Ex. No: 03 **A\* Search**

24/08/2022

**Code**:

import copy

def printState\_8p(state,nv):

ctr = 0

for i in range(nv):

for j in range(nv):

if state[ctr] == 0:

print('0', end = ' ')

else:

print(state[ctr], end=' ')

ctr += 1

print()

def matrix\_to\_list(x, y,nv):

counter = 0

for i in range(nv):

for j in range(nv):

if i == x and j == y:

return counter

counter += 1

return 'Index does not exist!'

def list\_to\_matrix(x,nv):

counter = 0

for i in range(nv):

for j in range(nv):

if counter == x:

return i,j,x

counter += 1

return 'Index does not exist!'

def findVehicle(state,x,nv):

ctr = 0

for i in state:

if i == x:

return list\_to\_matrix(ctr,nv)

ctr += 1

return 'x not found!'

def swap(state, x1, y1, x2, y2,nv):

temp = state[matrix\_to\_list(x1, y1,nv)]

state[matrix\_to\_list(x1, y1,nv)] = state[matrix\_to\_list(x2, y2,nv)]

state[matrix\_to\_list(x2, y2,nv)] = temp

def actionsF(state,x,nv1):

vehicle=findVehicle(state,x,nv1)

l=r=u=d=0

validActions = []

if vehicle[1] != 0 and state[matrix\_to\_list(vehicle[0], vehicle[1]-1,nv1)]==0:

validActions.append('left')

l=1

if vehicle[1] != nv1-1 and state[matrix\_to\_list(vehicle[0], vehicle[1]+1,nv1)]==0:

validActions.append('right')

r=1

if vehicle[0] != 0 and state[matrix\_to\_list(vehicle[0]-1, vehicle[1],nv1)]==0:

validActions.append('up')

u=1

if vehicle[0] != nv1-1 and state[matrix\_to\_list(vehicle[0]+1, vehicle[1], nv1)]==0:

validActions.append('down')

d=1

if vehicle[1]-2 >= 0 and state[matrix\_to\_list(vehicle[0], vehicle[1]-2,nv1)]==0 and l==0:

validActions.append('l-hop')

if vehicle[1]+2 <= nv1-1 and state[matrix\_to\_list(vehicle[0], vehicle[1]+2,nv1)]==0 and r==0:

validActions.append('r-hop')

if vehicle[0]-2 >= 0 and state[matrix\_to\_list(vehicle[0]-2, vehicle[1],nv1)]==0 and u==0:

validActions.append('u-hop')

if vehicle[0]+2 <= nv1-1 and state[matrix\_to\_list(vehicle[0]+2, vehicle[1],nv1)]==0 and d==0:

validActions.append('d-hop')

return validActions

def takeActionF(state, action,x,nv):

vehicle = findVehicle(state,x,nv)

state2 = copy.copy(state)

if action == 'left':

swap(state2, vehicle[0], vehicle[1], vehicle[0], vehicle[1] - 1,nv)

if action == 'right':

swap(state2, vehicle[0], vehicle[1], vehicle[0], vehicle[1] + 1,nv)

if action == 'up':

swap(state2, vehicle[0], vehicle[1], vehicle[0] - 1, vehicle[1],nv)

if action == 'down':

swap(state2, vehicle[0], vehicle[1], vehicle[0] + 1, vehicle[1],nv)

if action == 'l-hop':

swap(state2, vehicle[0], vehicle[1], vehicle[0], vehicle[1] - 2,nv)

if action == 'r-hop':

swap(state2, vehicle[0], vehicle[1], vehicle[0], vehicle[1] + 2,nv)

if action == 'u-hop':

swap(state2, vehicle[0], vehicle[1], vehicle[0] - 2, vehicle[1],nv)

if action == 'd-hop':

swap(state2, vehicle[0], vehicle[1], vehicle[0] + 2, vehicle[1],nv)

return state2

def heuristic(start,goal,nv):

h=0

#sum of all the heuristic values

for i in range(nv):

x=findVehicle(start,i+1,nv)

y=findVehicle(goal,i+1,nv)

h+=(abs(x[0]-y[0])+abs(x[1]-y[1]))

return h

def astar(state, goalState, actionsF, takeActionF,nv):

open\_list = set()

closed\_list = set()

open\_list.add(tuple(state))

cost={}

g = {}

total=0

g[tuple(state)] = 0

parents = {}

parents[tuple(state)] = state

while len(open\_list) > 0:

n = None

for v in open\_list:

c=g[v] + heuristic(v,goalState,nv)

cost[tuple(v)]= c

if n == None or c < g[n] + heuristic(n,goalState,nv):

n = v;

if n == None:

print('Path does not exist!')

return None

if list(n) == goalState:

path = []

while parents[tuple(n)] != n:

path.append(n)

n = parents[tuple(n)]

path.reverse()

print('Path found: ')

for s in path:

printState\_8p(s,nv)

print("Cost->",cost[tuple(s)])

print("\nCost for all the states:")

for key,val in cost.items():

printState\_8p(key,nv)

print("Cost ->",val)

return path

for i in range(nv):

for action in actionsF(n,i+1,nv):

childState = takeActionF(list(n), action,i+1,nv)

if tuple(childState) not in open\_list and tuple(childState) not in closed\_list:

open\_list.add(tuple(childState))

parents[tuple(childState)] = n

g[tuple(childState)] = g[n] + 1

else:

if g[tuple(childState)] > g[n] + 1:

g[tuple(childState)] = g[n] + 1

parents[tuple(childState)] = n

if tuple(childState) in closed\_list:

closed\_list.remove(tuple(childState))

open\_list.add(tuple(childState))

open\_list.remove(n)

closed\_list.add(n)

print('Path does not exist!')

return None

def hasPath(startState, goalState, actionsF, takeActionF,l,n):

l = astar(startState, goalState, actionsF, takeActionF,n)

n=int(input("No of vehicles:"))

state=[0]\*(n\*n)

goalState=[0]\*(n\*n)

fill1=0

fill2=len(state)-1

for i in range(n):

state[fill1]=i+1

goalState[fill2]=i+1

fill1+=n

fill2-=n

print("starting state:",state)

print("Goal state: ",goalState)

print()

l=[]

hasPath(state, goalState, actionsF, takeActionF,l,n)

**Output**:







