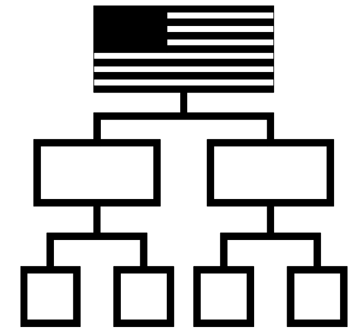
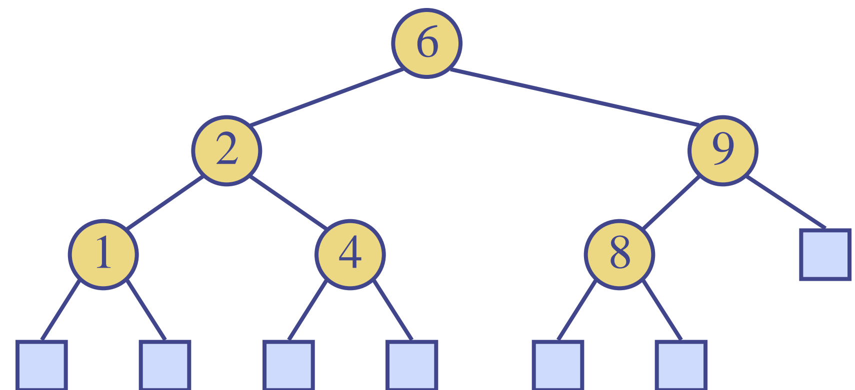
Árvore de pesquisa binária



- Uma árvore de pesquisa binária é uma árvore binária armazenando chaves (ou itens) em seus nós internos e satisfazendo a seguinte propriedade:
  - Seja u, v e w três nós tais que u é nó esquerdo de v e w é o nó direito. Temos $key(u) \leq key(v) \leq key(w)$
- Nós externos não armazenam itens (null)

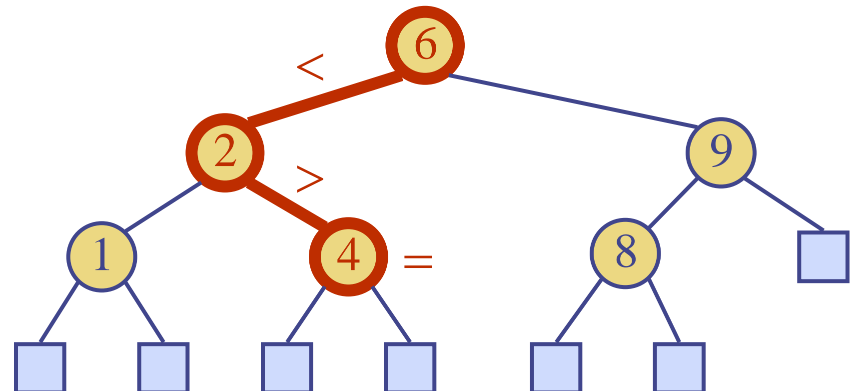
- Uma travessia em ordem visita as chaves em ordem crescente



Busca

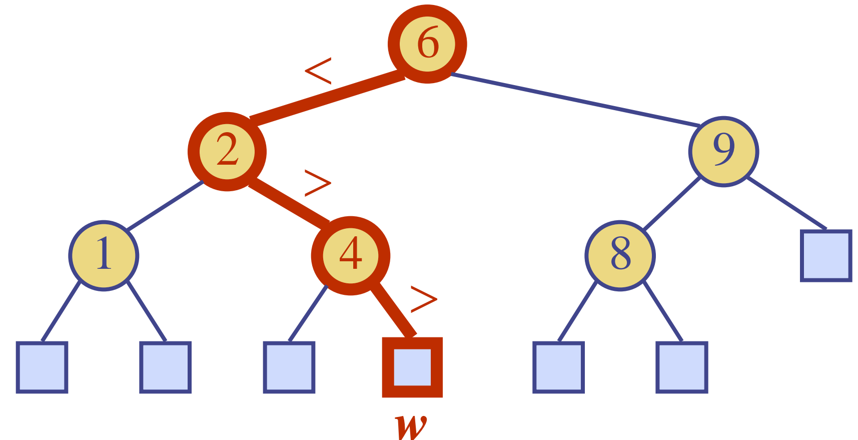
- Para procurar uma chave k, procuramos a partir da raiz comparando com a chave do nó.
- O próximo nó depende da comparação da chave pesquisada com a chave do nó atual
- Se chegar em uma folha e não encontrar a chave, retorna-se null
- Exemplo: find(4):
  - chama algoritmo TreeSearch(4,root)

