1. Problem Statement for Depth Limited Search: Design and develop a program in Python to print all the nodes reachable from a given starting node in a graph by using the Depth Limited Search method. Repeat the experiment for different Graphs.
   USN - 1NT18CS133

Algorithm: • The start node or node 1 is added to the beginning of the stack.
• Then it is marked as visited, and if node 1 is not the goal node in the search, then we push second node 2 on top of the stack.
• Next, we mark it as visited and check if node 2 is the goal node or not.
• If node 2 is not found to be the goal node, then we push node 4 on top of the stack.
• Now we search in the same depth limit and move along depth-wise to check for the goal nodes.
• If Node 4 is also not found to be the goal node and depth limit is found to be reached, then we retrace back to nearest nodes that remain unvisited or unexplored.
• Then we push them into the stack and mark them visited.
• We continue to perform these steps in iterative ways unless the goal node is reached or until all nodes within depth limit have been explored for the goal.

Depth-limited search is found to terminate under these two clauses:
• When the goal node is found to exist.
• When there is no solution within the given depth limit domain.

In [2]:
```python
from collections import defaultdict

class Graph:
    def __init__(self,vertices):
        self.V = vertices
        self.graph = defaultdict(list)

    def addEdge(self,u,v):
        self.graph[u].append(v)

    def DLS(self,source,target,maxDepth):
        if source == target : return True

        if maxDepth <= 0 : return False
           # recursively traversing the graph while searching
        for i in self.graph[source]:
                if(self.DLS(i, target, maxDepth-1)):
                    return True
        return False

g = Graph(9)# creating the graph
g.addEdge(0, 1)
g.addEdge(0, 2)
g.addEdge(1, 3)
g.addEdge(1, 4)
g.addEdge(2, 5)
g.addEdge(2, 6)
g.addEdge(3,7)
g.addEdge(3,8)

target = 3
maxDepth = 3
source = 0

if g.DLS(source, target, maxDepth) == True:
    print(f"Target {target} is reachable from source {source} within max depth {maxDepth}")
else:
    print(f"Target {target} is NOT reachable from source {source} within max depth {maxDepth}")
```

```
Target 3 is reachable from source 0 within max depth 3
```

In [ ]: