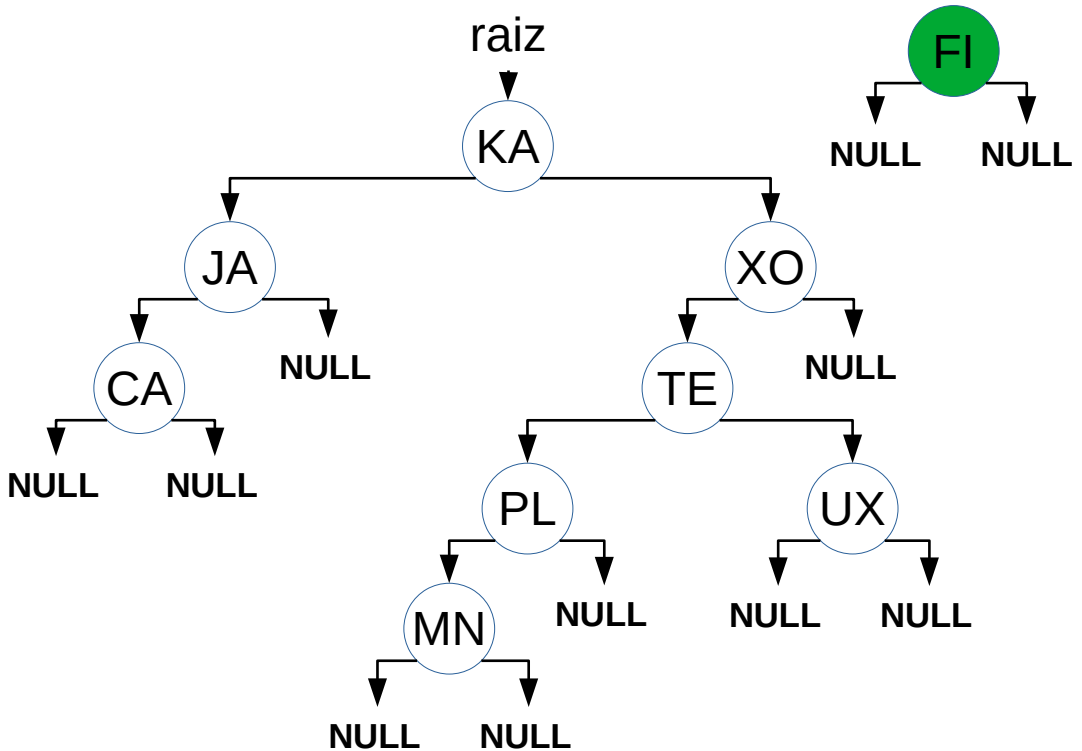

```
inserirIterativamente(umValor):
novo ← criar_noh(umValor);
se (raiz = NULL) {
  raiz ← novo;
}
senão {
  atual ← raiz;
  enquanto (atual ≠ NULL) {
    anterior ← atual;
    se (atual.valor > umValor) {
      atual ← atual.esquerdo;
    } senão {
      atual ← atual.direito;
    }
  }
novo.pai ← anterior;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

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



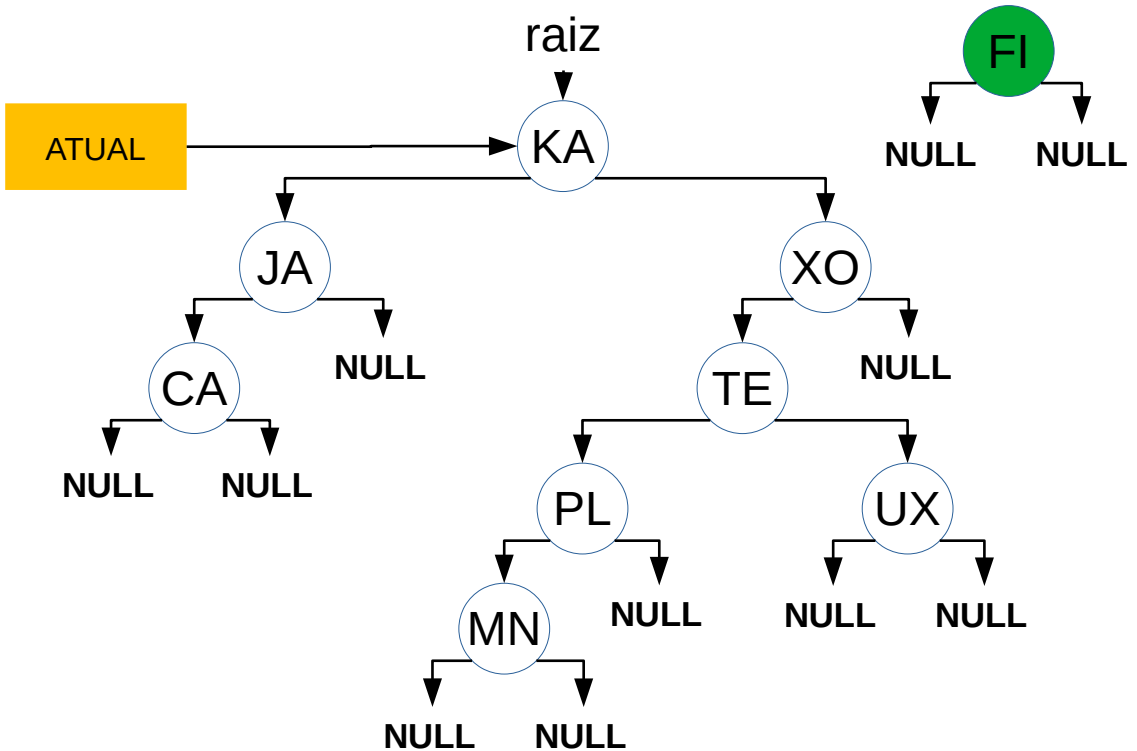

```
inserirIterativamente(umValor):
novo ← criar_noh(umValor);
se (raiz = NULL) {
  raiz ← novo;
}
senão {
  atual ← raiz;
  enquanto (atual ≠ NULL) {
    anterior ← atual;
    se (atual.valor > umValor) {
      atual ← atual.esquerdo;
    } senão {
      atual ← atual.direito;
    }
  }
novo.pai ← anterior;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

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



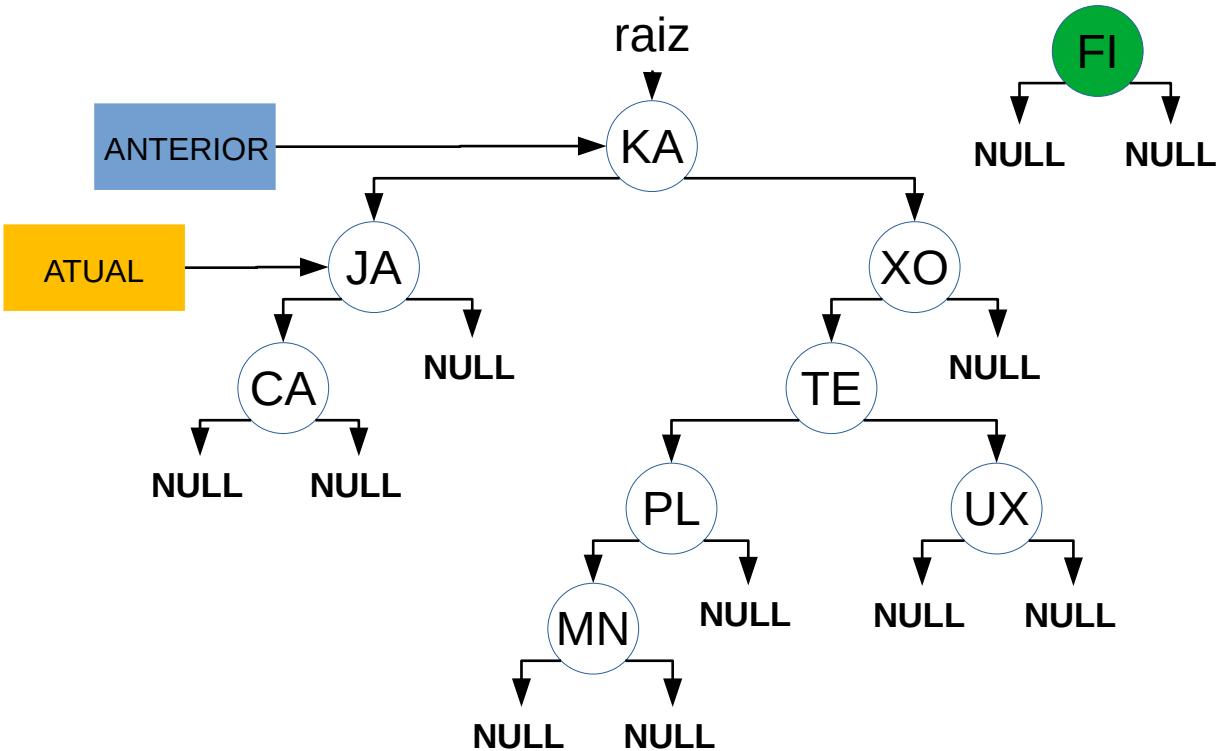

```
inserirIterativamente(umValor):
novo ← criar_noh(umValor);
se (raiz = NULL) {
  raiz ← novo;
}
senão {
  atual ← raiz;
  enquanto (atual ≠ NULL) {
    anterior ← atual;
    se (atual.valor > umValor) {
      atual ← atual.esquerdo;
    } senão {
      atual ← atual.direito;
    }
  }
novo.pai ← anterior;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

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



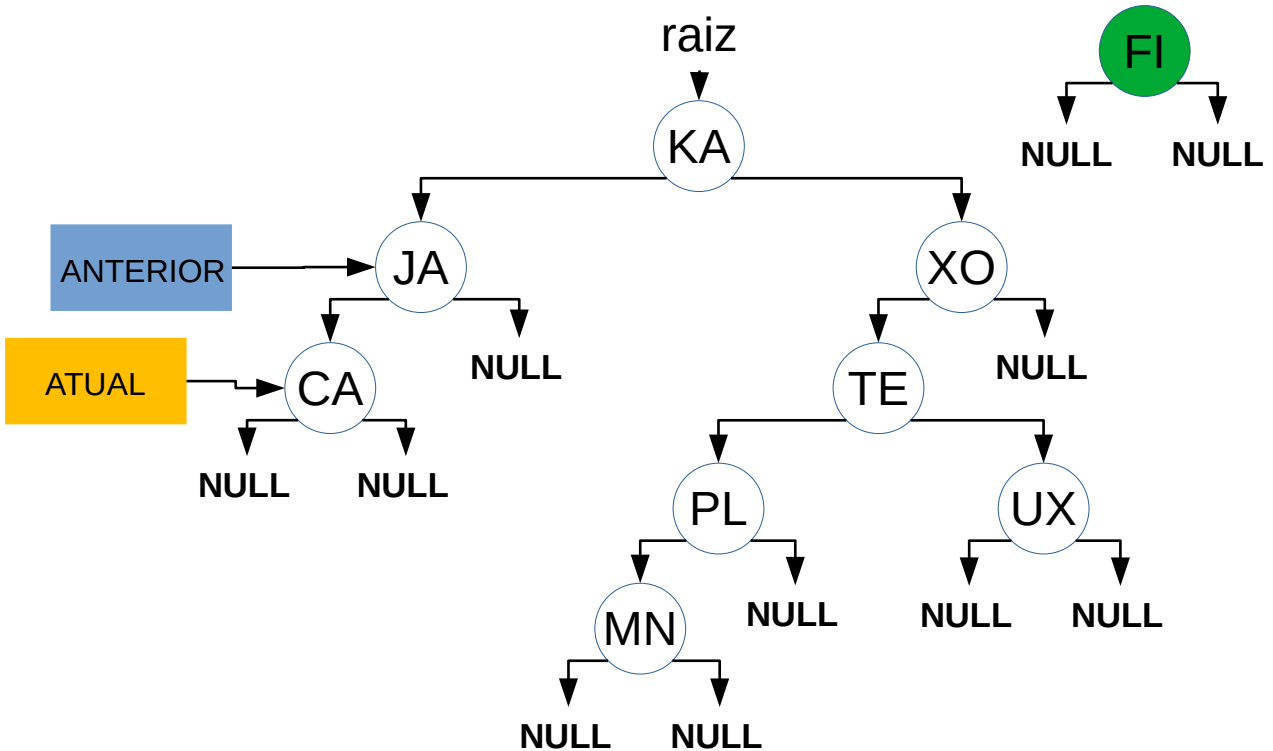

```
inserirIterativamente(umValor):
novo ← criar_noh(umValor);
se (raiz = NULL) {
  raiz ← novo;
}
senão {
  atual ← raiz;
  enquanto (atual ≠ NULL) {
    anterior ← atual;
    se (atual.valor > umValor) {
      atual ← atual.esquerdo;
    } senão {
      atual ← atual.direito;
    }
  }
novo.pai ← anterior;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

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



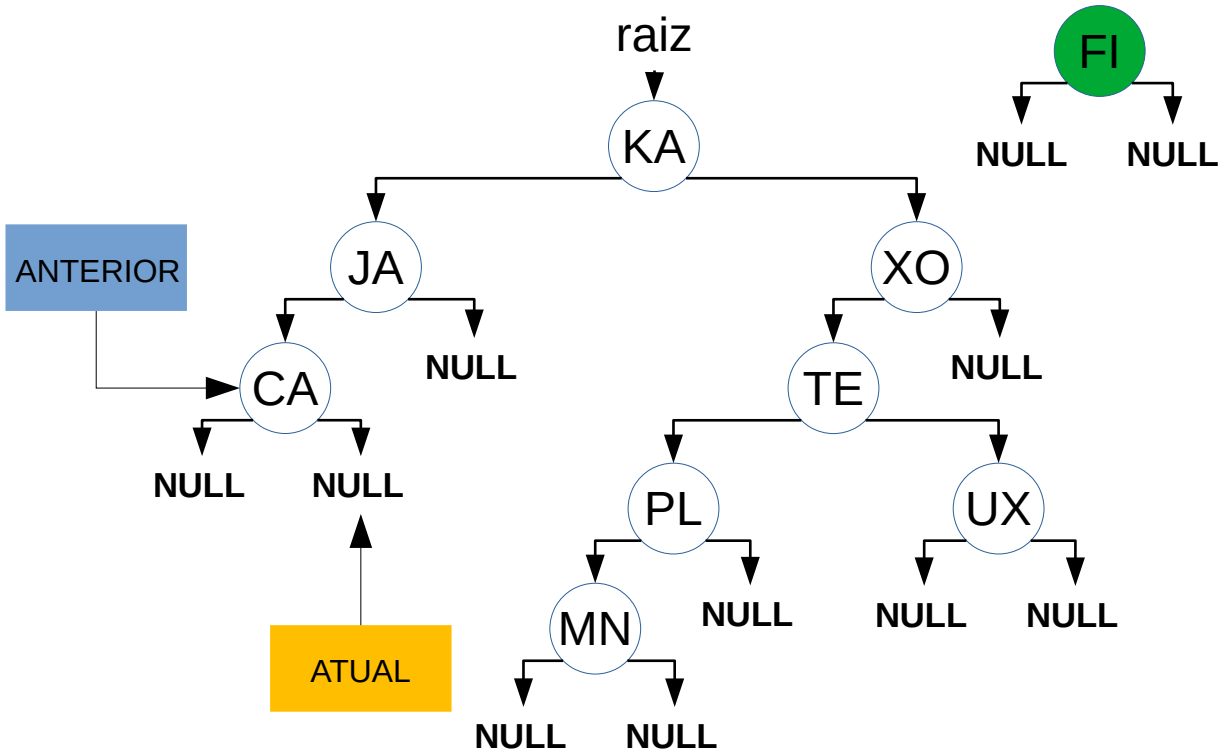

```
inserirIterativamente(umValor):
novo ← criar_noh(umValor);
se (raiz = NULL) {
  raiz ← novo;
}
senão {
  atual ← raiz;
  enquanto (atual ≠ NULL) {
    anterior ← atual;
    se (atual.valor > umValor) {
      atual ← atual.esquerdo;
    } senão {
      atual ← atual.direito;
    }
  }
novo.pai ← anterior;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

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



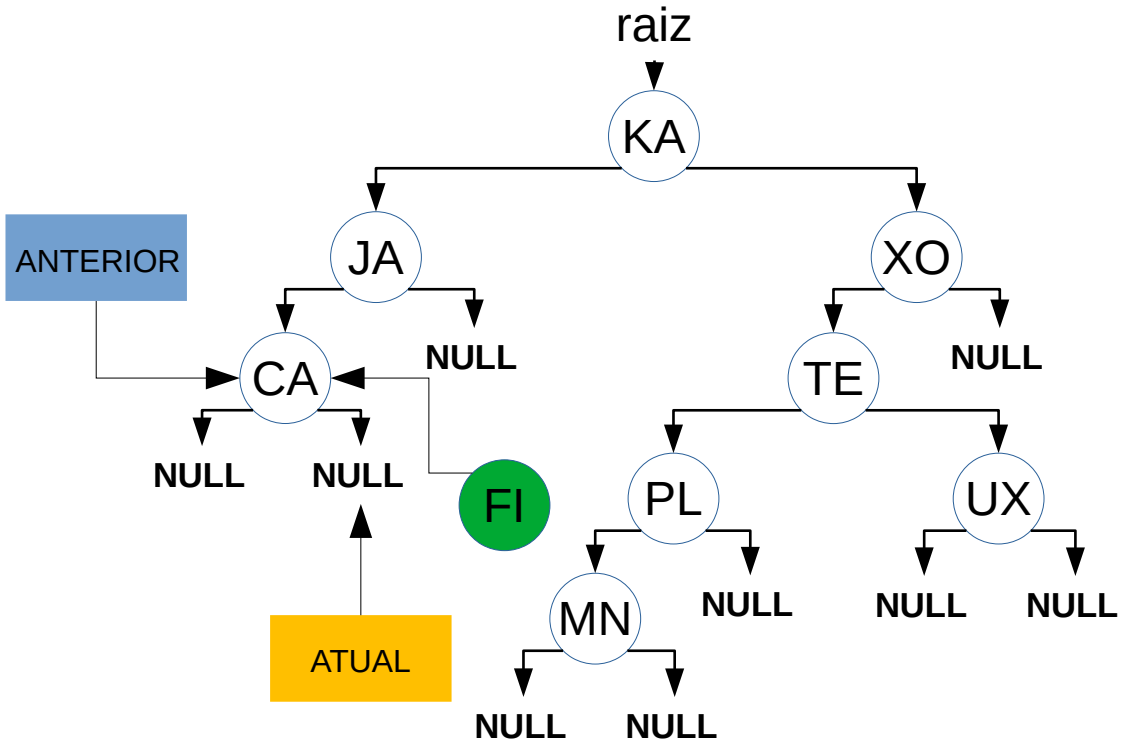

```
inserirIterativamente(umValor):
novo ← criar_noh(umValor);
se (raiz = NULL) {
  raiz ← novo;
}
senão {
  atual ← raiz;
  enquanto (atual ≠ NULL) {
    anterior ← atual;
    se (atual.valor > umValor) {
      atual ← atual.esquerdo;
    } senão {
      atual ← atual.direito;
    }
  }
novo.pai ← anterior;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

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



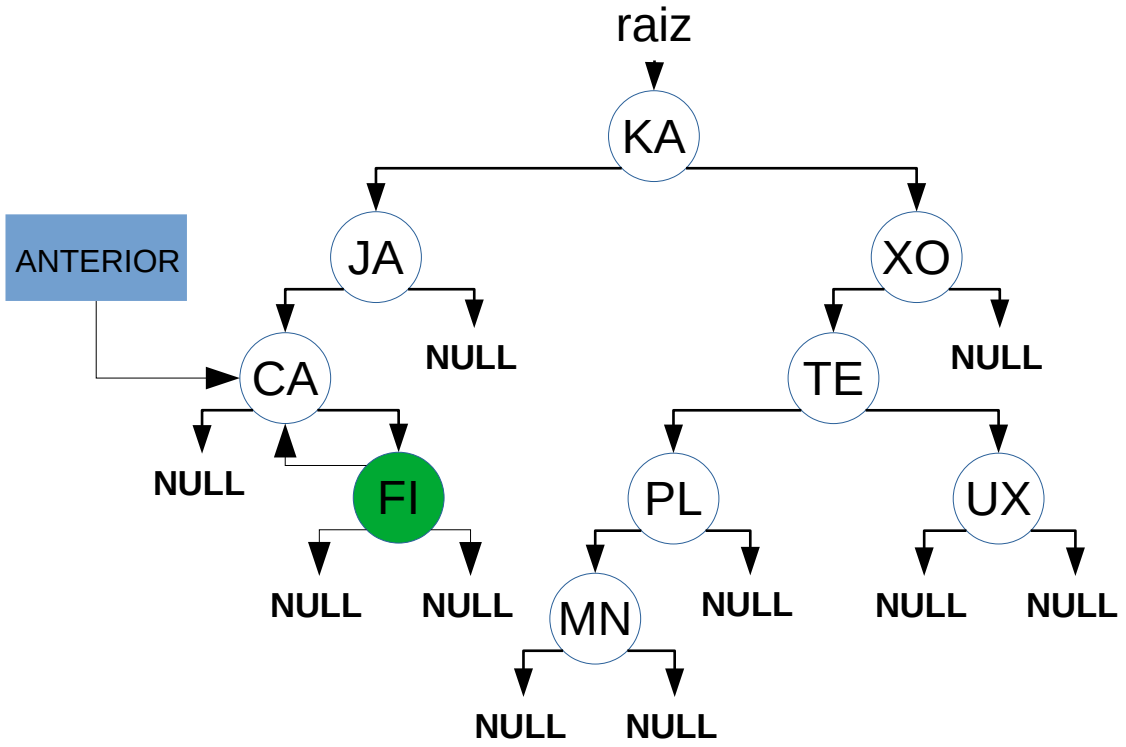

```
inserirIterativamente(umValor):
novo ← criar_noh(umValor);
se (raiz = NULL) {
  raiz ← novo;
}
senão {
  atual ← raiz;
  enquanto (atual ≠ NULL) {
    anterior ← atual;
    se (atual.valor > umValor) {
      atual ← atual.esquerdo;
    } senão {
      atual ← atual.direito;
    }
  }
novo.pai ← anterior;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

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



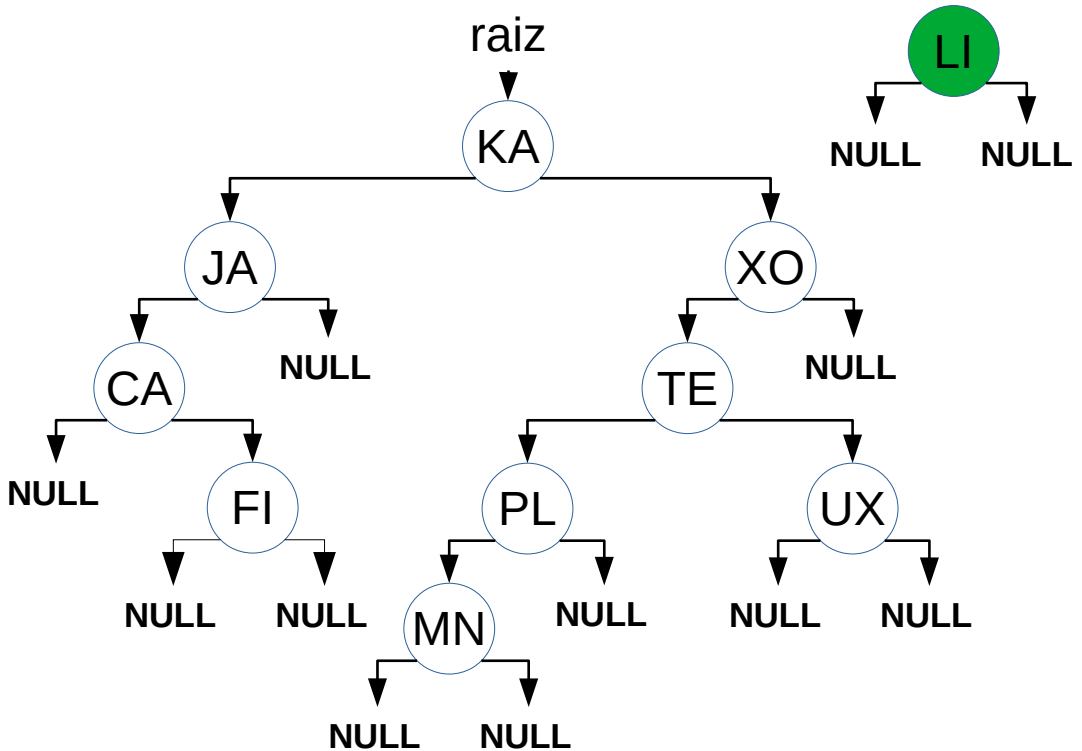

