Searching

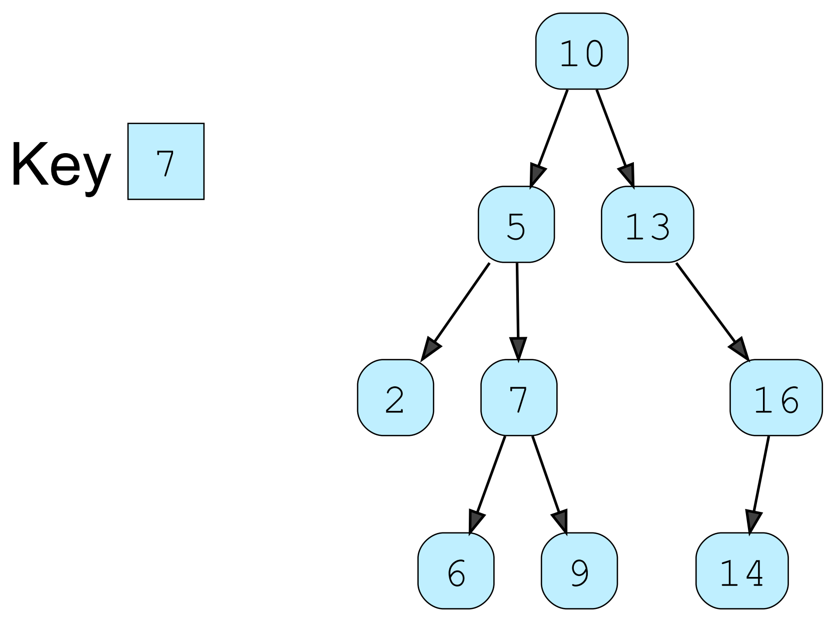
In this lesson, we'll see how to search a key in a BST.

Algorithm

Starting from the root, we compare the key. If the key is smaller than the root's key, we search in the left subtree. Similarly, we search in the right subtree if the key is greater than the root's key.

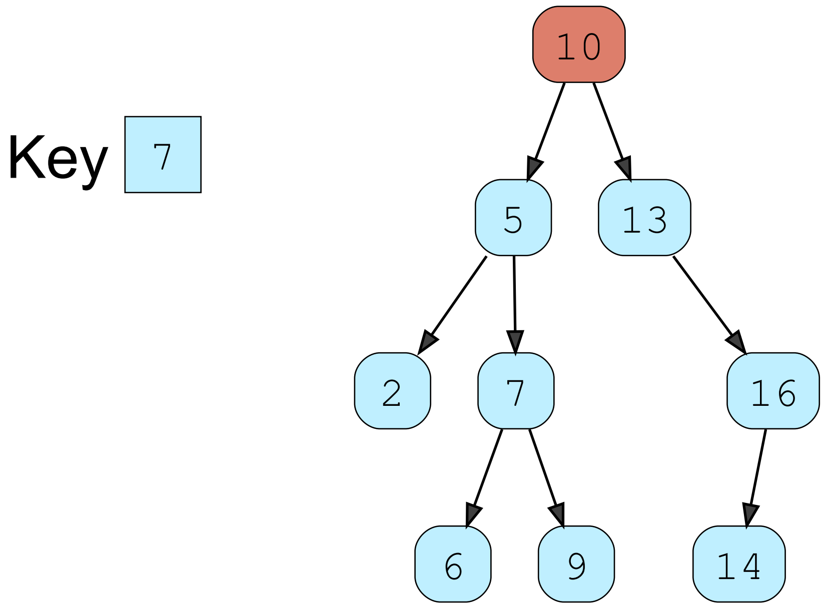
Time Complexity: In the worst case, the key could be the deepest node. So, the insertion time complexity is $O(height)$ or $O(N)$.

