CSE 3010 – Data Structures & Algorithms

Lecture #35

What will be covered today

- Operations on BST
  - Inserting into a BST … Contd.
  - Finding the node type
  - Finding if a node is a left or right child
  - Finding if a node has a left or right child
  - Finding the size of a BST

Example dataset

Mat, Bat, Pot, Hut, Cat, Lit, Hat, Sat, Met, Lot

Insert a node in BST

```
// Insert a node in the tree
BSTNODE* insertItem(BSTNODE *root, ITEM key) {

    if (root == NULL)
        root = createNode(key);
    else {
        if (key < root->key)
            root->left = insertItem(root->left, key);
        else
            root->right = insertItem(root->right, key);
    }
    return root;
}
```

Finding the type of a node

```
// Returns true if the node is an internal node
bool isInternal(BSTNODE* node) {
    if (node->left != NULL || node->right != NULL)
        return true;
    else
        return false;
}

// Returns true if the node is a leaf node
bool isLeaf(BSTNODE* node) {
    if (node->left == NULL && node->right == NULL)
        return true;
    else
        return false;
}
```

Does the node have a left or right child

```
// Returns true if the node has a left node
bool hasLeftChild(BSTNODE* node) {
    if (node->left != NULL)
        return true;
    else
        return false;
}
// Returns true if the node has a right node
bool hasRightChild(BSTNODE* node) {
    if (node->right != NULL)
        return true;
    else
        return false;
}
```

Getting left or right child of a node

```
// Returns the left child of the input node
BSTNODE* getLeftChild(BSTNODE* node) {
    return node->left;
}

// Returns the right child of the node
BSTNODE* getRightChild(BSTNODE* node) {
    return node->right;
}
```