```
inserirIterativamente(umValor):
novo ← criar_noh(umValor);
se (raiz = NULL) {
  raiz ← novo;
}
senão {
  atual ← raiz;
  enquanto (atual ≠ NULL) {
    anterior ← atual;
    se (atual.valor > umValor) {
      atual ← atual.esquerdo;
    } senão {
      atual ← atual.direito;
    }
  }
novo.pai ← anterior;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

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



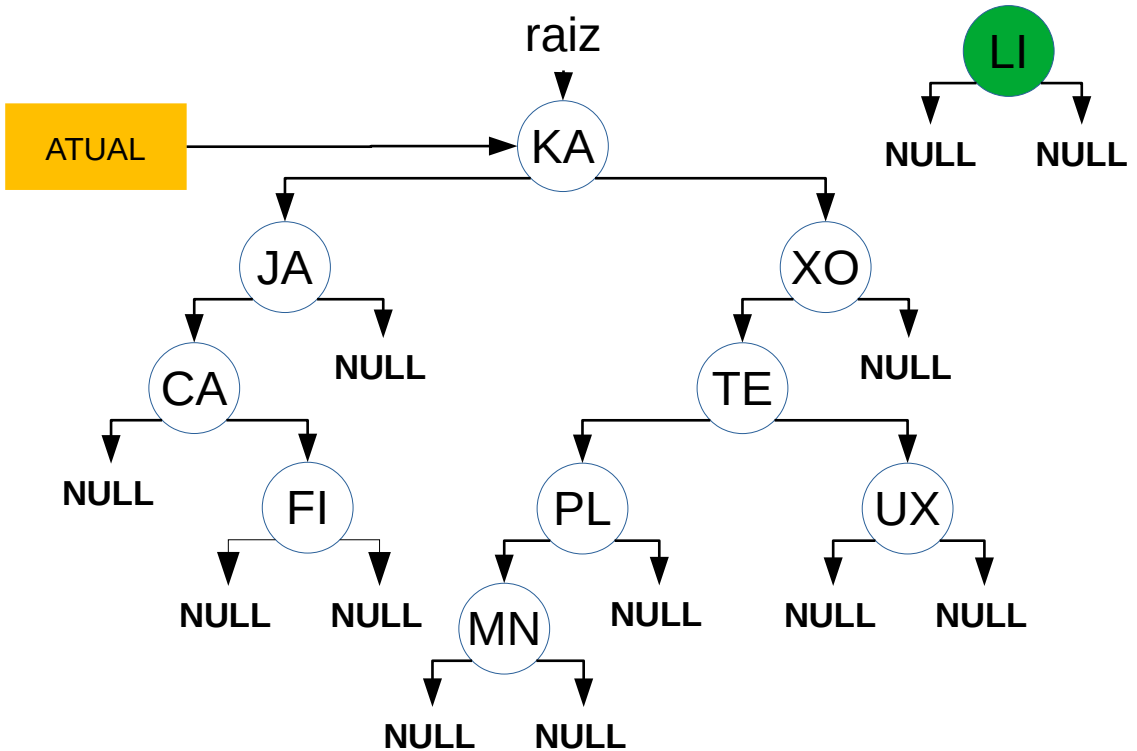

```
inserirIterativamente(umValor):
novo ← criar_noh(umValor);
se (raiz = NULL) {
  raiz ← novo;
}
senão {
  atual ← raiz;
  enquanto (atual ≠ NULL) {
    anterior ← atual;
    se (atual.valor > umValor) {
      atual ← atual.esquerdo;
    } senão {
      atual ← atual.direito;
    }
  }
novo.pai ← anterior;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

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



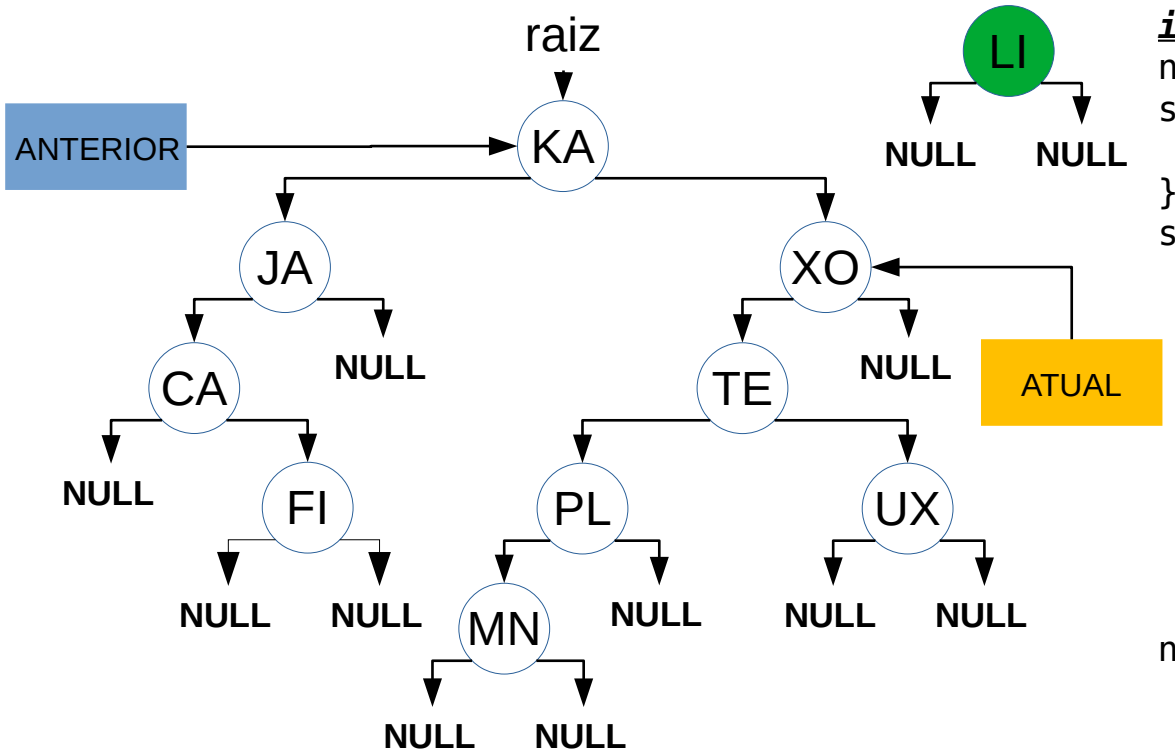

```
inserirIterativamente(umValor):
novo ← criar_noh(umValor);
se (raiz = NULL) {
  raiz ← novo;
}
senão {
  atual ← raiz;
  enquanto (atual ≠ NULL) {
    anterior ← atual;
    se (atual.valor > umValor) {
      atual ← atual.esquerdo;
    } senão {
      atual ← atual.direito;
    }
  }
novo.pai ← anterior;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

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



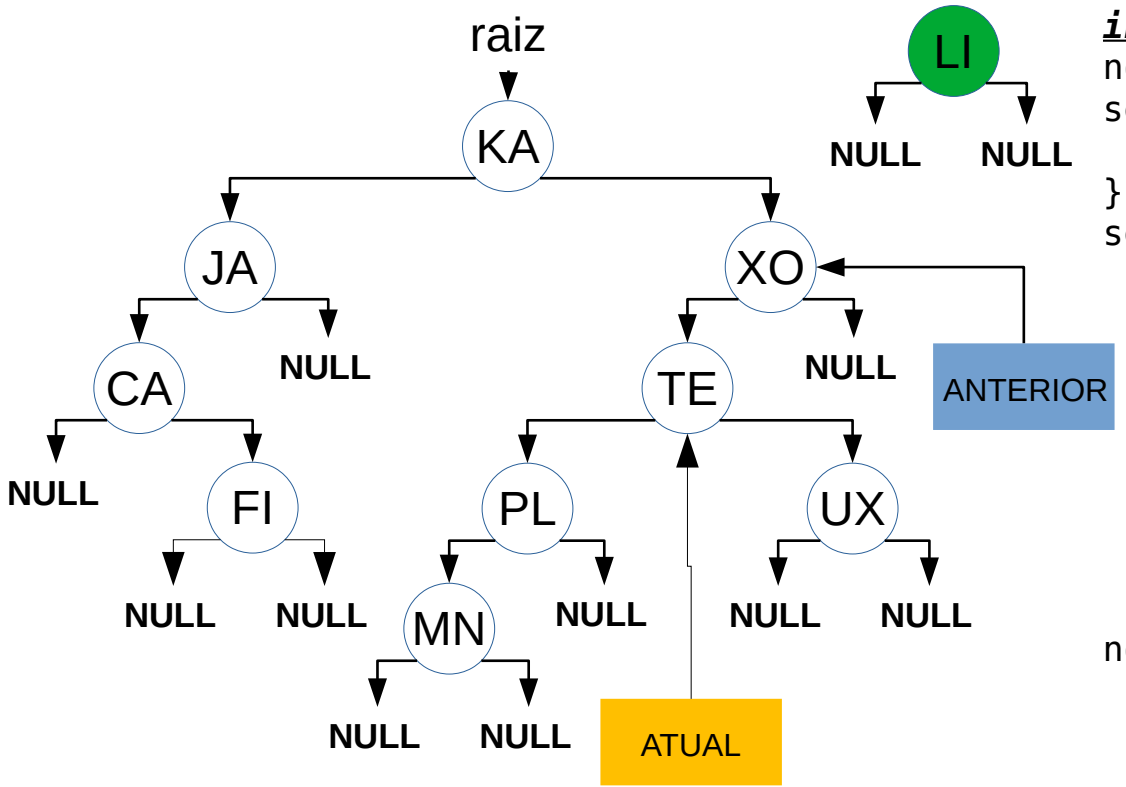

```
inserirIterativamente(umValor):
novo ← criar_noh(umValor);
se (raiz = NULL) {
  raiz ← novo;
}
senão {
  atual ← raiz;
  enquanto (atual ≠ NULL) {
    anterior ← atual;
    se (atual.valor > umValor) {
      atual ← atual.esquerdo;
    } senão {
      atual ← atual.direito;
    }
  }
novo.pai ← anterior;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

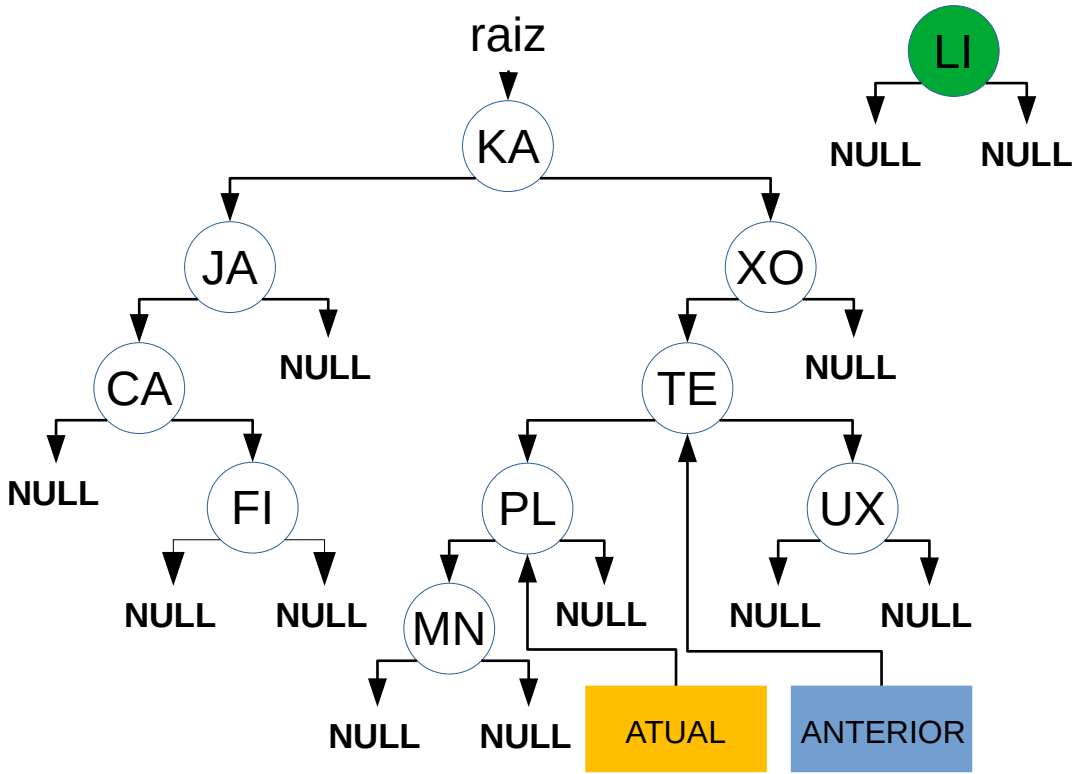
Inserir: KA JA XO CA TE UX PL MN FI LI MA



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;
  }
}
```

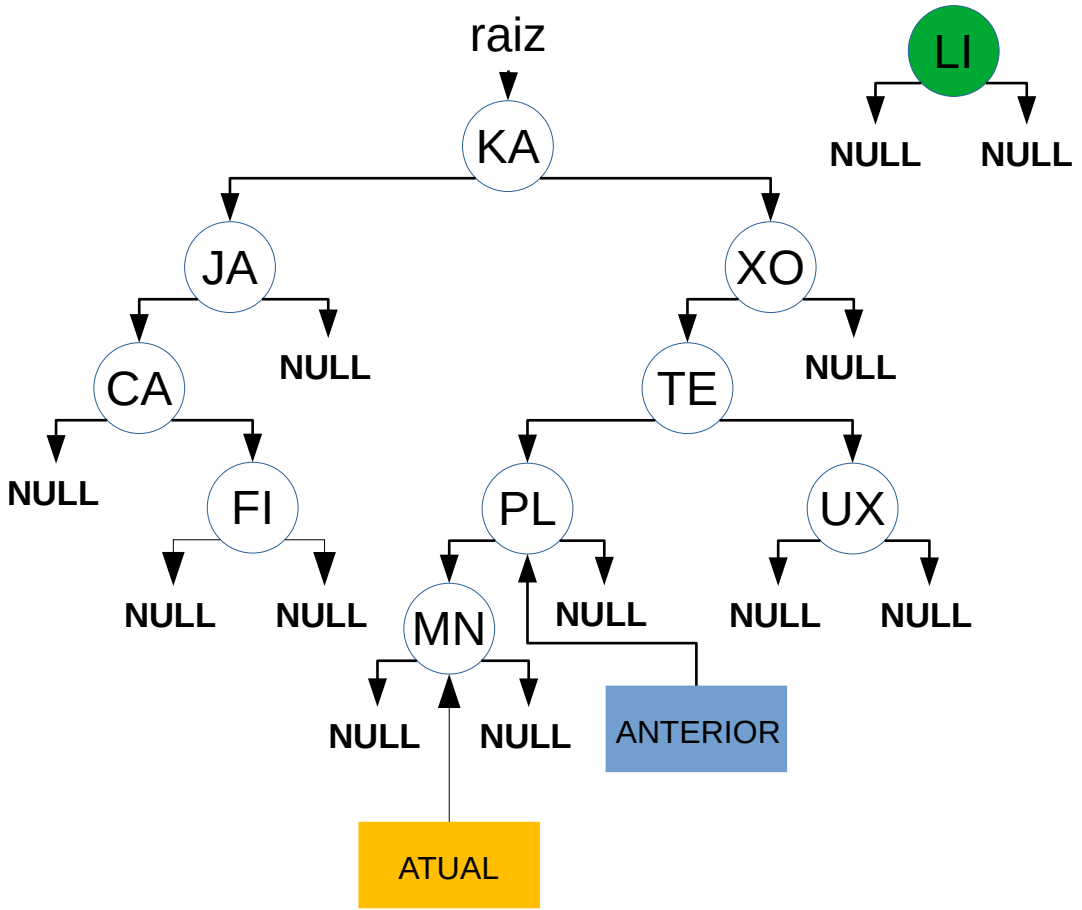
Inserir: KA JA XO CA TE UX PL MN FI LI MA



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;
  }
}
```

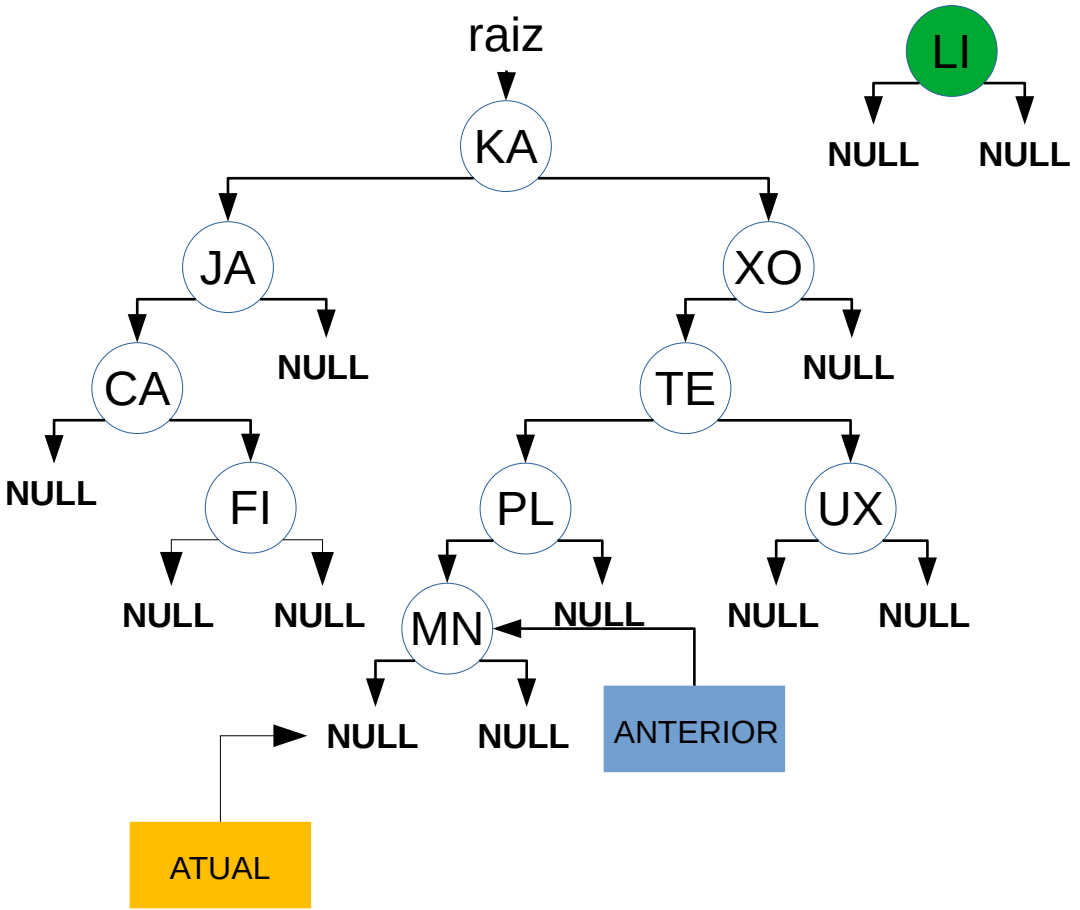
Inserir: KA JA XO CA TE UX PL MN FI LI MA



inserirIterativamente(umValor):

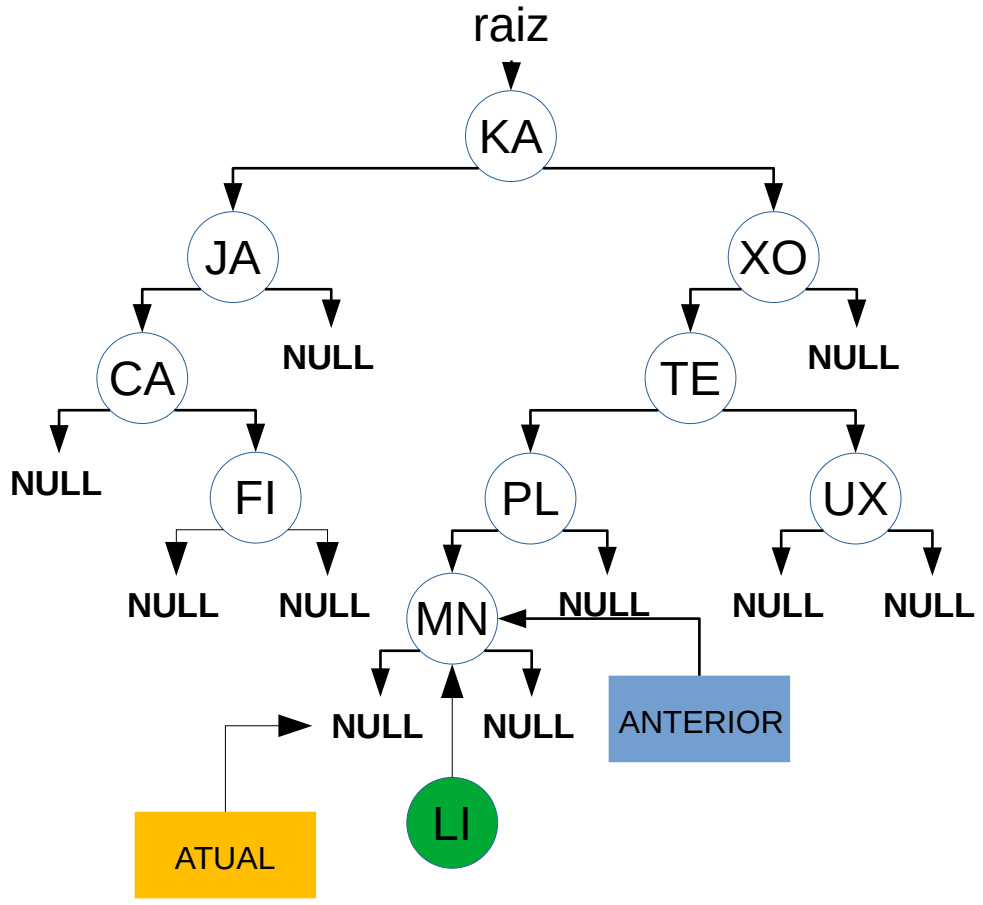
```
novo ← criar_noh(umValor);
se (raiz = NULL) {
  raiz ← novo;
}
senão {
  atual ← raiz;
  enquanto (atual ≠ NULL) {
    anterior ← atual;
    se (atual.valor > umValor) {
      atual ← atual.esquerdo;
    } senão {
      atual ← atual.direito;
    }
  }
novo.pai ← anterior;
  se (anterior.valor >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

