In [9]:

```
from neo4j import GraphDatabase
import bs4
from bs4 import BeautifulSoup as soup
from urllib.request import Request, urlopen
from termcolor import colored
#pip install termcolor
```

In [10]:

```
#CLASE PAR CREAR NODO CENTRAR-PARQUE CENTRAL
class CLASE_NEO4J(object):
    def __init__(self):
        self._driver = GraphDatabase.driver("bolt:neo4j://localhost:7687", auth=("neo4j
    def close(self):
        self._driver.close()
    def CREAR_LUGAR(self, message,lugar,latitud,longitud):
        with self._driver.session() as session:
            greeting = session.write_transaction(self._VALIDAR_LUGAR, message,lugar,lat
            print(greeting)
    def CREAR_RUTA(self,origen,destino,costo,hn):
        with self._driver.session() as session:
            greeting2 = session.write_transaction(self._VALIDAR_RUTA,origen,destino,cos
            print(greeting2)
    def CREAR_LUGAR_KNN(self, message,lugar,latitud,longitud,estrellas,infectados,tipo)
        with self._driver.session() as session:
            greeting = session.write_transaction(self._VALIDAR_LUGAR_KNN, message,lugar
            print(greeting)

    #METODO PARA CREAR LOS NODOS DE LUGARES PARA ALGORITMO COSTO
    @staticmethod
    def _VALIDAR_LUGAR(tx, message,lugar,latitud,longitud):
        #SE BUSCA SI EL LUGAR DEL ARREGLO EXISTE EN LA BASE NEO4J
        result2 = tx.run("match(l:Lugares {nombre:'"+lugar+"',latitud:"+latitud+",longi
        #CONDICION PARA VERIFICAR SI EXISTE
        if int(len(result2)) == 0:
            print("SE CREA EL LUGAR EN LA BASE.....")
            #SE CREA NODO LUGAR
            result = tx.run("CREATE("+lugar+":Lugares {nombre:'"+lugar+"',latitud:"+lat
                        "SET "+lugar+".message = $message "
                        "RETURN "+lugar+".message + ', from node ' + id("+lugar+")", me
        elif int(len(result2))  == 1:
            print("EL NODO LUGAR YA EXISTE, INGRESAR OTRO LUGAR........")

    #METODO PARA CREAR LOS NODOS DE LUGARES PARA EL ALGORITMO KNN
    @staticmethod
    def _VALIDAR_LUGAR_KNN(tx, message,lugar,latitud,longitud,estrellas,infectados,tipo
        #SE BUSCA SI EL LUGAR DEL ARREGLO EXISTE EN LA BASE N,estrellas,infectadosEO4J
        result2 = tx.run("match(l:"+tipo+" {nombre:'"+lugar+"',latitud:"+latitud+",long
        #CONDICION PARA VERIFICAR SI EXISTE
        if int(len(result2)) == 0:
            print("SE CREA EL LUGAR EN LA BASE.....")
            #SE CREA NODO LUGAR
            result = tx.run("CREATE("+lugar+":"+tipo+" {nombre:'"+lugar+"',latitud:"+la
                        "SET "+lugar+".message = $message "
                        "RETURN "+lugar+".message + ', from node ' + id("+lugar+")", me
        elif int(len(result2))  == 1:
            print("EL NODO LUGAR YA EXISTE, INGRESAR OTRO LUGAR........")

    #METODO PARA CREAR LAS RELACIONES CON EL COSTE  Y HN PARA LA RUTA
    @staticmethod
    def _VALIDAR_RUTA(tx,origen,destino,costo,hn):
        #SE BUSCA SI LA RUTA A CREAR YA DEL ARREGLO EXISTE EN LA BASE NEO4J
        result = tx.run("match(l1:Lugares{nombre:'"+origen+"'})-[r:RUTA_DE{costo:"+str(
        if int(len(result)) == 0:
            print("SE CREA LOS NODOS DE RELACION DE RUTAS ENTRE LOS LUGARES .....
