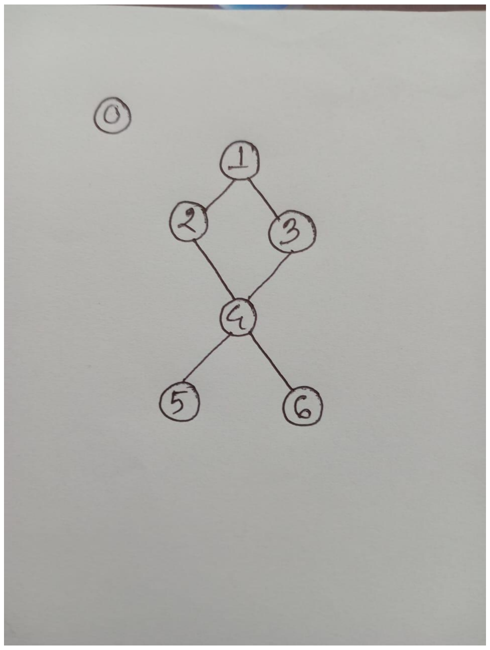
Answer to the question no-01

The matrix has 7 nodes.

Node connections are-> 1-3, 3-1, 1-2, 2-1, 2-4, 4-2, 3-4, 4-3, 4-5, 5-4, 4-6, 6-4.

It is an undirected and unweighted graph.

The graph is:



Adjacency list:

0->

1->2,3.

2->1,4.

3->1,4.

4->2,3,5,6.

5->4.

6->4.

Answer to the question no-02

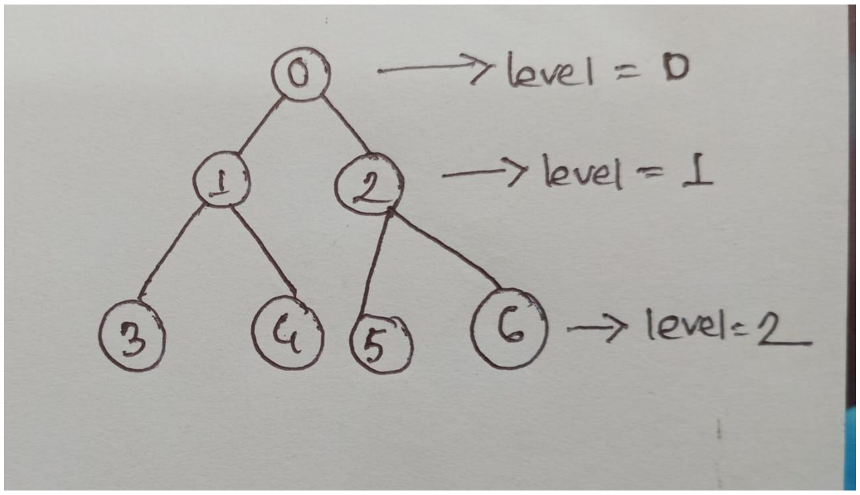
```
#include<bits/stdc++.h>
using namespace std;
int Pow(int n,int m)
{
   if(m==1)
   {
      return n;
   }
   return n*Pow(n,m-1);
}
int main()
{
        int n,m;
        cin>>n>>m;
        cout<<Pow(n,m);
}
```

Answer to the question no-03

BFS Algorithm	**DFS Algorithm**
1. Full form is Breadth First Search	1. Full form is Depth First Search
2. It is level wise traversal.	2. It is a pre-order traversal.
3. We have to use queue data structure to implement it	3. We have to use a recursive function to implement it.
4. It visits all the adjacent nodes.	4. It goes to the last end of a node.
5. It is useful for finding the smallest path.	5. It is useful for finding paths between 2 nodes.
6. It is only one type.	6. It has 3 types: pre-order, post-order, in-order.

