Tree (10A)

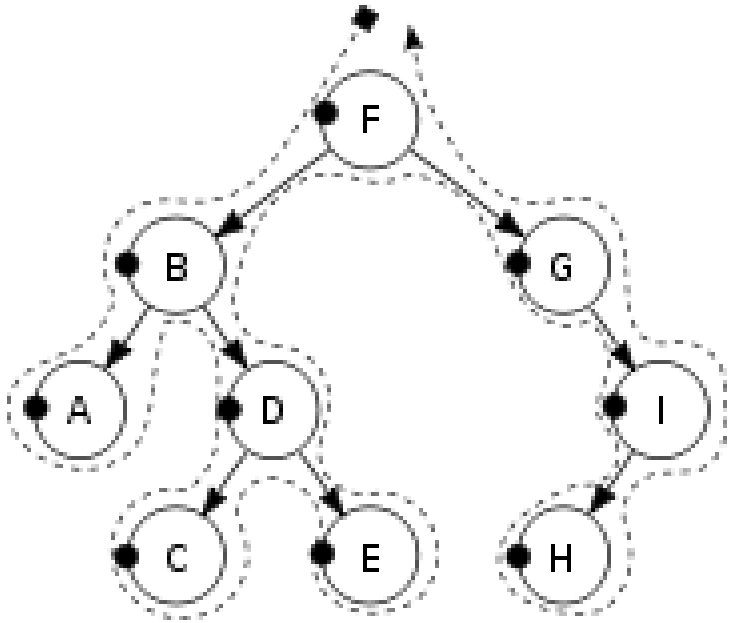
Young Won Lim
6/14/17

Please send corrections (or suggestions) to youngwlim@hotmail.com.

This document was produced by using OpenOffice and Octave.

In-Order

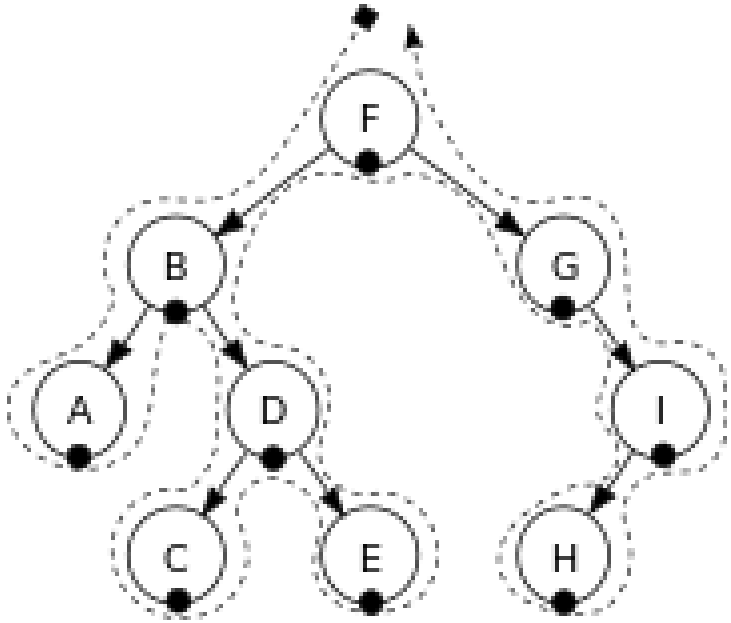
Check if the current node is empty / null.
Display the data part of the root (or current node).
Traverse the left subtree by recursively calling the pre-order function.
Traverse the right subtree by recursively calling the pre-order function.



https://en.wikipedia.org/wiki/Morphism

In-Order

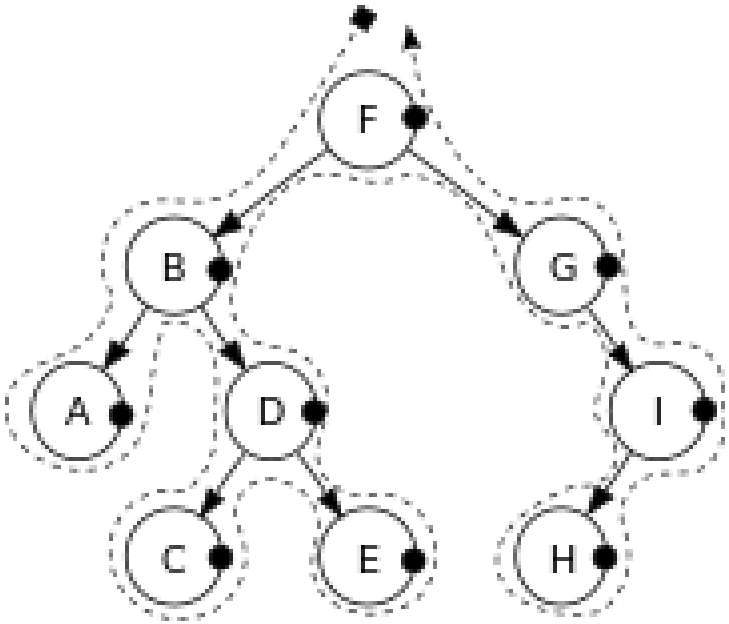
Check if the current node is empty / null.
Traverse the left subtree by recursively calling the in-order function.
Display the data part of the root (or current node).
Traverse the right subtree by recursively calling the in-order function.