            result2 = tx.run(" match("+origen+":Lugares {nombre:'"+origen+"'}) match("+
        elif int(len(result)) == 1:
            print(colored("YA EXISTE LA RUTA**********", 'green'))
```

```
60 # MATCH (n) OPTIONAL MATCH (n)-[r]-() DELETE n,r
61 #SE INICIALIZA LA CLASE DE LOS METODOS DE NEO4J
62 grafo=CLASE_NEO4J()
```

```
      2 class CLASE_NEO4J(object):
      3     def __init__(self):
----> 4         self._driver = GraphDatabase.driver("bolt:neo4j://localhos
t:7687", auth=("neo4j", "neo4jj"), encrypted=False)
      5     def close(self):
      6         self._driver.close()

~\anaconda3\lib\site-packages\neo4j\__init__.py in driver(cls, uri, auth,
 **config)
    181
    182         if driver_type == DRIVER_BOLT:
--> 183             return cls.bolt_driver(parsed.netloc, auth=auth, **con
fig)
    184         elif driver_type == DRIVER_NEO4j:
    185             routing_context = parse_routing_context(parsed.query)

~\anaconda3\lib\site-packages\neo4j\__init__.py in bolt_driver(cls, targe
t, auth, **config)
    194
    195         try:
```

In [8]:

```
#SE CREA LA LISTA DE NODOS LUGARES
listaL = (["HOTEL_MONTECARLO",-79.16885,-09.17885,2,10],
          ["PARQUE_CENTRAL",-0.22214,-0.34216,3,20],
          ["RESTAURANTE_CODELL",-0.2975,-79.16784,4,11],
          ["RESTAURANTE_PICHINCHA",-0.24665,-79.19275,1,9],
          ["RESTAURANTE_CHURRERIA",-0.2975,-79.1765,2,2],
          ["RESTAURANTE_QUINTA_DORADA",-0.23421,-79.16772,5,23],
          ["RESTAURANTE_CORBAN_COFFEE",-0.27227,-79.1402,3,15],
          ["RESTAURANTE_TENTACIONES",-0.25516,-79.18057,2,16],
          ["RESTAURANTE_DELIRIO",-0.25487,-79.1854,4,20],
          ["HOTEL_IMPERIAL",-0.24886, -79.15787,3,25],
          ["HOTEL_ESTACION",-0.25307,-79.17365,1,25],
          ["RESTAURANTE_BOM",-0.24781,-79.15229,3,14],
          ["RESTAURANTE_PAOLA",-0.26004,-79.16534,5,19],
          ["RESTAURANTE_CHILENO",-0.2476,-79.15319,3,23],
          ["RESTAURANTE_NELLY",-0.24873,-79.15046,2,34],
          ["HOTEL_RIOBAMBA",-0.2473,-79.15811,4,22],
          ["HOTEL_ROCIO",-0.24528,-79.18359,1,26],
          ["HOTEL_HUASI",-0.24936,-79.16421,4,27],
          ["HOTEL_TORIL",-0.2478,-79.15211,2,15],
          ["HOTEL_PRIMAVERA",-0.24848,-79.15261,3,6],
          ["HOTEL_GABRIEL",-0.24528,-79.15811,5,30],
          ["HOTEL_SOL",-0.26004,-79.16534,4,25],
          ["HOTEL_INTERNACIONAL",-0.24525,-79.16612,3,25],
          ["HOTEL_ACACIAS",-0.24986,-79.16322,3,45],
          ["HOTEL_MAYA",-0.24525,-79.16612,3,21],
          ["HOTEL_TROJE",-0.26004,-79.17629,2,45],
          ["HOTEL_PERUANA",-0.2473,-79.15811,1,23],
          ["HOTEL_MOLINO",-0.24528,-79.16534,5,25],
          ["HOTEL_SHALOM",-0.24525,-79.16322,2,15],
          ["RESTAURANTE_TAURO",-0.24759,-79.15341,4,26],
          ["HOTEL_CHACARERO",-0.24528,-79.18359,3,45],
          ["RESTAURANTE_JEKE",-0.25337,-79.15229,2,32],
          ["RESTAURANTE_XIMENITA",-0.24759,-79.19456,1,32],
          ["RESTAURANTE_ROSITA",-0.25077,-79.16322,3,28],
          ["HOTEL_DORADO",-0.24986,-79.15261,2,35],
          ["RESTAURANTE_SABOR",-0.2473,-79.15811,1,32],
          ["RESTAURANTE_BOEMIA",-0.26004,-79.15046,3,24],
          ["RESTAURANTE_PILSENER",-0.24848,-79.16322,4,37],
