```
# -*- coding: utf-8 -*-
"""Project_1.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1CiOb7HHcEVIVCmlBvL4BqcD4ZzaTLEzN

Project 1 : we'll directly use the csv file
"""

import csv
import pandas as pd
import math

from google.colab import drive
drive.mount("/content/drive")

path ="/content/drive/MyDrive/Projects_AI/100_nodes.csv"
df = pd.read_csv(path)

#Point on dataframes
#In order to access a column like x, y or node 9 juste type df['x'] for example
#For two it's df[['x','y']]

#df['1'][0] FROM node 0 to node 1

#df[str(9)] is ok too

"""GLOBAL VARIABLE: start, end, weight, frontier"""

start = 99
end = 0
weight = 0.25

reached = []
#element type: node
#node should have state, parent, action, cost
# node = {'STATE', 'PARENT', 'ACTION', 'COST'}

#selection function
def  SELECT_BEST_NODE(frontier,w):
  fbest = math.inf
  ibest = 0
  for i in range(len(frontier)):
    n = frontier[i]
    h = HEURISTIC_COST(n['STATE'], end)

    if w*n['COST']+(1-w)*h < fbest :
      fbest = w*n['COST']+(1-w)*h
      ibest = i
  node = frontier[ibest]
  frontier.remove(node)
  #print("Best node is", node, "with distance", fbest)
  return node

  #either return or return i

def HEURISTIC_COST(current_node, end):
```

```
  x_1 = df['x'][current_node]
  y_1 = df['y'][current_node]
  x_2 = df['x'][end]
  y_2 = df['y'][end]

  distance = abs(x_1-x_2) + abs(y_1-y_2)
  return distance

def EXPAND_GRAPH_SEARCH(node):
  child_list = []
  s = node['STATE']
  for i in range(100) :
    #search from node s to node [0;100]
    link_cost = df[str(i)][s]
    if link_cost != 0:
      c = node['COST'] + link_cost
      child = {'STATE': i, 'PARENT': node, 'ACTION': link_cost, 'COST': c}
      child_list.append(child)
  return child_list

def SEARCH_REACHED(reached, child):
  for index in range(len(reached)):
    if reached[index]['STATE'] == child['STATE']:
      return index
  return -1

def ASTAR_GRAPH_SEARCH(start, end, weight):
  node = {'STATE': start, 'PARENT': -1, 'ACTION': 0, 'COST': 0}
  frontier = []
  frontier.append(node)
  reached.append(node)

  while len(frontier) != 0 :
    node = SELECT_BEST_NODE(frontier, weight)
    if node['STATE'] == end:
      return node
    child_list = EXPAND_GRAPH_SEARCH(node)

    for child in child_list:
      index = SEARCH_REACHED(reached, child)
      if(index == -1):
        frontier.append(child)
        reached.append(child)
      elif (child['COST'] < reached[index]['COST']):
        reached[index] = child
        frontier.append(child)
  return -1

def PATH_FROM_NODE(node, reached):
  path = []
  path.append(node['STATE'])
  while path[-1] != start:
    parent = node['PARENT']
    path.append(parent['STATE'])
    node = parent
  return path

"""MAIN CODE"""
```

```
ans = ASTAR_GRAPH_SEARCH(start, end, weight)
print(ans)
print("Number of nodes generated :",len(reached))
print("Length of the path is :", ans['COST'])
print('Path is:', PATH_FROM_NODE(ans, reached)[::-1])
```