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



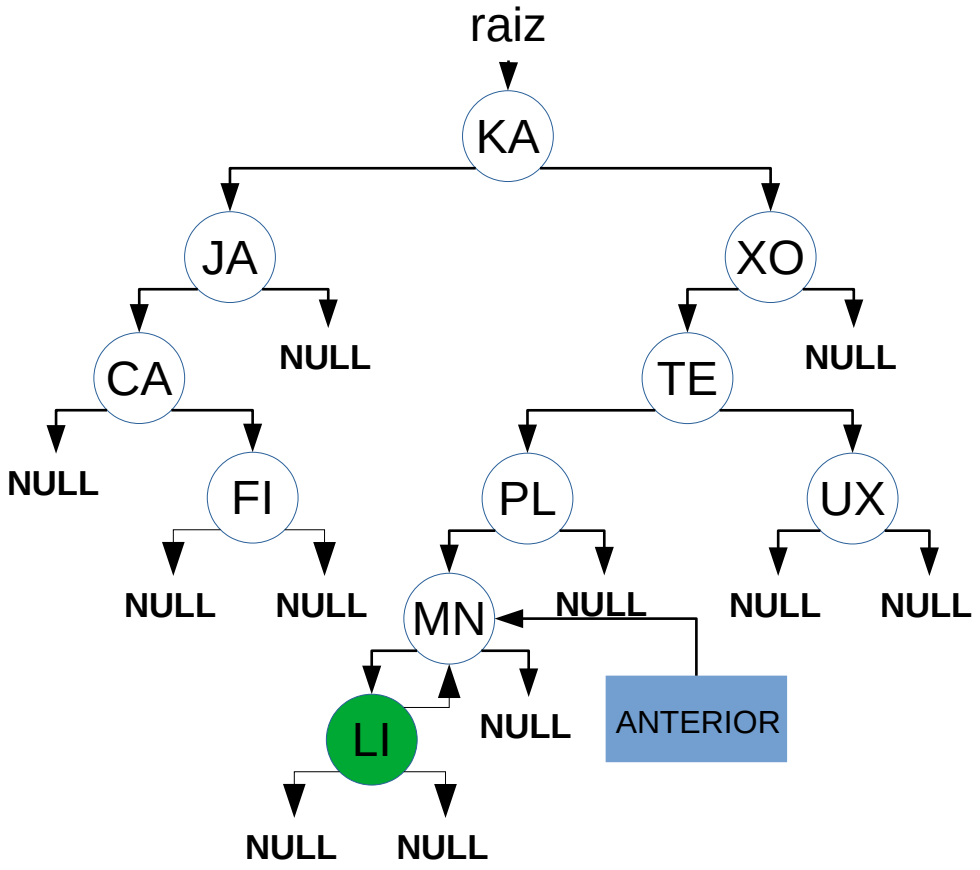

```
inserirIterativamente(umValor):
novo ← criar_noh(umValor);
se (raiz = NULL) {
  raiz ← novo;
}
senão {
  atual ← raiz;
  enquanto (atual ≠ NULL) {
    anterior ← atual;
    se (atual.valor > umValor) {
      atual ← atual.esquerdo;
    } senão {
      atual ← atual.direito;
    }
  }
novo.pai ← anterior;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

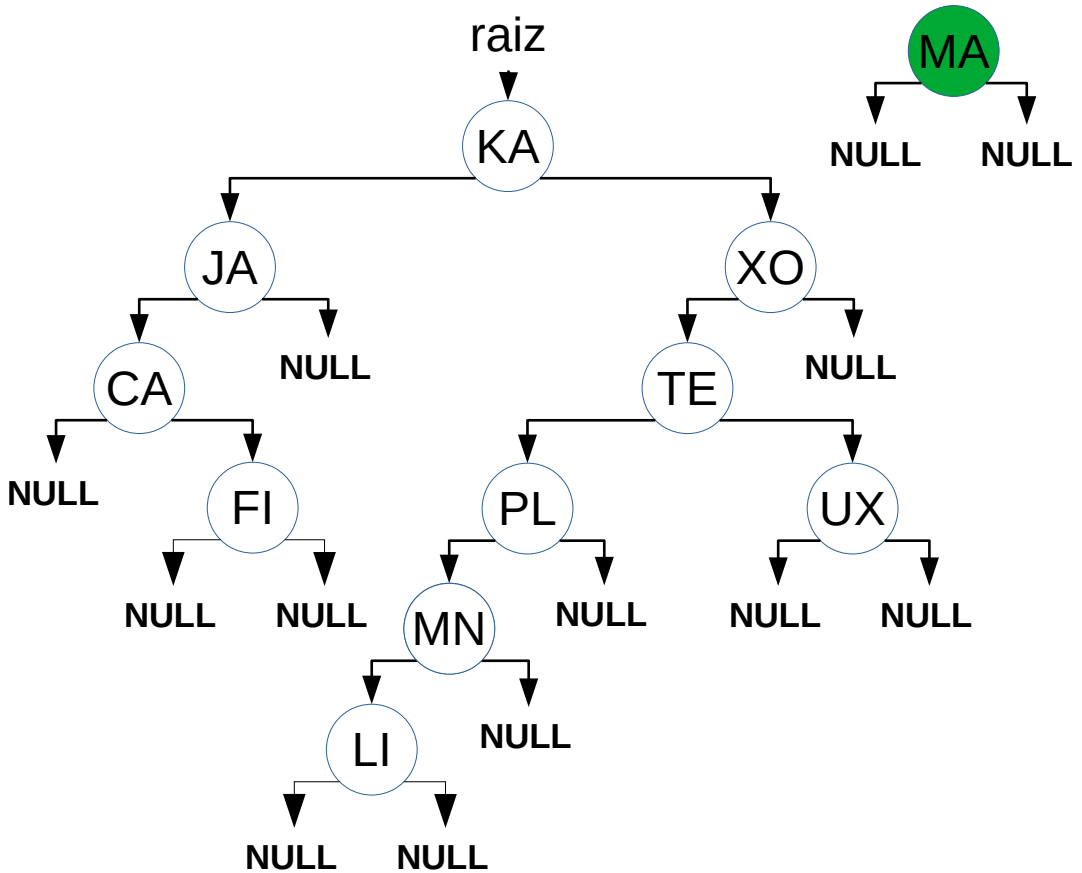
Inserir: KA JA XO CA TE UX PL MN FI LI MA



inserirIterativamente(umValor):

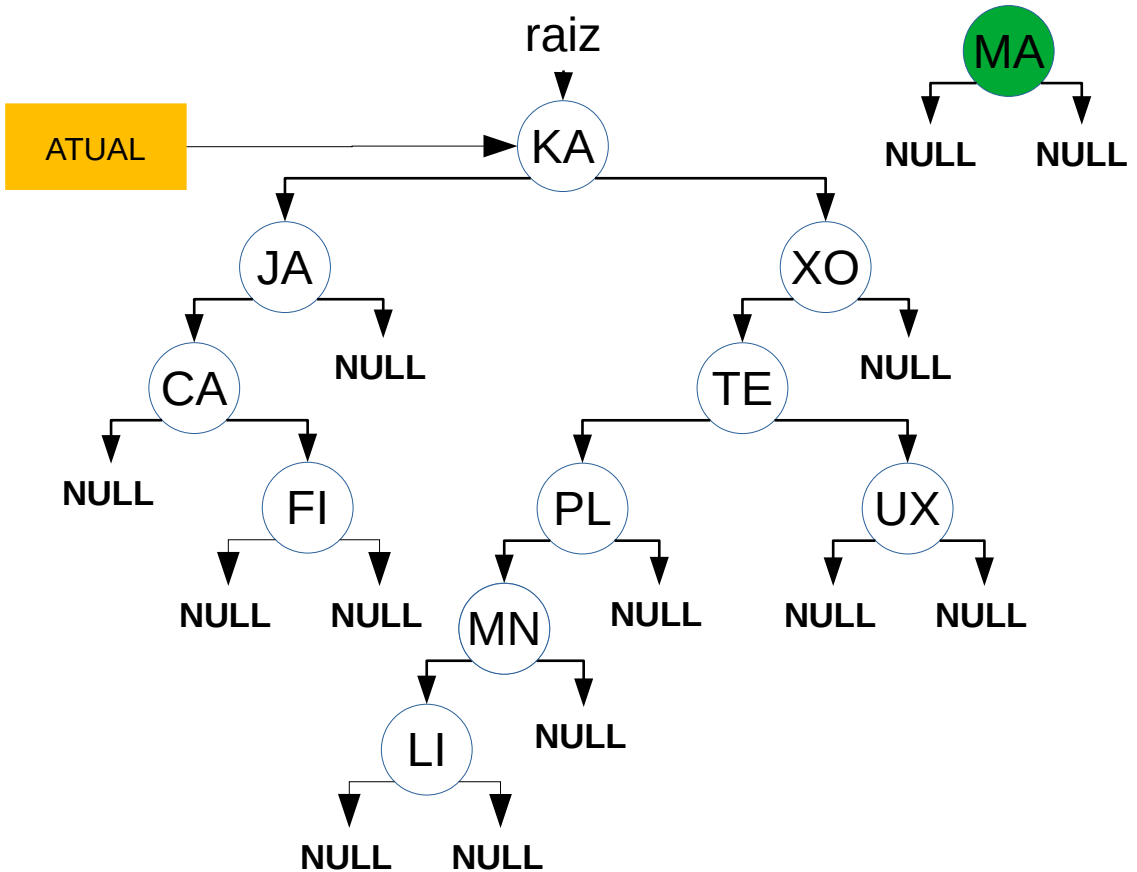
```
novo ← criar_noh(umValor);
se (raiz = NULL) {
  raiz ← novo;
}
senão {
  atual ← raiz;
  enquanto (atual ≠ NULL) {
    anterior ← atual;
    se (atual.valor > umValor) {
      atual ← atual.esquerdo;
    } senão {
      atual ← atual.direito;
    }
  }
novo.pai ← anterior;
  se (anterior.valor >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

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



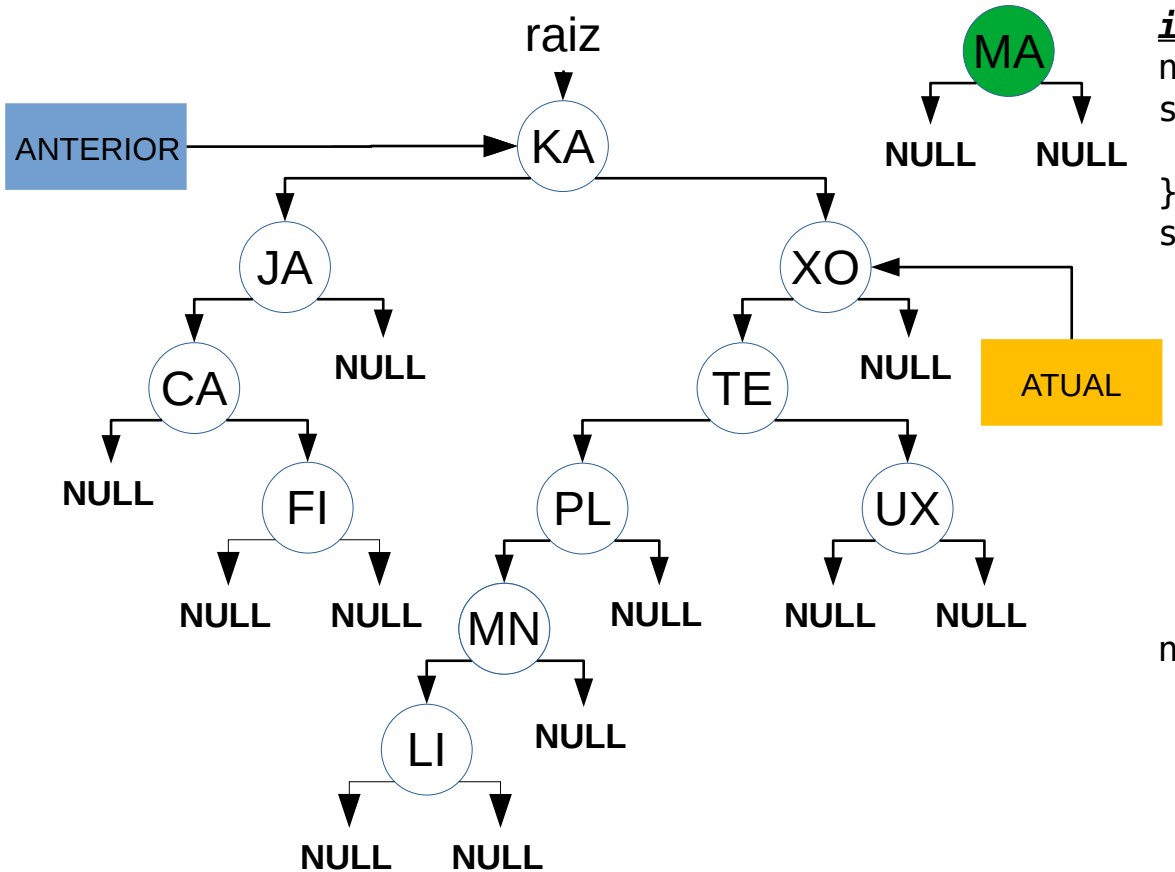

```
inserirIterativamente(umValor):
novo ← criar_noh(umValor);
se (raiz = NULL) {
  raiz ← novo;
}
senão {
  atual ← raiz;
  enquanto (atual ≠ NULL) {
    anterior ← atual;
    se (atual.valor > umValor) {
      atual ← atual.esquerdo;
    } senão {
      atual ← atual.direito;
    }
  }
novo.pai ← anterior;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

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



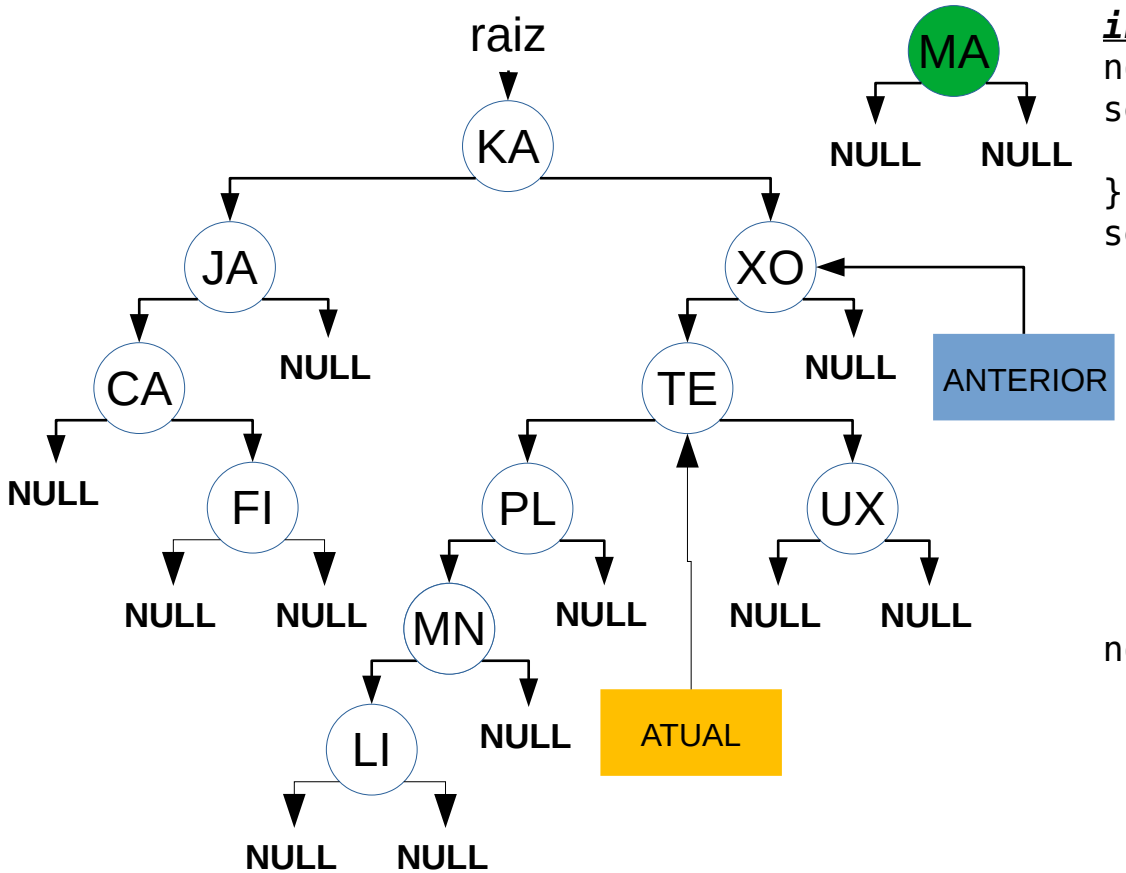

```
inserirIterativamente(umValor):
novo ← criar_noh(umValor);
se (raiz = NULL) {
  raiz ← novo;
}
senão {
  atual ← raiz;
  enquanto (atual ≠ NULL) {
    anterior ← atual;
    se (atual.valor > umValor) {
      atual ← atual.esquerdo;
    } senão {
      atual ← atual.direito;
    }
  }
novo.pai ← anterior;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

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



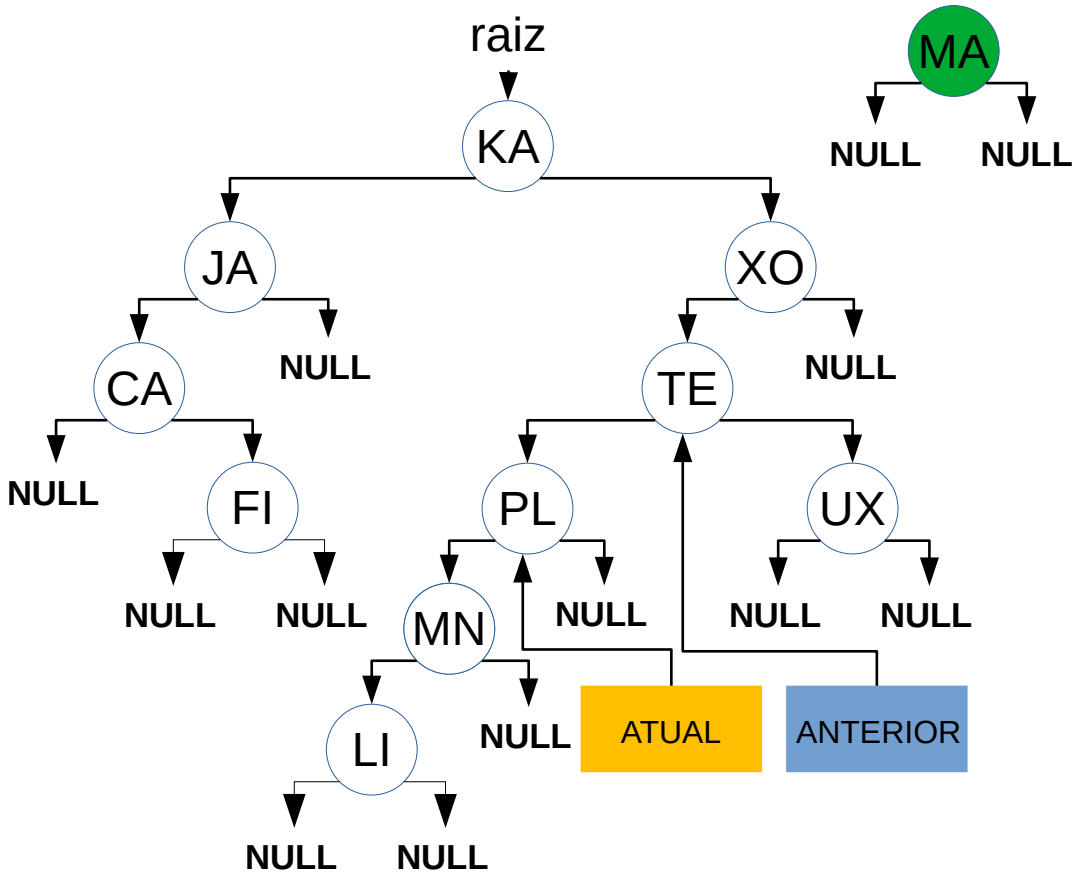

```
inserirIterativamente(umValor):
novo ← criar_noh(umValor);
se (raiz = NULL) {
  raiz ← novo;
}
senão {
  atual ← raiz;
  enquanto (atual ≠ NULL) {
    anterior ← atual;
    se (atual.valor > umValor) {
      atual ← atual.esquerdo;
    } senão {
      atual ← atual.direito;
    }
  }
novo.pai ← anterior;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

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



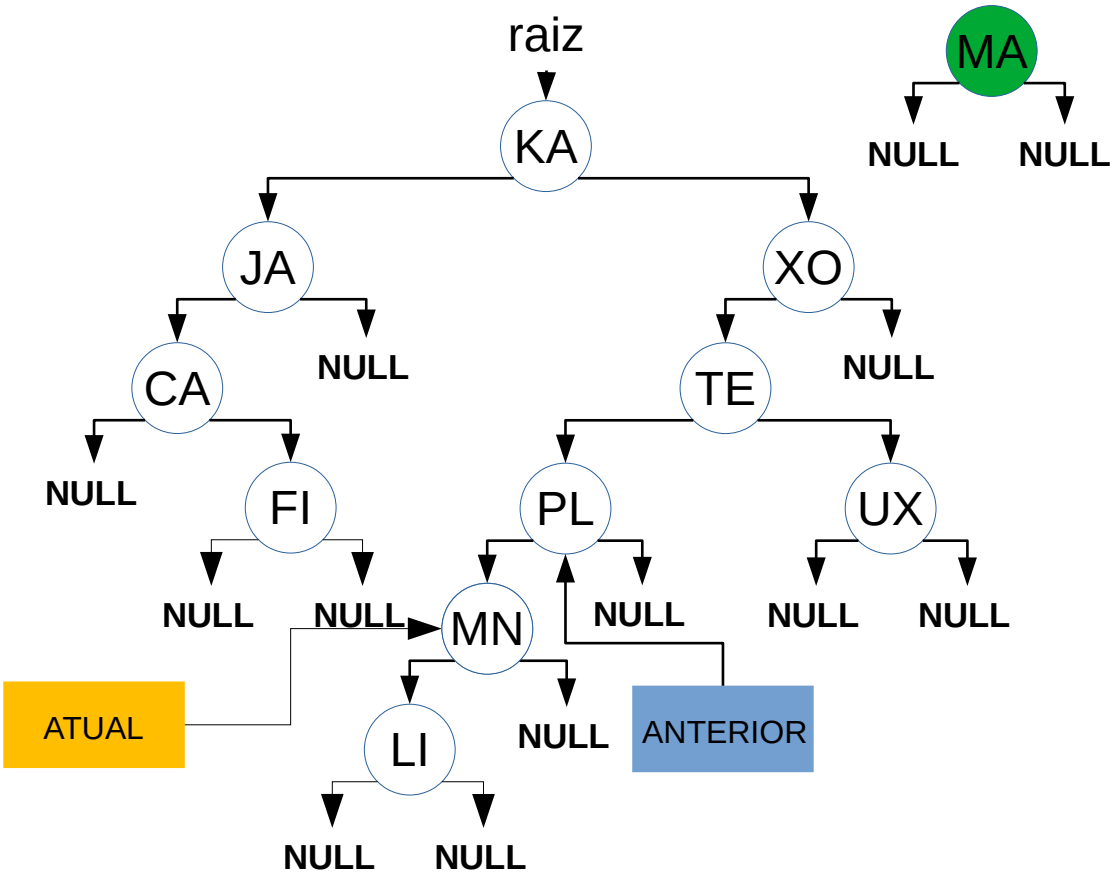