          ["RESTAURANTE_ANDESBULL",-0.25578,-79.18003,1,45],
          ["RESTAURANTE_QUINCHOS",-0.25197,-79.15253,2,36],
          ["RESTAURANTE_NORTENO",-0.24936,-79.18003,3,56],
          ["RESTAURANTE_YAGUAR",-0.24886,-79.1402,4,23],
          ["RESTAURANTE_BURGER",-0.25382,-79.16772,5,67],
          ["RESTAURANTE_LINA",-0.24759,-79.13482,3,43],
          ["RESTAURANTE_COFFEREX",-0.26618,-79.11198,4,76],
          ["RESTAURANTE_BONNY",-0.25337,-79.1681,14])
for ll in listaL:
    #SE INICIA EL METODO DE GENERAR NODOS LUGARES
    print(str(ll[0]),str(ll[1]),str(ll[2]))
    grafo.CREAR_LUGAR("NODO CREADO",str(ll[0]),str(ll[1]),str(ll[2]))

# SE GENERA LAS RELACIONES DE LOS LUGARES
#SE CREA LA LISTA DE LAS RELACIONES Y LOS NODOS
listaL = (['PARQUE_CENTRAL','HOTEL_MONTECARLO',0.10371,0.350],
          ['PARQUE_CENTRAL','RESTAURANTE_CODELL',0.9416,0.600],
          ['RESTAURANTE_CODELL','RESTAURANTE_PICHINCHA',0.11894,0.130],
          ['HOTEL_MONTECARLO','RESTAURANTE_CHURRERIA',0.6768,0.9100],
          ['HOTEL_MONTECARLO','RESTAURANTE_QUINTA_DORADA',0.4681,0.3500],
```

```
            ['RESTAURANTE_CORBAN_COFFEE','RESTAURANTE_TENTACIONES',0.5173,0.7700],
            ['RESTAURANTE_CORBAN_COFFEE','HOTEL_LIBERTADOR',0.1041,0.1100],
            ['RESTAURANTE_CORBAN_COFFEE','RESTAURANTE_DELIRIO',0.10271,0.350],
            ['HOTEL_LIBERTADOR','HOTEL_IMPERIAL',0.37,0.350],
            ['HOTEL_LIBERTADOR','HOTEL_ESTACION',0.121,0.400],
            ['RESTAURANTE_CORBAN_COFFEE','RESTAURANTE_BOM',0.5184,0.350],
            ['RESTAURANTE_BOM','RESTAURANTE_PAOLA',0.184,0.190],
            ['RESTAURANTE_PICHINCHA','RESTAURANTE_CHILENO',0.131,0.500],
            ['RESTAURANTE_PICHINCHA','RESTAURANTE_NELLY', 0.51,0.290],
            ['RESTAURANTE_NELLY','RESTAURANTE_DELIRIO', 0.115,0.130],
            ['RESTAURANTE_CHILENO','RESTAURANTE_NELLY', 0.116,0.500],
            ['RESTAURANTE_DELIRIO','HOTEL_DORADO', 0.101,0.140],
            ['RESTAURANTE_DELIRIO','HOTEL_LIBERTADOR', 0.121,0.550],
            ['HOTEL_LIBERTADOR','HOTEL_DORADO', 0.117,0.270],
            ['HOTEL_DORADO','HOTEL_ESTACION', 0.131,0.140],
            ['RESTAURANTE_CHILENO','HOTEL_DORADO', 0.294,0.450],
            ['RESTAURANTE_PAOLA','HOTEL_LIBERTADOR', 0.150,0.250],
            ['RESTAURANTE_PAOLA','HOTEL_ESTACION', 0.191,0.450],
            ['HOTEL_RIOBAMBA','RESTAURANTE_CHURRERIA', 0.209,0.650],
            ['RESTAURANTE_RIOBAMBA','RESTAURANTE_PAOLA', 0.5300,0.6400],
            ['RESTAURANTE_QUINTA_DORADA','RESTAURANTE_PAOLA', 0.187,0.400],
            ['RESTAURANTE_CHURRRIA','HOTEL_ESTACION', 0.338,0.400],
            ['RESTAURANTE_CODELL','HOTEL_DORADO', 0.358,0.450],
            ['HOTEL_ESTACION','PARQUE_CENTRAL',0.351,0.450],
            ['RESTAURANTE_NELLY','RESTAURANTE_TENTACIONES', 0.158,0.170],
            ['RESTAURANTE_DELIRIO','RESTAURANTE_PAOLA', 0.263,0.500],
             ['HOTEL_RIOBAMBA','RESTAURANTE_LINA', 0.205,0.500],
             ['RESTAURANTE_LINA','RESTAURANTE_PAOLA', 2.41,3.30],
             ['RESTAURANTE_LINA','RESTAURANTE_BUGER', 2.6,2.5],
             ['RESTAURANTE_CHURRERIA','RESTAURANTE_NORTENO', 1.04,1.9],
             ['RESTAURANTE_NORTENO','RESTAURANTE_LINA',0.189,0.260],