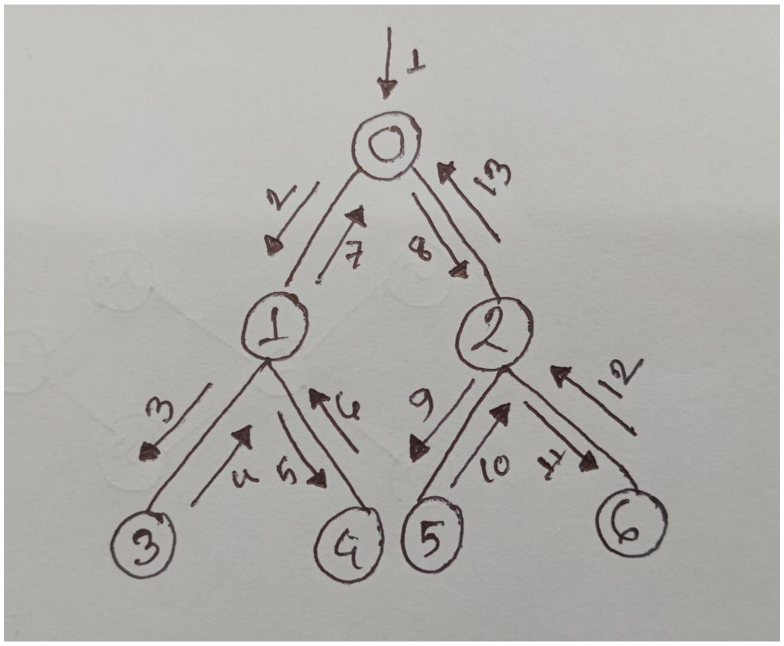
Answer to the question no-04

BFS stands for Breadth first search, which is an algorithm for traversal among nodes in a graph. It works level wise on a graph.
If I show a graph:



In level 0->node is 0;
In level 1->nodes are 1,2.
In level 2->nodes are 3,4,5,6.
As BFS traverses level wise first it goes level 0,then level 1,then level 2.
So, BFS traversal result will be - 0 1 2 3 4 5 6.

DFS stands for Depth First Search, which is an algorithm for traversal among nodes in a graph. It works pre-order wise traversal on a graph. It goes to a node then explores its branches and then explores that branch. It means that it goes to the last end of a node.
If I show a graph:



In a graph, it is noted that traversal by numbering. It starts from node 0, then explores branch node 1 and then goes to node 1. After that node 1 explores and finds node 3. Then

goes to node 3. Node 3 has no branch so it goes to the next branch of node 1 and explores this process step by step.
So, DFS traversal will be- 0 1 3 4 2 5 6.

Answer to the question no-05

Steps for BFS traversal output:

1. Source node is 2. So first it will store 2 in the queue and print 2.
2. By exploring node 2 we get node 1 3 6, stored in the queue.
3. After finishing node 2 exploration, pop the queue and we go to node 1 and explore it and store 4 5 in the queue it will become 1 3 6 4 . As node is visited it will not store 2 in the queue.
4. Print 1 and pop the queue. It will be 3 6 4 .
5. Then explore the node 3 and find that every branch of node 3 is visited so, print 3 and pop the queue. It will be 6 4.
6. Then we explore 6 and find every branch of node 6 is visited so, print 6 and pop the queue. It will be 4 .
7. Then explore 4 and push 5 in the queue. It will be 4,5. Because branches are already visited.
8. Print 4 and pop the queue. It will be 5.
9. Now explore 5 and find every branch of node 5 is visited so, print 5.pop the queue and finish the BFS traversal.

BFS traversal is **2 1 3 6 4 5.**

Steps for DFS traversal output:

1. Source node is 2. Print 2 and explore it.
2. Firstly we find 1, print 1 and explore it.
3. After that we find node 4 as a branch of node 1, print 4 and explore it.
4. We find node 3 as a branch of node 4, so, print 3 and explore it.
5. We find node 6 as a branch of node 3, so print 6 and explore it.
6. When exploring node 6 we see that node 2 and node 1 are already visited. So it returns to node 3. Node 3's branch node 2 is visited so it returns to node 4.
7. Node 4 has a branch of node 5, so print 5 and there is no branch of node 5. So it returns to node 4. Then it returns to root node 4 which is node 1. As every node is visited, it also returns to node 2 as it is a source node and finishes the DFS traversal.

DFS traversal is **2 1 4 3 6 5.**

When exploring the nodes it will go to the smallest node firstly as an adjacency matrix’s indices.