Searches

→ attempt to locate a particular key in a BST by viewing it as a decision tree

→ in the algorithm, if k occurs in a subtree rooted at p, a call to TreeSearch results in the position at which the key is found

```
Algorithm TreeSearch(T, p, k):
  if k == p.key() then
    return p
  else if k < p.key() and T.left(p) is not None
    return TreeSearch(T, T.left(p), k)
  else if k > p.key() and T.right(p) is not None
    return TreeSearch(T, T.right(p), k)
  return p
```

Insertion

→ the map command M[k] = v begins with a search for k

  → if found, reassign

  → otherwise, insert a new node for the item into underlying tree T in place of empty subtree that was reached at end of failed search

```
Algorithm TreeInsert(T, k, v):
  p = TreeSearch(T, T.root(), k)
  if k = p.key then
    set p value to v
  else if k < p.key() then
    add node with item (k, v) as left child of p
  else
    add node with item (k, v) as right child of p
```