             ['RESTAURANTE_QUINTA_DORADA','RESTAURANTE_NORTENO', 0.189,0.260],
             ['HOTEL_IMPERIAL','RESTAURANTE_TENTACIONES', 0.95,0.400],
             ['RESTAURANTE_CHURRERIA','RESTAURANTE_YAGUAR', 0.699,0.850],
             ['RESTAURANTE_YAGUAR','RESTAURANTE_BONNY', 1.33,2.1],
             ['RESTAURANTE_YAGUAR','RESTAURANTE_BURGER', 1.26,2.0],
             ['RESTAURANTE_PILSENER','RESTAURANTE_YAGUAR', 1.30,1.5],
             ['RESTAURANTE_PILSENER','RESTAURANTE_QUINCHOS', 1.23,2.0],
             ['HOTEL_MONTECARLO','RESTAURANTE_PILSENER',0.654,0.800],
            ['RESTAURANTE_BURGER','HOTEL_RIOBAMBA',0.6768,0.9100],
            ['HOTEL_MONTECARLO','RESTAURANTE_CHURRERIA',0.6768,0.9100],
            ['RESTAURANTE_BONNY','RESTAURANTE_BURGER',0.6768,0.9100],
            ['RESTAURANTE_BONNY','RESTAURANTE_QUINCHOS',0.6768,0.9100],
            ['HOTEL_IMPERIAL','RESTAURANTE_CHILENO',0.284,0.500],
            ['RESTAURANTE_HUASI','HOTEL_IMPERIAL',0.676,0.650],
            ['RESTAURANTE_QUINCHOS','RESTAURANTE_BOEMIA',0.6768,0.9100],
            ['RESTAURANTE_BOEMIA','RESTAURANTE_ANDESBULL',0.6768,0.9100],
            ['RESTAURANTE_BOEMIA','RESTAURANTE_XIMENITA',0.6768,0.9100],
            ['RESTAURANTE_SABOR','RESTAURANTE_PILSENER',1.54,2.4],
            ['RESTAURANTE_SABOR','RESTAURANTE_BOEMIA',0.6768,0.9100],
            ['RESTAURANTE_ROSITA','RESTAURANTE_SABOR',2.26,3.0],
            ['HOTEL_CHACARRERO','RESTAURANTE_SABOR',0.65,0.950],
            ['HOTEL_CHACARRERO','RESTAURANTE_XIMENITA',0.688,0.918],
            ['RESTAURANTE_QUINTA_DORADA','HOTEL_CHACARRERO',0.67,0.9100],
            ['RESTAURANTE_XIMENITA','RESTAURANTE_ROSITA',0.67,0.874],
            ['RESTAURANTE_ROSITA','RESTAURANTE_COFFEREX',0.54,0.74],
            ['RESTAURANTE_QUINTA_DORADA','HOTEL_CHACARRERO',0.68,0.470],
            ['HOTEL_ESTACION','RESTAURANTE_PICHINCA',0.57,0.74],
            ['RESTAURANTE_CODELL','HOTEL_RIOBAMBA',0.812,0.74],
            ['HOTEL_RIOBAMBA','HOTEL_PRIVAMERA',0.550,0.7420],
            ['RESTAURANTE_COFFEREX','HOTEL_HUASI',0.57,0.74],
```

```
          ['RESTAURANTE_COFFEREX','HOTEL_SOL',0.57,0.74],
          ['HOTEL_SOL','RESTAURANTE_PILSENER',0.57,0.74],
          ['HOTEL_ESTACION','RESTAURANTE_PICHINCA',0.57,0.74],
          ['HOTEL_SOL','HOTEL_ROCIO',0.47,0.78],
          ['HOTEL_SOL','RESTAURANTE_CODELL',0.74,0.65],
          ['RESTAURANTE_ANDESBULL','RESTAURANTE_XIMENITA',0.57,0.74],
          ['HOTEL_ROCIO','RESTAURANTE_CODELL',0.57,0.74],
          ['HOTEL_CHACARRERO','HOTEL_ROCIO',0.57,0.74],
          ['HOTEL_PRIMAVERA','HOTEL_ROCIO',0.57,0.74],
          ['RESTAURANTE_CHILENO','HOTEL_PRIMAVERA',0.57,0.74],
          ['HOTEL_PRIMAVERA','RESTAURANTE_JEKE',0.57,0.74],
          ['HOTEL_HUASI','RESTAURANTE_JEKE',0.57,0.74],
          ['HOTEL_HUASI','RESTAURANTE_CHACARRERO',0.57,0.74],
          ['HOTEL_HUASI','HOTEL_ROCIO','0.57','0.74'],
          ['RESTAURANTE_YAGUAR','RESTAURANTE_QUINCHOS',0.25,0.50],
          ['RESTAURANTE_JEKE','HOTEL_IMPERIAL',0.30,0.38]

         )

#          ['RESTAURANTE_QUINTA_DORADA','RESTAURANTE_CORBAN_COFFEE','0.6079','0.8000']
for ll in listaL:
    #SE INICIA EL METODO DE GENERAR NODOS LUGARES
    grafo.CREAR_RUTA(str(ll[0]),str(ll[1]),str(ll[2]),str(ll[3]))

