```
class Node(object):
    def __init__(self, label: str=None):
        self.label = label
        self.children = []

    def __lt__(self,other):
        return (self.label < other.label)

    def __gt__(self,other):
        return (self.label > other.label)

    def __repr__(self):
        return '{} -> {}'.format(self.label, self.children)

    def add_child(self, node, cost=1):
        if type(node) is list:
            [self.add_child(sub_node) for sub_node in node]
            return
        edge = Edge(self, node, cost)
        self.children.append(edge)
class Edge(object):
    def __init__(self, source: Node, destination: Node, cost: int=1, bidirectional: bool=False):
        self.source = source
        self.destination = destination
        self.cost = cost
        self.bidirectional = bidirectional

    def __repr__(self):
        return '{}: {}'.format(self.cost, self.destination.label)



A = Node('A')
B = Node('B')
C = Node('C')
D = Node('D')
E = Node('E')
F = Node('F')
G = Node('G')


A.add_child([B, C, E])
B.add_child([A, D, F])
C.add_child([G, A])
D.add_child(B)
E.add_child([F, A])
F.add_child([E, B])
G.add_child(C)



_ = [print(node) for node in [A, B, C, D, E, F, G]]
```

```
 A -> [1: B, 1: C, 1: E]
 B -> [1: A, 1: D, 1: F]
 C -> [1: G, 1: A]
 D -> [1: B]
 E -> [1: F, 1: A]
 F -> [1: E, 1: B]
 G -> [1: C]
```

```
def iddfs(root: Node, goal: str, maximum_depth: int=10):
    for depth in range(0, maximum_depth):
        result = _dls([root], goal, depth)
        if result is None:
            continue
        return result

    raise ValueError('goal not in graph with depth {}'.format(maximum_depth))

def _dls(path: list, goal: str, depth: int):
    current = path[-1]
    if current.label == goal:
        return path
    if depth <= 0:
        return None
    for edge in current.children:
        new_path = list(path)
        new_path.append(edge.destination)
        result = _dls(new_path, goal, depth - 1)
        if result is not None:
            return result
```

```
iddfs(D, 'G')
```

```
     [D -> [1: B],
      B -> [1: A, 1: D, 1: F],
      A -> [1: B, 1: C, 1: E],
      C -> [1: G, 1: A],
      G -> [1: C]]
```


0s completed at 9:04 AM