Basics of Graphs & Trees

What is Graph?

Imagine a country consisting of some cities and some roads connecting some pairs of cities.

If the country is represented as a graph:

- ➔ Each city is considered as a **node** or **vertex** of the graph.
- ➔ Each road is considered as an **edge** of the graph.

A graph is a pair of sets (V, E), where V is the set of vertices and E is the set of edges, connecting the pairs of vertices.

Different Types of Graphs

- Directed Graph
- Undirected Graph
- Weighted Graph

Visualization: https://csacademy.com/app/graph_editor/

Basic Terminologies

1. Adjacency & Adjacent Nodes
2. Degree of a Vertex
    a. Indegree
    b. Outdegree
3. Path
4. Cycle
5. Connected Components

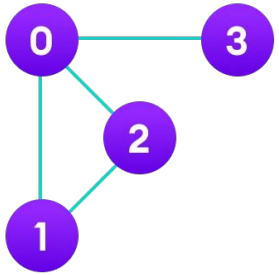
Graph Representation

Two common ways of graph representation:

1. Adjacency Matrix Representation
2. Adjacency List Representation

Adjacency Matrix Representation

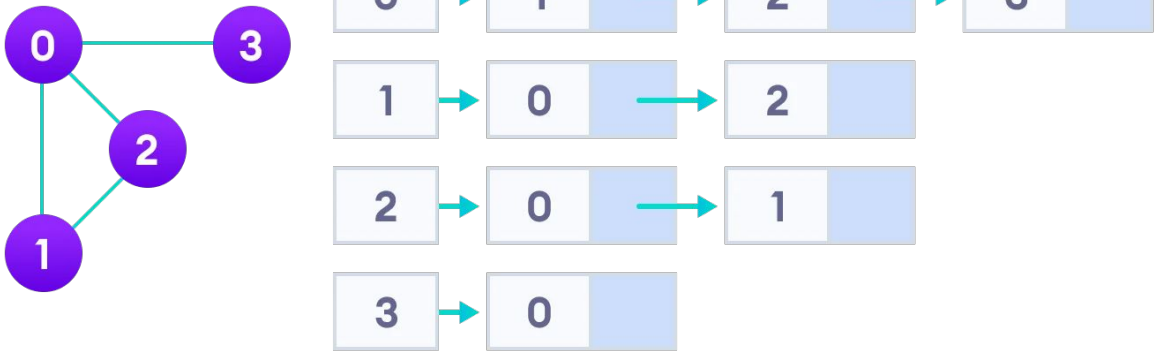
An adjacency matrix is a 2D array of size N x N where N is the number of vertices. If the array is a, then a[i][j] is 1 if there is an edge connecting vertex i and vertex j.



	0	1	2	3
0	0	1	1	1
1	1	0	1	0
2	1	1	0	0
3	1	0	0	0

Adjacency List Representation

An array of list is used. i-th list contains the adjacent vertices of vertex i.



Which one of the two representation methods would you like to use and why?

Graph Input & Store

Graph input (implementation will be shown):

- Undirected Graph
- Directed Graph
- Weighted Graph

Graph Traversal

Graph traversal is the process of visiting or checking each node of the graph.

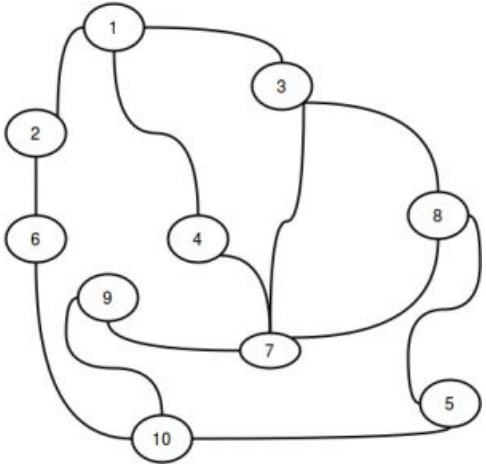
Two commonly used algorithms for graph traversing:

1. Breadth First Search (BFS)
2. Depth First Search (DFS)

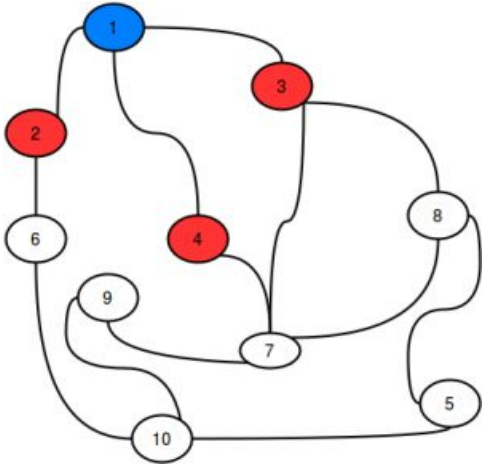
Breadth First Search (BFS)