#------------------------------NODOS PARA ALGORITMO KNN----------------
listar= (["RESTAURANTE_CODELL",-0.2975,-79.16784,4,11],
          ["RESTAURANTE_PICHINCHA",-0.24665,-79.19275,1,9],
          ["RESTAURANTE_CHURRERIA",-0.2975,-79.1765,2,2],
          ["RESTAURANTE_QUINTA_DORADA",-0.23421,-79.16772,5,23],
          ["RESTAURANTE_CORBAN_COFFEE",-0.27227,-79.1402,3,15],
          ["RESTAURANTE_TENTACIONES",-0.25516,-79.18057,2,16],
          ["RESTAURANTE_DELIRIO",-0.25487,-79.1854,4,20],
          ["RESTAURANTE_BOM",-0.24781,-79.15229,3,14],
          ["RESTAURANTE_PAOLA",-0.26004,-79.16534,5,19],
          ["RESTAURANTE_CHILENO",-0.2476,-79.15319,3,23],
          ["RESTAURANTE_NELLY",-0.24873,-79.15046,2,34],
          ["RESTAURANTE_TAURO",-0.24759,-79.15341,4,26],
          ["RESTAURANTE_SABOR",-0.2473,-79.15811,1,32],
          ["RESTAURANTE_BOEMIA",-0.26004,-79.15046,3,24],
          ["RESTAURANTE_PILSENER",-0.24848,-79.16322,4,37],
          ["RESTAURANTE_ANDESBULL",-0.25578,-79.18003,1,45],
          ["RESTAURANTE_QUINCHOS",-0.25197,-79.15253,2,36],
          ["RESTAURANTE_NORTENO",-0.24936,-79.18003,3,56],
          ["RESTAURANTE_YAGUAR",-0.24886,-79.1402,4,23],
          ["RESTAURANTE_BURGER",-0.25382,-79.16772,5,67],
          ["RESTAURANTE_LINA",-0.24759,-79.13482,3,43],
          ["RESTAURANTE_COFFEREX",-0.26618,-79.11198,4,76],
          ["RESTAURANTE_BONNY",-0.25337,-79.1681,1,14],
          ["RESTAURANTE_JEKE",-0.25337,-79.15229,2,32],
          ["RESTAURANTE_XIMENITA",-0.24759,-79.19456,1,32],
          ["RESTAURANTE_ROSITA",-0.25077,-79.16322,3,28])
st ="Restaurantes"
for ll in listar:
    #SE INICIA EL METODO DE GENERAR NODOS LUGARES
    print(str(ll[0]),str(ll[1]),str(ll[2]))
    grafo.CREAR_LUGAR_KNN("NODO CREADO",str(ll[0]),str(ll[1]),str(ll[2]),str(ll[3]),st
#SE CREA LA LISTA DE NODOS LUGARES
listaL = (["HOTEL_MONTECARLO",-79.16885,-09.17885,2,10],
          ["PARQUE_CENTRAL",-0.22214,-0.34216,3,20],
          ["HOTEL_IMPERIAL",-0.24886, -79.15787,3,25],
          ["HOTEL_ESTACION",-0.25307,-79.17365,1,25],
```

```
            ["HOTEL_RIOBAMBA",-0.2473,-79.15811,4,22],
            ["HOTEL_ROCIO",-0.24528,-79.18359,1,26],
            ["HOTEL_HUASI",-0.24936,-79.16421,4,27],
            ["HOTEL_TORIL",-0.2478,-79.15211,2,15],
            ["HOTEL_PRIMAVERA",-0.24848,-79.15261,3,6],
            ["HOTEL_GABRIEL",-0.24528,-79.15811,5,30],
            ["HOTEL_SOL",-0.26004,-79.16534,4,25],
            ["HOTEL_INTERNACIONAL",-0.24525,-79.16612,3,25],
            ["HOTEL_ACACIAS",-0.24986,-79.16322,3,45],
            ["HOTEL_MAYA",-0.24525,-79.16612,3,21],
            ["HOTEL_TROJE",-0.26004,-79.17629,2,45],
            ["HOTEL_PERUANA",-0.2473,-79.15811,1,23],
            ["HOTEL_MOLINO",-0.24528,-79.16534,5,25],
            ["HOTEL_SHALOM",-0.24525,-79.16322,2,15],
            ["HOTEL_CHACARERO",-0.24528,-79.18359,3,45],
            ["HOTEL_DORADO",-0.24986,-79.15261,2,35])
st ="Hoteles"
for ll in listaL:
    #SE INICIA EL METODO DE GENERAR NODOS LUGARES
    print(str(ll[0]),str(ll[1]),str(ll[2]))
    grafo.CREAR_LUGAR_KNN("NODO CREADO",str(ll[0]),str(ll[1]),str(ll[2]),str(ll[3]),st
```

```
HOTEL_MONTECARLO -79.16885 -9.17885
EL NODO LUGAR YA EXISTE, INGRESAR OTRO LUGAR........