```
inserirIterativamente(umValor):
novo ← criar_noh(umValor);
se (raiz = NULL) {
  raiz ← novo;
}
senão {
  atual ← raiz;
  enquanto (atual ≠ NULL) {
    anterior ← atual;
    se (atual.valor > umValor) {
      atual ← atual.esquerdo;
    } senão {
      atual ← atual.direito;
    }
  }
novo.pai ← anterior;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

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



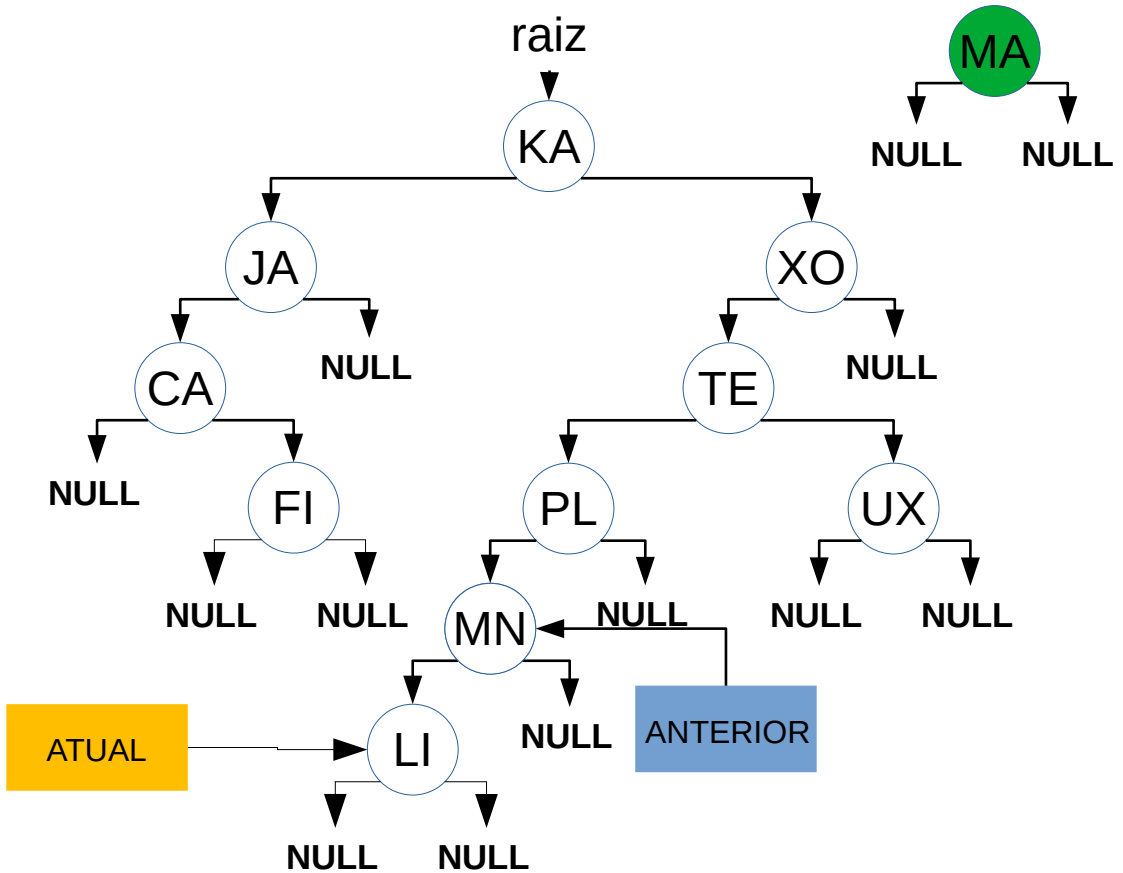

```
inserirIterativamente(umValor):
novo ← criar_noh(umValor);
se (raiz = NULL) {
  raiz ← novo;
}
senão {
  atual ← raiz;
  enquanto (atual ≠ NULL) {
    anterior ← atual;
    se (atual.valor > umValor) {
      atual ← atual.esquerdo;
    } senão {
      atual ← atual.direito;
    }
  }
novo.pai ← anterior;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

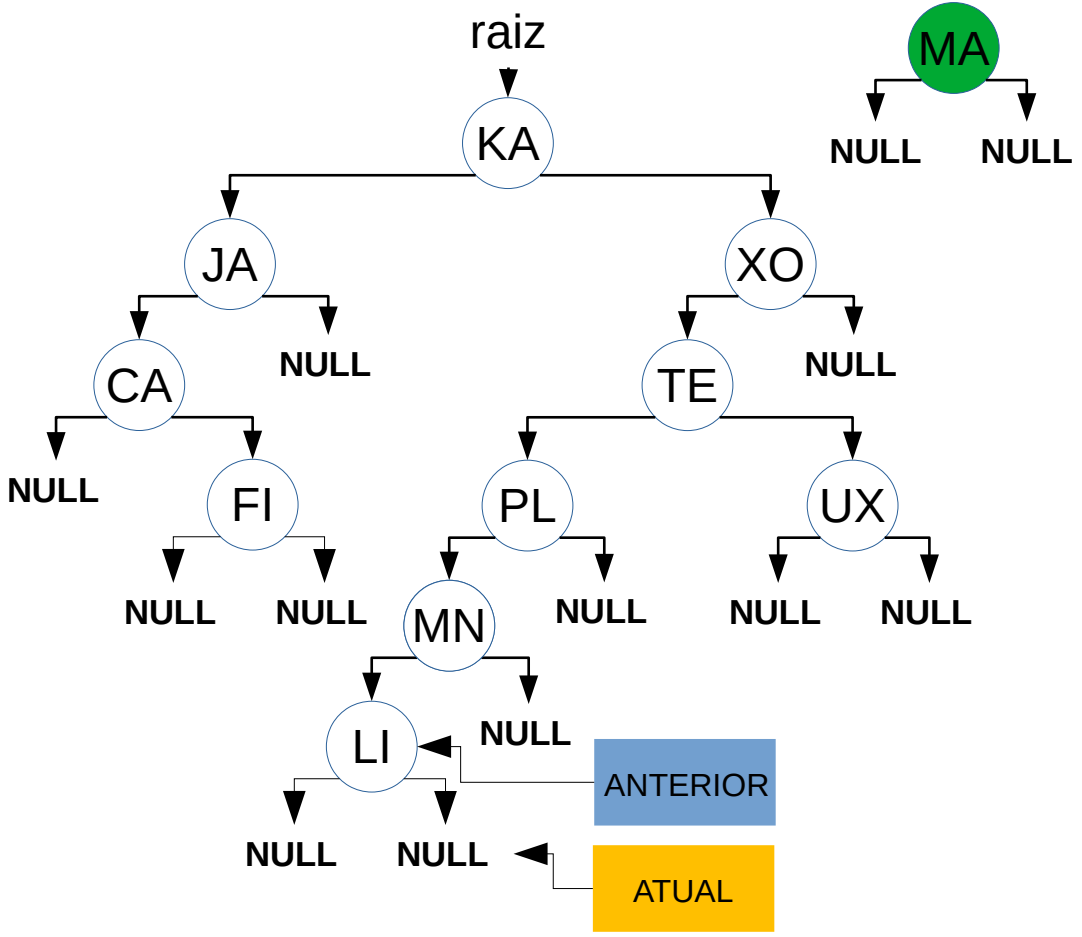
Inserir: KA JA XO CA TE UX PL MN FI LI MA



inserirIterativamente(umValor):

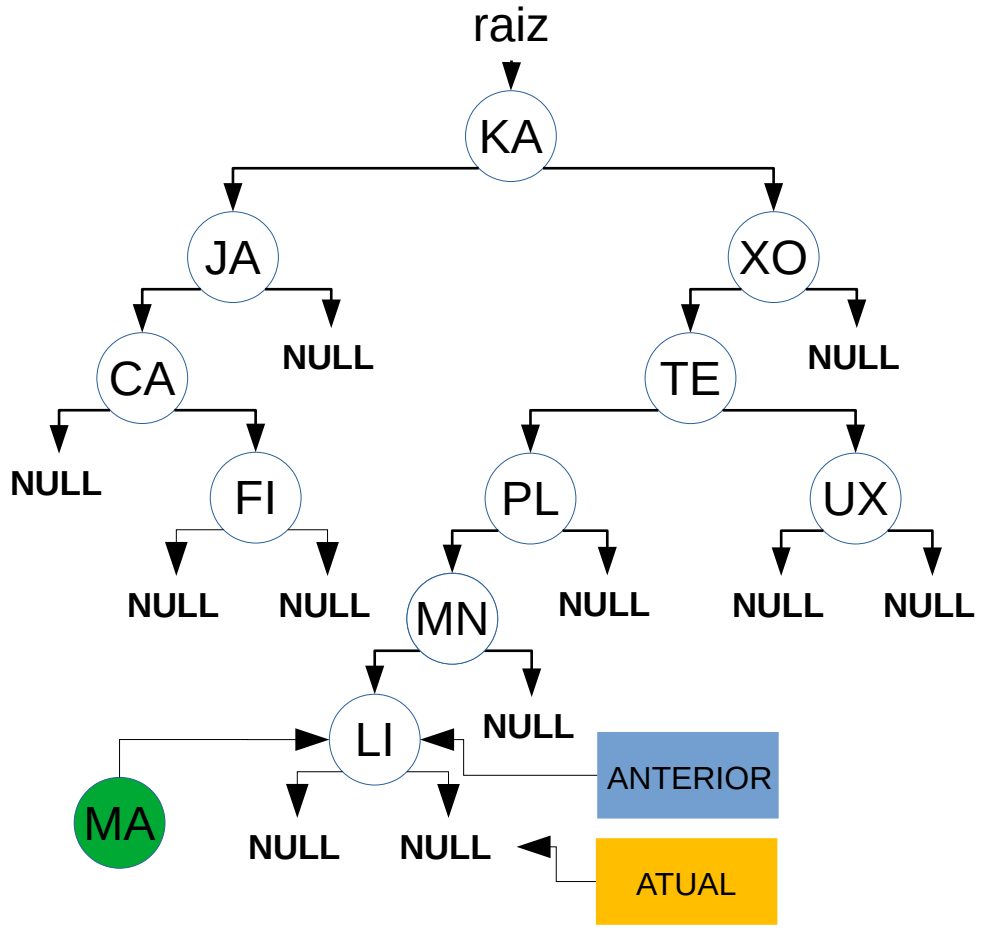
```
novo ← criar_noh(umValor);
se (raiz = NULL) {
  raiz ← novo;
}
senão {
  atual ← raiz;
  enquanto (atual ≠ NULL) {
    anterior ← atual;
    se (atual.valor > umValor) {
      atual ← atual.esquerdo;
    } senão {
      atual ← atual.direito;
    }
  }
novo.pai ← anterior;
  se (anterior.valor >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

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



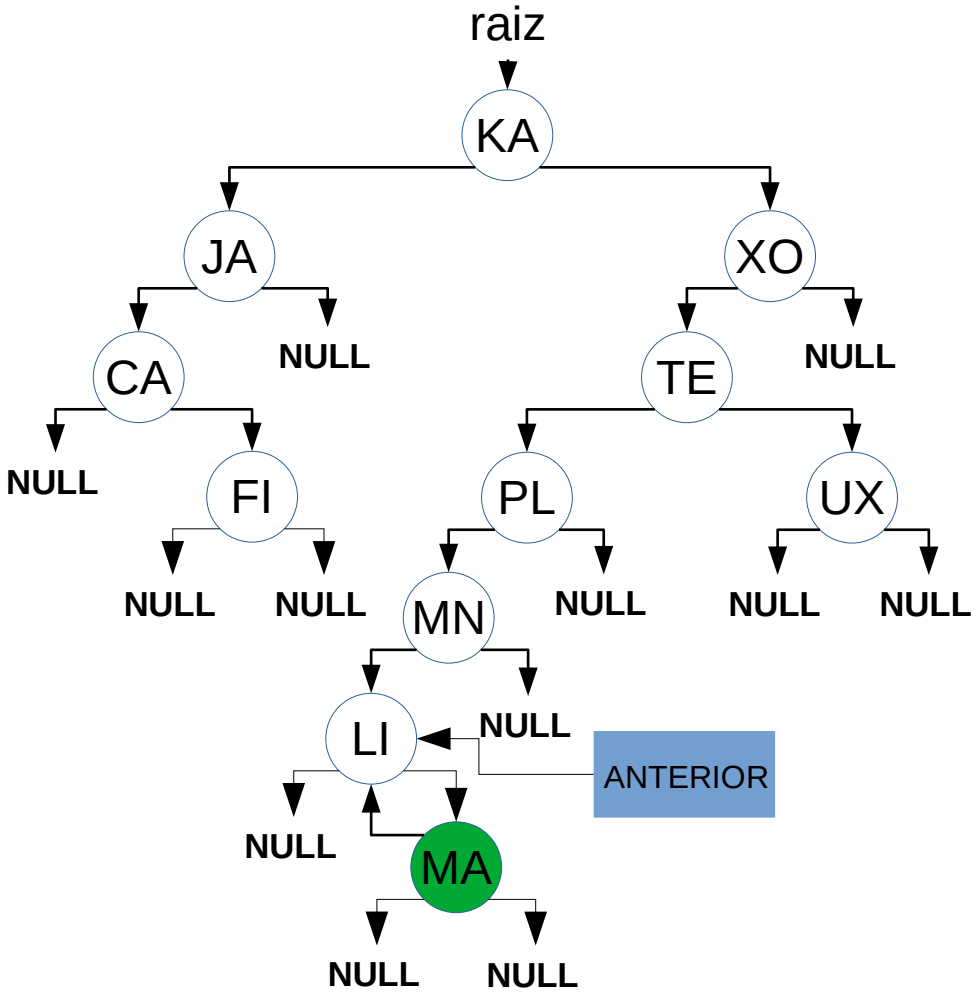

```
inserirIterativamente(umValor):
novo ← criar_noh(umValor);
se (raiz = NULL) {
  raiz ← novo;
}
senão {
  atual ← raiz;
  enquanto (atual ≠ NULL) {
    anterior ← atual;
    se (atual.valor > umValor) {
      atual ← atual.esquerdo;
    } senão {
      atual ← atual.direito;
    }
  }
novo.pai ← anterior;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

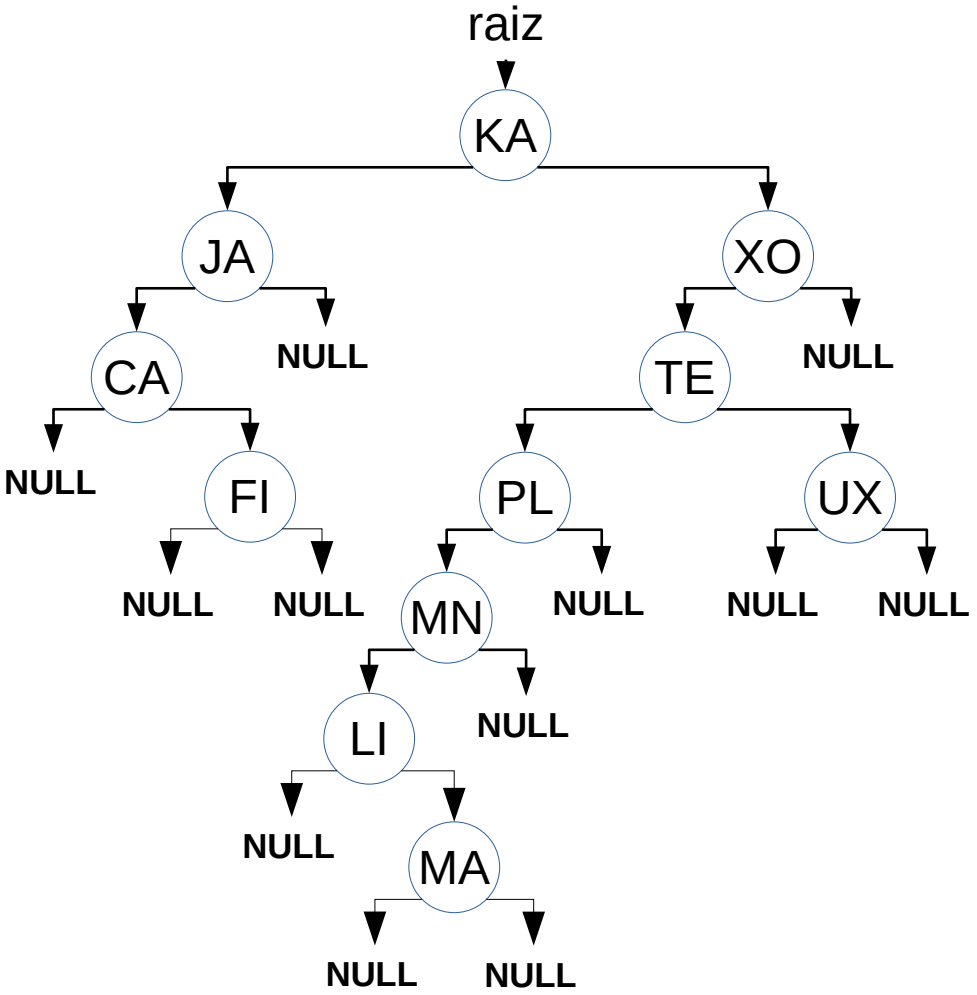
Inserir: KA JA XO CA TE UX PL MN FI LI MA



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;
  }
}
```

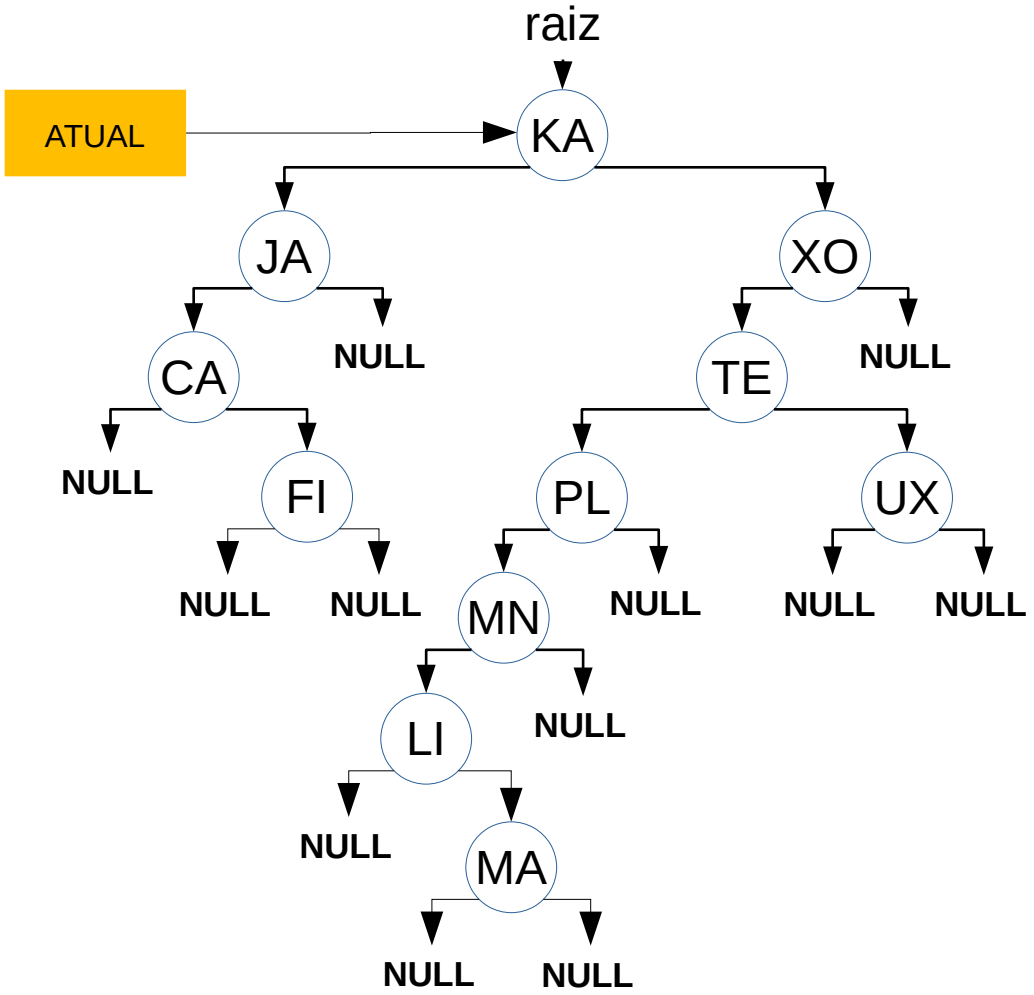
Remover PL CA TE



remove(umValor):

