Lab Assignment no # 03  
Course: Intro to Artificial Intelligence

Reg: FA22-BAI-014

DATE: JUNE,5-2023

From: HASEEB AHMED

To: Sir QAZI ZIA

****A-STAR Search Method****

import math

class Node:

def \_init\_ (self, state, parent, actions, heuristic, totalCost ):

self.state = state

self.parent = parent

self.actions = actions

self.totalCost = totalCost

self.heuristic = heuristic

def findMin(frontier):

minValue = math.inf

node = ''

for i in frontier:

if minValue > frontier[i][1]:

minValue = frontier[i][1]

node = i

return node

def actionSequence(graph, initialState, goalState):

solution = [goalState]

currentParent = graph[goalState].parent

while currentParent != None:

solution.append(currentParent)

currentParent = graph[currentParent].parent

solution.reverse()

return solution

def Astar():

initialState = 'A'

goalState = 'Y'

graph = {'A': Node('A', None, [('F',1)], (0,0), 0),

'B': Node('B', None, [('G',1), ('C',1)], (2,0), 0),

'C': Node('C', None, [('B',1), ('D',1)], (3,0), 0),

'D': Node('D', None, [('C',1), ('E',1)], (4,0), 0),

'E': Node('E', None, [('D',1)], (5,0), 0),

'F': Node('F', None, [('A',1), ('H',1)], (0,1), 0),

'G': Node('G', None, [('B',1), ('J',1)], (2,1), 0),

'H': Node('H', None, [('F',1), ('I',1), ('M',1)], (0,2), 0),

'I': Node('I', None, [('H',1), ('J',1), ('N',1)], (1,2), 0),

'J': Node('J', None, [('G',1), ('I',1)], (2,2), 0),

'K': Node('K', None, [('L',1), ('P',1)], (4,2), 0),

'L': Node('L', None, [('K',1), ('Q',1)], (5,2), 0),

'M': Node('M', None, [('H',1), ('N',1), ('R',1)], (0,3), 0),

'N': Node('N', None, [('I',1), ('M',1), ('S',1)], (1,3), 0),

'O': Node('O', None, [('P',1), ('U',1)], (3,3), 0),

'P': Node('P', None, [('O',1), ('Q',1)], (4,3), 0),

'Q': Node('Q', None, [('L',1), ('P',1), ('V',1)], (5,3), 0),

'R': Node('R', None, [('M',1), ('S',1)], (0,4), 0),

'S': Node('S', None, [('N',1), ('R',1), ('T',1)], (1,4), 0),

'T': Node('T', None, [('S',1), ('U',1), ('W',1)], (2,4), 0),

'U': Node('U', None, [('O',1), ('T',1)], (3,4), 0),

'V': Node('V', None, [('Q',1), ('Y',1)], (5,4), 0),

'W': Node('W', None, [('T',1)], (2,5), 0),

'X': Node('X', None, [('Y',1)], (4,5), 0),

'Y': Node('Y', None, [('V',1), ('X',1)], (5,5), 0)}

frontier = dict()

heuristicCost = math.sqrt(((graph[goalState].heuristic[0]- graph[initialState].heuristic[0])\*\*2)\

+((graph[goalState].heuristic[1]-graph[initialState].heuristic[1])\*\*2))

frontier[initialState] = (None, heuristicCost)

explored = dict()

while len(frontier) != 0:

currentNode = findMin(frontier)

print(currentNode)

del frontier[currentNode]

if graph[currentNode].state == goalState:

return actionSequence(graph, initialState, goalState)

heuristicCost = math.sqrt(((graph[goalState].heuristic[0]-graph[currentNode].heuristic[0])\*\*2)\

+((graph[goalState].heuristic[1]-graph[currentNode].heuristic[1])\*\*2))

currentCost = graph[currentNode].totalCost

explored[currentNode] = (graph[currentNode].parent, heuristicCost+currentCost)

for child in graph[currentNode].actions:

currentCost = child[1] + graph[currentNode].totalCost

heuristicCost = math.sqrt(((graph[goalState].heuristic[0]-graph[child[0]].heuristic[0])\*\*2)\