None
PARQUE_CENTRAL -0.22214 -0.34216
EL NODO LUGAR YA EXISTE, INGRESAR OTRO LUGAR........
None
RESTAURANTE_CODELL -0.2975 -79.16784
EL NODO LUGAR YA EXISTE, INGRESAR OTRO LUGAR........
None
RESTAURANTE_PICHINCHA -0.24665 -79.19275
EL NODO LUGAR YA EXISTE, INGRESAR OTRO LUGAR........
None
RESTAURANTE_CHURRERIA -0.2975 -79.1765
EL NODO LUGAR YA EXISTE, INGRESAR OTRO LUGAR........
None
RESTAURANTE_QUINTA_DORADA -0.23421 -79.16772
EL NODO LUGAR YA EXISTE, INGRESAR OTRO LUGAR........
None
RESTAURANTE CORBAN COFFEE -0.27227 -79.1402
```

In [5]:

```
#IMPLEMENTAR EL ALGORITMO DE BUSQUEDAD POR AMPLITUD

# Creamos la clase Nodo
class Node:
    def __init__(self, data, child=None): # Constructor de la clase
        self.data = data
        self.child = None
        self.fathr = None
        self.cost = None
        self.set_child(child)

    def set_child(self, child): # Agregar hijos
        self.child = child
        if self.child is not None:
            for ch in self.child:
                ch.fathr = self

    def equal(self, node): # Igual al equals de Java
        if self.data == node.data:
            return True
        else:
            return False

    def on_list(self, node_list): # Verfiicar su el nodo esta en la lista
        listed = False
        for n in node_list:
            if self.equal(n):
                listed = True
        return listed

    def __str__(self): # Igual al toString Java
        return str(self.data)







# Implementacion del metodo de busqueda por amplitud
def search_Amplitud_solution(connections, init_state, solution):
    solved = False # Variable para almacenar el estado de la busqueda
    visited_nodes = [] # Nodos visitados
    frontrs_nodes = [] # Nodos en busqueda o lista nodos

    init_node = Node(init_state) # Nodo inicial
    frontrs_nodes.append(init_node)
    while (not solved) and len(frontrs_nodes) != 0:
        node = frontrs_nodes[0]
        # extraer nodo y añadirlo a visitados
        visited_nodes.append(frontrs_nodes.pop(0))
        if node.data == solution: # Preguntar se el nodo obtenido es la solucion
            solved = True
            return node # Retornamos el nodo de la solucion
        else:
            # expandir nodos hijo - ciudades con conexion
            node_data = node.data
            child_list = []
            for chld in connections[node_data]:
```

```
                child = Node(chld)
                child_list.append(child)
                if not child.on_list(visited_nodes) and not child.on_list(frontrs_node
                    frontrs_nodes.append(child)
            node.set_child(child_list)

if __name__ == "__main__":
