**IMPLEMENTATION OF UNIFORM COST SEARCH &**

**A\* SEARCH**

**EXERCISE:**

**Object 01:**

Write a program to implement uniform cost search & A\* for the graph given below.



**UNIFORM COST SEARCH**

**SOURCE CODE**

import re

def is\_word\_in\_text(word, text):

"""

Check if a word is in a text.

Parameters

----------

word : str

text : str

Returns

-------

bool : True if word is in text, otherwise False.

Examples

--------

>>> is\_word\_in\_text("Python", "python is awesome.")

True

>>> is\_word\_in\_text("Python", "camelCase is pythonic.")

False

>>> is\_word\_in\_text("Python", "At the end is Python")

True

"""

pattern = r'(^|[^\w]){}([^\w]|$)'.format(word)

pattern = re.compile(pattern, re.IGNORECASE)

matches = re.search(pattern, text)

return bool(matches)

list1={'S':['A','B'],'A':['B','C','D'],'B':'C','C':'D'}

path={'SB':4,'SA':1,'AB':2,'BC':2,'CD':3,'AC':5,'AD':12}

heuristic={'S':7,'A':6,'B':2,'C':1,'D':0}

initial='S'

initial+=' 0'

goal='D'

openlist=[initial]

closelist=[]

a=1;

print("Openlist\t\t\t\t\t\t\t\t\t\tCloselist")

print(openlist,"\t\t\t\t\t\t\t\t\t\t\t",closelist);

while a:

no=[]

ab=[]

for b in openlist:

v=''

for c in b:

if (c=='0') | (c=='1') | (c=='2') | (c=='3') | (c=='4') | (c=='5') | (c=='6') | (c=='7') | (c=='8') | (c=='9'):

v+=c

v=int(v)

no.append(v)

no.sort()

x=no.pop(0)

x=(str)(x)

for b in openlist:

c=is\_word\_in\_text(x,b)

if c==True:

ab.append(b)

x2=ab.pop(0)

x3=openlist.index(x2)

x4=openlist.pop(x3)

closelist.append(x4)

if x4.find(goal)>0:

break

x5=''

a1=0

for c in x4:

if c==' ':

break;

x5+=c

a1+=1

a1-=1

l1=list(x5)

l2=l1[a1]

key=list1[l2]

for c in key:

con=x5+c

l3=list(con)

p1=0

p2=1

cost=0

for b in range(len(con)):

if p1>len(con)-1:

break

if p2>len(con)-1:

break

if p1>p2:

a=l3[p2]+l3[p1]

else:

a=l3[p1]+l3[p2]

cost+=path[a]

if b%2==0:

p1+=2

else:

p2+=2

openlist.append(con+' '+str(cost))

print(openlist, "\t\t\t\t\t", closelist);

print(openlist, "\t\t\t\t\t", closelist);

**OUTPUT**

Openlist Closelist

['S 0'] []

['SA 1', 'SB 4'] ['S 0']

['SB 4', 'SAB 3', 'SAC 6', 'SAD 13'] ['S 0', 'SA 1']

['SB 4', 'SAC 6', 'SAD 13', 'SABC 5'] ['S 0', 'SA 1', 'SAB 3']

['SAC 6', 'SAD 13', 'SABC 5', 'SBC 6'] ['S 0', 'SA 1', 'SAB 3', 'SB 4']

['SAC 6', 'SAD 13', 'SBC 6', 'SABCD 8'] ['S 0', 'SA 1', 'SAB 3', 'SB 4', 'SABC 5']

['SAD 13', 'SBC 6', 'SABCD 8', 'SACD 9'] ['S 0', 'SA 1', 'SAB 3', 'SB 4', 'SABC 5', 'SAC 6']

['SAD 13', 'SABCD 8', 'SACD 9', 'SBCD 9'] ['S 0', 'SA 1', 'SAB 3', 'SB 4', 'SABC 5', 'SAC 6', 'SBC 6']

['SAD 13', 'SACD 9', 'SBCD 9'] ['S 0', 'SA 1', 'SAB 3', 'SB 4', 'SABC 5', 'SAC 6', 'SBC 6', 'SABCD 8']

**A\* SEARCH**

**SOURCE CODE**

import re

def is\_word\_in\_text(word, text):

"""

Check if a word is in a text.

