EXPERIMENT – 01 8-Puzzle (A\* search)

Aim:

Solve the 8-puzzle using A\* with Manhattan distance.

Code:

# 8puzzle\_a\_star.py

import heapq

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

def manhattan(state):

dist=0

for i,val in enumerate(state):

if val==0: continue

gi = (val-1)//3; gj=(val-1)%3

i0=i//3; j0=i%3

dist+=abs(gi-i0)+abs(gj-j0)

return dist

def neighbors(state):

s=list(state)

i=s.index(0)

r,c=divmod(i,3)

moves=[]

for dr,dc in [(1,0),(-1,0),(0,1),(0,-1)]:

nr,nc=r+dr,c+dc

if 0<=nr<3 and 0<=nc<3:

j=nr\*3+nc

t=list(s); t[i],t[j]=t[j],t[i]

moves.append(tuple(t))

return moves

def a\_star(start):

open\_heap=[(manhattan(start),0,start,None)]

came={}

g={start:0}

while open\_heap:

f,gcost,state,parent = heapq.heappop(open\_heap)

if state==goal:

path=[state]

p=parent

while p: path.append(p[0]); p=p[1]

return list(reversed(path))

if state in came: continue

came[state]=(parent)

for nb in neighbors(state):

ng=gcost+1

if nb not in g or ng<g[nb]:

g[nb]=ng

heapq.heappush(open\_heap,(ng+manhattan(nb),ng,nb,(state,parent)))

return None

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

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

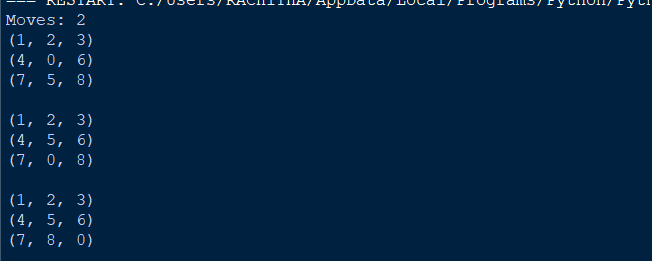
path=a\_star(start)

print("Moves:",len(path)-1)

for s in path:

print(s[:3]); print(s[3:6]); print(s[6:]); print()

Output:



Result:

8 puzzle is successfully implemented using python.