COMP26120 Algorithms and Data Structures
Topic 2: Data Structures

Traversing a Tree

Dr. Thomas Carroll
thomas.carroll@manchester.ac.uk
All information on Blackboard

Learning Outcomes

- Understand how Trees can be:
  - Traversed
  - Searched

Tree Traversal

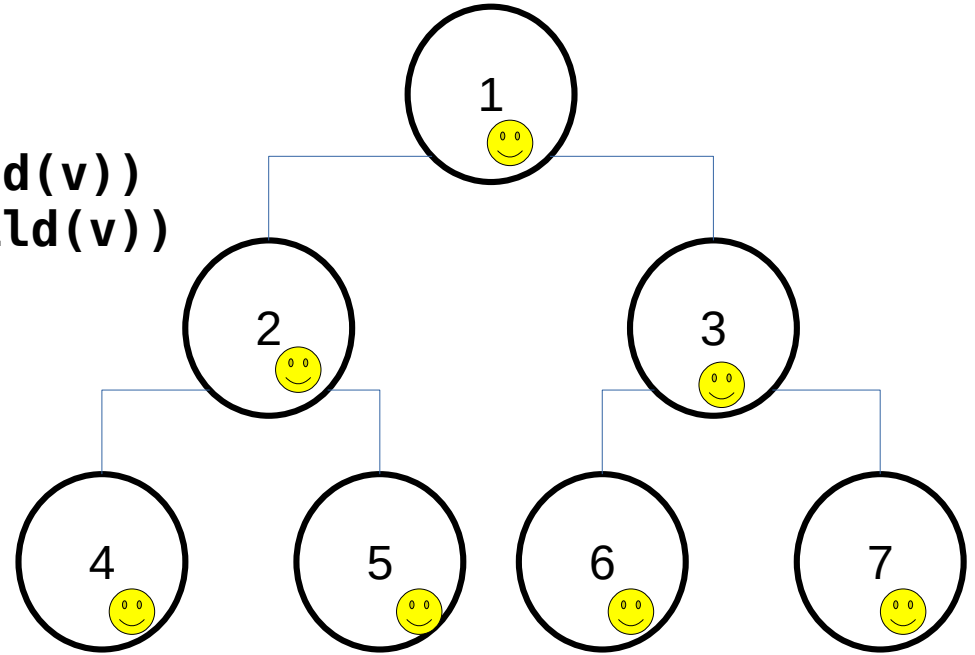
- Pre order
- In order
- Post order

The order refers to when we explore/visit the current node

Pre Order Traversal

```
preorder(T,v):
    visit(v)
    If T.isInternal(v):
          preorder(T, T.leftChild(v))
          preorder(T, T.rightChild(v))
```

1,2,4,5,3,6,7



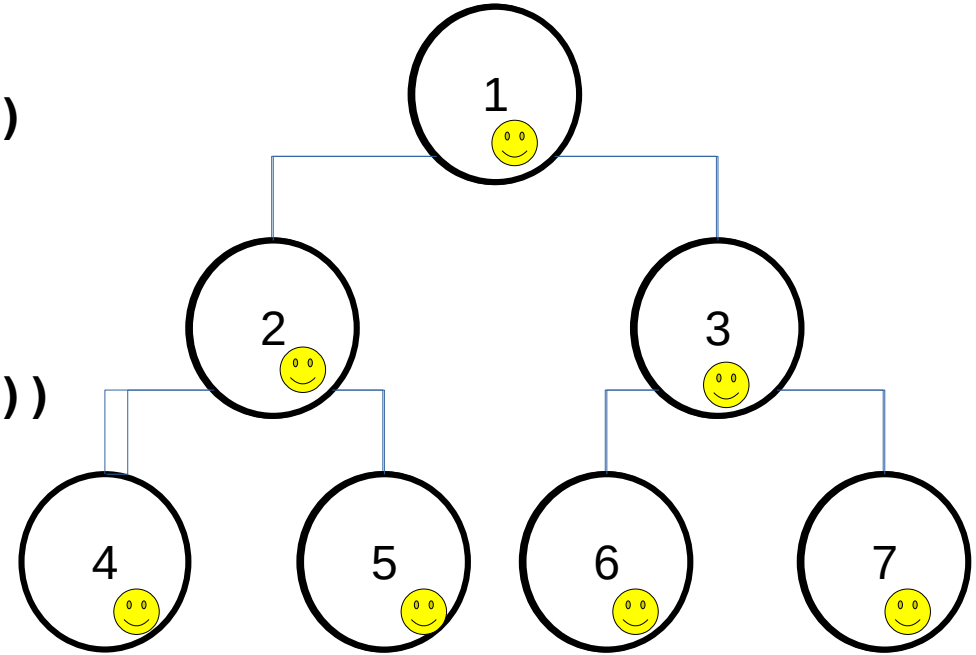
In Order Traversal

```
inorder(T,v):
    If T.isInternal(v):
        inorder(T, T.leftChild(v))

    visit(v)

    If T.isInternal(v):
        inorder(T, T.rightChild(v))
```