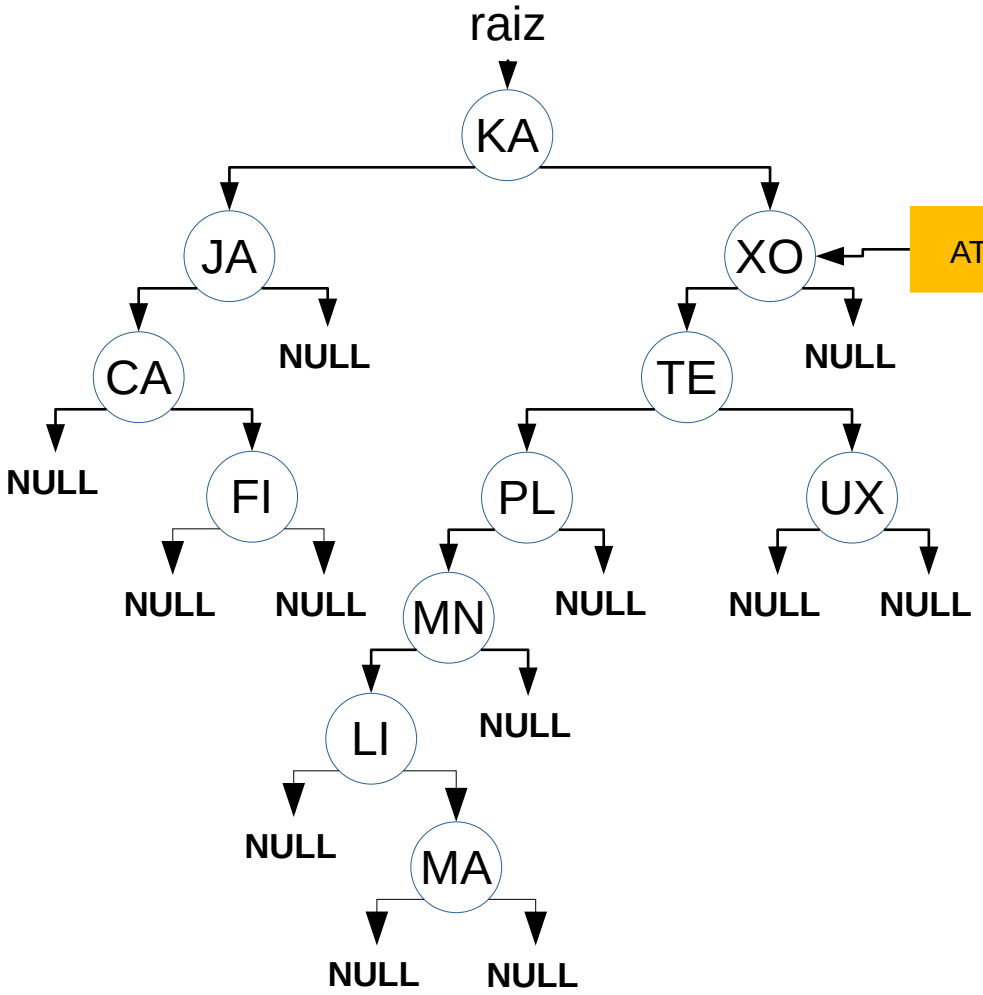
```
nohRemover ← buscaAux(umValor);
```

Remover PL CA TE



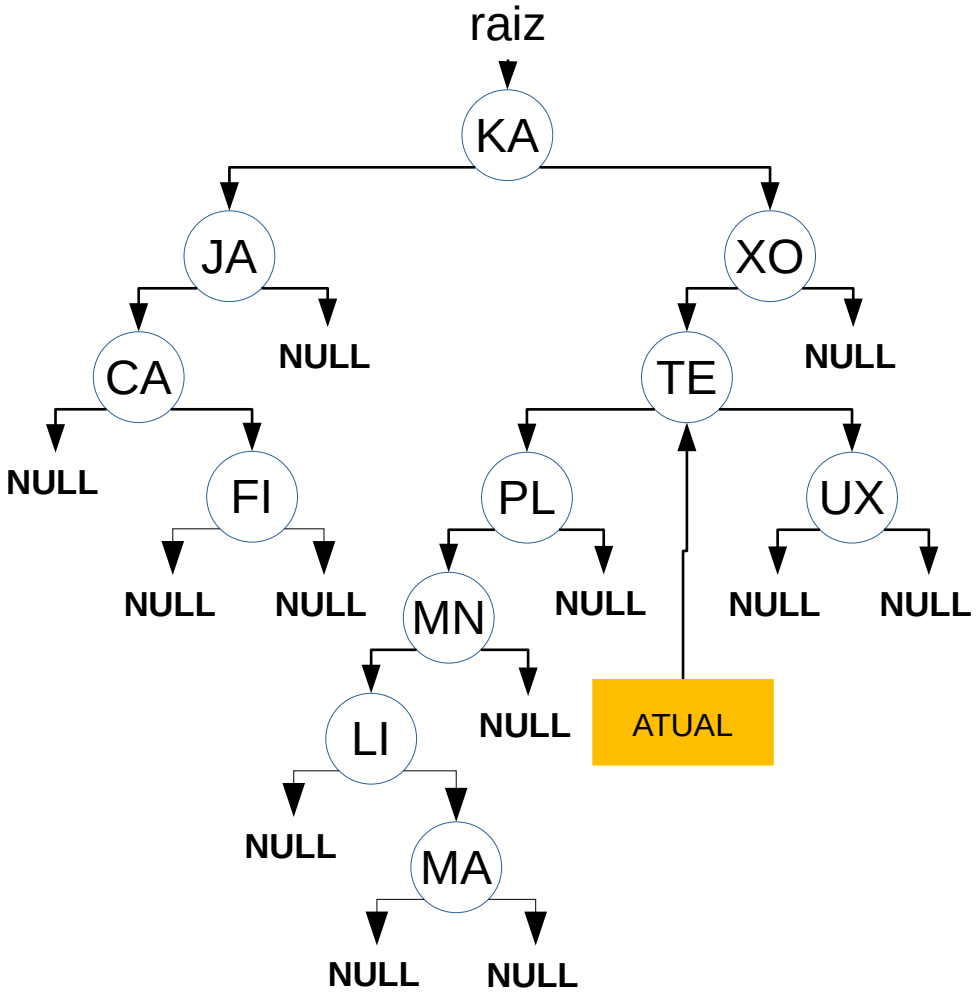

```
buscaAux(umValor):
atual ← raiz;
enquanto (atual ≠ NULL) {
     se (atual.valor = umValor) {
          retorna atual;
     } senão se (atual.valor > umValor) {
          atual ← atual->esquerdo;
     } senão {
          atual ← atual.direito;
     }
}
retorna atual;
```

Remover PL CA TE



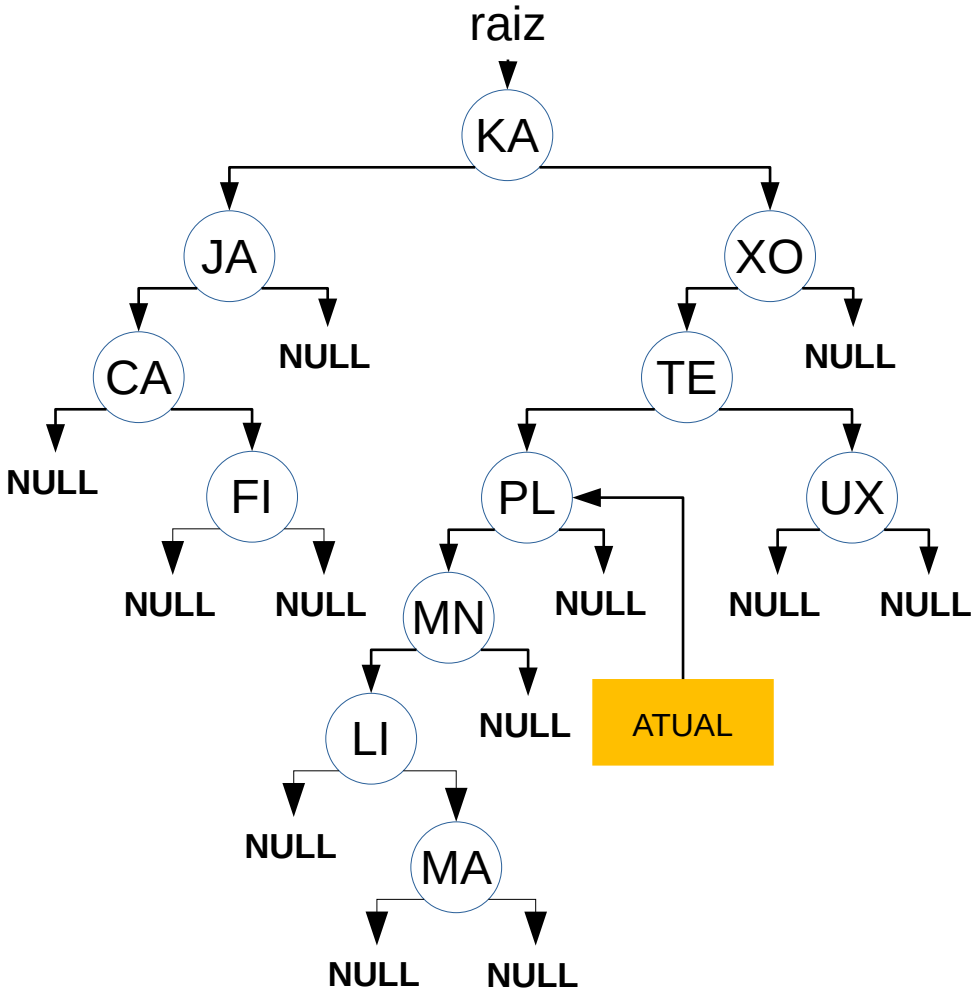

```
buscaAux(umValor):
atual ← raiz;
enquanto (atual ≠ NULL) {
     se (atual.valor = umValor) {
          retorna atual;
     } senão se (atual.valor > umValor) {
          atual ← atual->esquerdo;
     } senão {
          atual ← atual.direito;
     }
}
retorna atual;
```

Remover PL CA TE



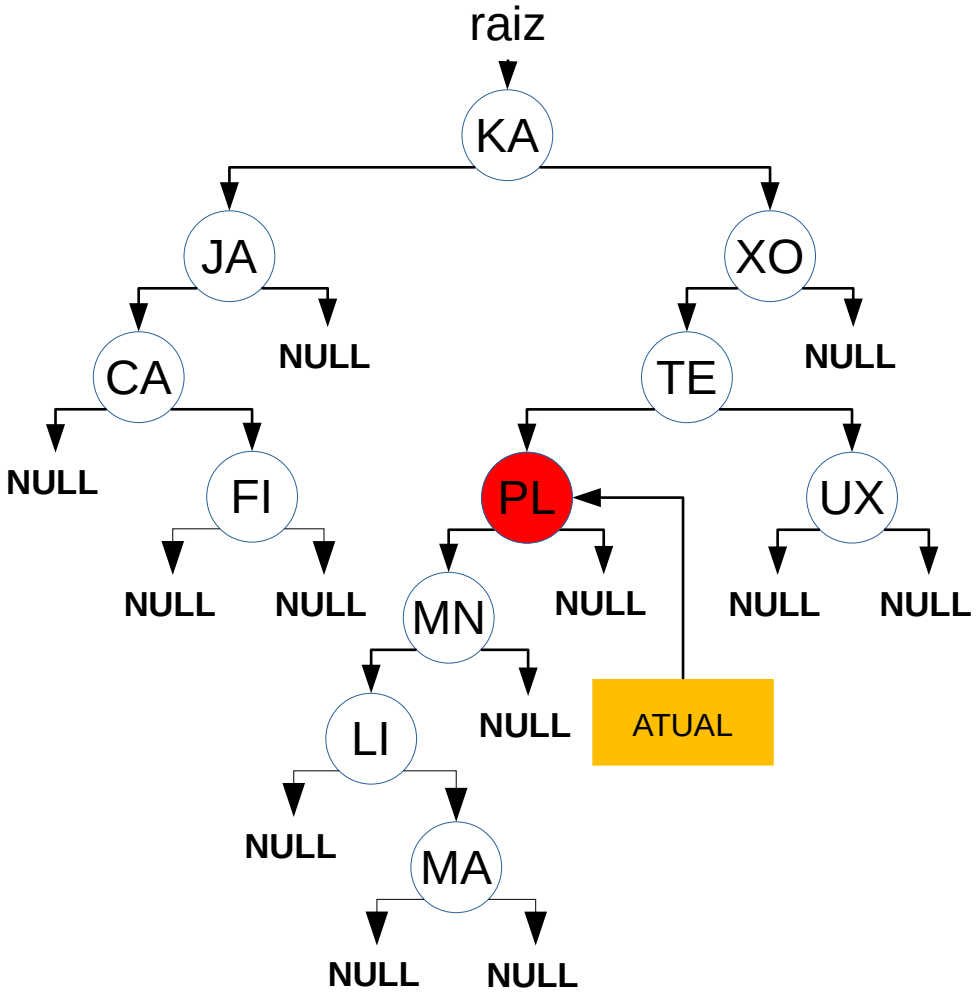

```
buscaAux(umValor):
atual ← raiz;
enquanto (atual ≠ NULL) {
     se (atual.valor = umValor) {
          retorna atual;
     } senão se (atual.valor > umValor) {
          atual ← atual->esquerdo;
     } senão {
          atual ← atual.direito;
     }
}
retorna atual;
```

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;
```

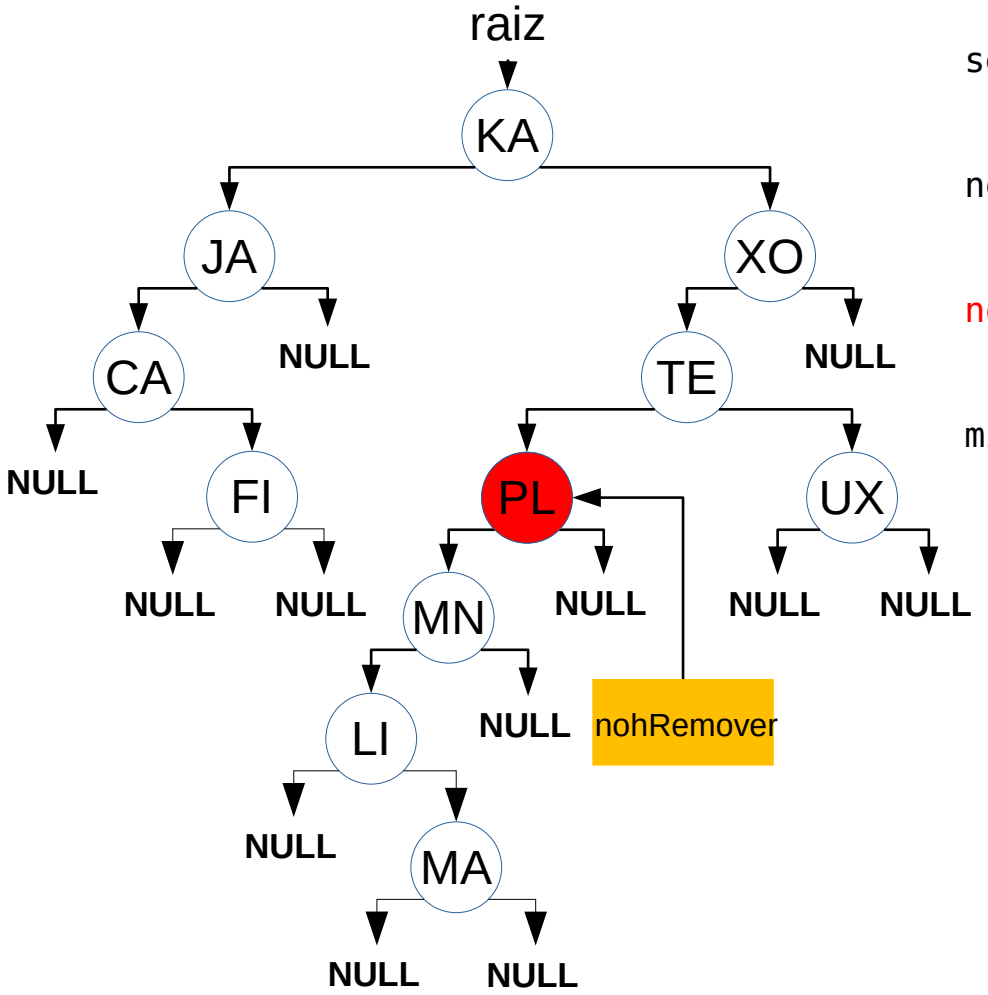
Remover PL CA TE



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;
```