https://en.wikipedia.org/wiki/Morphism

Post-Order

Check if the current node is empty / null.
Traverse the left subtree by recursively calling the post-order function.
Traverse the right subtree by recursively calling the post-order function.
Display the data part of the root (or current node).



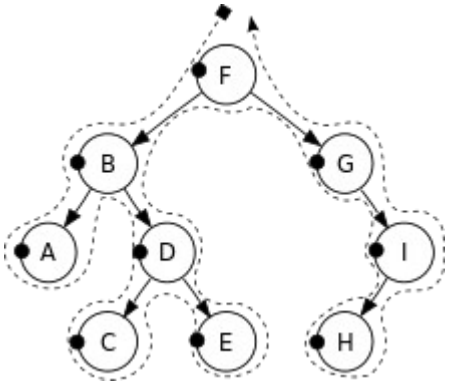
https://en.wikipedia.org/wiki/Morphism

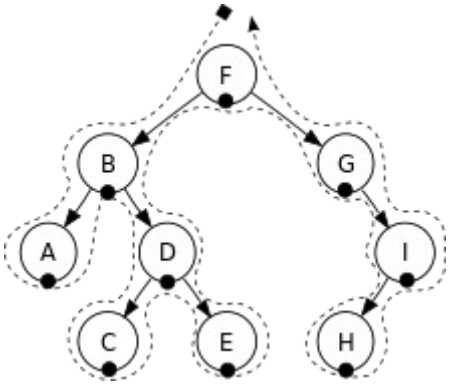
Recursive Algorithms

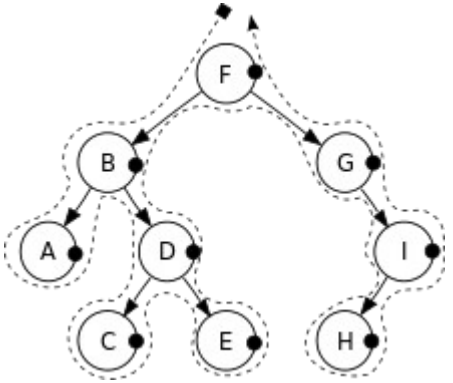
```
preorder(node)
  if (node = null)
    return
  visit(node)
  preorder(node.left)
  preorder(node.right)
```

```
inorder(node)
  if (node = null)
    return
  inorder(node.left)
  visit(node)
  inorder(node.right)
```

```
postorder(node)
  if (node = null)
    return
  postorder(node.left)
  postorder(node.right)
  visit(node)
```





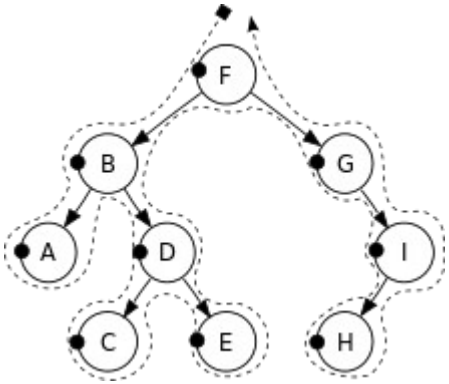


https://en.wikipedia.org/wiki/Tree_traversal

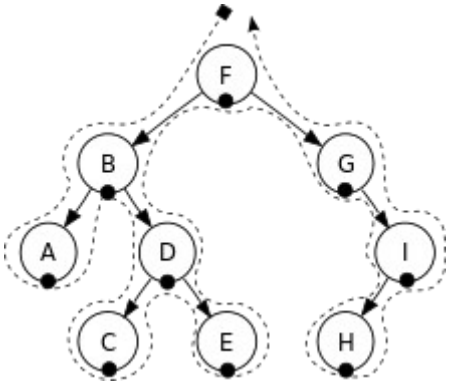
Iterative Algorithms

```
iterativePreorder(node)
  if (node = null)
    return
  s ← empty stack
  s.push(node)
  while (not s.isEmpty())
    node ← s.pop()
    visit(node)
    //right child is pushed first so that
left is processed first
    if (node.right ≠ null)
      s.push(node.right)
    if (node.left ≠ null)
      s.push(node.left)
```

