Data Structure

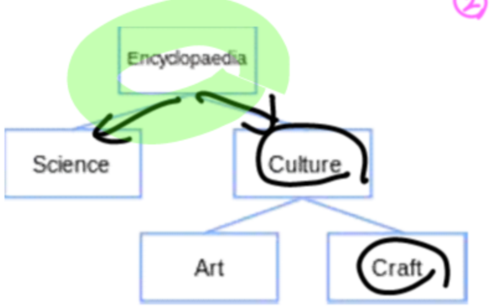
Tree

jintaeks@dongseo.ac.kr

December 3th, 2018

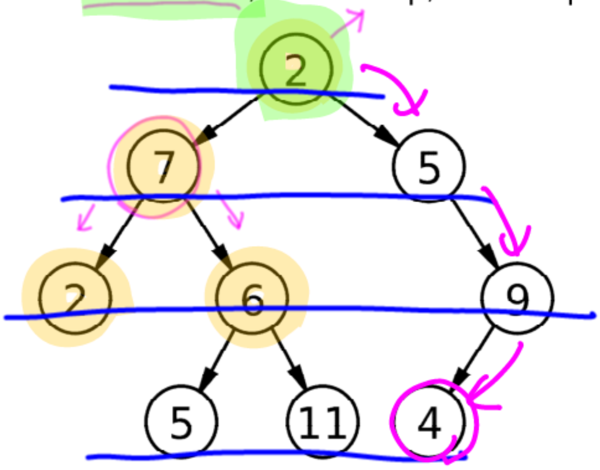
Tree

✓ A tree data structure can be defined recursively (locally) as a collection of nodes (starting at a root node), where each node is a data structure consisting of a **value**, together with a list of **references** to nodes (the "children"), with the constraints that ① no reference is duplicated, and ② none points to the root.



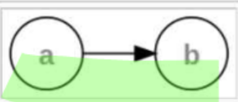

```
struct Node
{
    char name[20];
    Node* left;
    Node* right;
};
```

✓ A simple **unordered tree**; in this diagram, the node labeled 7 has two **children**, labeled 2 and 6, and one **parent**, labeled 2. The **root node**, at the top, has no parent.



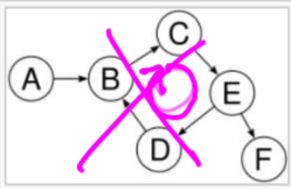
node
edge (link)

✓ A tree is a data structure made up of **nodes or vertices** and **edges without having any cycle**.
✓ The tree with no nodes is called the **null** or **empty** tree.



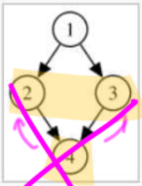
Each linear list is trivially a tree

Not a tree: cycle A→A. A is the root but it also has a parent.

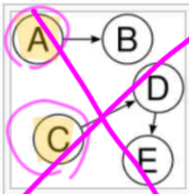


Not a tree: cycle B→C→E→D→B. B has more than one parent (inbound edge).

- A tree that is not empty consists of **a root node** and potentially many **levels** of additional nodes that form a **hierarchy**.



Not a tree: undirected cycle 1-2-4-3. 4 has more than one parent (inbound edge).



Not a tree: two non-connected parts, A→B and C→D→E. There is more than one root.

Terms

- Root
  - The top node in a tree.
- Child
  - A node directly connected to another node when moving away from the root.
- Parent
  - The converse notion of a child.
- Siblings
  - A group of nodes with the same parent.
- Descendant
  - A node reachable by repeated proceeding from parent to child. Also known as subchild.
- Ancestor
  - A node reachable by repeated proceeding from child to parent.

Terms

- ✓ Leaf
- ✓ External node (not common)
  - A node with no children.
- ✓ Branch node
- ✓ Internal node
  - A node with at least one child.
- ✓ Degree
  - For a given node, its number of children. A leaf is necessarily degree zero.
- ✓ Edge
  - The connection between one node and another.
- ✓ Path
  - A sequence of nodes and edges connecting a node with a descendant.