    connections = {
        'PARQUE_CENTRAL': {'RESTAURANTE_CODELL', 'HOTEL_MONTECARLO'},
        'HOTEL_MONTECARLO': {'RESTAURANTE_PILSENER', 'RESTAURANTE_CHURRERIA','RESTAURA
        'RESTAURANTE_QUINTA_DORADA': {'RESTAURANTE_PAOLA','RESTAURANTE_NORTENO'},
        'RESTAURANTE_NORTENO': {'RESTAURANTE_LINA'},
        'RESTAURANTE_CODELL': {'HOTEL_RIOBAMBA', 'HOTEL_DORADO', 'RESTAURANTE_PICHINCH
        'HOTEL_RIOBAMBA': {'RESTAURANTE_CHURRERIA', 'RESTAURANTE_LINA'},
        'RESTAURANTE_LINA': {'RESTAURANTE_PAOLA'},
        'RESTAURANTE_PILSENER': {'RESTAURANTE_YAGUAR', 'RESTAURANTE_QUINCHOS'},
        'HOTEL_DORADO': {'HOTEL_ESTACION',''},
        'HOTEL_ESTACION': {'PARQUE_CENTRAL'},
        'RESTAURANTE_PAOLA': {'HOTEL_ESTACION'},
        'RESTAURANTE_PICHINCHA': {'RESTAURANTE_NELLY','RESTAURANTE_CHILENO'},
        'RESTAURANTE_NELLY': {'RESTAURANTE_DELIRIO','RESTAURANTE_TENTACIONES'},
        'RESTAURANTE_CHILENO': {'RESTAURANTE_NELLY','HOTEL_PRIMAVERA'},
        'HOTEL_IMPERIAL': {'RESTAURANTE_CHILENO','RESTAURANTE_TENTACIONES'},
        'RESTAURANTE_CORBAN_COFFEE': {'RESTAURANTE_BOM','RESTAURANTE_DELIRIO'},
        'RESTAURANTE_BOM': {'RESTAURANTE_PAOLA'},
        'RESTAURANTE_DELIRIO': {'RESTAURANTE_PAOLA','HOTEL_DORADO'},
        'RESTAURANTE_CHURRERIA': {'RESTAURANTE_YAGUAR','RESTAURANTE_NORTENO'},
        'RESTAURANTE_QUINCHOS': {'RESTAURANTE_BOEMIA'},
        'RESTAURANTE_SABOR': {'RESTAURANTE_BOEMIA','RESTAURANTE_PILSENER'},
        'RESTAURANTE_BOEMIA': {'RESTAURANTE_ANDESBULL','RESTAURANTE_XIMENITA'},
        'RESTAURANTE_ANDESBULL': {'RESTAURANTE_XIMENITA'},
        'RESTAURANTE_XIMENITA': {'RESTAURANTE_ROSITA'},
        'RESTAURANTE_ROSITA': {'RESTAURANTE_SABOR','RESTAURANTE_COFFEREX'},
        'HOTEL_SOL': {'RESTAURANTE_CODELL','HOTEL_ROCIO'},
        'RESTAURANTE_COFFEREX': {'HOTEL_HUASI','HOTEL_SOL'},
        'HOTEL_ROCIO': {'RESTAURANTE_CODELL'},
        'HOTEL_HUASI': {'HOTEL_ROCIO','RESTAURANTE_JEKE'},
        'HOTEL_PRIMAVERA': {'RESTAURANTE_JEKE','HOTEL_ROCIO'},
        'RESTAURANTE_YAGUAR': {'RESTAURANTE_QUINCHOS'},
        'RESTAURANTE_JEKE': {'HOTEL_IMPERIAL'}
    }

    init_state = 'HOTEL_MONTECARLO'
    solution = 'RESTAURANTE_PAOLA'
    solution_node = search_Amplitud_solution(connections, init_state, solution)
    # mostrar resultado
    result = []
    node = solution_node
    if node is not None:
        while node.fathr is not None:
            result.append(node.data)
            node = node.fathr
        result.append(init_state)
        result.reverse() # Reverso el resultado (Solo para presentar)
        print(result)
    else:
        print("No hay solucion !!!!")