+((graph[goalState].heuristic[1]-graph[child[0]].heuristic[1])\*\*2))

if child[0] in explored:

if graph[child[0]].parent == currentNode or child[0]==initialState or explored[child[0]][1] <= currentCost + heuristicCost:

continue

if child[0] not in frontier:

graph[child[0]].parent = currentNode

graph[child[0]].totalCost = currentCost

frontier[child[0]] = (graph[child[0]].parent, currentCost + heuristicCost)

else:

if frontier[child[0]][1] < currentCost + heuristicCost:

graph[child[0]].parent = frontier[child[0]][0]

graph[child[0]].totalCost = frontier[child[0]][1] - heuristicCost

else:

frontier[child[0]] = (currentNode, currentCost + heuristicCost)

graph[child[0]].parent = frontier[child[0]][0]

graph[child[0]].totalCost = currentCost

sol = Astar()

print(sol)

**OUTPUT:**

A

F

H

I

J

M

N

R

S

T

U

G

W

O

P

B

Q

V

Y

['A', 'F', 'H', 'M', 'R', 'S', 'T', 'U', 'O', 'P', 'Q', 'V', 'Y']

## ****Machine Learning/Deep Learning****

import numpy as np

import matplotlib.pyplot as plt

x\_data=[1.0,2.0,3.0]

y\_data=[2.0,4.0,6.0]

w=1.0 #random guess

def forward(x):

return x\*w

def loss(x,y):

y\_pred=forward(x)

return (y\_pred-y)\*(y\_pred-y)

w\_list=[]

mse\_list=[]

for w in np.arange(0.0, 4.1, 0.1):

print("w=", w)

l\_sum=0

for x\_val,y\_val in zip(x\_data,y\_data):

y\_pred\_val=forward(x\_val)

l=loss(x\_val,y\_val)

l\_sum=l\_sum+l

print("\t",x\_val,y\_val,y\_pred\_val,l)

print("MSE=", l\_sum/3)

w\_list.append(w)

mse\_list.append(l\_sum/3)

plt.plot(w\_list,mse\_list)

plt.ylabel("loss")

plt.xlabel("w")

plt.show()