7

DIVISION OF DIGITAL CONTENTS DONGSEO UNIVERSITY

Terms

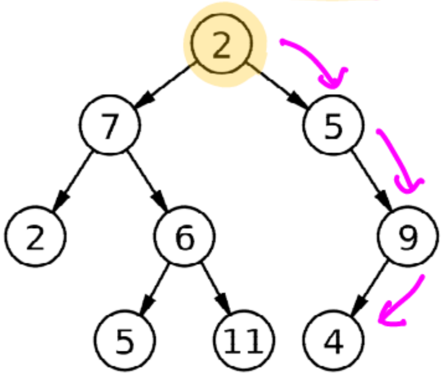
- ✓ Level
  - The level of a node is defined as: 1 + the number of edges between the node and the root.
- ✓ Depth
  - The depth of a node is defined as: the number of edges between the node and the root.
- ✓ Height of node
  - The height of a node is the number of edges on the longest path between that node and a leaf.
- ✓ Height of tree
  - The height of a tree is the height of its root node.
- ✓ Forest
  - A forest is a set of $n \geq 0$ disjoint trees.

8

DIVISION OF DIGITAL CONTENTS DONGSEO UNIVERSITY

Binary Tree

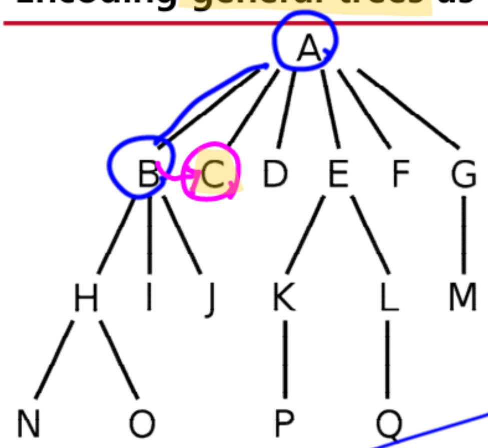
- ✓ A **binary tree** is a tree data structure in which each node has at most two children, which are referred to as the *left child* and the *right child*.

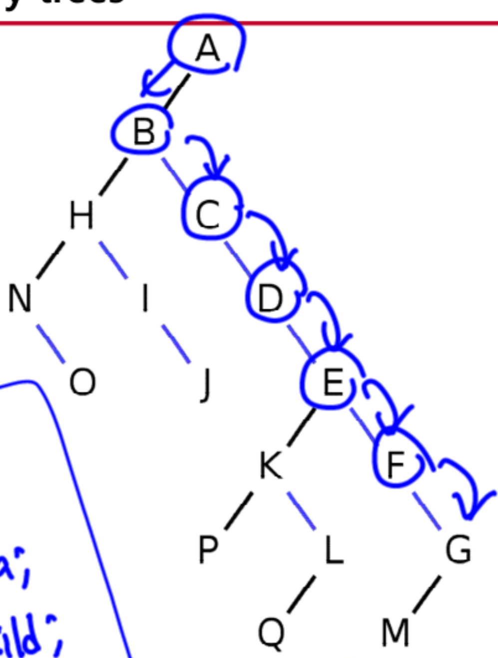


struct Node {
Node* left;
Node* right;
Data m_data;
};

- A labeled binary tree of size 9 and height 3, with a root node whose value is 2.

