Implement Iterative depth first search algorithm

Code:

```
class MyGraph:
  def __init__(self, cmap, i, g):
    self.citymap = cmap
    self.init = i
    self.goal = g
  def goal_test(self,anode):
    if anode==self.goal:
      return True
    else:
      return False
  def getLinks(self, anode):
    return list(self.citymap[anode].keys())

def recursive_dls2(node, citygraph, limit,nodelist):
  if citygraph.goal_test(node):
    return node
  elif limit == 0:
    return 'cutoff'
  else:
    cutoff_occurred = False
    for child in citygraph.getLinks(node):
      #print("child :", child)
      if not child in nodelist :
        nodelist.append(child)
        result = recursive_dls2(child, citygraph, limit - 1, nodelist)
        if result == 'cutoff':
          cutoff_occurred = True
        elif result is not None:
          return result
    return 'cutoff' if cutoff_occurred else 'Not found'

def depth_limited_search(citymap, limit=50):
  return recursive_dls2(citymap.init, citymap, limit,[citymap.init])

def iterative_deepening_search(citymap, limit):
  for depth in range(0,limit):
    print("checking with depth :", depth)
```

```
    result = depth_limited_search(citymap, depth)
    print("result : ", result)

# graph with cycles
romania_map =  {'Arad': {'Zerind': 75, 'Sibiu': 140, 'Timisoara': 118},
        'Bucharest': {'Urziceni': 85, 'Pitesti': 101, 'Giurgiu': 90, 'Fagaras': 211},
        'Craiova': {'Drobeta': 120, 'Rimnicu': 146, 'Pitesti': 138},
        'Drobeta': {'Mehadia': 75, 'Craiova': 120},
        'Eforie': {'Hirsova': 86},
        'Fagaras': {'Sibiu': 99, 'Bucharest': 211},
        'Hirsova': {'Urziceni': 98, 'Eforie': 86},
        'Iasi': {'Vaslui': 92, 'Neamt': 87},
        'Lugoj': {'Timisoara': 111, 'Mehadia': 70},
        'Oradea': {'Zerind': 71, 'Sibiu': 151},
        'Pitesti': {'Rimnicu': 97, 'Bucharest': 101, 'Craiova': 138},
        'Rimnicu': {'Sibiu': 80, 'Craiova': 146, 'Pitesti': 97},
        'Urziceni': {'Vaslui': 142, 'Bucharest': 85, 'Hirsova': 98},
        'Zerind': {'Arad': 75, 'Oradea': 71},
        'Sibiu': {'Arad': 140, 'Fagaras': 99, 'Oradea': 151, 'Rimnicu': 80},
        'Timisoara': {'Arad': 118, 'Lugoj': 111},
        'Giurgiu': {'Bucharest': 90},
        'Mehadia': {'Drobeta': 75, 'Lugoj': 70},
        'Vaslui': {'Iasi': 92, 'Urziceni': 142},
        'Neamt': {'Iasi': 87}}

print("----searching from arad to bucharest with level 6...")
romania_problem = MyGraph(romania_map, 'Arad','Bucharest' )
iterative_deepening_search(romania_problem, 6)

print("---searching from arad to neamt with level 2...")
romania_problem = MyGraph(romania_map, 'Arad','Neamt' )
iterative_deepening_search(romania_problem, 10)

print("---searching from arad to kurla with level 50...")
romania_problem = MyGraph(romania_map, 'Arad','Kurla' )
iterative_deepening_search(romania_problem, 7)
```

Output:

```
========================== RESTART: D:\TY\AI\prac2.py ==========================
----searching from arad to bucharest with level 6...
checking with depth : 0
result :  cutoff
checking with depth : 1
result :  cutoff
checking with depth : 2
result :  cutoff
checking with depth : 3
result :  cutoff
checking with depth : 4
result :  cutoff
checking with depth : 5
result :  Bucharest
---searching from arad to neamt with level 2...
checking with depth : 0
result :  cutoff
checking with depth : 1
result :  cutoff
---searching from arad to kurla with level 7...
checking with depth : 0
result :  cutoff
checking with depth : 1
result :  cutoff
checking with depth : 2
result :  cutoff
checking with depth : 3
result :  cutoff
checking with depth : 4
result :  cutoff
checking with depth : 5
result :  Not found
checking with depth : 6
result :  Not found
>
```