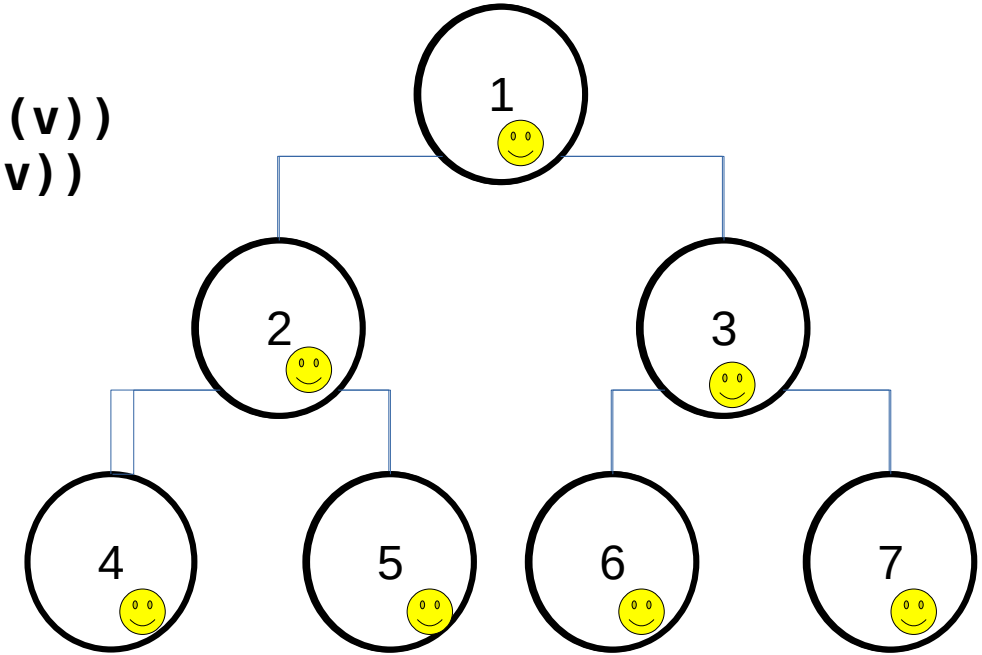
4,2,5,1,6,3,7



Post Order Traversal

```
postorder(T,v):
    If T.isInternal(v):
        postorder(T, T.rightChild(v))
        postorder(T, T.leftChild(v))
    visit(v)
```

