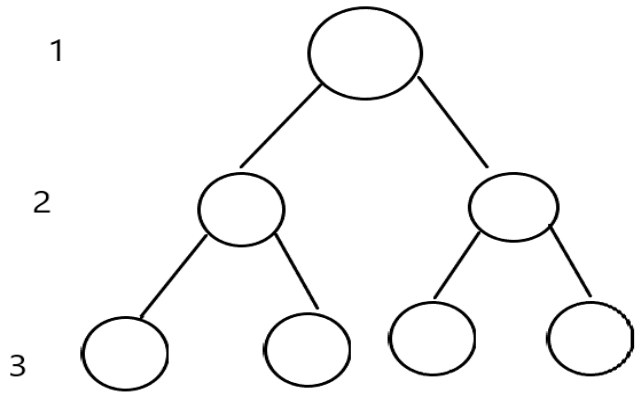
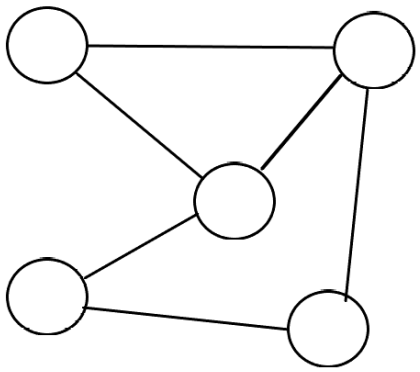
By now, you have all learned what a binary tree is, as well as how to structure a binary tree and understand its anatomy. In simple words, it is a data structure where every NODE has one or more child NODES, which can follow up to having its own child NODE and so on.



In the above diagram, the node in level 1 is the root note as in, the tree starts at that Node. The nodes in level 2 are the children of Node in level 1. Following this scenario, the nodes in level 3 are the children of nodes in level 2. And the beauty of a tree structure is that, this pattern can go on as long as it needs to go and everything can be solved recursively. (which I have already discussed in class before Covid-19 took over our lives).

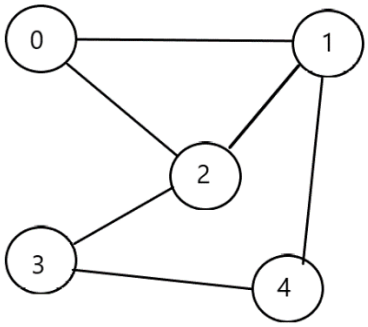
Now, I want you to picture a scenario. In the above diagram, you can see all the nodes are following the same pattern, that is, for every node, there can be at most 2 child nodes and one parent node. But what if this wasn't the situation? Look at the next diagram



What do you think? It kind of looks like a tree, but then again, it is not as it does not follow the rules of binary tree. So what is it? Let's think of the above example in this way. Suppose there are 5 individual islands and some of these islands have a road between them, some does not. So we can say that each of

these nodes represent an island, and the connection between these nodes (represented by a line) is a bridge between two respective nodes. Now if I ask you what kind of structure can be used to represent this situation, what would you say? It certainly cannot be represented using a binary tree, as you can see the middle node has three connections to three other nodes. And also the top-leftmost and bottom-leftmost nodes are not even connected to each other.

Here, we use a data structure which is known as GRAPH. It is just like what you learned back in school and college maths. A 2 dimensional plane where you have values in X axis, and also in Y axis. So how can you relate that knowledge in this scenario? Well lets have a look. First, we will label our nodes using numbers.



We have put labeled our nodes as 0,1,2,3,4. Now, we can create a table with Rows and Columns as these numbers.

	0	1	2	3	4
0					
1					
2					
3					
4					

Here, you see that both rows and columns are marked as 0,1,2,3,4. Just like the labels in the nodes. But how can you represent the connection between these nodes? Well what you can do is: **if there is connection between node 0 and 1, we will place a numeric value (like 1) in the table on 0th row and 1th column.**

	0	1	2	3	4
0		1			
1					
2					
3					
4					

Thus, we have placed 1 in the 0^{th} row and 1^{st} column representing that there is a connection between Nodes 0 and 1. But, according to the diagram we can also say that, we have connection between Node 1 and 0 as well. What shall you do in this case? Well that is simple, put 1 in 1^{st} row and 0^{th} column.

	0	1	2	3	4
0		1			
1	1				
2					
3					
4					

Hence, if there is connection between Node X and Node Y, we will put 1 in (X,Y) place and also in (Y,X) since it goes both ways. And fill-up the non-connected nodes with 0. Following this logic, you can now fill up the whole table, which will be a PERFECT representation of the graph image.

	0	1	2	3	4
0	0	1	1	0	0
1	1	0	1	0	0
2	1	1	0	1	0
3	0	0	1	0	1
4	0	0	0	1	0

Now for 5 nodes, we are getting this 5x5 table. If there are 10 nodes, then the table will be 10x10. This table can be easily represented using a 2D array in C++. If you have an array which is **int graph[5][5]** and you want to say there is connection between Node **X** and Node **Y** then you can simple write;
graph[X][Y] = 1;
graph[Y][X] = 1;

Connection between two nodes is formally known as an **Edge.** And the nodes are called **Vertex (plural: vertices)**

Graphs can be represented in another way instead of using a 2D array, which is known as Adjacency List representation. But we will not go into that method, since 2D representation is much more convenient. There are different kind of graphs, for example weighted graphs, unweighted graphs, unidirectional, bidirectional. We will discuss in brief about all these.

A coded image of a graph is given below:

```
#include<iostream>
using namespace std;
void addEdge(int arr[][5], int vertex1, int vertex2);

int main(){
  // A graph with 5 vertices can be initialized as the following 2D array with No edges
  int graph[5][5] = {
  {0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0}
  };

  //Example
  addEdge(graph, 0, 1); //Put edge between Node 0 and 1
  addEdge(graph, 2, 3); // Put edge between Node 2 and 3

  //Print the Graph
  for(int i=0;i<5;i++){
    for(int j=0;j<5;j++){
      cout<<graph[i][j]<<" ";
    }
    cout<<endl;
  }
}

// A function that will take in the graph, and 2 vertices and make an edge (connection) between them
void addEdge(int arr[][5], int vertex1, int vertex2){
  arr[vertex1][vertex2] = 1;
  arr[vertex2][vertex1] = 1;
}
```