DFS

```
goal = [[1,2,3],[4,5,6],[7,8,0]]

def find_blank(s):
    for i in range(3):
        for j in range(3):
            if s[i][j] == 0: return i,j

def neighbors(s):
    x,y=find_blank(s)
    moves=[(1,0),(-1,0),(0,1),(0,-1)]
    result=[]
    for dx,dy in moves:
        nx,ny=x+dx,y+dy
        if 0<=nx<3 and 0<=ny<3:
            ns=[row[:] for row in s]
            ns[x][y],ns[nx][ny]=ns[nx][ny],ns[x][y]
            result.append(ns)
    return result

def dfs(start):
    stack=[(start,[start])]
    visited=set()
    while stack:
        state,path=stack.pop()
        if state==goal: return path
        visited.add(str(state))
        for nxt in neighbors(state):
            if str(nxt) not in visited:
                stack.append((nxt,path+[nxt]))
    return None

start=[[1,2,3],[4,0,6],[7,5,8]]
sol=dfs(start)
if sol:
    for step in sol:
        for row in step: print(row)
        print("----")
else:
    print("No solution")
```

```
[0, 5, 3]
```

```
[0, 5, 3]
[7, 8, 6]
----
[0, 1, 2]
[4, 5, 3]
[7, 8, 6]
----
[1, 0, 2]
[4, 5, 3]
[7, 8, 6]
----
[1, 2, 0]
[4, 5, 3]
[7, 8, 6]
----
[1, 2, 3]
[4, 5, 0]
[7, 8, 6]
----
[1, 2, 3]
[4, 5, 6]
[7, 8, 0]
----
```

IDDFS

```
def depth_limited_dfs(start,limit=5):
    stack=[(start,[start])]
    visited=set()
    while stack:
        state,path=stack.pop()
        if state==goal: return path
        if len(path)<limit:
            visited.add(str(state))
            for nxt in neighbors(state):
                if str(nxt) not in visited:
                    stack.append((nxt,path+[nxt]))
    return None

sol=depth_limited_dfs(start,limit=6)
if sol:
    for step in sol:
        for row in step: print(row)
        print("----")
else:
    print("No solution within depth limit")
```

```
[1, 2, 3]
[4, 0, 6]
[7, 5, 8]
----
[1, 2, 3]
[4, 5, 6]
[7, 0, 8]
----
[1, 2, 3]
[4, 5, 6]
[7, 8, 0]
----
```

```
def dls(state,limit):
    if state==goal: return [state]
    if limit<=0: return None
    for nxt in neighbors(state):
        res=dls(nxt,limit-1)
        if res: return [state]+res
    return None

def iddfs(start,max_depth=10):
    for depth in range(max_depth+1):
        result=dls(start,depth)
        if result: return result
    return None
sol=iddfs(start,6)
if sol:
    for step in sol:
        for row in step: print(row)
        print("----")
```

```
else:
    print("No solution within max depth")
```

```
[1, 2, 3]
[4, 0, 6]
[7, 5, 8]
----
[1, 2, 3]
[4, 5, 6]
[7, 0, 8]
----
[1, 2, 3]
[4, 5, 6]
[7, 8, 0]
----
```