#SE CREA EL GRAFO CON LAS RUTAS COMO LAS RELACIONES
#CALL gds.graph.create('migrafo', 'Lugares', 'RUTA_DE', { relationshipProperties: 'cos

```

```
# MATCH (HOTEL_MONTECARLO:Lugares{nombre:'HOTEL_MONTECARLO'}), (RESTAURANTE_PAOLA:Luga
#  WITH id(HOTEL_MONTECARLO) AS startNode, [id(RESTAURANTE_PAOLA)] AS targetNodes
# CALL gds.alpha.bfs.stream('migrafo', {startNode: startNode, targetNodes: targetNodes
#  YIELD path UNWIND [ n in nodes(path) | n.nombre ] AS tags  RETURN tags

```

```
['HOTEL_MONTECARLO', 'RESTAURANTE_QUINTA_DORADA', 'RESTAURANTE_PAOLA']
```

In [17]:

```
#SE IMPLEMENTE ALGORITMO BUSQUEDAD POR PROFUNDIDA
# Busqueda en Profundidad

# Creamos la clase Nodo
class Node:
    def __init__(self, data, child=None): # Constructor de la clase
        self.data = data
        self.child = None
        self.fathr = None
        self.cost = None
        self.set_child(child)

    def set_child(self, child): # Agregar hijos
        self.child = child
        if self.child is not None:
            for ch in self.child:
                ch.fathr = self

    def equal(self, node): # Igual al equals de Java
        if self.data == node.data:
            return True

        else:
            return False

    def on_list(self, node_list): # Verfiicar su el nodo esta en la lista
        listed = False
        for n in node_list:
            if self.equal(n):
                listed = True
        return listed

    def __str__(self): # Igual al toString Java
        return str(self.data)

# Implementacion del metodo de busqueda por profundidad
def search_profundidad(init_node, solution, visited):
    visited.append(init_node.data) #Lista de visitados
    if init_node.data == solution: # Condicion de salida recursividad (Encontro la solu
        return init_node # Retorno el nodo resultado
    else:
        # Expandir nodos sucesores (hijos)
        node_data = init_node.data
        son = [node_data[1], node_data[0], node_data[2], node_data[3]]
        izq = Node(son)
        son = [node_data[0], node_data[2], node_data[1], node_data[3]]
        centro = Node(son)
        son = [node_data[0], node_data[1], node_data[3], node_data[2]]
        der = Node(son)
        init_node.set_child([izq, centro, der])

        for node_son in init_node.child: # Recorrer los nodos hijos
            if not node_son.data in visited: # No deben estar en los nodos visitados
                # Llamada Recursiva
                Solution = search_profundidad(node_son, solution, visited)
                if Solution is not None: # Cuando encuentra una solucion
                    return Solution # Retornamos la solucion encontrada
        return None

```

```
init_state = [4, 2, 3, 1] # Creamos un estado inicial
solution = [1, 2, 3, 4] # La solucion que debe buscar
#Inicializamos las variables
solution_node = None
visited = []
init_node = Node(init_state)
node = search_profundidad(init_node, solution, visited) # Llamamos la metodo de busqued

# Mostrar Resultado
result = []
while node.fathr is not None:
    result.append(node.data)
    node = node.fathr
result.append(init_state)
result.reverse() # Reverso el resultado (Solo para presentar)
print(result)


#
# CALL gds.graph.create('myGraph', 'Lugares', 'RUTA_DE', { relationshipProperties: 'cos

# ALGORITMO DE BUSQUEDA POR PROFUNDIDAD
# MATCH (HOTEL_MONTECARLO:Lugares{nombre:'HOTEL_MONTECARLO'}), (RESTAURANTE_PAOLA:Lugar
# WITH id(HOTEL_MONTECARLO) AS startNode, [id(RESTAURANTE_PAOLA)] AS targetNodes
# CALL gds.alpha.dfs.stream('myGraph', {startNode: startNode, targetNodes: targetNodes}
# YIELD path
# UNWIND [ n in nodes(path) | n.nombre ] AS tags
# RETURN tags

```

```
[[4, 2, 3, 1], [2, 4, 3, 1], [2, 3, 4, 1], [3, 2, 4, 1], [3, 4, 2, 1], [4,
3, 2, 1], [4, 3, 1, 2], [3, 4, 1, 2], [3, 1, 4, 2], [1, 3, 4, 2], [1, 4, 3,
2], [4, 1, 3, 2], [4, 1, 2, 3], [1, 4, 2, 3], [1, 2, 4, 3], [2, 1, 4, 3],
[2, 1, 3, 4], [1, 2, 3, 4]]
```

In [ ]:

```
#SE IMPLEMENTE ALGORITMO BUSQUEDAD POR COSTO DE A*
# ALGORITMO DE BUSQUEDA A*
# MATCH (start:Lugares {nombre:
# 'HOTEL_MONTECARLO'}), (end:Lugares {nombre: 'RESTAURANTE_PAOLA'})
# CALL gds.alpha.shortestPath.astar.stream({
#   nodeProjection: {
#     Lugares: {
#       properties: ['longitud', 'latitud']
#     }
#   },
#   relationshipProjection: {
#     CONNECTION: {
#       type: 'RUTA_DE',
#       orientation: 'UNDIRECTED',
#       properties: 'costo'
#     }
#   },
#   startNode: start,
#   endNode: end,
#   propertyKeyLat: 'latitud',
#   propertyKeyLon: 'longitud'
# })
# YIELD nodeId, cost
# RETURN gds.util.asNode(nodeId).nombre AS lugares, cost

```

In [ ]:

```

```