Encoding general trees as binary trees



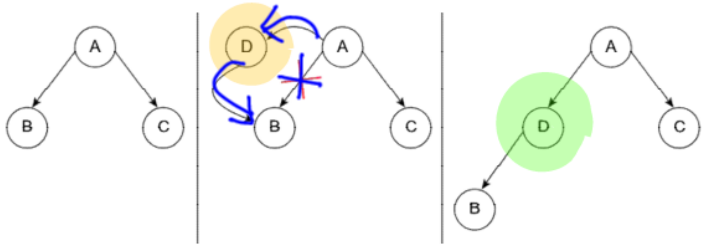


* Data Structure.

struct Node {
Data m_data;
Node* firstChild;
Node* nextSibling;
Node* parent;
};

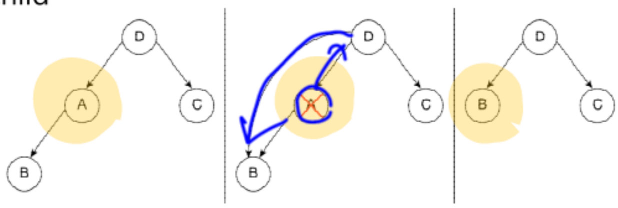
Common operations

- ✓ **Insertion**
  - Nodes can be inserted into binary trees in between two other nodes or added after a leaf node.
  - In binary trees, a node that is inserted is specified as to which child it is
- ✓ Leaf nodes
  - To add a new node after leaf node A, A assigns the new node as one of its children and the new node assigns node A as its parent.
- ✓ Internal nodes



Deletion

- ✓ Node with zero or one children
  - Suppose that the node to delete is node A.
  - If A has no children, deletion is accomplished by setting the child of A's parent to null.
  - If A has one child, set the parent of A's child to A's parent and set the child of A's parent to A's child



- ✓ Node with two children
  - In a binary tree, a node with two children cannot be deleted unambiguously.
  - However, in certain binary trees (including binary search trees) these nodes *can* be deleted, though with a rearrangement of the tree structure

ex) Heapify

Traversal: Depth-first search

✓ These searches are referred to as *depth-first search* (DFS), as the search tree is deepened as much as possible on each child before going to the next sibling.

- (L) Recursively traverse its left subtree. This step is finished at the node N again.
- (R) Recursively traverse its right subtree. This step is finished at the node N again.
- (N) Process N itself.

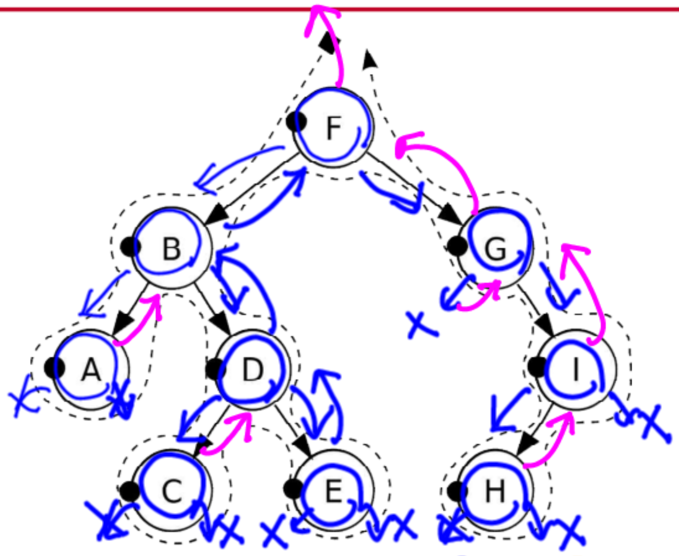
① DFS
② BFS

DIVISION OF DIGITAL CONTENTS DONGSEO UNIVERSITY

Pre-order(NLR)

