# In class notes - trees 1

CS 14 - Data Structures

April 22, 2013

Given this code for a tree structure:

```
template<typename T> struct Tree
{
  T val;
  Tree<T> *left, *right;
}
```

We can perform a depth first search like this:

```
depthFirstSearch(Tree<T> *t)
{
  if (t) {
    // do something with t->val
    depthFirstSearch(t->left);
    depthFirstSearch(t->right);
  }
}
```

When we do this, we are implicitly creating a stack using the stack in memory. We can make this stack explicit by removing the recursion from our algorithm:

```
depthFirstSearch(Tree<T> *t)
{
  stack<Tree<T>*> search;
  search.push(t);

  while (!search.isEmpty()) {
    Tree *tmpTree = search.pop();
    if (tmpTree != NULL) {
      // do something with tmpTree->val
      search.push(tmpTree->left);
      search.push(tmpTree->right);
    }
  }
}
```

We can also perform a "breadth first search" by replacing the explicit stack with a queue:

```
breadthFirstSearch(Tree<T> *t)
{
  queue<Tree<T>*> search;
  search.enqueue(t);

  while (!search.isEmpty()) {
    Tree *tmpTree = search.dequeue();
    if (tmpTree != NULL) {
      // do something with tmpTree->val
      search.enqueue(tmpTree->left);
      search.enqueue(tmpTree->right);
    }
  }
}
```

There is no way to use an implicit queue on modern computers.