- The algorithm starts traversing the graph from a specific node called the source node. The level of source node is 0.
- Each node is visited exactly once.
- Adjacent nodes of level 0 node is marked as level 1 node.
- Adjacent (unmarked) nodes of any level 1 node is marked as level 2 node.
- Traversal starts from level 0 (source) node. Then level 1 node is visited. Then level 2 and so on.

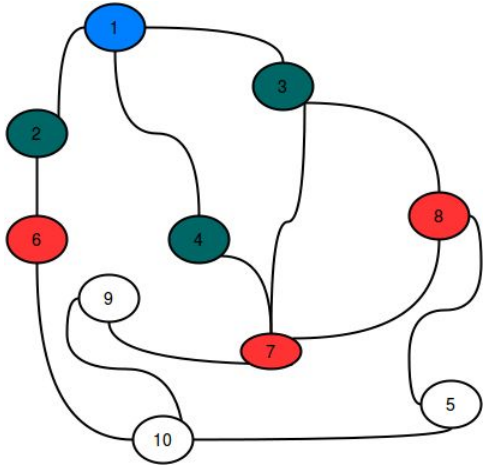
Breadth First Search (BFS)



Node 1 is source

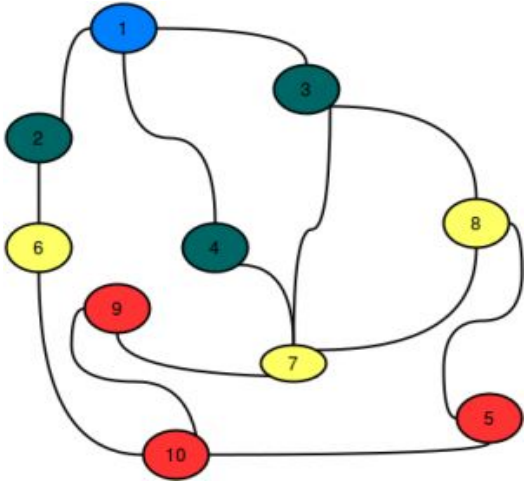


Node 2, 3 & 4 are level 1 nodes

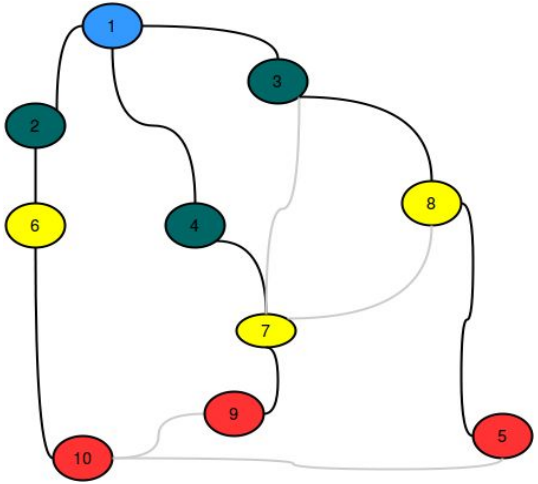


Node 6, 7 & 8 are level 2 nodes

Breadth First Search (BFS)



Node 5, 9 & 10 are level 3 nodes



BFS Tree

Breadth First Search (BFS)

```
vector<int> vis(n+1, 0);
vector<int> level(n+1, 0);
queue<int> q;
int src = 1; // starting bfs from node 1
vis[src] = 1; // source node is visited by default
level[src] = 0; // level of source node is 0
q.push(u);

while(!q.empty()){
    int u = q.front();
    q.pop();

    for(int v: adj[u]){ // iterating over the list of adjacent nodes
        if(vis[v]) continue; // node v is already visited
        level[v] = level[u]+1;
        vis[v] = 1;
        q.push(v);
    }
}
```

Breadth First Search (BFS)

