A satr-8 puzzle

g=0 # keeps track of the number of moves taken so far.

def print\_board(elements):

for i in range(9):

if i%3 == 0:

print()

if elements[i]==-1:

print("\_", end = " ")

else:

print(elements[i], end = " ")

print()

# prints the board in a neat 3×3 grid. Replaces -1 with \_ for readability.

def solvable(start):

inv=0

for i in range(9):

if start[i] <= 1:

continue

for j in range(i+1,9):

if start[j]==-1:

continue

if start[i]>start[j]:

inv+=1

if inv%2==0:

return True

return False

# Checks if a given board configuration is solvable or not.or each pair of tiles, if a tile with a higher number comes before a tile with a lower number, it's counted as an inversion.If the total number of inversions is even → solvable.

def heuristic(start,goal):

global g

h = 0

for i in range(9):

for j in range(9):

if start[i] == goal[j] and start[i] != -1:

h += (abs(j-i))//3 + (abs(j-i))%3

return h + g

# Heuristic = sum of distances for each tile from its goal position.

def moveleft(start,position):

start[position],start[position-1]= start[position-1],start[position]

# Swap the empty space -1 with a neighboring tile moveleft orr right

def moveright(start,position):

start[position],start[position+1]= start[position+1],start[position]

def moveup(start,position):

start[position],start[position-3]= start[position-3],start[position]

def movedown(start,position):

start[position],start[position+3]= start[position+3],start[position]

def movetile(start,goal):

emptyat= start.index(-1)

row = emptyat//3

col = emptyat%3

t1,t2,t3,t4 = start[:],start[:],start[:],start[:]

f1,f2,f3,f4 = 100,100,100,100

# Finds the position of the empty tile.

if col -1 >=0:

moveleft(t1, emptyat)

f1 = heuristic(t1, goal)

if col+1<3:

moveright(t2, emptyat)

f2 = heuristic(t2, goal)

if row + 1 <3:

movedown(t3, emptyat)

f3 = heuristic(t3, goal)

if row-1>=0:

moveup(t4, emptyat)

f4 = heuristic(t4, goal)

min\_heuristic = min(f1, f2,f3,f4)

if f1==min\_heuristic:

moveleft(start, emptyat)

elif f2==min\_heuristic:

moveright(start, emptyat)

elif f3==min\_heuristic:

movedown(start, emptyat)

elif f4 == min\_heuristic:

moveup(start, emptyat)

def solveEight(start,goal):

global g

g+=1

movetile(start,goal)

print\_board(start)

f = heuristic(start,goal)

if f == g:

print("Solved in {} moves".format(f))

return

solveEight(start,goal)

# Increments g (number of moves). Calls movetile() to pick and make the best move.

def main():

global g

start = list()

goal = list()

print("Enter the start state:(Enter -1 for empty):")

for i in range(9):

start.append(int(input()))

print("Enter the goal state:(Enter -1 for empty):")

for i in range(9):

goal.append(int(input()))

print\_board(start)

# To check if solvable

if solvable(start):

solveEight(start,goal)

print("Solved in {} moves".format(g))

else:

print("Not possible to solve")

if \_\_name\_\_ == '\_\_main\_\_':

main()

# Enter the start state:(Enter -1 for empty):

# 1

# 2

# -1

# 4

# 5

# 3

# 7

# 8

# 6

# Enter the goal state:(Enter -1 for empty):

# 1

# 2

# 3

# 4

# 5

# 6

# 76

# 8

# -1

# 1 2 \_

# 4 5 3

# 7 8 6

# ...

# 4 5 6

# 7 8 \_

# Solved in 2 moves

# Solved in 2 moves