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

DIVISION OF DIGITAL CONTENTS DONGSEO UNIVERSITY



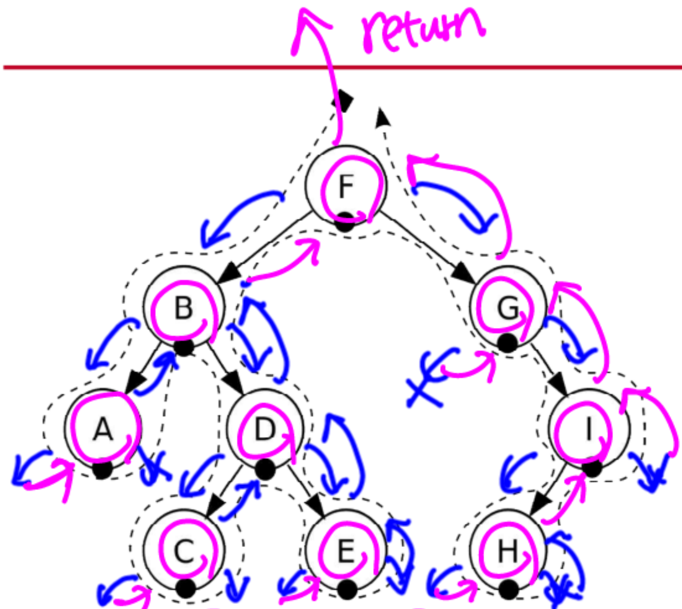
visit (node)
NLR { display node
visit left;
visit right;
}

- Pre-order: F B A D C E G I H

In-order(LNR)

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

✓ In a binary search tree, in-order traversal retrieves data in sorted order



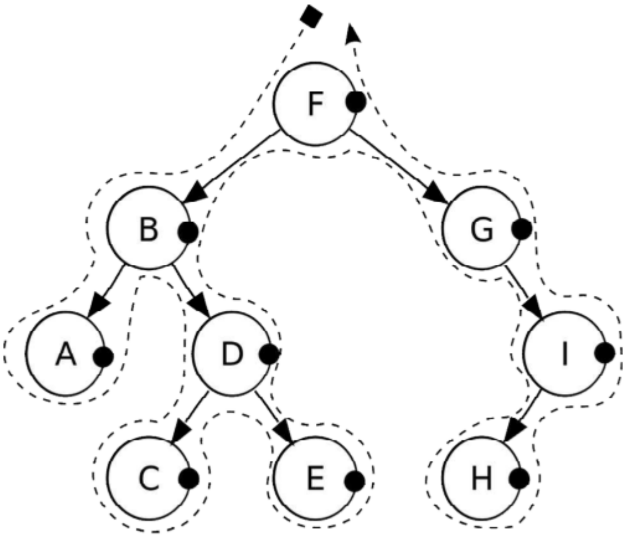
Visit(node)
{ visit left;
Display node;
Visit right
}

- In-order: A, B, C, D, E, F, G, H, I

Post-order(LRN)

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

Visit (node)
{ Visit left;
Visit right;
Display node;
}



* Do it!

- Post-order: A, C, E, D, B, H, I, G, F

Generic tree

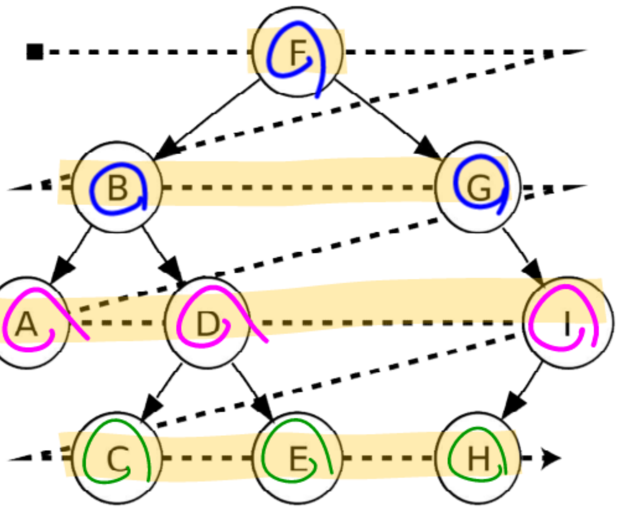
1. Perform pre-order operation.
2. For each *i* from 1 to the number of children do:
   1. Visit *i*-th, if present.
   2. Perform in-order operation.
3. Perform post-order operation.