Parameters

----------

word : str

text : str

Returns

-------

bool : True if word is in text, otherwise False.

Examples

--------

>>> is\_word\_in\_text("Python", "python is awesome.")

True

>>> is\_word\_in\_text("Python", "camelCase is pythonic.")

False

>>> is\_word\_in\_text("Python", "At the end is Python")

True

"""

pattern = r'(^|[^\w]){}([^\w]|$)'.format(word)

pattern = re.compile(pattern, re.IGNORECASE)

matches = re.search(pattern, text)

return bool(matches)

list1={'S':['A','B'],'A':['B','C','D'],'B':'C','C':'D'}

path={'SB':4,'SA':1,'AB':2,'BC':2,'CD':3,'AC':5,'AD':12}

heuristic={'S':7,'A':6,'B':2,'C':1,'D':0}

initial='S'

initial+=' 0'

goal='D'

openlist=[initial]

closelist=[]

a=1;

print("Openlist\t\t\t\t\t\tCloselist")

print(openlist,"\t\t\t\t\t",closelist);

while a:

no=[]

ab=[]

for b in openlist:

v=''

for c in b:

if (c=='0') | (c=='1') | (c=='2') | (c=='3') | (c=='4') | (c=='5') | (c=='6') | (c=='7') | (c=='8') | (c=='9'):

v+=c

v=int(v)

no.append(v)

no.sort()

x=no.pop(0)

x=(str)(x)

for b in openlist:

c=is\_word\_in\_text(x,b)

if c==True:

ab.append(b)

x2=ab.pop(0)

x3=openlist.index(x2)

x4=openlist.pop(x3)

closelist.append(x4)

if x4.find(goal)>0:

break

x5=''

a1=0

for c in x4:

if c==' ':

break;

x5+=c

a1+=1

a1-=1

l1=list(x5)

l2=l1[a1]

key=list1[l2]

for c in key:

con=x5+c

l3=list(con)

p1=0

p2=1

cost=0

cost+=heuristic[c]

for b in range(len(con)):

if p1>len(con)-1:

break

if p2>len(con)-1:

break

if p1>p2:

a=l3[p2]+l3[p1]

else:

a=l3[p1]+l3[p2]

cost+=path[a]

if b%2==0:

p1+=2

else:

p2+=2

openlist.append(con+' '+str(cost))

print(openlist, "\t\t\t\t\t", closelist);

print(openlist, "\t\t\t\t\t", closelist);

**OUTPUT**

Openlist Closelist

['S 0'] []

['SA 7', 'SB 6'] ['S 0']

['SA 7', 'SBC 7'] ['S 0', 'SB 6']

['SBC 7', 'SAB 5', 'SAC 7', 'SAD 13'] ['S 0', 'SB 6', 'SA 7']

['SBC 7', 'SAC 7', 'SAD 13', 'SABC 6'] ['S 0', 'SB 6', 'SA 7', 'SAB 5']

['SBC 7', 'SAC 7', 'SAD 13', 'SABCD 8'] ['S 0', 'SB 6', 'SA 7', 'SAB 5', 'SABC 6']

['SAC 7', 'SAD 13', 'SABCD 8', 'SBCD 9'] ['S 0', 'SB 6', 'SA 7', 'SAB 5', 'SABC 6', 'SBC 7']

['SAD 13', 'SABCD 8', 'SBCD 9', 'SACD 9'] ['S 0', 'SB 6', 'SA 7', 'SAB 5', 'SABC 6', 'SBC 7', 'SAC 7']

['SAD 13', 'SBCD 9', 'SACD 9'] ['S 0', 'SB 6', 'SA 7', 'SAB 5', 'SABC 6', 'SBC 7', 'SAC 7', 'SABCD 8']