Visualization



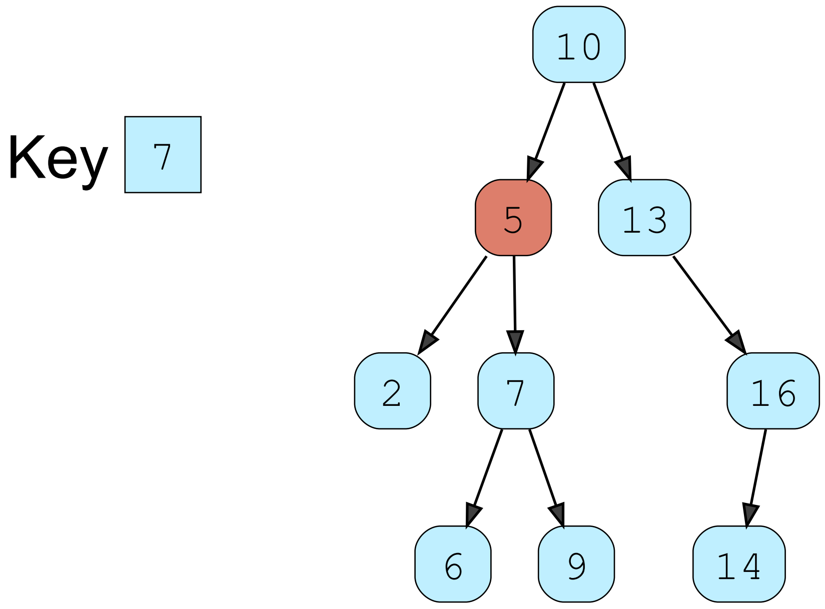
Searching for key X =7

1 of 5



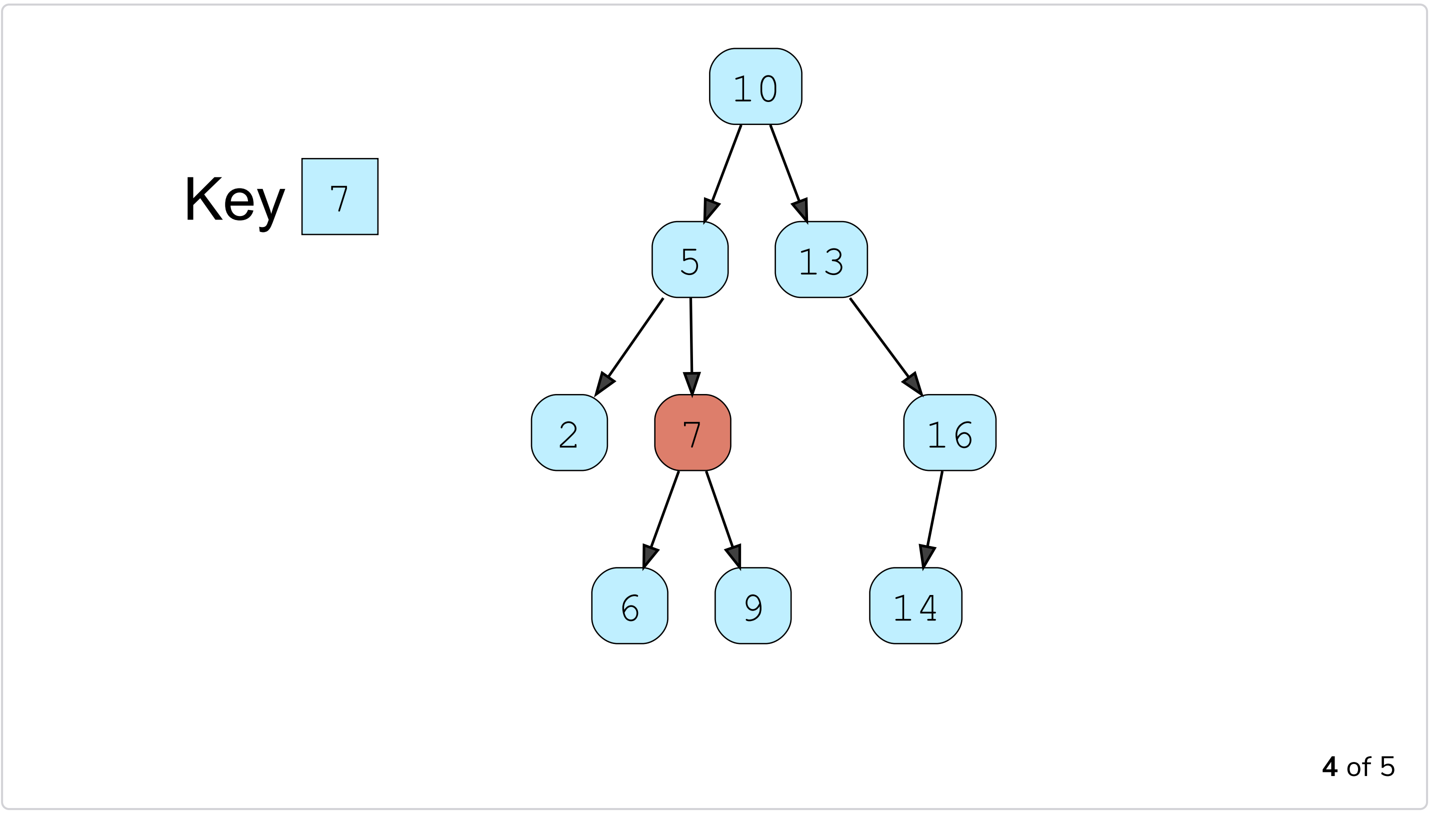
X < 10, so we go to left subtree

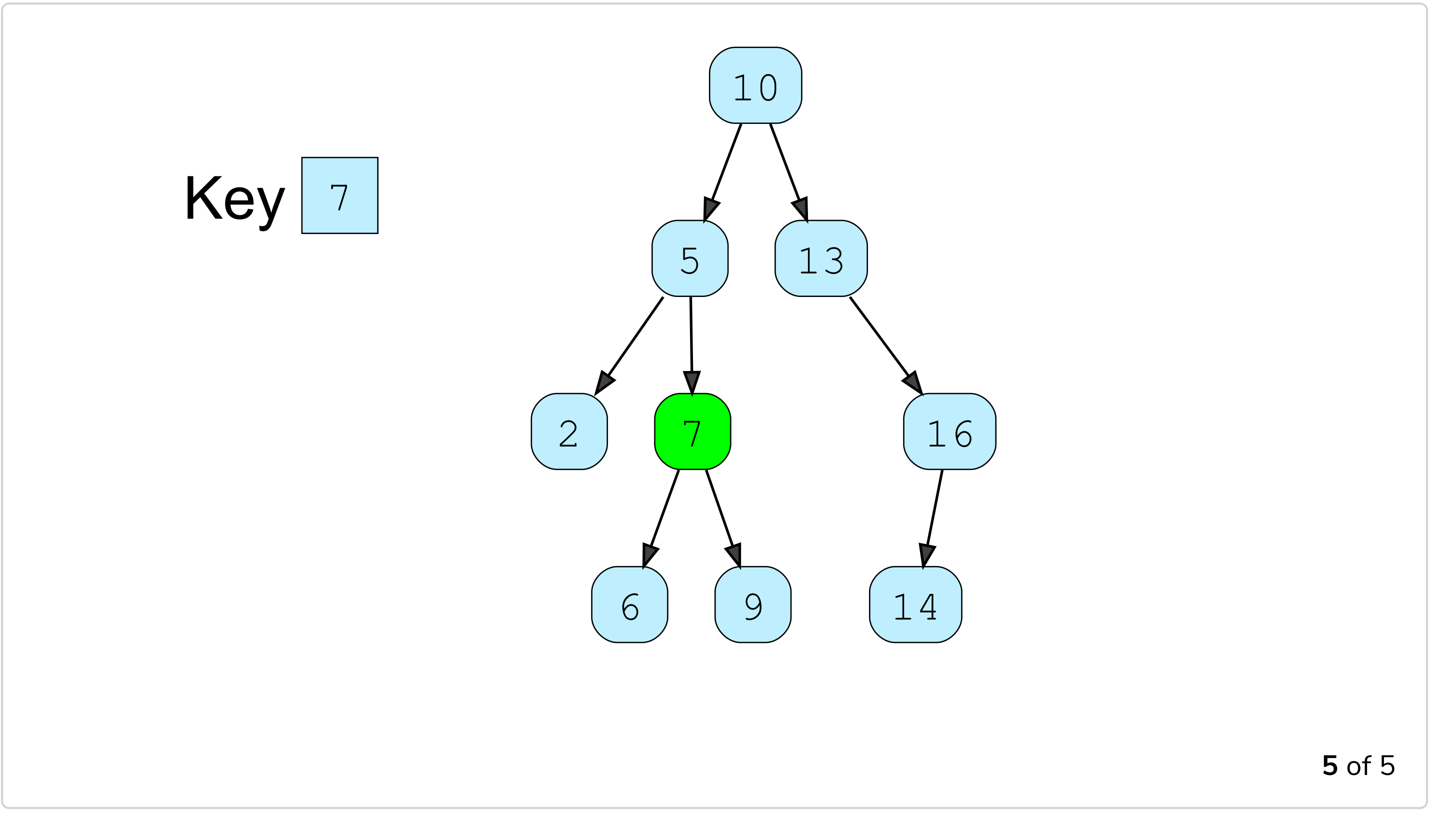
2 of 5



X > 5, so we go to right subtree

3 of 5





Code

```
struct Node* search(struct Node* pCrawl, int key) {
    // Base Cases
    if (pCrawl == NULL || pCrawl->key == key)
      return pCrawl;
    if (pCrawl->key < key)
      return search(pCrawl->right, key);
    else

      return search(pCrawl->left, key);
}
```

In the next lesson, we’ll see how to insert a key in a BST.