Unify.

Breadth-first search

- ✓ Trees can also be traversed in ***level-order***, where we visit every node on a level before going to a lower level.
- ✓ This search is referred to as ***breadth-first search* (BFS)**, as the search tree is broadened as much as possible on each depth before going to the next depth.

Queue.



- Level-order: F, B, G, A, D, I, C, E, H

Implementation

✓ Pre-order

```
preorder(node)
  if (node = null)
    return
  visit(node)

preorder(node.left)

preorder(node.right)
```

23

in-order

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

24

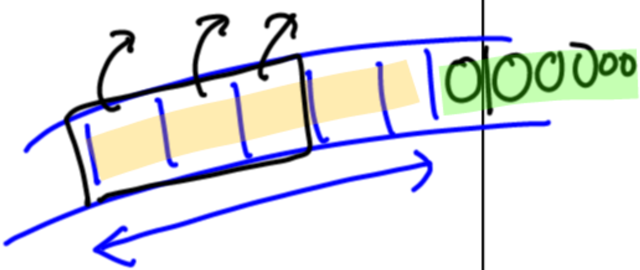
post-order

```
postorder(node)
  if (node = null)
    return

postorder(node.left)

postorder(node.right)
  visit(node)
```

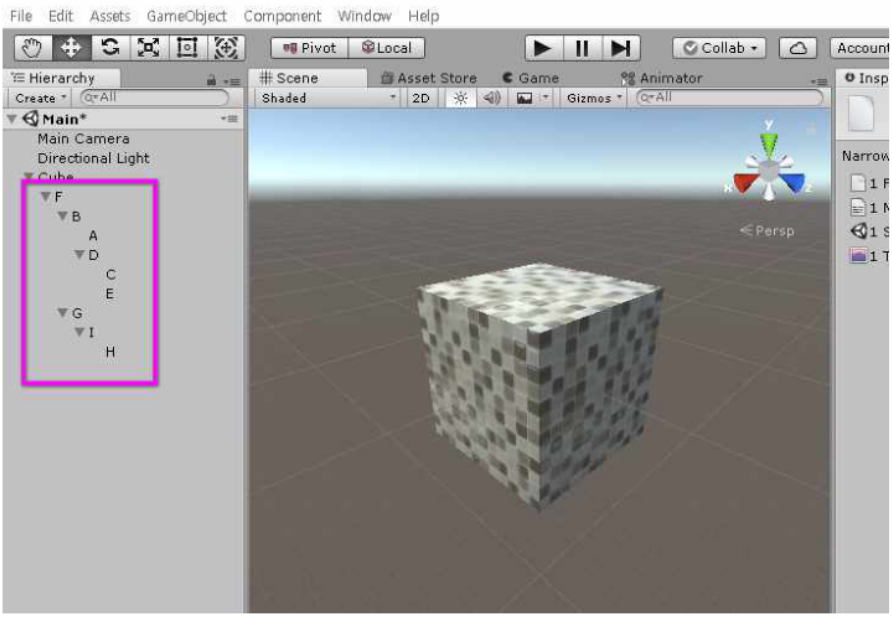

level-order

BFS

```
levelorder(root)
  q ← empty queue
  q.enqueue(root)
  while (not q.isEmpty())
    node ← q.dequeue()
    visit(node)
      if (node.left ≠ null)
        q.enqueue(node.left)
      if (node.right ≠ null)
        q.enqueue(node.right)
```