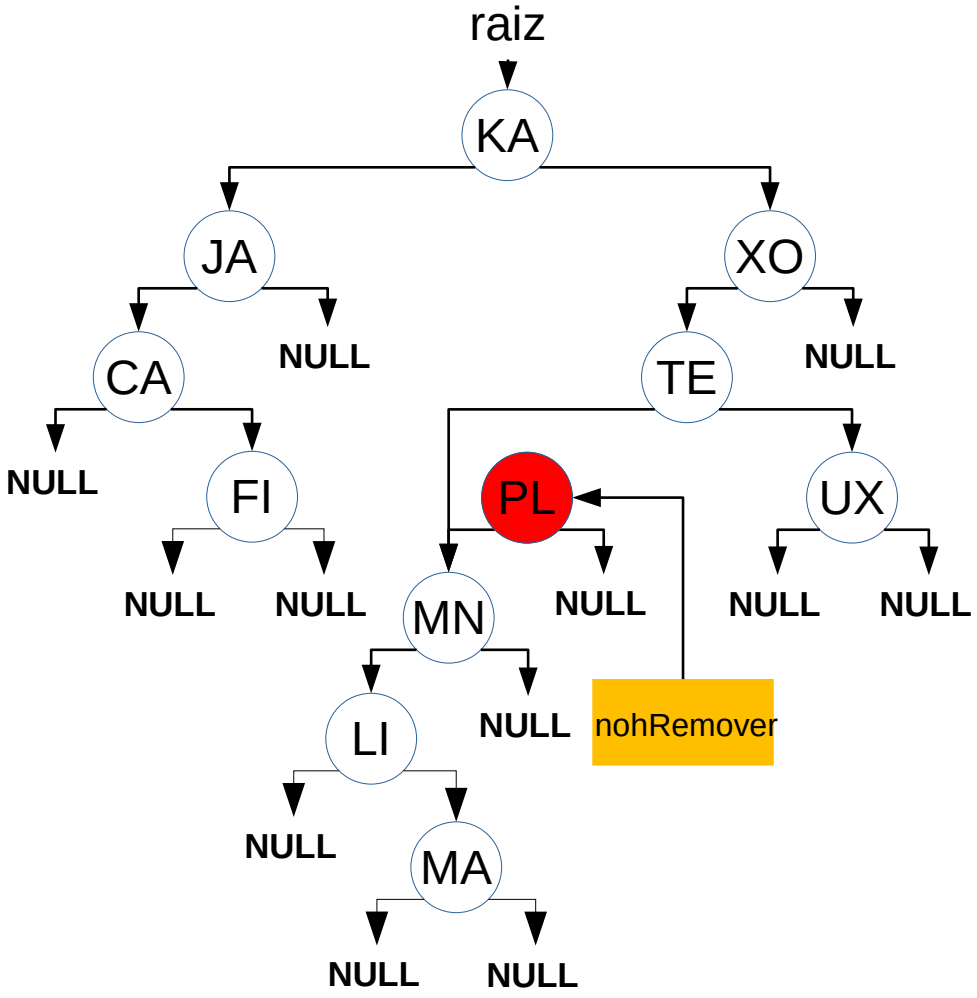
Remover PL CA TE



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);
```

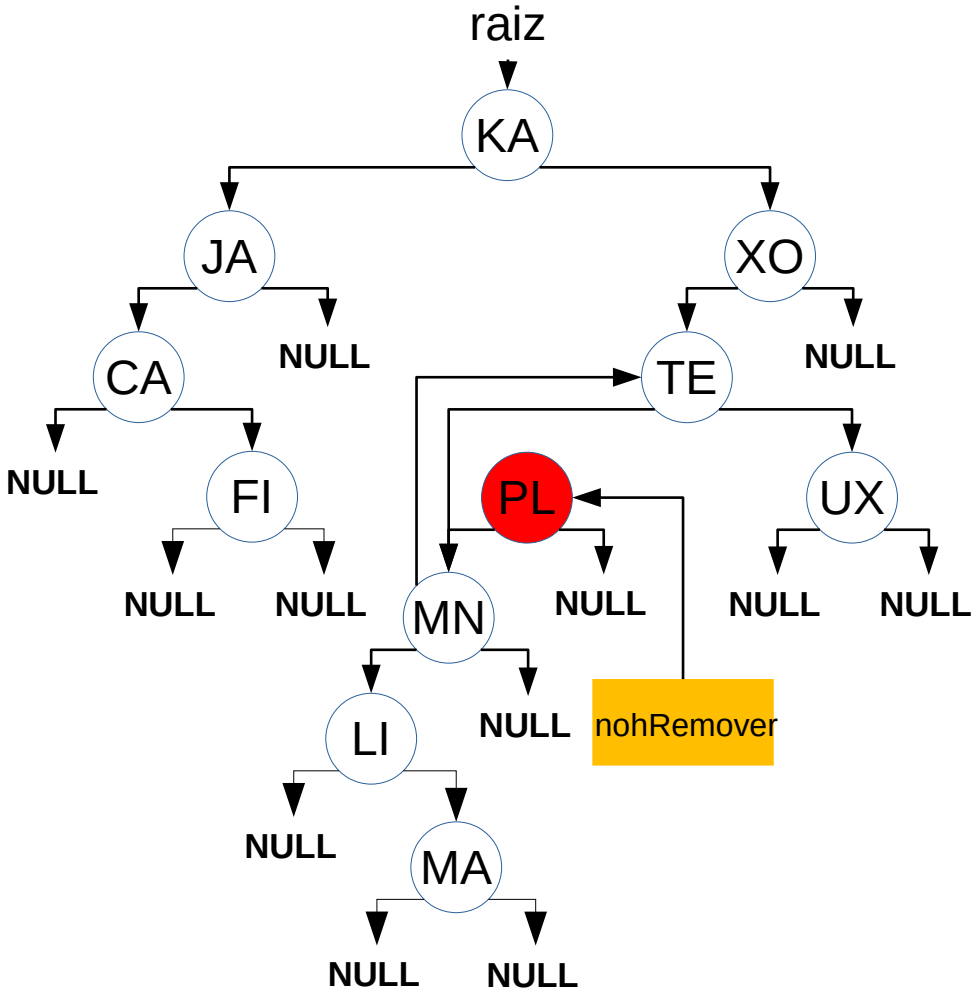
Remover PL CA TE



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

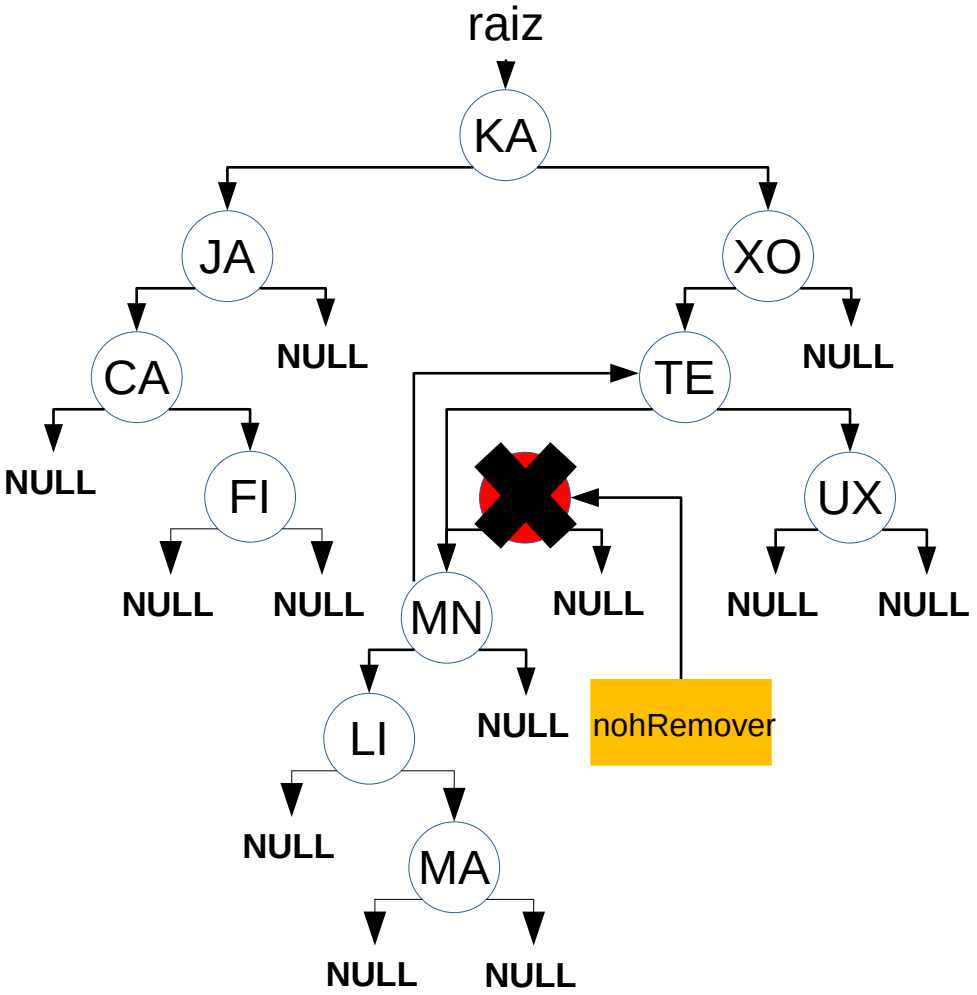


transplanta(antigo, novo):

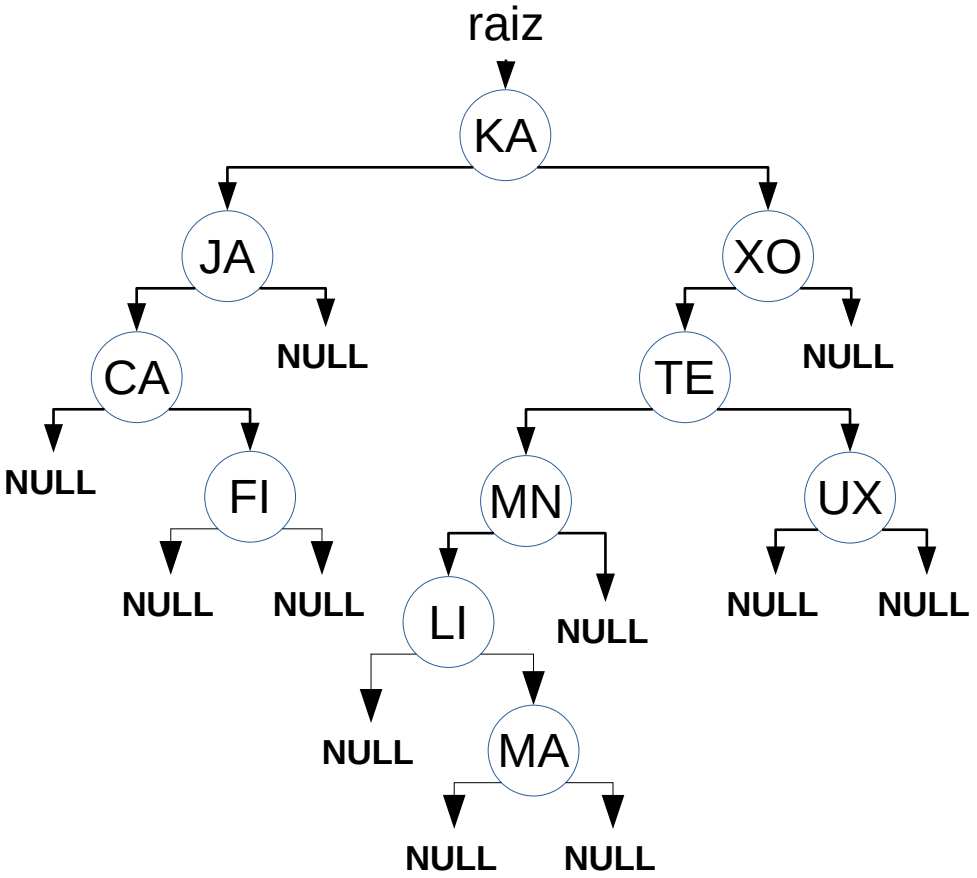
```
se (raiz = antigo) {
     raiz ← novo;
} senão se (antigo = antigo.pai.esquerdo) {
     antigo.pai.esquerdo ← novo;
} senão { // antigo = antigo.pai.direito
     antigo.pai.direito ← novo;
}
se (novo ≠ NULO) {
     novo.pai ← antigo.pai;
}
```

Remover PL CA TE

remove(umValor):
```
apagar(nohRemover);
```



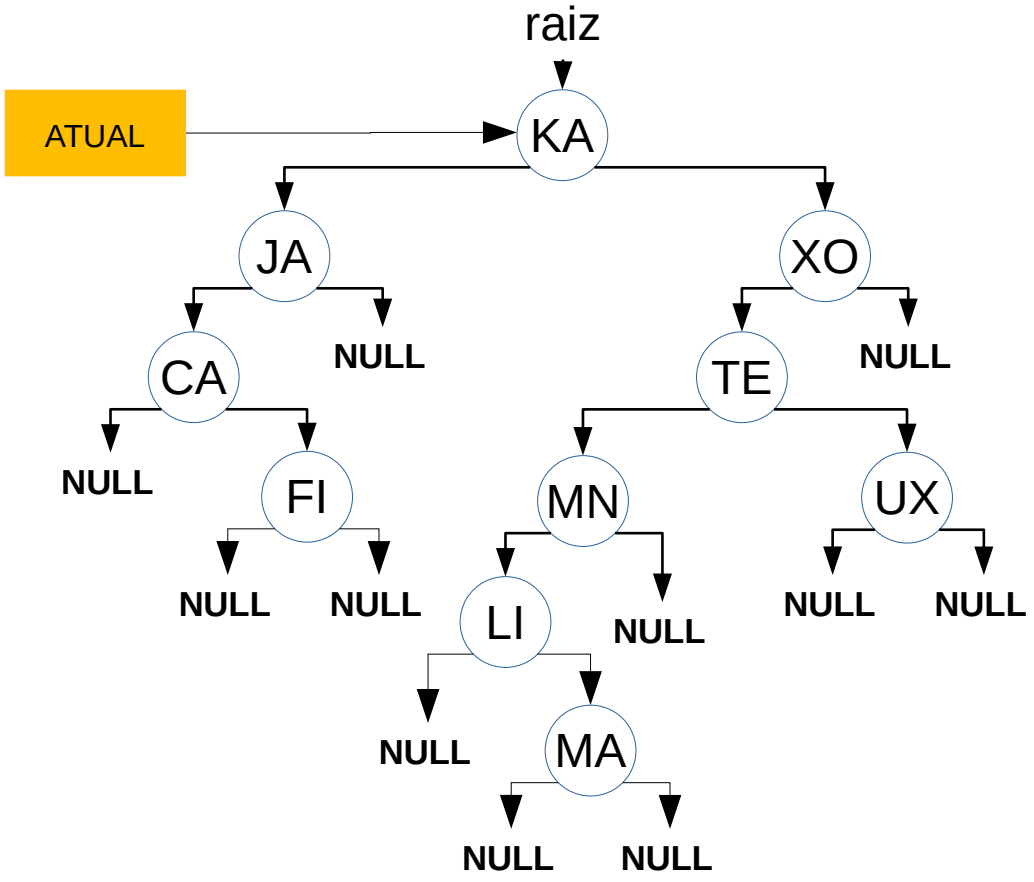
Remover PL CA TE



remove(umValor):

```
nohRemover ← buscaAux(umValor);
```

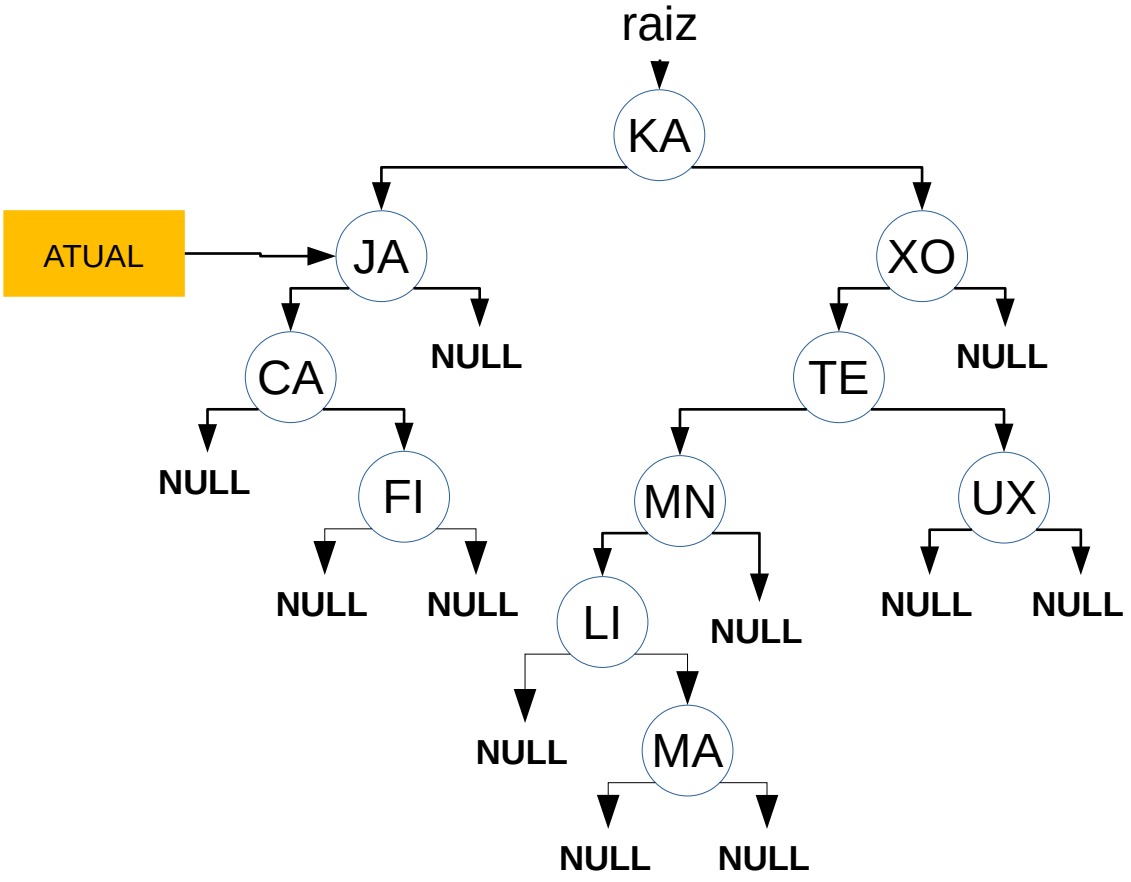
Remover PL CA TE



buscaAux(umValor):

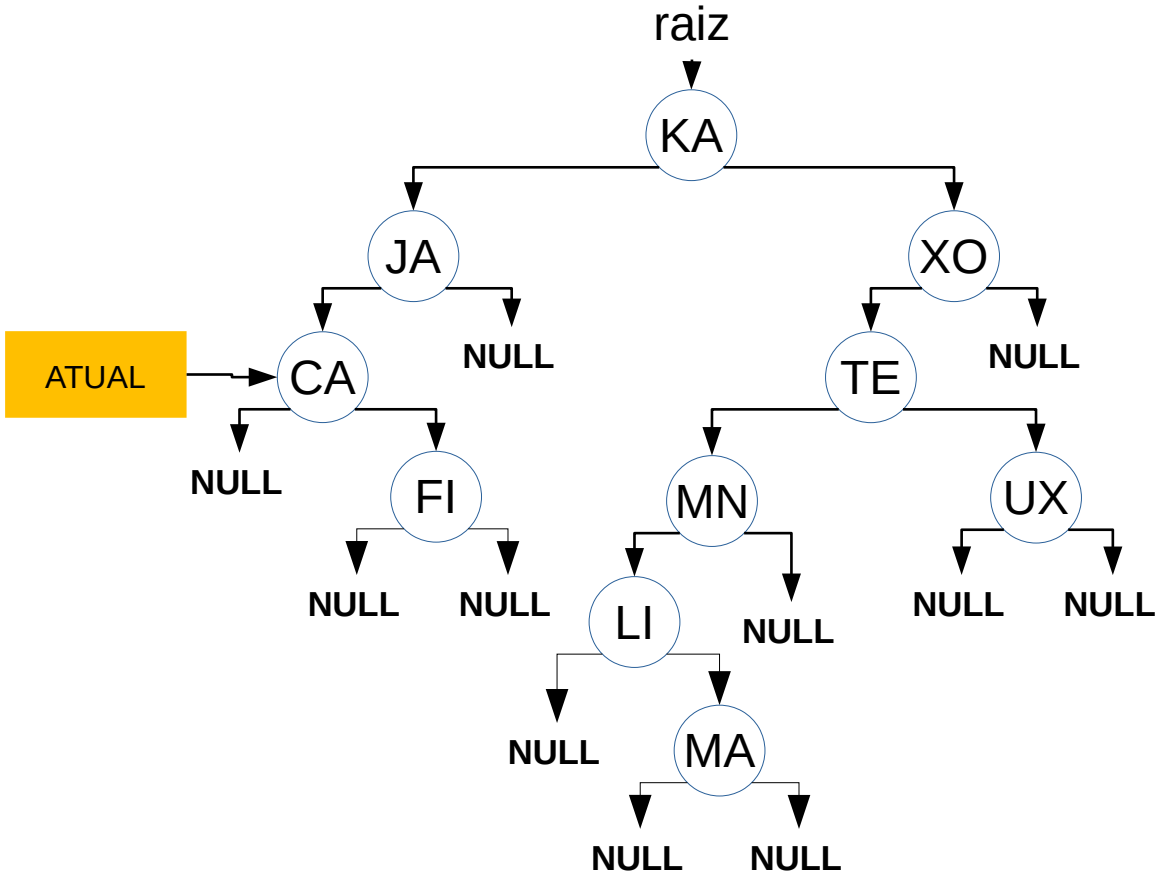
```
atual ← raiz;
enquanto (atual ≠ NULL) {
     se (atual.valor = umValor) {
          retorna atual;
     } senão se (atual.valor > umValor) {
          atual ← atual->esquerdo;
     } senão {
          atual ← atual.direito;
     }
}
retorna atual;
```

Remover PL CA TE



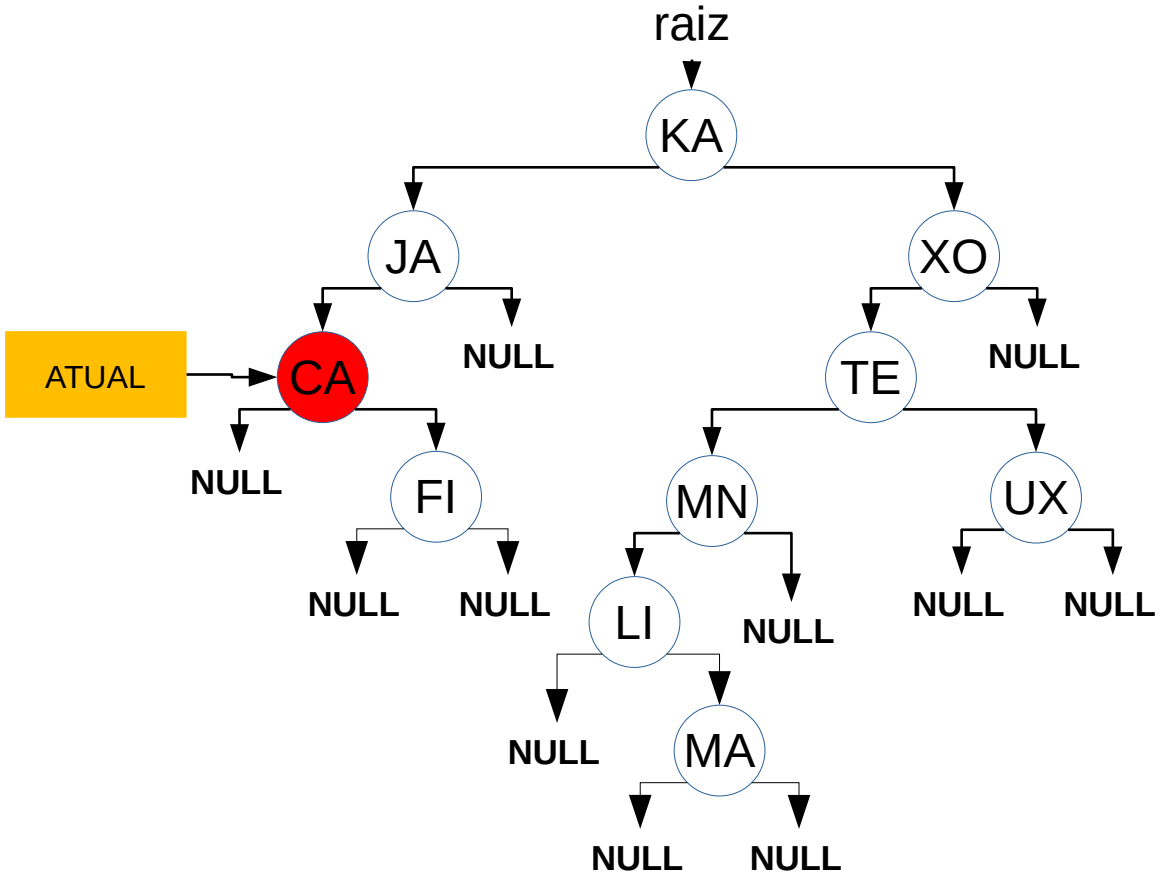

```
buscaAux(umValor):
atual ← raiz;
enquanto (atual ≠ NULL) {
     se (atual.valor = umValor) {
          retorna atual;
     } senão se (atual.valor > umValor) {
          atual ← atual->esquerdo;
     } senão {
          atual ← atual.direito;
     }
}
retorna atual;
```

Remover PL CA TE



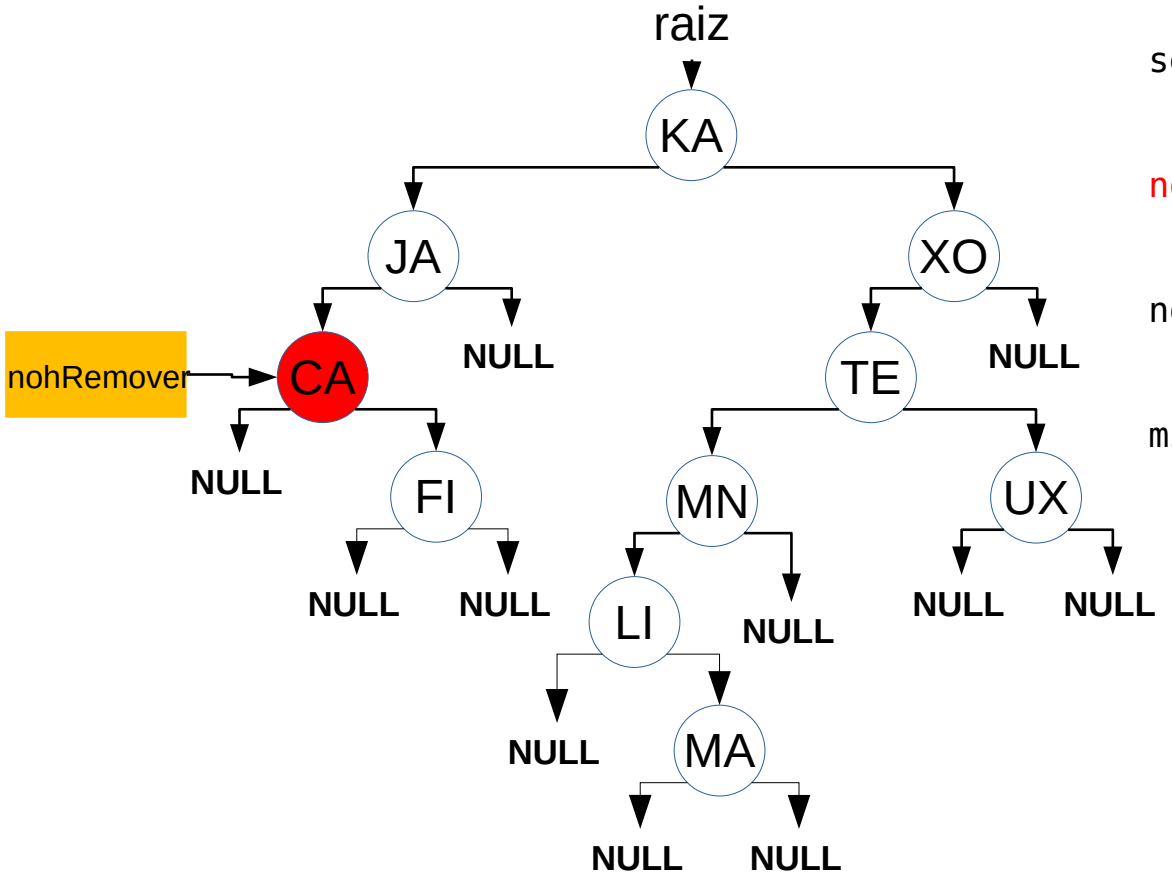