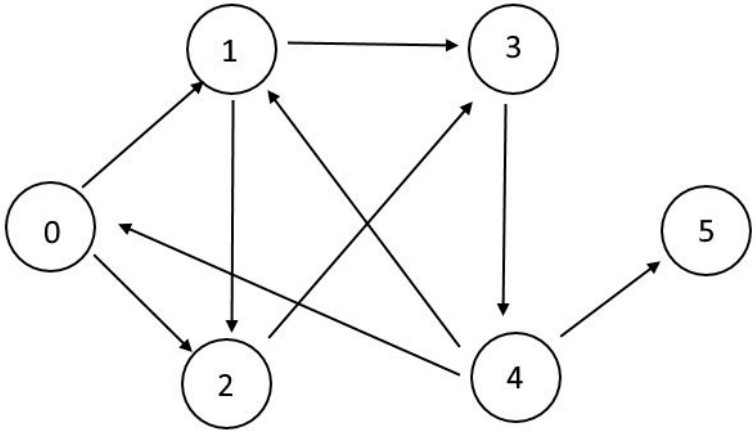
For an unweighted graph, the level of each node is the length of shortest path (minimum distance) from source node.

The time complexity of BFS algorithm is O(V+E) where V is the number of nodes and E is the number of edges. It is because, each node and edge of the graph is visited at most once.

Depth First Search (DFS)

DFS is a recursive algorithm. The basic idea is to start from the root or any arbitrary node and mark the node as visited and move to the adjacent unmarked node and continue this loop until there is no unmarked adjacent node. Then backtrack and check for other unmarked nodes and traverse them.

Depth First Search (DFS)



Depth First Traversal - 0 1 3 4 5 2

Is it the only possible way of traversing the graph using DFS?

Depth First Search (DFS)

```
int vis[N];

void dfs(int u){
   if(vis[u]) return;

   vis[u] = 1;
   // node u is visited
   for(int v: adj[u]){
       dfs(v);
   }
}
```

Time Complexity: O(V+E)

Problems on Graph Traversal

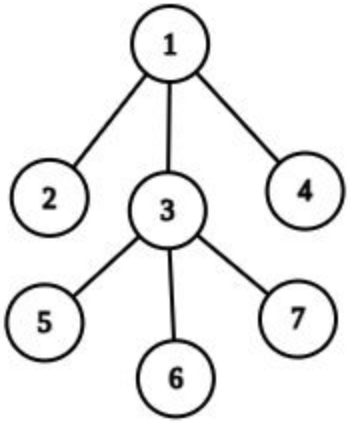
- Counting number of connected components of a graph.

Problems on Graph Traversal

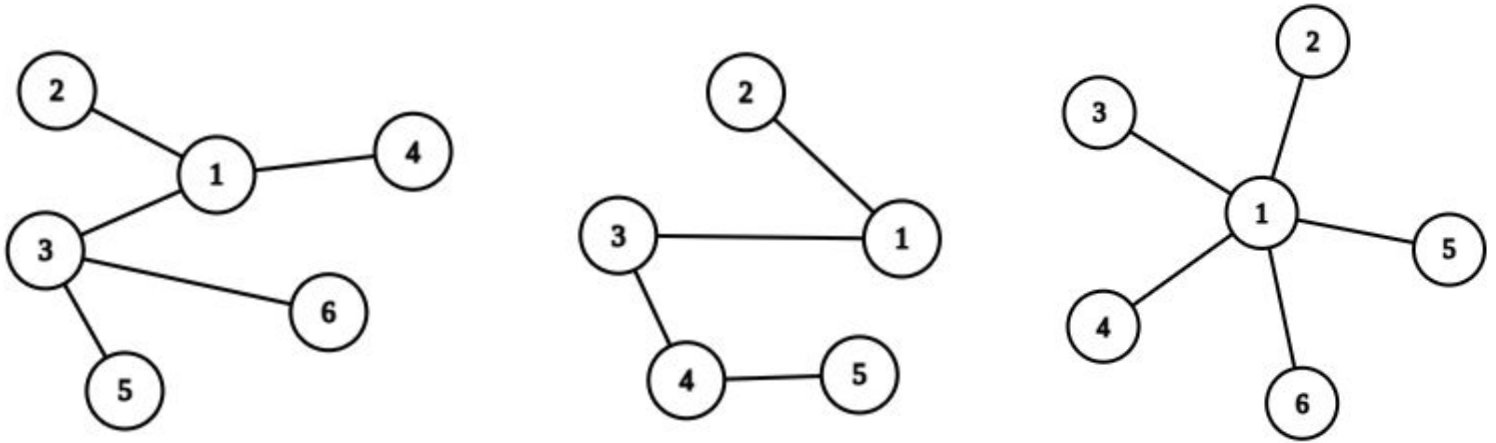
- Finding shortest distance between two nodes of an unweighted graph.

What is a Tree?