**Expert System**

|  |
| --- |
| % is used to add comments in code in prolog language |
|  | suggest(S) :- write('What is your personality type?: '),read(P),write('How is your mood?: '),read(M), song(S,\_,M,P). |
|  |  |
|  | %Happy\_Mood |
|  | song('https://www.youtube.com/watch?v=c8YIlU\_30Kk',jazz,M,P):- M = happy ,(P= (entj) ; P=(enfj) ; P=(enfp)),!. |
|  | song('https://www.youtube.com/watch?v=SsZRci3sA4I',classical,M,P):- M = happy ,(P= (entj) ; P=(intj) ; P=(entp) ; P=(infj)),!. |
|  | song('https://www.youtube.com/watch?v=XYk2kt8K6E0',electronica,M,P):- M = happy ,(P= (entj) ; P=(estp) ; P=(enfp)),!. |
|  | song('https://www.youtube.com/watch?v=VguED7BfpgU',metal,M,P):- M = happy ,(P= (intj) ; P=(istp) ; P=(intp) ; P=(estp)),!. |
|  | song('https://www.youtube.com/watch?v=5f-wQBh-zbQ',alternative\_rock,M,P):- M = happy ,(P= (intj) ; P=(entp) ; P=(infj) ; P=(infp) ; P=(istj) ; P=(isfj) ; P=(istp)),!. |
|  | song('https://www.youtube.com/watch?v=lPIiB02uqXM',rock,M,P):- M = happy ,(P= (entp) ; P=(intp) ; P=(infp) ; P=(istj) ; P=(isfj)),!. |
|  | song('https://www.youtube.com/watch?v=PIfJ7nYQFTM',punk,M,P):- M = happy ,(P= (intp) ; P=(infp) ; P=(istp)),!. |
|  | song('https://www.youtube.com/watch?v=eWyeAIlaYUY',world,M,P):- M = happy ,(P= (infj) ; P=(enfj)),!. |
|  | song('https://www.youtube.com/watch?v=qAqKsw4GjB0',blues,M,P):- M = happy ,(P= (enfj)),!. |
|  | song('https://www.youtube.com/watch?v=w47D1Fqn\_sA',ambient,M,P):- M = happy ,(P= (enfp) ; P=(isfp) ; P=(esfp)),!. |
|  | song('https://www.youtube.com/watch?v=HA06Rr3bRVc',pop\_songs,M,P):- M = happy ,(P= (istj) ; P=(estj) ; P=(isfp) ; P=(esfp)),!. |
|  | song('https://www.youtube.com/watch?v=qCZAynQU\_-8',religious,M,P):- M = happy ,(P= (isfj) ; P=(estj)),!. |
|  | song('https://www.youtube.com/watch?v=hvVPMIqRulE',hip\_hop,M,P):- M = happy ,(P= (estj) ; P=(estp) ; P=(esfp)),!. |
|  | song('https://www.youtube.com/watch?v=X7ses5rI5U4',soul,M,P):- M = happy ,(P= (esfj)),!. |
|  | song('https://www.youtube.com/watch?v=NKzyyxvNiFc',country,M,P):- M = happy ,(P= (esfj)),!. |
|  | song('https://www.youtube.com/watch?v=oWQpQW95Ru8',reggae,M,P):- M = happy ,(P= (isfp)),!. |
|  | %Sad\_Mood |
|  | song('https://www.youtube.com/watch?v=McxPJ3RYY4Y',jazz,M,P):- M = sad ,(P= (entj) ; P=(enfj) ; P=(enfp)),!. |
|  | song('https://www.youtube.com/watch?v=R6OElQVVlLo',classical,M,P):- M = sad ,(P= (entj) ; P=(intj) ; P=(entp) ; P=(infj)),!. |
|  | song('https://www.youtube.com/watch?v=ilTbMVG5t6M',electronica,M,P):- M = sad ,(P= (entj) ; P=(estp) ; P=(enfp)),!. |
|  | song('https://www.youtube.com/watch?v=SWkKvDD-Gu4',metal,M,P):- M = sad ,(P= (intj) ; P=(istp) ; P=(intp) ; P=(estp)),!. |
|  | song('https://www.youtube.com/watch?v=-fvBrKeobyA',alternative\_rock,M,P):- M = sad ,(P= (intj) ; P=(entp) ; P=(infj) ; P=(infp) ; P=(istj) ; P=(isfj) ; P=(istp)),!. |
|  | song('https://www.youtube.com/watch?v=qGxO2YNFj1o',rock,M,P):- M = sad ,(P= (entp) ; P=(intp) ; P=(infp) ; P=(istj) ; P=(isfj)),!. |
|  | song('https://www.youtube.com/watch?v=2MRdtXWcgIw',punk,M,P):- M = sad ,(P= (intp) ; P=(infp) ; P=(istp)),!. |
|  | song('https://www.youtube.com/watch?v=F90ymkS2dt4',world,M,P):- M = sad ,(P= (infj) ; P=(enfj)),!. |
|  | song('https://www.youtube.com/watch?v=6R9nWRWgl90',blues,M,P):- M = sad ,(P= (enfj)),!. |
|  | song('https://www.youtube.com/watch?v=sbX\_aElB2dI',ambient,M,P):- M = sad ,(P= (enfp) ; P=(isfp) ; P=(esfp)),!. |
|  | song('https://www.youtube.com/watch?v=T8-96tqFCFU&vl=en',pop\_songs,M,P):- M = sad ,(P= (istj) ; P=(estj) ; P=(isfp) ; P=(esfp)),!. |
|  | song('https://www.youtube.com/watch?v=K\_-e99oLp4Y',religious,M,P):- M = sad ,(P= (isfj) ; P=(estj)),!. |
|  | song('https://www.youtube.com/watch?v=0pZmHnj3-jQ',hip\_hop,M,P):- M = sad ,(P= (estj) ; P=(estp) ; P=(esfp)),!. |
|  | song('https://www.youtube.com/watch?v=KVIwQFEyZQE',soul,M,P):- M = sad ,(P= (esfj)),!. |
|  | song('https://www.youtube.com/watch?v=ygxmpkHgaC8',country,M,P):- M = sad ,(P= (esfj)),!. |
|  | song('https://www.youtube.com/watch?v=FqQjAUB6DJY',reggae,M,P):- M = sad ,(P= (isfp)),!. |
|  |  |
|  | %! is called cut and is used to reduce backtracking |