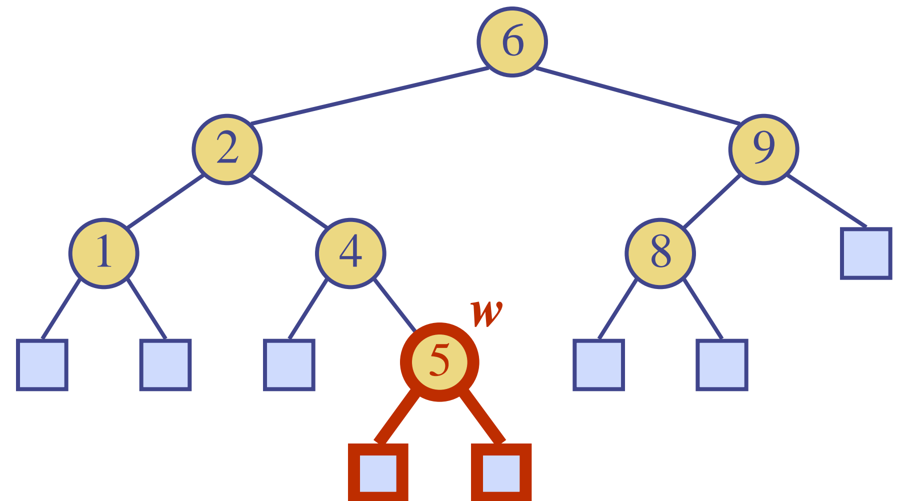
Algoritmo *TreeSearch*(k, v)
    se *T.isExternal* (v)
        retorne v
    se $k < key(v)$
        retorne *TreeSearch*$(k, T.left(v))$
    senão se $k = key(v)$
        retorne v
    senão $\{ k > key(v) \}$
        retorne *TreeSearch*$(k, T.right(v))$



inserção

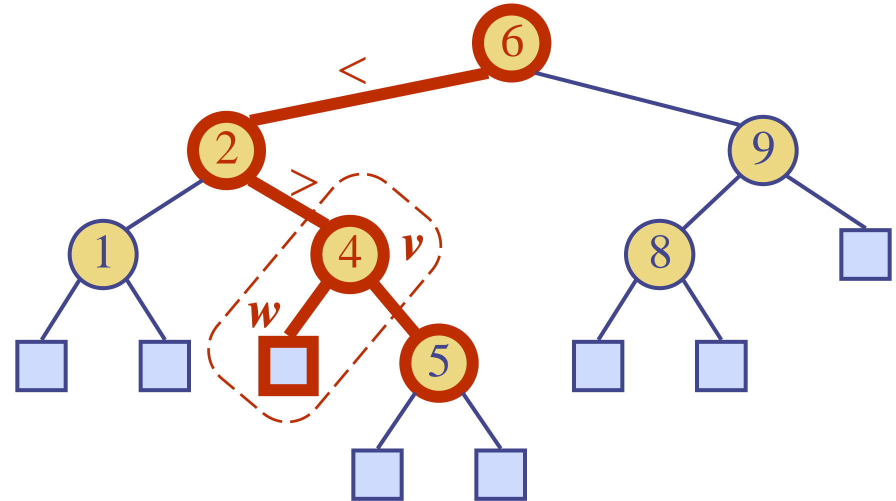
- Para executar inser(k, o), procura-se pela chave k
- Assumindo que k ainda não está na árvore, w será a folha encontrada pela busca
- Inserimos k no nó w
- Exemplo: inserir 5

