****

**Ing. En Ciencias De La Computación**

**Sistemas de bbúsqueda y razonamiento**

**Busqueda Local**

326820 Domínguez Márquez Erick Rolando

31/10/2020

Ricardo Martínez Ricardo

DESCRIPCIÓN:

En este tipo de busquedas intentamos buscar el estado mas efectivo según el criterio de cada una de las tres busquedas e imprimimos el camino, se representa el grafo en un arreglo con sus uniones y sus costos según cada nodo para despues ser enviado a las funciones necesarias que devolveran un resultadoA screenshot of a computer screen

Description automatically generated

CODIGO:

import random as rnd

import numpy as np

## Esta funcion regresa a los hijos de un estado

def getHijos(edo,proble):

return proble[edo]

## Esto regresa al mejor estado en la lista

def siguiente(lista ,hs):

hmejor =1000

mejor= ''

for edo in lista:

if hs[edo]<hmejor:

hmejor=hs[edo]

mejor=edo

return mejor,hmejor

## Esto implementa la busqueda por ascenso de colina

def ascCol(edo,hedo,proble,hs):

hijos=getHijos(edo,proble)

nuevo ,hnuevo=siguiente(hijos ,hs)

if hnuevo >hedo:

soln=edo

else:

soln=ascCol(nuevo ,hnuevo ,proble ,hs)

return soln

def siguientet(lEdos ,lTabu ,hs):

hmejor=1000

mejor=' '

listo=False

indi=0

donde=0

for edo in lEdos:

if hs[edo]<hmejor:

mejor=edo

hmejor=hs[edo]

donde=indi

indi=indi+1

if mejor not in lTabu:

soln=(mejor ,hmejor)

else:

del lEdos[donde]

soln=siguientet(lEdos ,lTabu ,hs)

return soln

def tabu(actual ,lTabu ,proble ,hs):

print(actual)

hactual=hs[actual]

hijos=getHijos(actual ,proble)

nuevo,hnuevo=siguientet(hijos ,lTabu ,hs)

if hactual <hnuevo:

soln=actual

else:

lTabu.append(actual)

soln=tabu(nuevo,lTabu,proble,hs)

return soln

def siguienteSim(lista ,hs):

hmejor =1000

mejor=' '

donde=0

indi=0

for edo in lista:

if hs[edo]<hmejor:

mejor=edo

hmejor=hs[edo]

donde=indi

indi=indi+1

lista =[]

return lista ,mejor ,hmejor

## Esto implementa el templado simulado

def temSim(actual ,k,T,lista ,proble ,hs):

hactual=hs[actual]

if k>35:

soln=actual

else:

lista=getHijos(actual ,proble)

lista ,mejor ,hmejor=siguienteSim(lista ,hs)

if hmejor >hactual:

p=rnd.uniform (0,1)

razon=np.exp((hactual -hmejor)/T)

if p<razon:

nuevo=mejor

else:

nuevo=actual

else:

nuevo=mejor

soln=temSim(nuevo ,k+1 ,0.75\*T,lista ,proble ,hs)

return soln

def main():

proble={'A':['B','C','D','E','F'],'B':['A','C'],'C':['A','B','D','G','H'],'D':['A','C','E','I'],'E':['A','D','F','J','K'],'F':['A','E'],'G':['C','H'],'H':['G','C','I','L'],'I':['D','J','N','M','L','H'],'J':['K','E','Ñ','I'],'K':['J','E'],'L':['M','I','H','O'],'M':['I','N','O','L'],'N':['I','M','Ñ','O'],'Ñ':['J','N'],'O':['L','M','N']}

hs={'A':10,'B':11,'C':10.5,'D':10.25,'E':8,'F':12,'G':9,'H':7.5,'I':10,'J':12.25,'K':12,'L':9,'M':7,'N':10,'Ñ':11,'O':8.5}

print("{'A':10,'B':11,'C':10.5,'D':10.25,'E':8,'F':12,'G':9,'H':7.5,'I':10,'J':12.25,'K':12,'L':9,'M':7,'N':10,'Ñ':11,'O':8.5} ")

inicial=input("INICIO: ")

tam=int(input("TAMAÑO: "))

soln=tabu(inicial,[],proble ,hs)

print('SOLUCION TABU:'+soln)

solnc=ascCol(inicial,tam,proble,hs)

print('SOLUCION DE COLINA:'+solnc)

solns=temSim(inicial,0,45,[],proble ,hs)

print('SOLUCION SIMULADO:'+solns)

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

main()