```
# Depth first search in search of target - Using Recursion
def dfs(src,target,limit,visited_states):

    if src==target:
        return True

    if limit<=0:
        return False

    visited_states.append(src)



    adj = possible_moves(src,visited_states)

    for move in adj:
        if dfs(move,target,limit-1,visited_states):
            return True
    return False

def possible_moves(state,visited_states):
    # Find index of empty spot and assign it to b

    b = state.index('-1')

    #'d' for down, 'u' for up, 'r' for right, 'l' for left - directions array
    d = []

    #Add all possible direction into directions array - Hint using if statements

    if b+3 in range(9):
        d.append('d')
```

```
    if b-3 in range(9):
        d.append('u')
    if b not in [0,3,6]:
        d.append('l')
    if b not in [2,5,8]:
        d.append('r')

    # If direction is possible then add state to move
    pos_moves = []

    # for all possible directions find the state if that move is played
    ### Jump to gen function to generate all possible moves in the given directions
    for move in d:
        pos_moves.append(gen(state,move,b))

    # return all possible moves only if the move not in visited_states
    return [move for move in pos_moves if move not in visited_states]



def gen(state, m, b): # m(move) is direction to slide, b(blank) is index of empty spot
    # create a copy of current state to test the move
    temp = state.copy()

    # if move is to slide empty spot to the left and so on

    if m=='d':
        a = temp[b+3]
        temp[b+3]=temp[b]
        temp[b]=a
    elif m=='u':
        a = temp[b-3]
```

```
        temp[b-3]=temp[b]
        temp[b]=a
    elif m=='l':
        a = temp[b-1]
        temp[b-1]=temp[b]
        temp[b]=a
    elif m=='r':
        a = temp[b+1]
        temp[b+1]=temp[b]
        temp[b]=a

    # return new state with tested move to later check if "src == target"
    return temp


def iddfs(src,target,depth):
    visited_states = []
    # Return Min depth at which the target was found
    for i in range(1, depth+1):
        if dfs(src, target, i, visited_states): return True
    return False


src = ['*','2','4','*','-1','*','*','3','1']
target = ['-1','1','2','3','4','*','*','*','*']
```

depth = 4

print(iddfs(src, target, depth))

```
src = ['*','2','4','*','-1','*','*','3','1']
target = ['-1','1','2','3','4','*','*','*','*']



depth = 4
print(iddfs(src, target, depth))
```

```
False
```