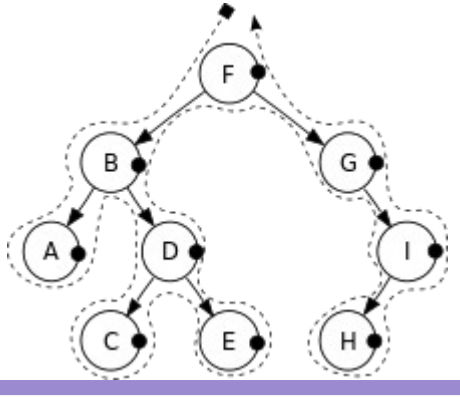
https://en.wikipedia.org/wiki/Tree_traversal




```
iterativeInorder(node)
  s ← empty stack
  while (not s.isEmpty() or
          node ≠ null)
    if (node ≠ null)
      s.push(node)
      node ← node.left
    else
      node ← s.pop()
      visit(node)
      node ← node.right
```

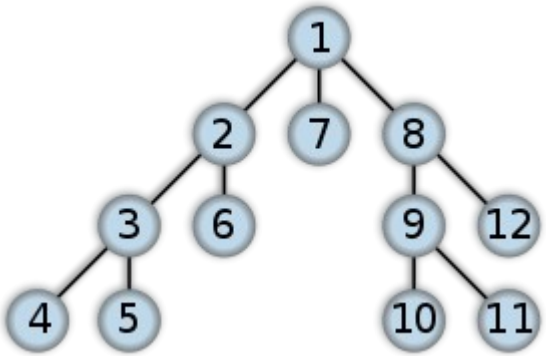



```
iterativePostorder(node)
  s ← empty stack
  lastNodeVisited ← null
  while (not s.isEmpty() or node ≠ null)
    if (node ≠ null)
      s.push(node)
      node ← node.left
    else
      peekNode ← s.peek()
      // if right child exists and traversing
node
      // from left child, then move right
      if (peekNode.right ≠ null and
          lastNodeVisited ≠ peekNode.right)
        node ← peekNode.right
      else
        visit(peekNode)
        lastNodeVisited ← s.pop()
```

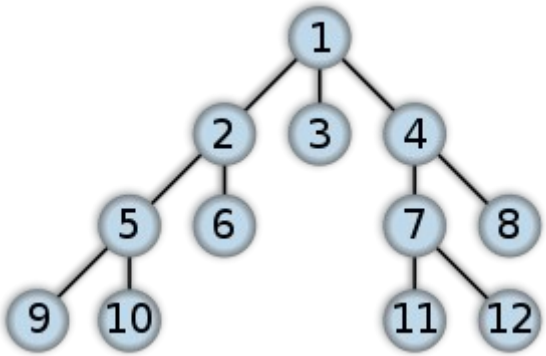


Search Algorithms

DFS (Depth First Search)



BFS (Breadth First Search)



https://en.wikipedia.org/wiki/Breadth-first_search, /Depth-first_search

DFS Algorithm

A recursive implementation of DFS:

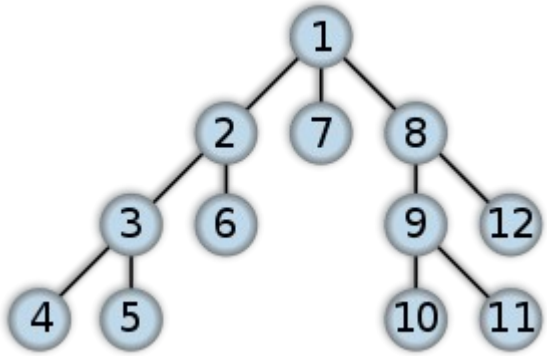
```
1  procedure DFS(G,v):
2      label v as discovered
3      for all edges from v to w in G.adjacentEdges(v) do
4          if vertex w is not labeled as discovered then
5              recursively call DFS(G,w)
```

A non-recursive implementation of DFS:

```
1  procedure DFS-iterative(G,v):
2      let S be a stack
3      S.push(v)
4      while S is not empty
5          v = S.pop()
6          if v is not labeled as discovered:
7              label v as discovered
8              for all edges from v to w in G.adjacentEdges(v) do
9                  S.push(w)
```

https://en.wikipedia.org/wiki/Breadth-first_search, /Depth-first_search

DFS (Depth First Search)

