Binary Search Trees

```
• TreeSearch(x, k)              // pg 257
  // Search for key k in tree rooted at x
      if ((x = NIL) or (k = key[x]))
            return x
      if (k < key[x])
            return TreeSearch(left[x], k)
      else
            return TreeSearch(right[x], k)
```


Binary Search Trees

```
TreeInsert (T,z)        // pg 261,  Insert node z in tree T
        y = NIL
        x = root[T]             // y follows x down the tree
                                // when x is NIL, y points to a leaf
        while (x ≠ NIL) do
                y = x
                if (key[z] < key[x])
                        x = left[x]
                else    x = right[x]
        p[z] = y
        if (y == NIL)
                root[T] = z
        else if (key[z] < key[y])
                left[y] = z
        else right[y] = z
```



```
TreeDelete(T,z) // delete node z in tree T
        if (left[z] == NIL) or (right[z] == NIL)       then
                y = z
        else    y = TreeSuccessor(z)      // y has at most 1 child
        if (left[y] ≠ NIL) then
                x = left[y]
        else    x = right[y]              // x points to a child of y
        if (x ≠ NIL)  then
                p[x] = p[y]
        if (p[y] == NIL) then
                root[T] = x
        else    if (y == left[p[y]]) then
                        left[p[y]] = x
                else    right[p[y]] = x
        if (y ≠ z) then
                key[z] = key[y]
                copy y's data into z
        return y
```

Binary Search Trees

Animations

- BST:
  http://babbage.clarku.edu/~achou/cs160/examples/bst_animation/BST-Example.html
- Rotations:
  http://babbage.clarku.edu/~achou/cs160/examples/bst_animation/index2.html
- RB-Trees:
  http://babbage.clarku.edu/~achou/cs160/examples/bst_animation/RedBlackTree-Example.html