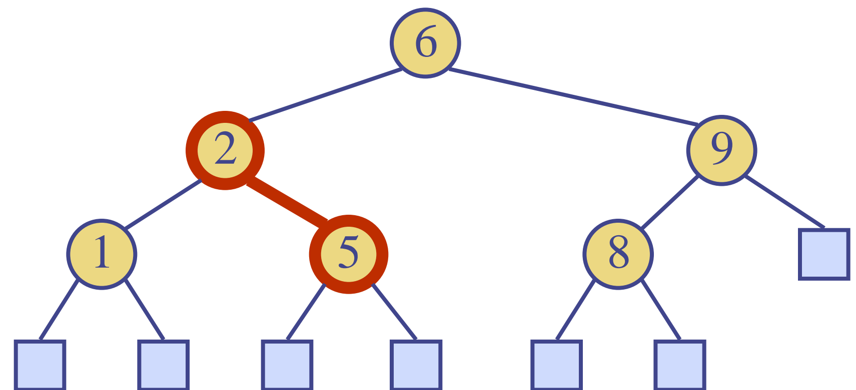




remoção

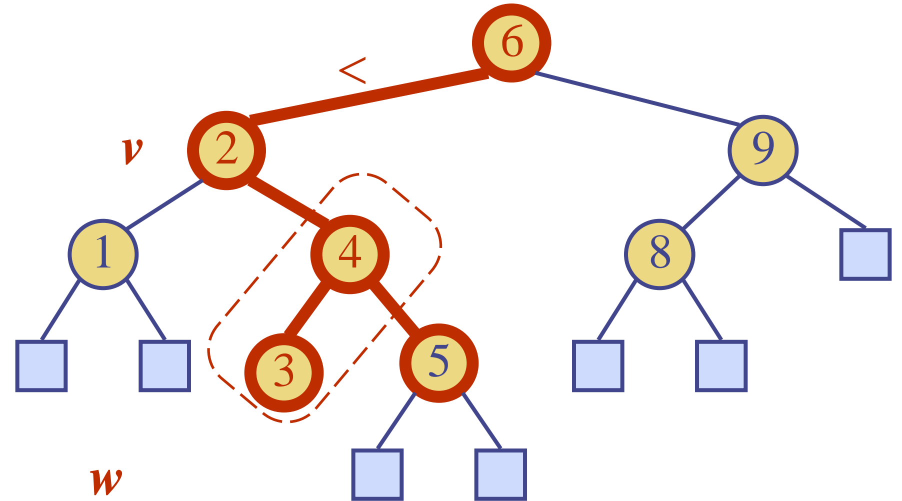
- Para executar remove(*4*), procuramos pela chave *4*

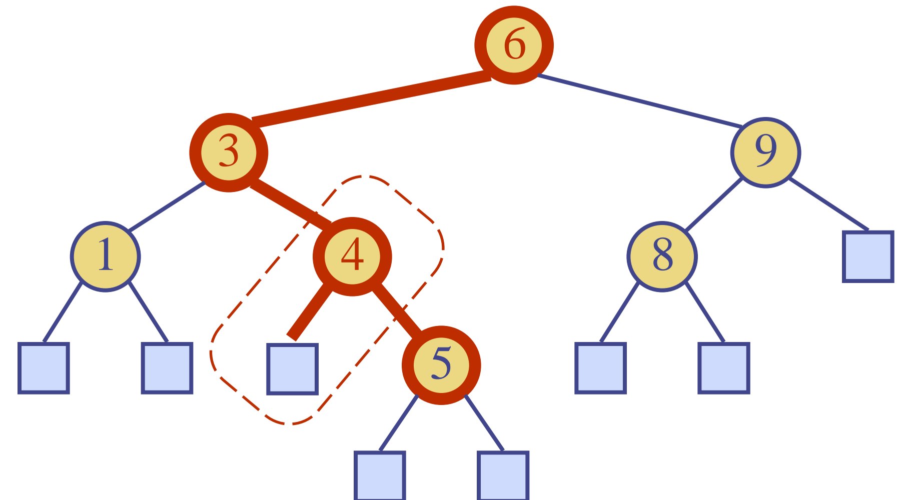




remoção

- Para executar remove(***2***), procuramos pela chave ***2***
- Fazemos um caminhamento em ordem na subárvore direita do nó 2 e até encontrar o primeiro nó





Desempenho

- Considere um dicionário com n itens, implementado com uma árvore binária de pesquisa de altura h
  - o espaço usado é $O(n)$
  - métodos find, insert e remove executam em tempo $O(h)$
- A altura h é $O(n)$ no pior caso e $O(\log n)$ no melhor caso