A tree is a connected graph without any cycle.



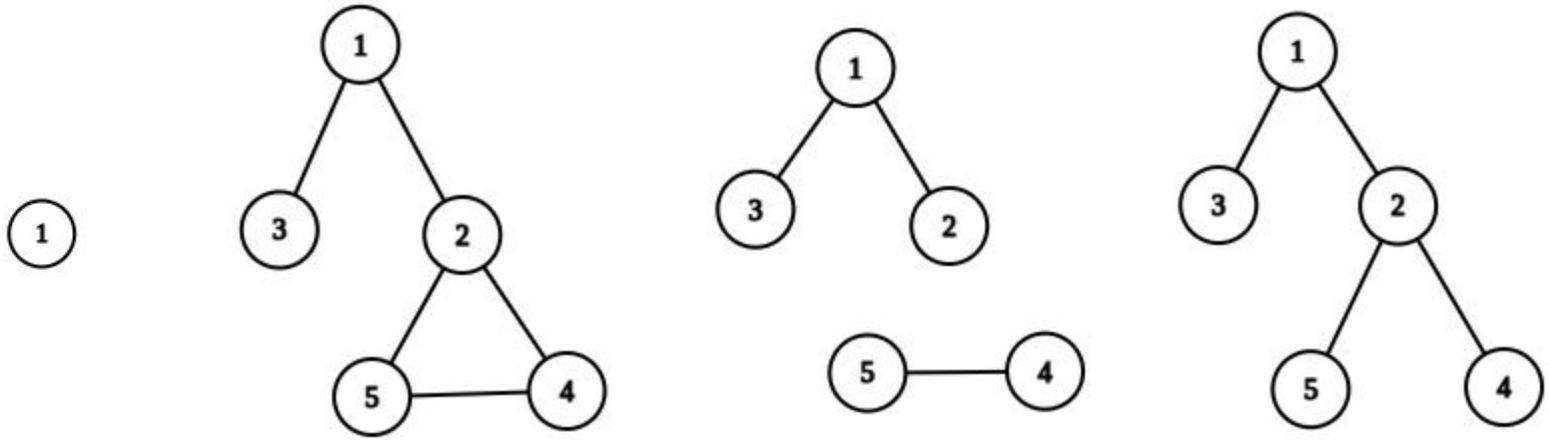
What is a Tree?



Examples of some trees

What is a Tree?

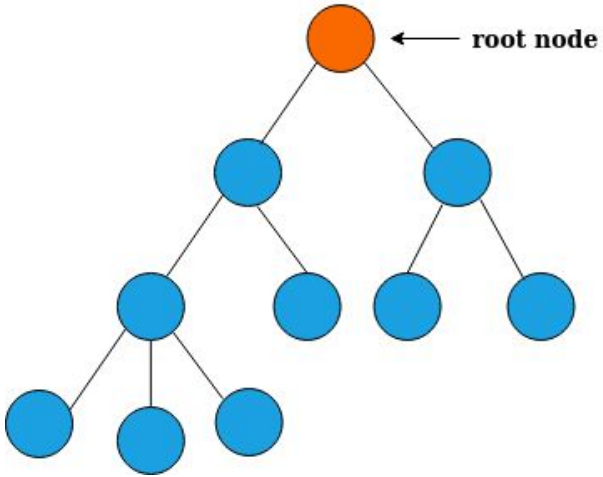
Which of the following graphs are tree?

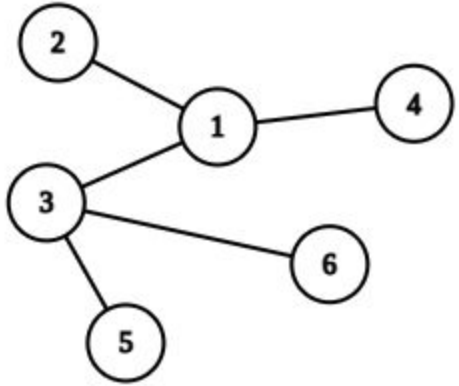


Tree Properties

- Number of Edges in a Tree for N nodes
  - N - 1
- Number of paths between 2 nodes
  - 1
- Sum of Degree of all nodes
  - 2 * (N - 1)
- Can there be less than 2 leaf nodes in a Tree with >= 2 nodes
  - No, except for the case when there is just one node in the entire tree

