A\*

import math  
  
  
def shortest\_path(M, start, goal):  
 sx = M.intersections[start][0]  
 sy = M.intersections[start][1]  
 gx = M.intersections[goal][0]  
 gy = M.intersections[goal][1]  
 h = math.sqrt((sx - gx) \* (sx - gx) + (sy - gy) \* (sy - gy))  
 closedSet = set()  
 openSet = set()  
 openSet.add(start)  
 gScore = {}  
 gScore[start] = 0  
 fScore = {}  
 fScore[start] = h  
 cameFrom = {}  
 sumg = 0  
 NEW = 0  
 BOOL = False  
 while len(openSet) != 0:  
 MAX = 1000  
 for new in openSet:  
 print("new", new)  
 if fScore[new] < MAX:  
 MAX = fScore[new]  
 # print("MAX=",MAX)  
 NEW = new  
 current = NEW  
 print("current=", current)  
 if current == goal:  
 return reconstruct\_path(cameFrom, current)  
 openSet.remove(current)  
 closedSet.add(current)  
 # dafult=M.roads(current)  
 for neighbor in M.roads[current]:  
 BOOL = False  
 print("key=", neighbor)  
 a = {neighbor}  
 if len(a & closedSet) > 0:  
 continue  
 print("key is not in closeSet")  
 if len(a & openSet) == 0:  
 openSet.add(neighbor)  
 else:  
 BOOL = True  
 x = M.intersections[current][0]  
 y = M.intersections[current][1]  
 x1 = M.intersections[neighbor][0]  
 y1 = M.intersections[neighbor][1]  
 g = math.sqrt((x - x1) \* (x - x1) + (y - y1) \* (y - y1))  
 h = math.sqrt((x1 - gx) \* (x1 - gx) + (y1 - gy) \* (y1 - gy))  
  
 new\_gScore = gScore[current] + g  
 if BOOL == True:  
 if new\_gScore >= gScore[neighbor]:  
 continue  
 print("new\_gScore", new\_gScore)  
 cameFrom[neighbor] = current  
 gScore[neighbor] = new\_gScore  
 fScore[neighbor] = new\_gScore + h  
 print("fScore", neighbor, "is", new\_gScore + h)  
 print("fScore=", new\_gScore + h)  
  
 print("\_\_\_\_\_\_\_\_\_\_++--------------++\_\_\_\_\_\_\_\_\_")  
  
  
def reconstruct\_path(cameFrom, current):  
 print("已到达lllll")  
 total\_path = []  
 total\_path.append(current)  
 for key, value in cameFrom.items():  
 print("key", key, ":", "value", value)  
  
 while current in cameFrom.keys():  
 current = cameFrom[current]  
 total\_path.append(current)  
 total\_path = list(reversed(total\_path))  
 return total\_path