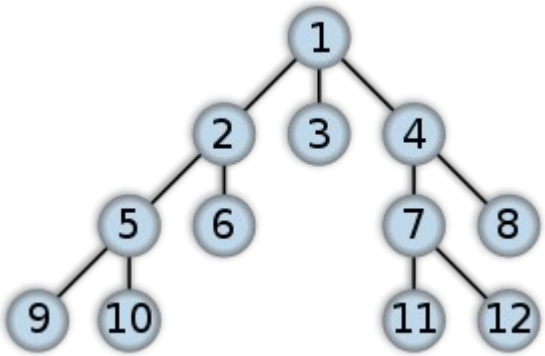


BFS Algorithm

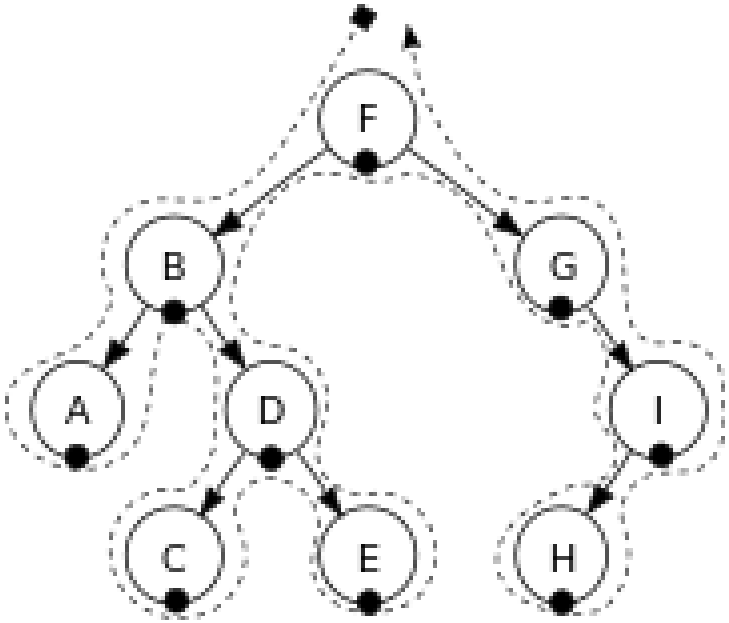
```
Breadth-First-Search(Graph, root):

    create empty set S
    create empty queue Q

    add root to S
    Q.enqueue(root)

    while Q is not empty:
        current = Q.dequeue()
        if current is the goal:
            return current
        for each node n that is adjacent to current:
            if n is not in S:
                add n to S
                n.parent = current
                Q.enqueue(n)
```

https://en.wikipedia.org/wiki/Breadth-first_search, /Depth-first_search

BFS (Breadth First Search)

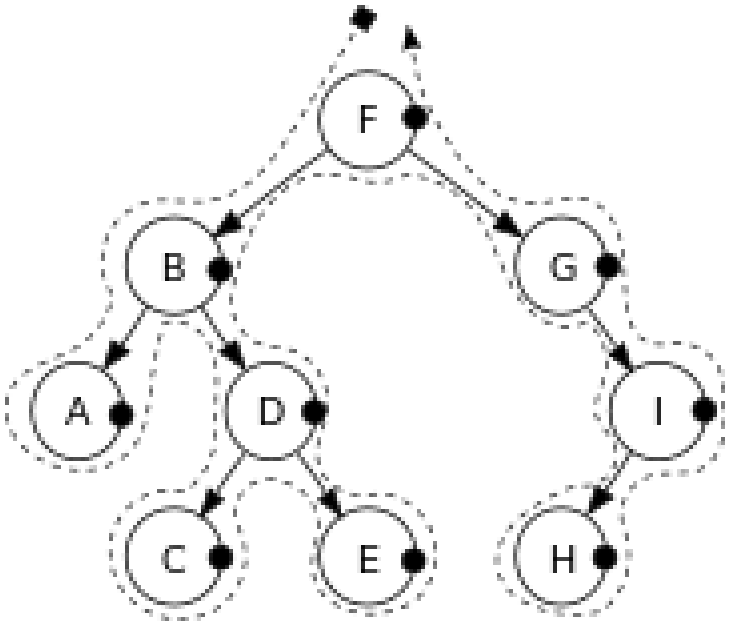


In-Order



https://en.wikipedia.org/wiki/Morphism

Post-Order



https://en.wikipedia.org/wiki/Morphism

References

[1] http://en.wikipedia.org/
[2]