LAB-2

Project Title:
8-puzzle

Algorithm:

function BFS (start state, goal state):
queue = [(start state, [ ])]
(starting state) ↳ (path)
visited = { }

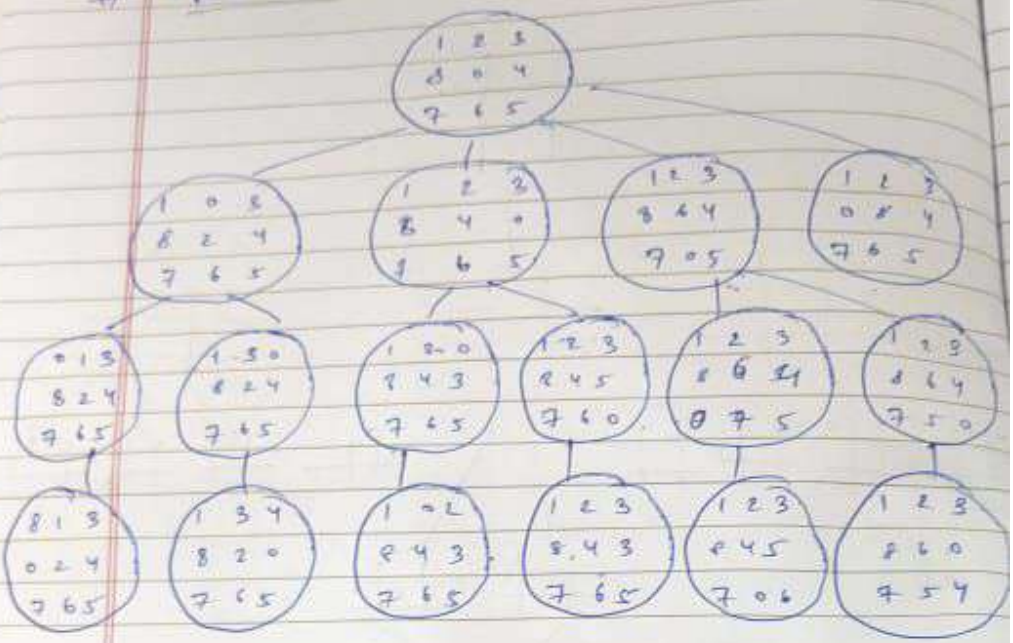
while queue is not empty:
current state, path = dequeue (queue)

if current state == goal state
return path
visited.add (current_state)

for each valid move:
new_state = apply move (current state, move)
if new state is not in visited
enqueue (queue, new state, path + [move])

Return None.

8 puzzle DFS



classmate
Date
Page 05

Project title :
Iterative deepening search

Algorithm :
function iterative deepening search (initial state, goal state, max depth)

for depth from 0 to max depth :
Set result = depth-limited-search (initial state, goal state, depth)
if result is not none, then
return result
if node.depth >= limit then
return none
for each child in expand (node):
Set result = depth-limited-search (child, goal state, limit)
if result is not none then
return result
return none

Set initial_state, goal_state, max_depth
Set solution = iterative_deepening_search (initial_state, goal state, max depth)
if solution is not none then print solution
else print "No solution found".