**Source code:**

import heapq

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

def manhattan(s):

return sum(abs((val-1)//3 - i) + abs((val-1)%3 - j)

for i,row in enumerate(s) for j,val in enumerate(row) if val)

def neighbors(s):

i,j = next((i,j) for i in range(3) for j in range(3) if s[i][j]==0)

moves = []

for dx,dy in [(-1,0),(1,0),(0,-1),(0,1)]:

x,y = i+dx,j+dy

if 0<=x<3 and 0<=y<3:

new = [list(r) for r in s]

new[i][j], new[x][y] = new[x][y], new[i][j]

moves.append(tuple(tuple(r) for r in new))

return moves

def solve(start):

heap = [(manhattan(start), 0, start, [])]

seen = set ()

while heap:

est, cost, state, path = heapq. heappop(heap)

if state in seen: continue

seen.add(state)

if state == tuple(goal): return path + [state]

for n in neighbors(state):

heapq.heappush(heap, (cost+1+manhattan(n), cost+1, n, path+[state]))

# Example usage

start = ((1,2,3), (4,0,6), (7,5,8))

path = solve(start)

for step in path:

for row in step: print(row)

print ()