```
buscaAux(umValor):
atual ← raiz;
enquanto (atual ≠ NULL) {
     se (atual.valor = umValor) {
          retorna atual;
     } senão se (atual.valor > umValor) {
          atual ← atual->esquerdo;
     } senão {
          atual ← atual.direito;
     }
}
retorna atual;
```

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;
```

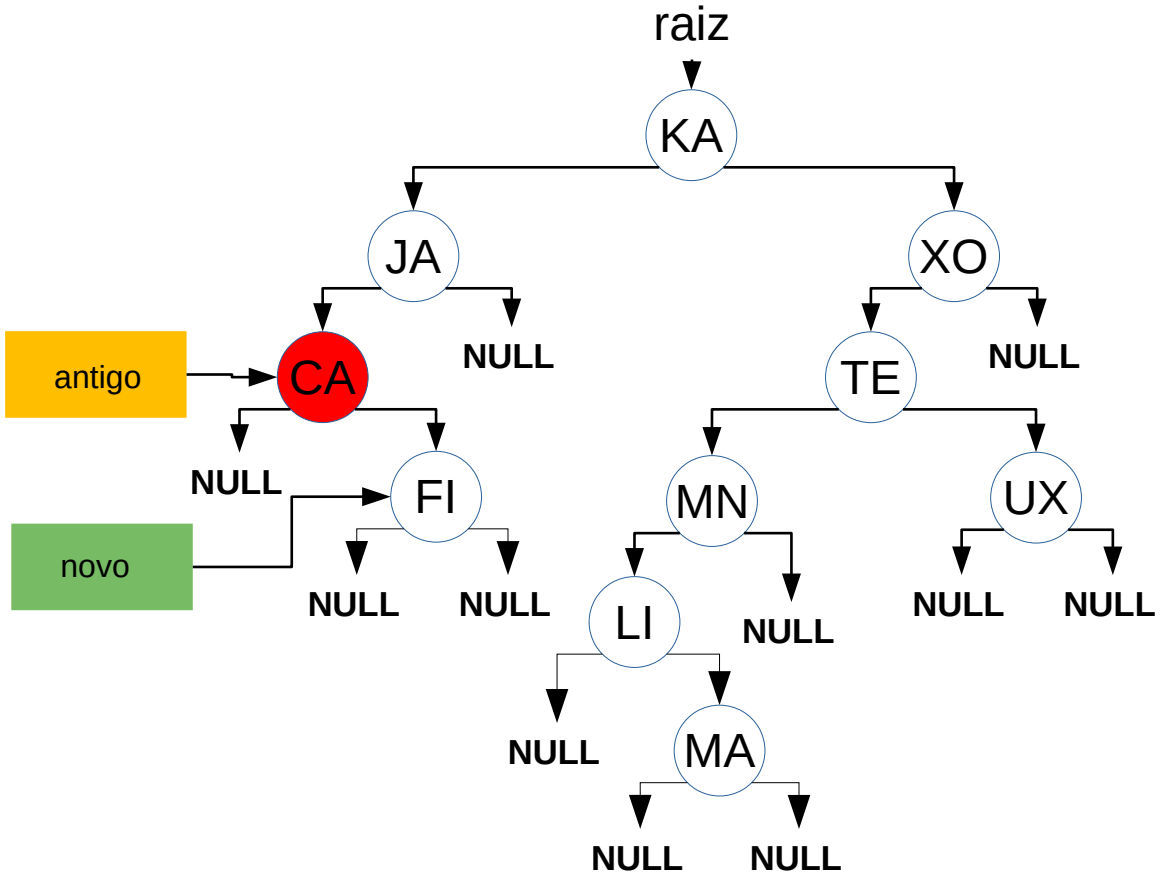
Remover PL CA TE



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);
```

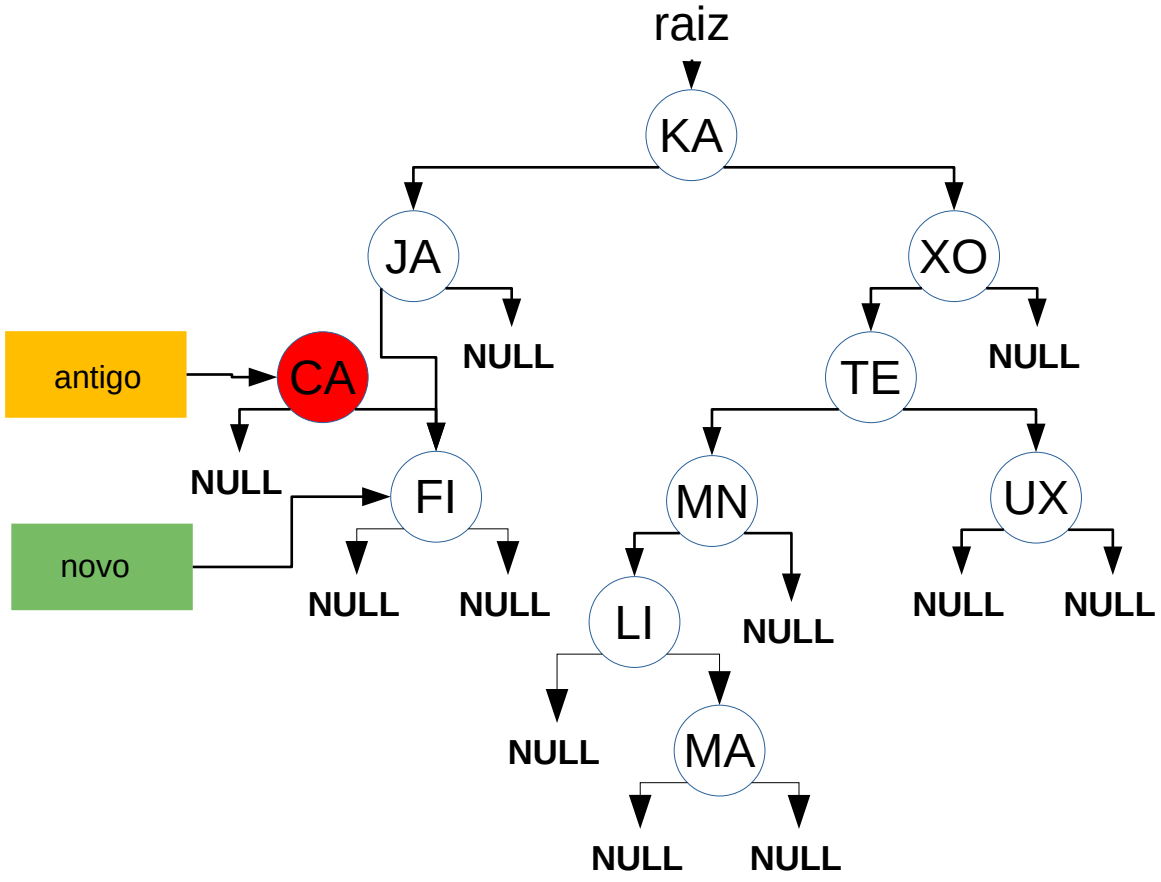
Remover PL CA TE



transplanta(antigo, novo):

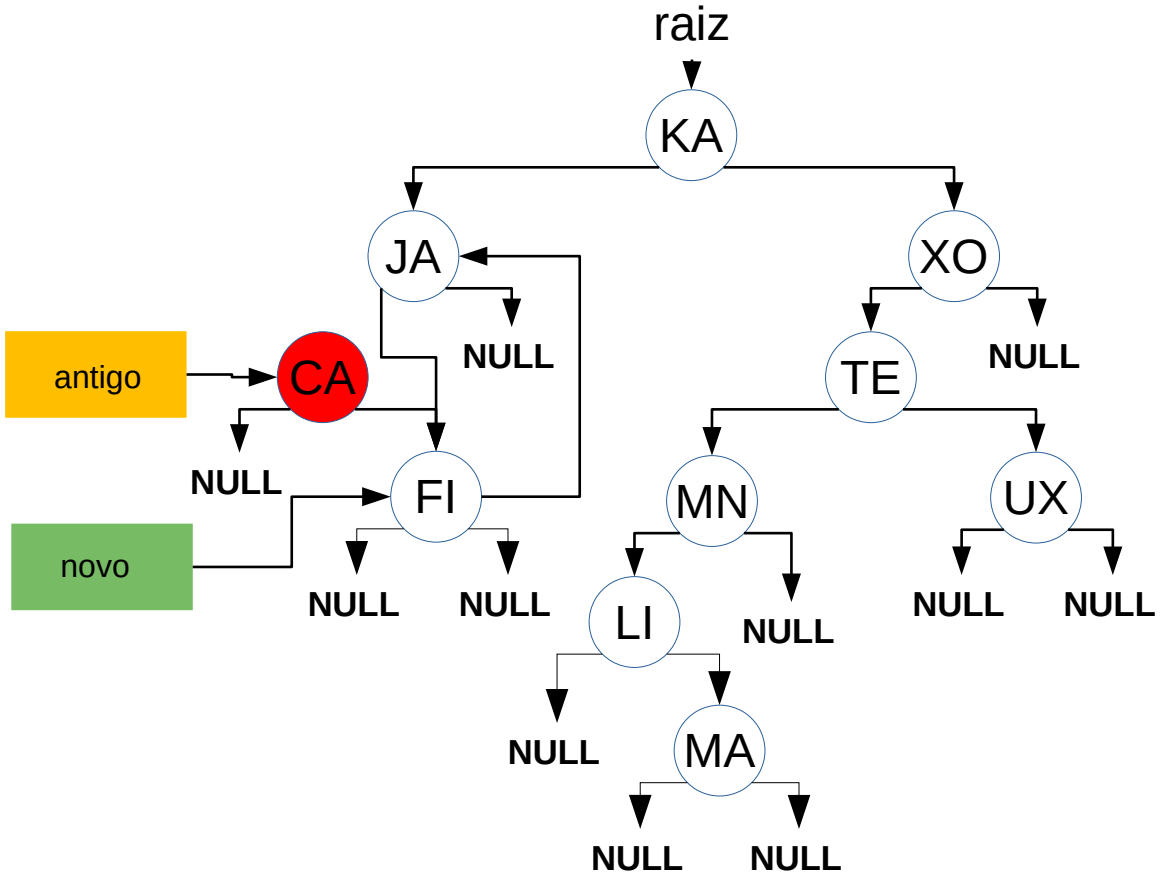
```
se (raiz = antigo) {
     raiz ← novo;
} senão se (antigo = antigo.pai.esquerdo) {
     antigo.pai.esquerdo ← novo;
} senão { // antigo = antigo.pai.direito
     antigo.pai.direito ← novo;
}
se (novo ≠ NULO) {
     novo.pai ← antigo.pai;
}
```

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

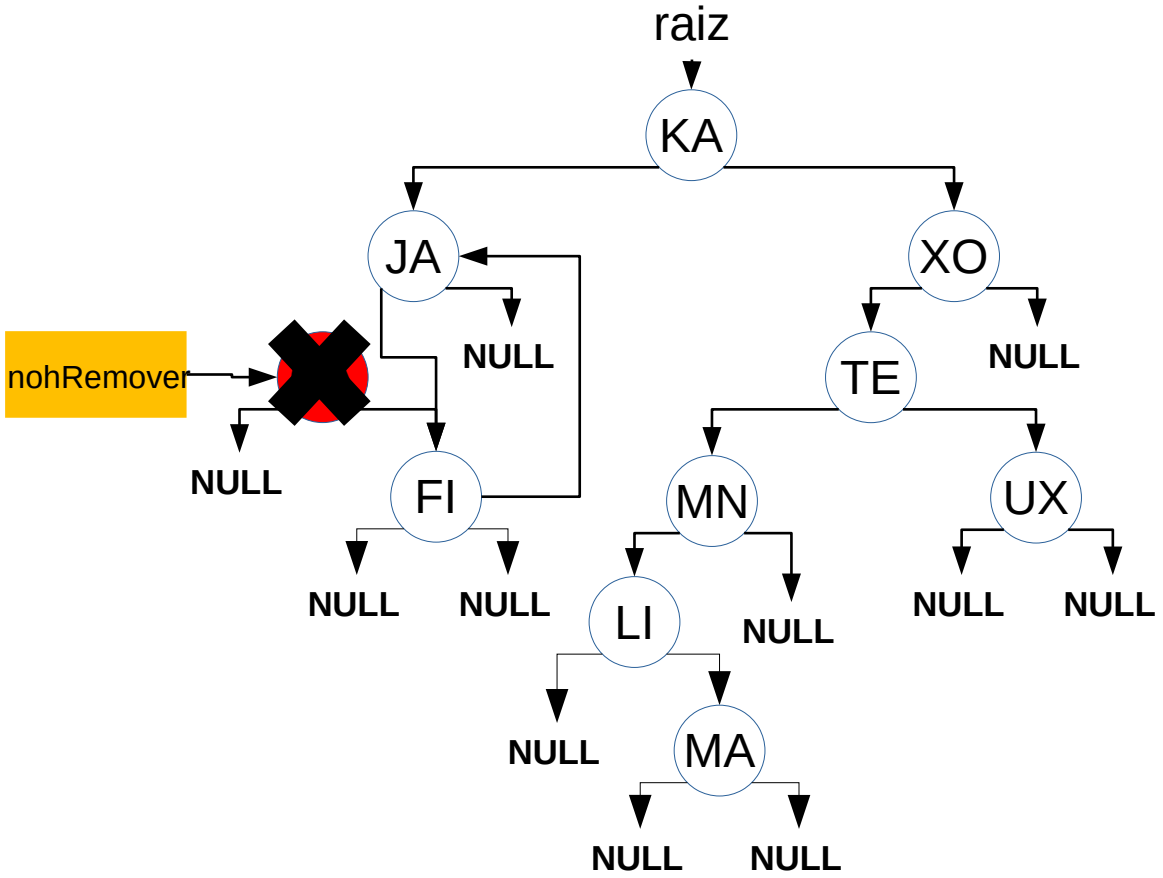


transplanta(antigo, novo):

```
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

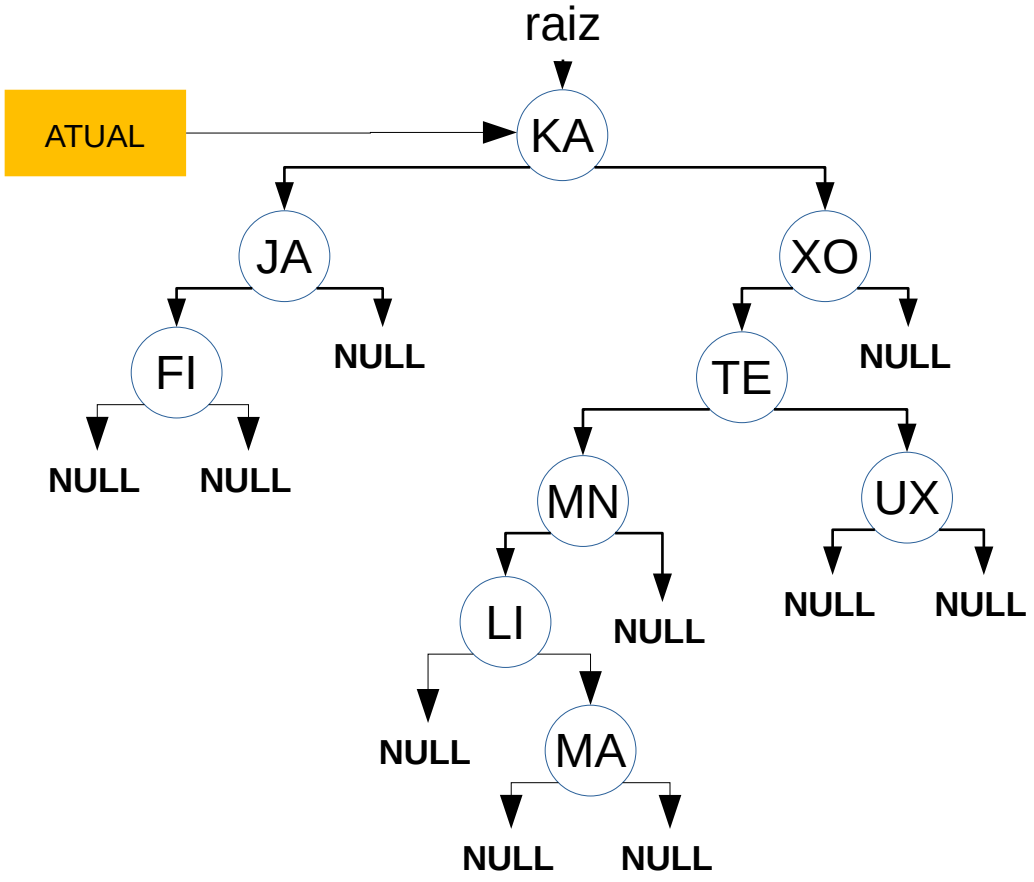
PL CA TE



remove(umValor):

```
apagar(nohRemover);
```

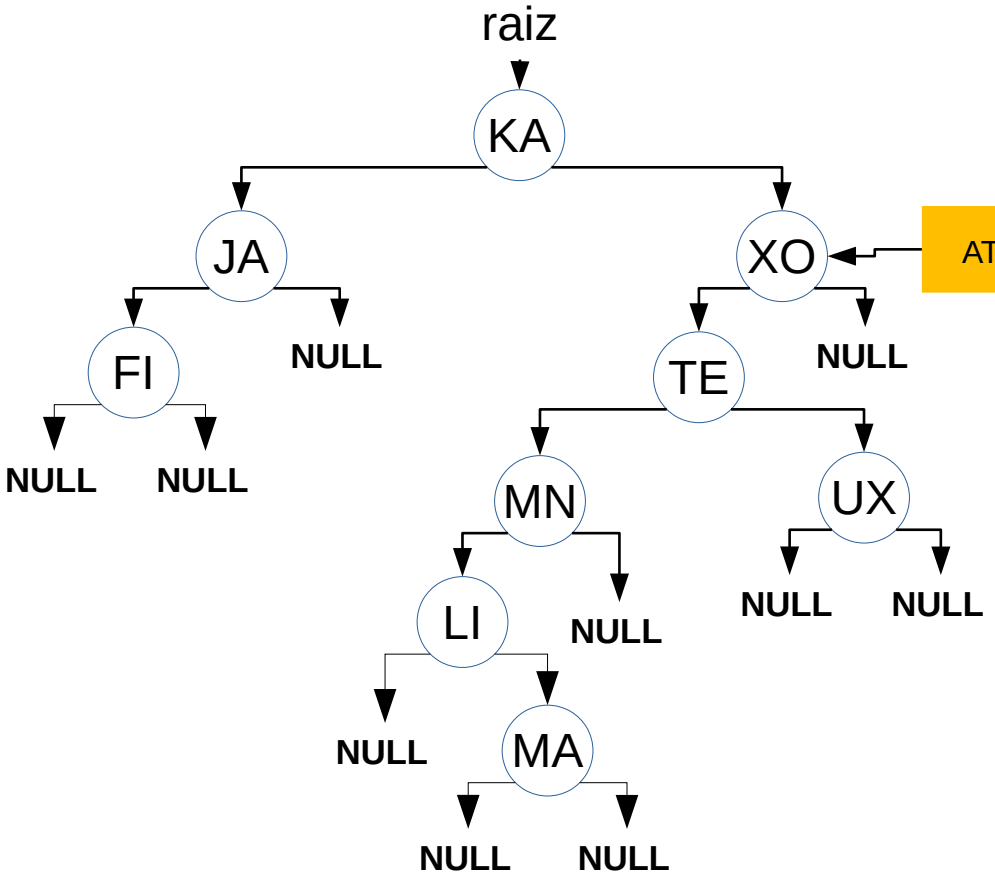
Remover PL CA TE



buscaAux(umValor):