4,5,2,6,7,3,1



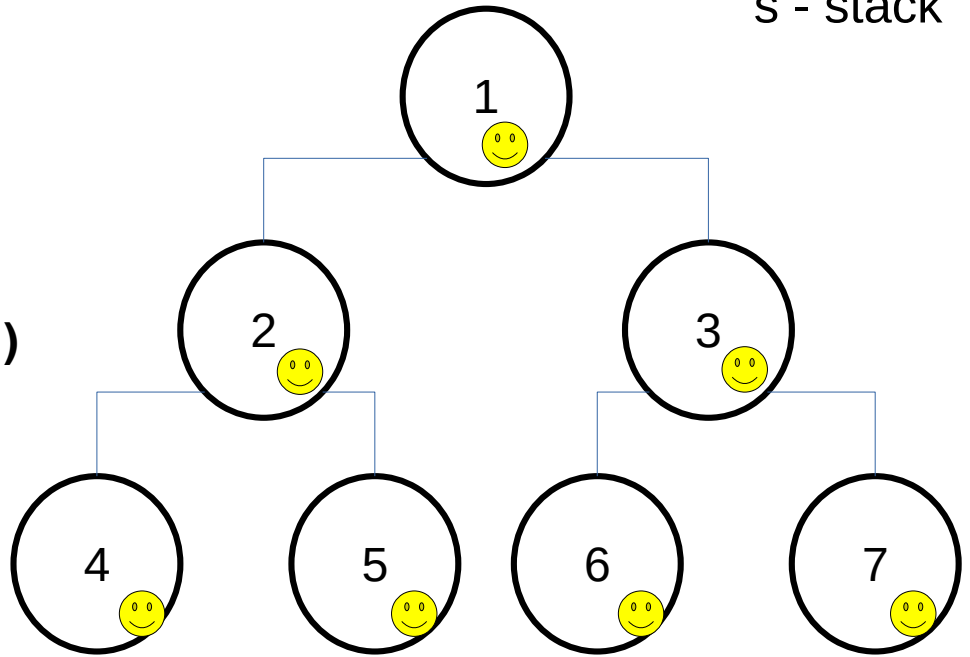
Tree Searching

- Depth First Search
- Breadth First Search

Depth First Search

```
DFS(T,v):
    s.push(v)
    while(s is not empty):
        v = s.pop()
        if v is not discovered:
            label v as discovered
            s.push(T.rightChild(v))
            s.push(T.leftChild(v))
```

s - stack



6
7
4
5
2
3
1

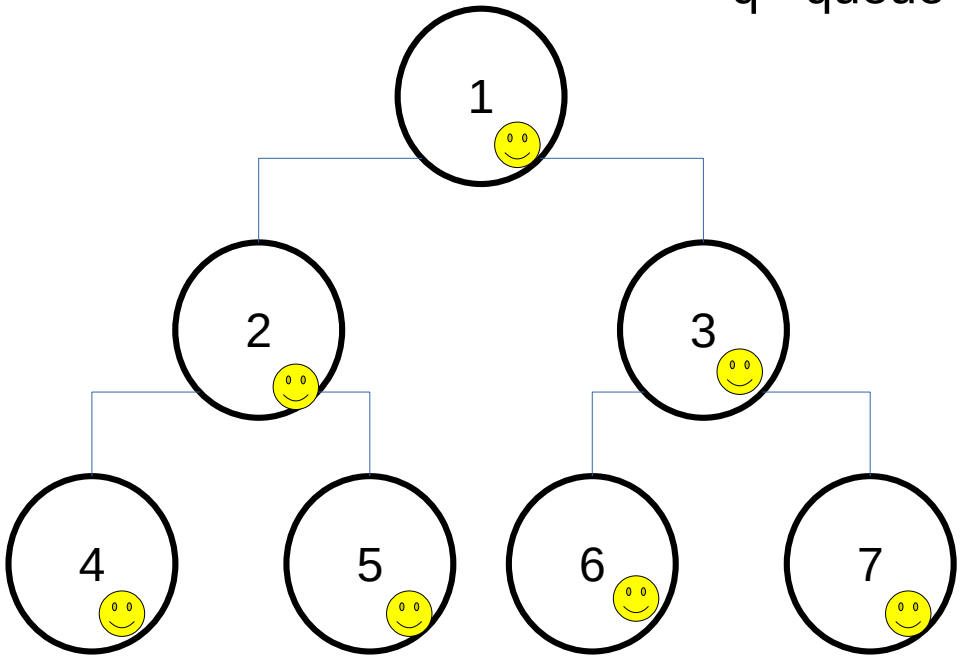
Could be used for file searching because we know a file is always a leaf,

DFS pushes itself down to the leaves faster than BFS

Breadth First Search

```
BFS(T,root):
    q.push(root)
    label root as discovered
    while(q is not empty):
        v = q.pop()
        r = T.rightChild(v)
        l = T.leftChild(v)
        if l is not discovered:
            label l as discovered
            q.push(l)
        if r is not discovered:
            label r as discovered
            q.push(r)
```

q - queue



1 2 3 4 5 6 7

BFS might be more useful for medical records where you're the root of the tree and the child nodes are your literal parents, the level below; grandparents etc