Rooted & Unrooted Trees





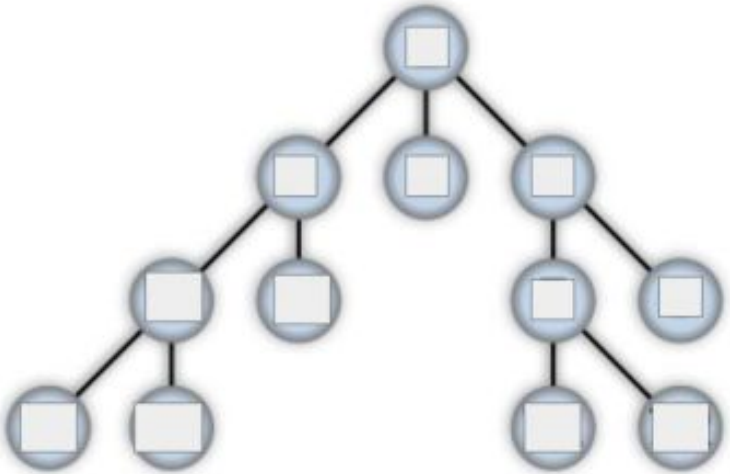
Tree Terminologies

- Root
- Parent
- Child
- Ancestor
- Descendant
- Level of Node
- Subtree
- Subtree Size
- Height of Tree
- Lowest Common Ancestor

More Tree Properties

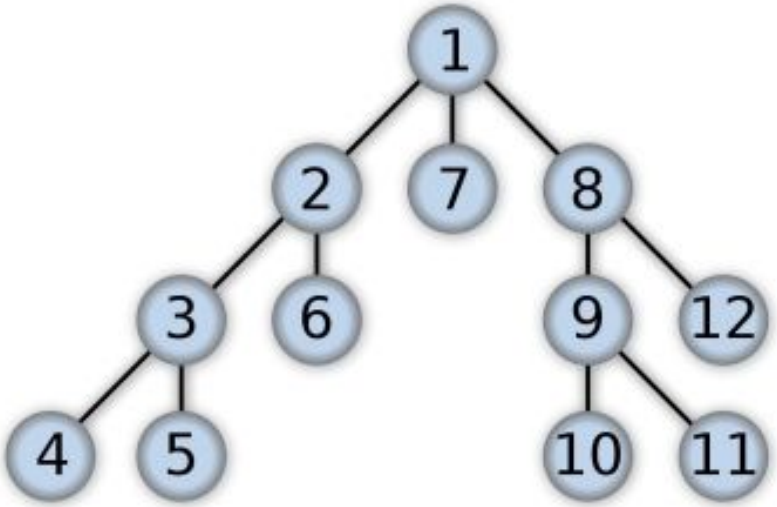
- Can there be more than 2 parents of a single node in a rooted tree
  - No
- Path property. What are the only 3 types of paths possible?
  - 1. Go Up, Come Down 2. Go Up 3. Come Down
- How to color a Tree with just 2 colors such that no two neighbours have the same color
  - Just root the tree and color level wise

Traversal in a Tree



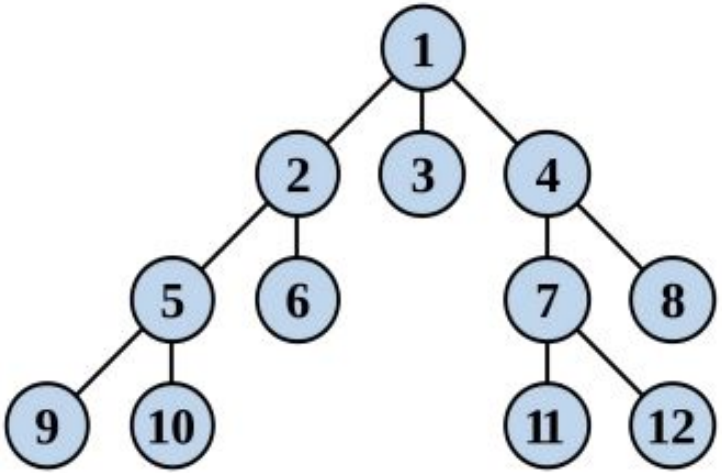
How to traverse the tree in a way such that we visit all nodes?

DFS Traversal in a Tree



Nodes are numbered in the order in which they are visited.

BFS Traversal in a Tree



Nodes are numbered in the order in which they are visited.

Problems on Tree Traversals

- Finding the parent of each node of a rooted tree.

Problems on Tree Traversals

- Finding the level of each node of a rooted tree.

Problems on Tree Traversals

- Finding the height of a rooted tree.

Problems on Tree Traversals

- Finding the size of all the subtrees of a rooted tree.