Tree

Class Tree {

Public Node node ;

}

Class Node {

Public String nodeValue ;

Node [] childes ;

}

Binary Tree vs. Binary Search Tree

Binary Search Tree

for all nodes, the left children are less than or equal to the current node, which is less than all the right nodes

Balanced vs. Unbalanced

Balance Tree takes log n time to insert and find.

Graph

Class Graph {

Node [] nodes ;

}

Grarph Search

DFS

It uses recursion,

void search(Node root) {

if (root == null) return;

visit(root);

root.visited = true;

foreach (Node n in root.adjacent) {

if (n.visited == false) {

search(n);

}

}

}

BFS

BFS uses a queue

If you are asked to implement BFS, the key thing to remember is the use of the queue.

The rest of the algorithm flows from this fact.

BFS first visit adjust of root node and after visiting them , it puts them in a queue .

DFS is typically the easiest if we want to visit every node in the graph,

or at least visit every node until we find whatever we're looking for. However, if we have

a very large tree and want to be prepared to quit when we get too far from the original

node, DFS can be problematic; we might search thousands of ancestors of the node,

but never even search all of the node's children

Q Given a directed graph, design an algorithm to find out whether there is a route

Between two nodes.

So just traversal the tree will solve the problem

Q Given a linked list\array , designed a binary search tree with minimal height

1st sort the array, then mid of array will be root , then recursively add 1st half on the left and second half on the right .

Q List at each level of the tree

DFS – pass the level at each node

BFS – Hard to understand , but visit childs and then add the childs in a linked list then for element in the linked list get left and right and add them in a new linked list . follow this in a loop

Q Check of the tree is balanced or not ? or Balanced tree is one where hight of left node – height of right node is less than one