



Tree Nodes (Binary & Ternary Trees)

Ben Yan ✨ ✨ ✨

```
struct treeNode { // for binary tree
```

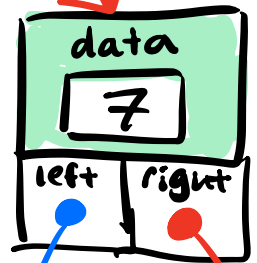
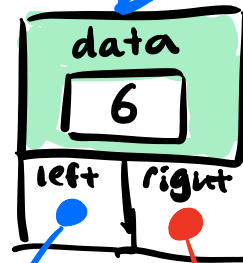
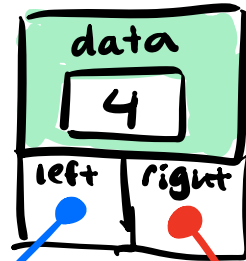
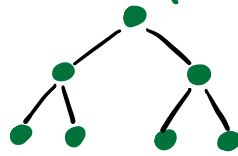
```
    int data;
```

```
    treeNode* left;
```

```
    treeNode* right;
```

```
// default constructor  
// does not initialize
```

```
treeNode() {}
```



null

null

null

null

// 3-ary constructor

// that initializes all fields from inputs

```
treeNode(int d, treeNode* l, treeNode* r) {
```

```
    data = d;
```

```
    left = l;
```

```
    right = r;
```

// Sample code to set up 3-node binary tree above

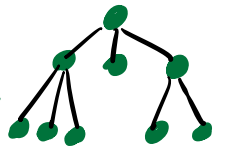
```
treeNode* root = new treeNode();
```

```
root->data = 4;
```

```
root->left = new treeNode(6, nullptr, nullptr);
```

```
root->right = new treeNode(7, nullptr, nullptr);
```

Struct TernaryTreeNode { // for ternary tree

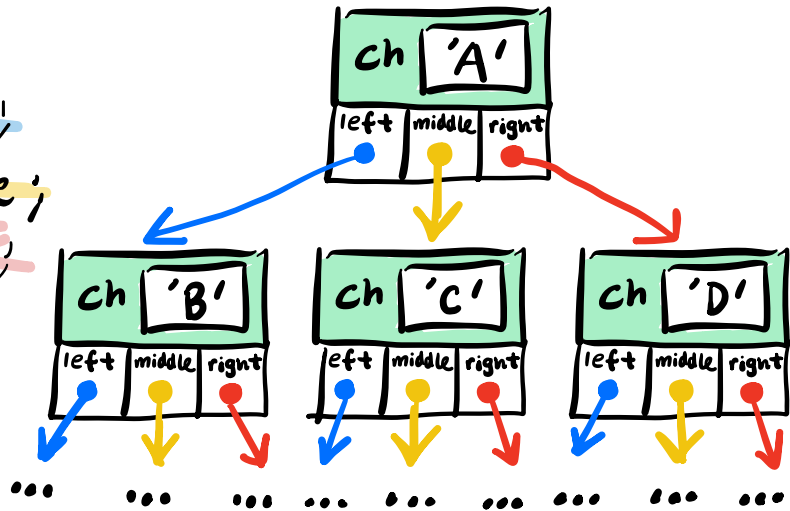


Char ch;

TernaryTreeNode * left;

TernaryTreeNode * middle;

TernaryTreeNode * right;



// default constructor, no initialization

TernaryTreeNode() { }

// initializes char data, and at first, no node children

TernaryTreeNode(char c) {

ch = c;

left = middle = right = nullptr;

// initializes char data and pointers to child nodes

TernaryTreeNode(char c, TernaryTreeNode * l,

TernaryTreeNode * m, TernaryTreeNode * r) {

ch = c;

left = l;

middle = m;

right = r;

// checks if node is leaf, ie. no node children
bool isleaf() {

return left == nullptr && middle == nullptr

&& right == nullptr;

}