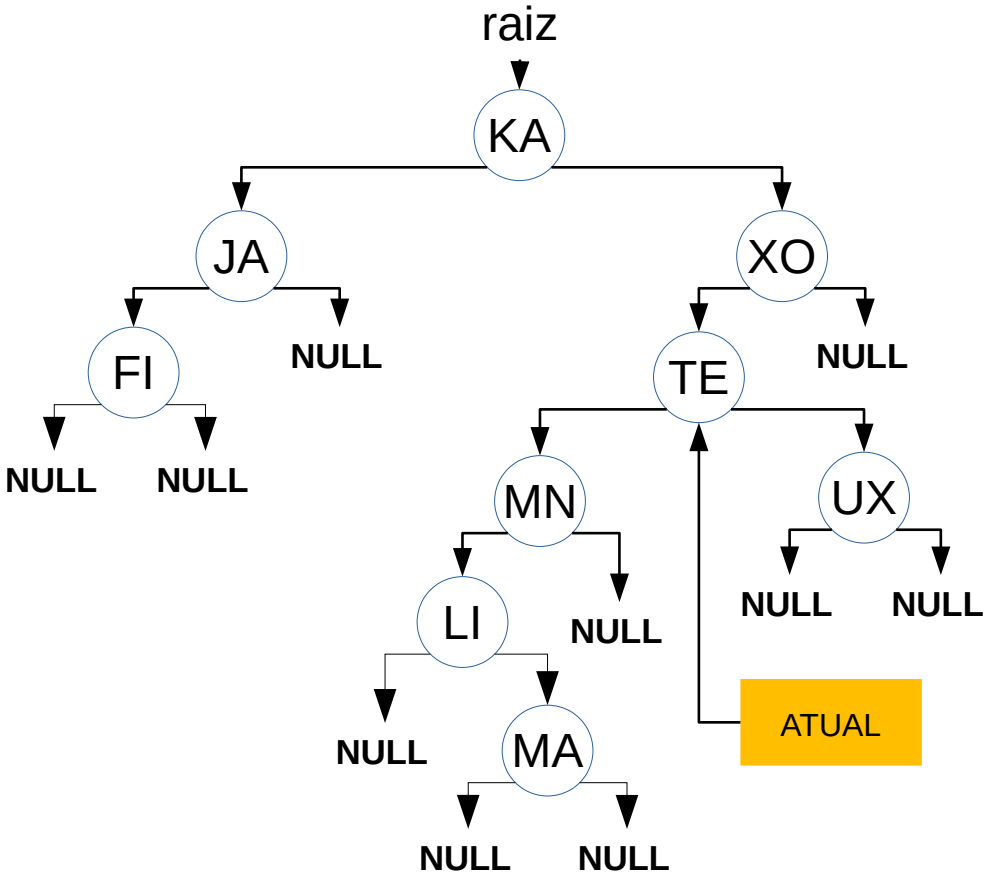
```
atual ← raiz;
enquanto (atual ≠ NULL) {
     se (atual.valor = umValor) {
          retorna atual;
     } senão se (atual.valor > umValor) {
          atual ← atual->esquerdo;
     } senão {
          atual ← atual.direito;
     }
}
retorna atual;
```

Remover PL CA TE



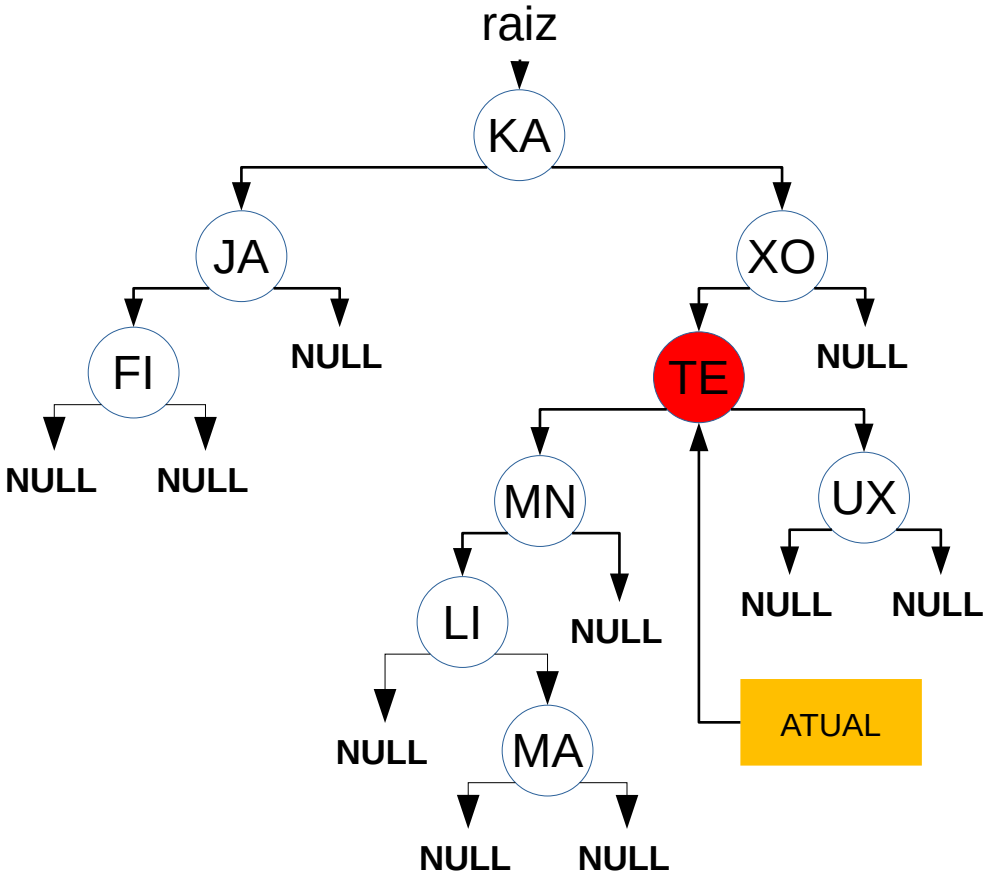

```
buscaAux(umValor):
atual ← raiz;
enquanto (atual ≠ NULL) {
     se (atual.valor = umValor) {
          retorna atual;
     } senão se (atual.valor > umValor) {
          atual ← atual->esquerdo;
     } senão {
          atual ← atual.direito;
     }
}
retorna atual;
```

Remover PL CA TE



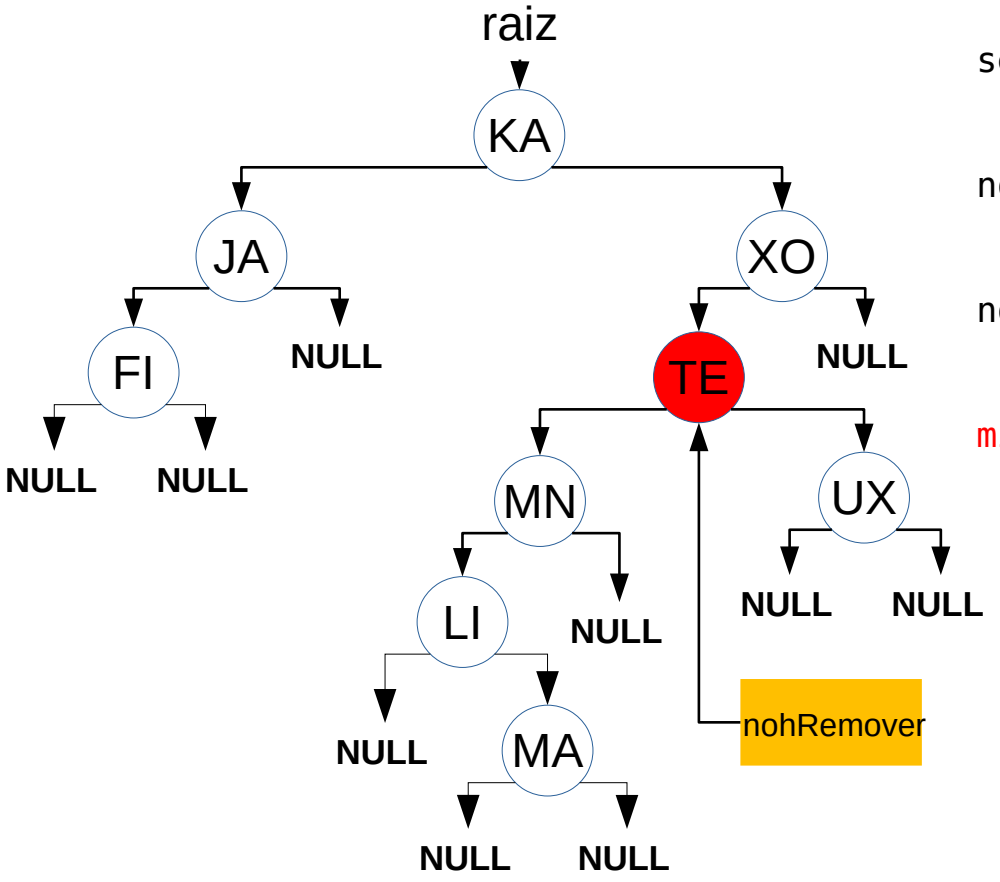

```
buscaAux(umValor):
atual ← raiz;
enquanto (atual ≠ NULL) {
     se (atual.valor = umValor) {
          retorna atual;
     } senão se (atual.valor > umValor) {
          atual ← atual->esquerdo;
     } senão {
          atual ← atual.direito;
     }
}
retorna atual;
```

Remover PL CA TE



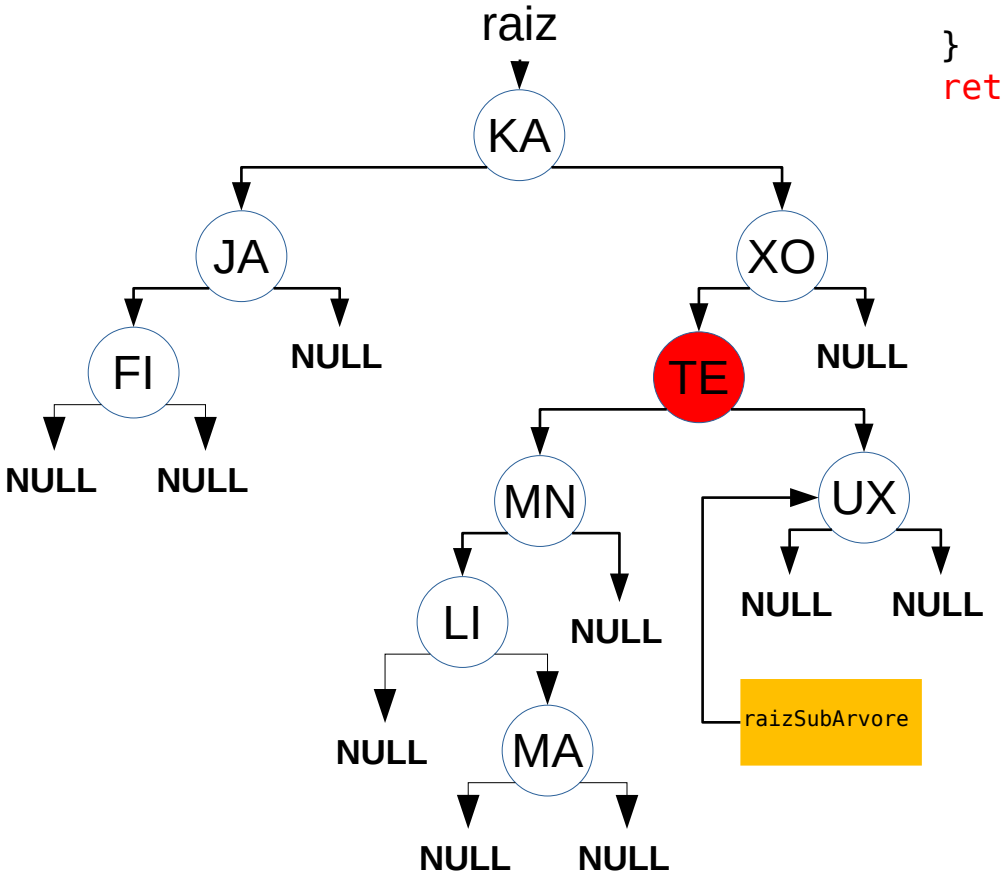

```
buscaAux(umValor):
atual ← raiz;
enquanto (atual ≠ NULL) {
     se (atual.valor = umValor) {
          retorna atual;
     } senão se (atual.valor > umValor) {
          atual ← atual->esquerdo;
     } senão {
          atual ← atual.direito;
     }
}
retorna atual;
```

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);
```

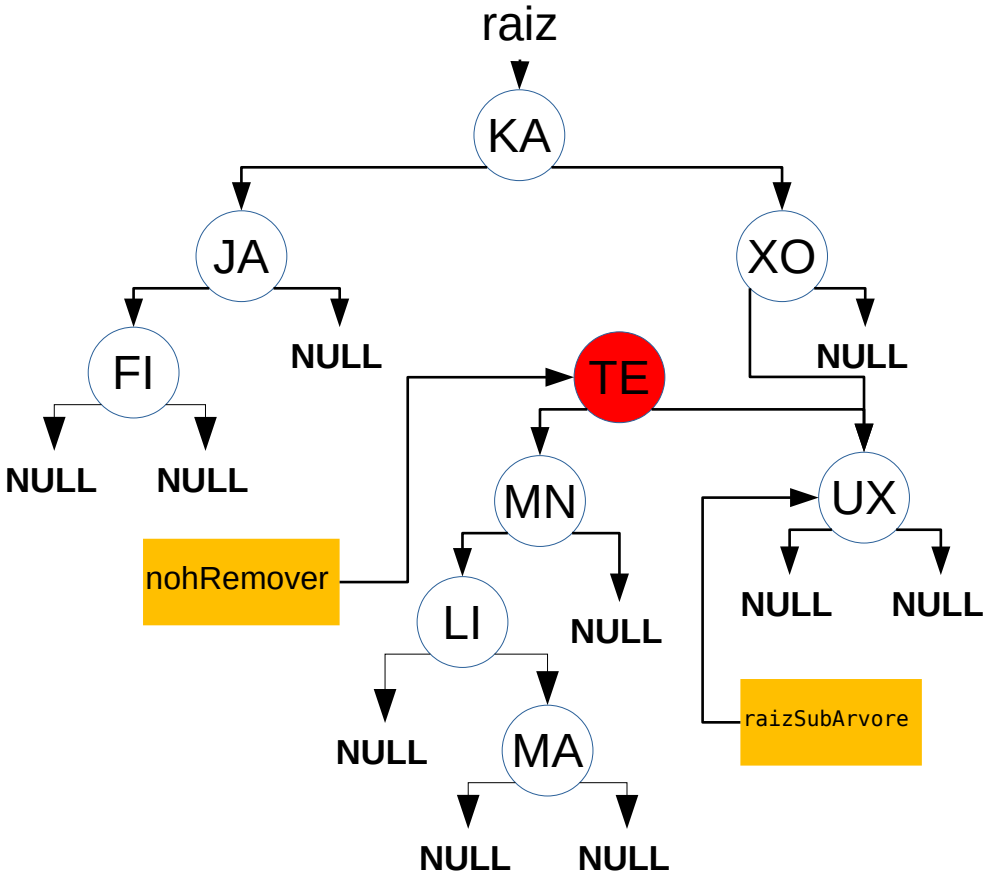
Remover PL CA TE



minimoAux(raizSubArvore):

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

Remover PL CA TE

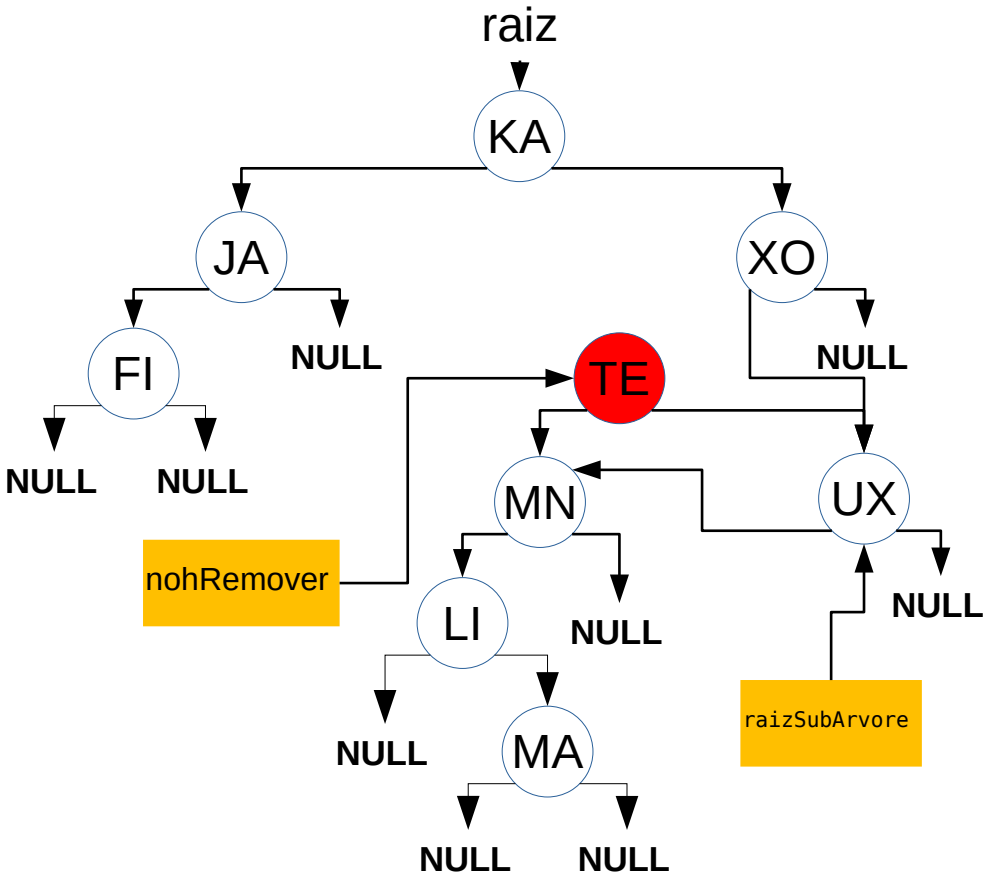


transplanta(antigo, novo):

```
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

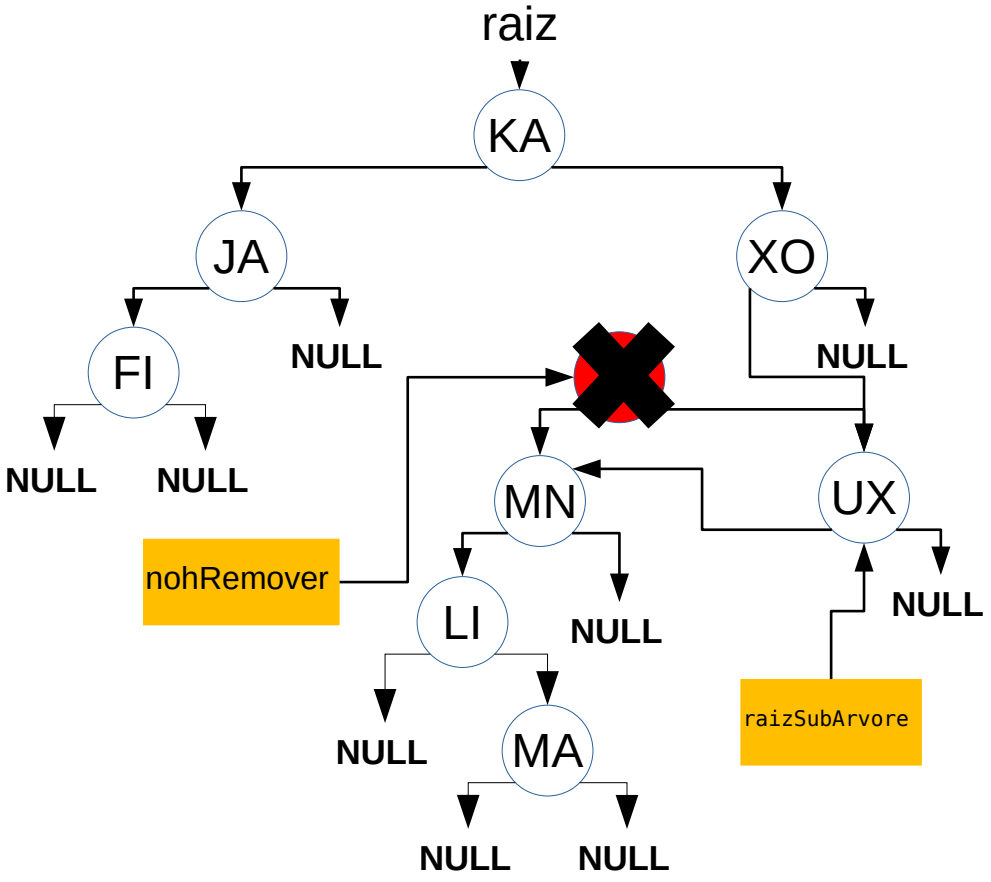
```
...sucessor.esquerdo ← nohRemover.esquerdo;

Sucessor.esquerdo.pai ← sucessor;
```



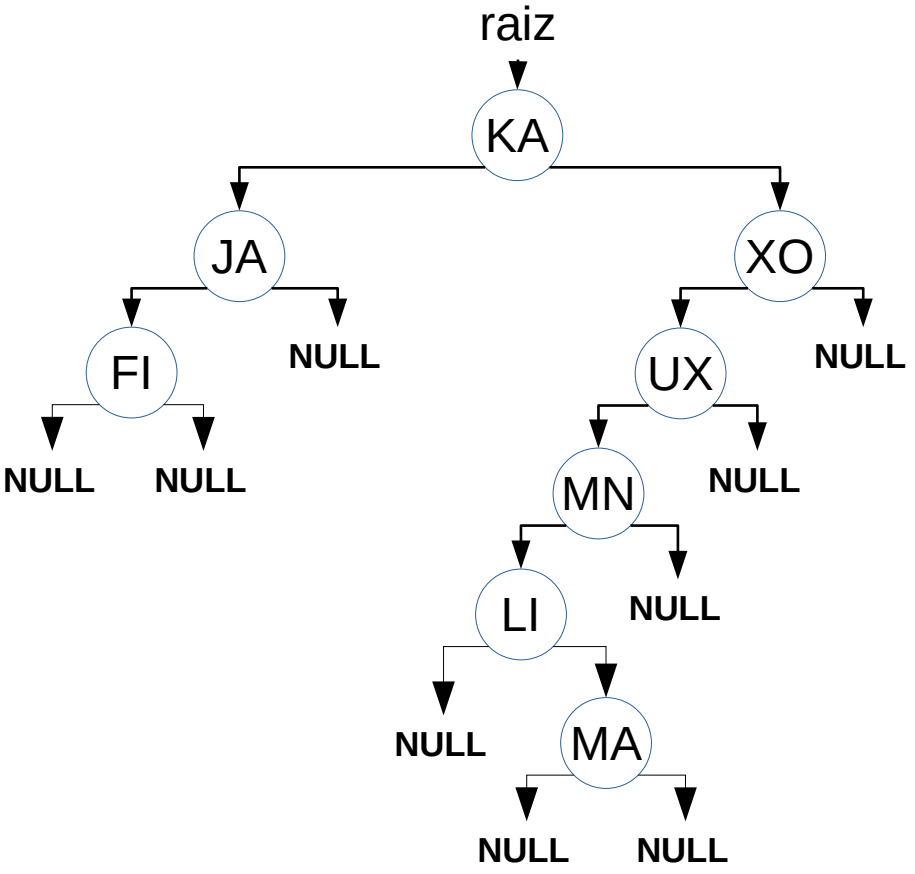
Remover PL CA TE

remove(umValor):

```
apagar(nohRemover);
```



Resultado:



remove(umValor):

```
apagar(nohRemover);
```