Practice in Unity



Pre-order and Post-order

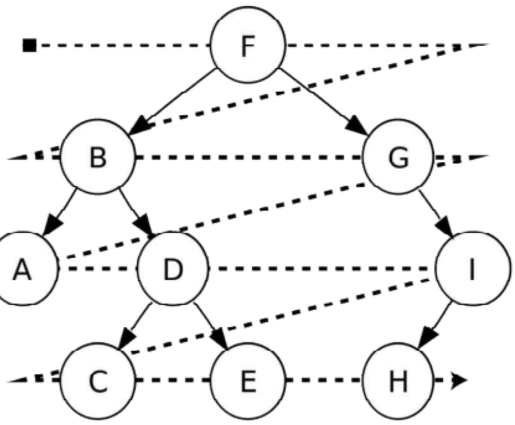
```
void TreeTraverse( Transform go )
  {
     foreach( Transform child in go )
     {
        Debug.Log( child.name ); // pre-order
        TreeTraverse( child );
        //Debug.Log( child.name ); // post-order
     }
  }

  // Use this for initialization
  void Start () {
     TreeTraverse( transform );
  }
```

C# generic: Queue

```
Queue<string> numbers = new Queue<string>();
        numbers.Enqueue("one");
        numbers.Enqueue("two");
        numbers.Enqueue("three");
        numbers.Enqueue("four");
        numbers.Enqueue("five");

        foreach( string number in numbers )
        {
            Console.WriteLine(number);
        }
```


Practice: Implement LevelOrder

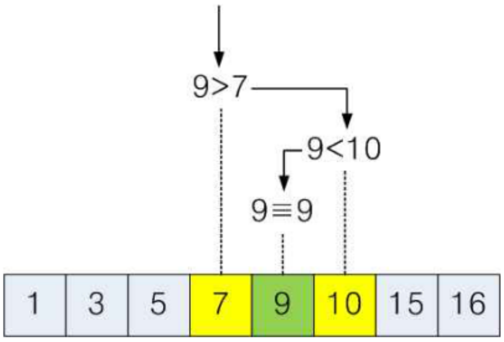
✓ @see "TreeTraverse_20181203_jintaeks.unitypackage"

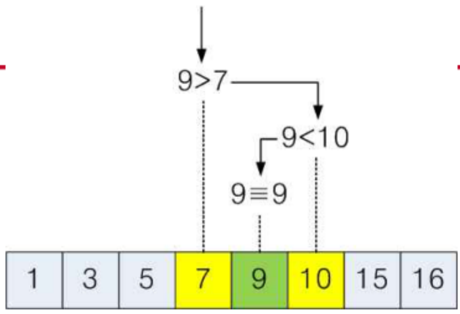


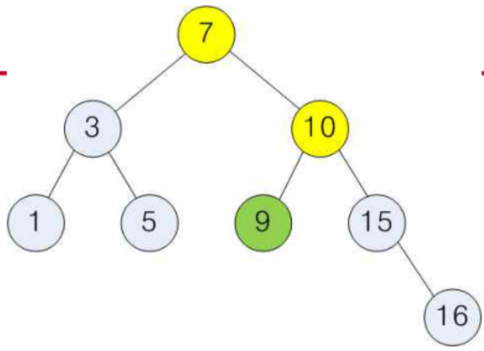
- Level-order: F, B, G, A, D, I, C, E, H

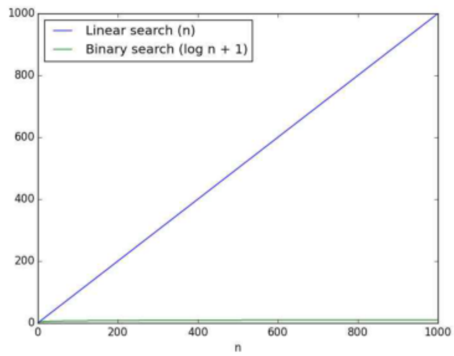
Binary search

- ✓ In computer science, **binary search**, also known as **half-interval search** or **logarithmic search**,[1] is a search algorithm that finds the position of a target value within a sorted array.









: Number of iterations for binary and linear search for *n* from 1 to 1000.

QnA

MY BRIGHT FUTURE

DSU Dongseo University 동서대학교

DIVISION OF DIGITAL CONTENTS DONGSEO UNIVERSITY