(7) * Implementing iterative deepening search algorithm.

Algorithm

⇒ class Graph:
    function __init__():
        graph = empty dictionary

function add_edge(u,v):
    if u is not in graph:
        graph[u] = empty list
    graph[u].append(v)

⇒ function depthSearch(source, target, limit):

if source == target:
    return True
if limit <= 0:
    return False
if source in graph:
    for each neighbor in graph[source]:
        if depthSearch(neighbor, target, limit-1):
            return True // found

return False // Not found

⇒ function iterativedep(src, target, max depth):

for depth in range(0, max depth + 1):
print("depth limit")

if depthsearch(src, target, depth):
print "Target" + target + "found at depth level" + depth.

return True.
return False.

⇒ function main():
graph = new Graph()

num-edges = input
print to enter edge pair (space separated)

for i in range(0, num-edges):
u, v = input("Enter edge:").split()
graph.add-edge(int(u), int(v)).

src = input("Enter the source node:")
target = input("Enter the target node:")
max-depth = input("Enter max depth limit")

if iterativedepening(int(src), int(target),
int(max-depth)):
print "Reachable target"
else
print "Not reachable target".

Output:

Enter the no. of edges in graph : 5
Enter each edge as a pair (source, destination):
0 1
0 2
1 3
1 4
2 5
Enter the source node: 0
Enter the target node : 5
Enter the max depth limit: 3

Checking with depth limit: 0
Checking with depth limit : 1
Checking with depth limit : 2
Target 5 found at depth level : 2

Target 5 is reachable from source 0 with depth 3.

Rm
13/10/24