Iterative deepening dfs with using variable limit and return the path to traverse till input node

In [1]:
```
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,src,target,maxDepth):
        if src == target : return True
        if maxDepth <= 0 : return False

        for i in self.graph[src]:
                if(self.DLS(i,target,maxDepth-1)):
                    return True
        return False
    def IDDFS(self,src, target, maxDepth):
        for i in range(maxDepth):
            if (self.DLS(src, target, i)):
                return True
        return False


g = Graph (7);
g.addEdge(0, 1)
g.addEdge(0, 2)
g.addEdge(1, 3)
g.addEdge(1, 4)
g.addEdge(2, 5)
g.addEdge(2, 6)

target = int(input("enter the node to be searched "));
maxDepth = int(input("enter the depth "));
src = 0
found = 1
while(found):
    if g.IDDFS(src, target, maxDepth) == True:
        print ("Target is reachable from source " +
        "within max depth : ")
        print(maxDepth)
        found = 0
    else :
        maxDepth = maxDepth +1
```

```
enter the node to be searched 3
enter the depth 5
Target is reachable from source within max depth :
5
```

In [ ]:
```

```