# import packages

##############################################################################

import numpy as np

import heapq

import matplotlib.pyplot as plt

from matplotlib.pyplot import figure

##############################################################################

# plot grid

##############################################################################

grid = np.array([

[0, 0, 0, 0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0, 1, 1, 1],

[0, 0, 0, 0, 0, 0, 0, 1, 0, 0],

[0, 0, 0, 0, 0, 0, 0, 0, 0, 0],

])

# start point and goal

start = (0,0)

goal = (9,9)

##############################################################################

# heuristic function for path scoring

##############################################################################

def heuristic(a, b):

return np.sqrt((b[0] - a[0]) \*\* 2 + (b[1] - a[1]) \*\* 2)

##############################################################################

# path finding function

##############################################################################

def astar(array, start, goal):

neighbors = [(0,1),(0,-1),(1,0),(-1,0),(1,1),(1,-1),(-1,1),(-1,-1)]

close\_set = set()

came\_from = {}

gscore = {start:0}

fscore = {start:heuristic(start, goal)}

oheap = []

heapq.heappush(oheap, (fscore[start], start))

while oheap:

current = heapq.heappop(oheap)[1]

if current == goal:

data = []

while current in came\_from:

data.append(current)

current = came\_from[current]

return data

close\_set.add(current)

for i, j in neighbors:

neighbor = current[0] + i, current[1] + j

tentative\_g\_score = gscore[current] + heuristic(current, neighbor)

if 0 <= neighbor[0] < array.shape[0]:

if 0 <= neighbor[1] < array.shape[1]:

if array[neighbor[0]][neighbor[1]] == 1:

continue

else:

# array bound y walls

continue

else:

# array bound x walls

continue

if neighbor in close\_set and tentative\_g\_score >= gscore.get(neighbor, 0):

continue

if tentative\_g\_score < gscore.get(neighbor, 0) or neighbor not in [i[1]for i in oheap]:

came\_from[neighbor] = current

gscore[neighbor] = tentative\_g\_score

fscore[neighbor] = tentative\_g\_score + heuristic(neighbor, goal)

heapq.heappush(oheap, (fscore[neighbor], neighbor))

return False

route = astar(grid, start, goal)

route = route + [start]

route = route[::-1]

print(route)

print("reached")

##############################################################################

# plot the path

##############################################################################

#extract x and y coordinates from route list

x\_coords = []

y\_coords = []

for i in (range(0,len(route))):

x = route[i][0]

y = route[i][1]

x\_coords.append(x)

y\_coords.append(y)

# plot map and path

fig, ax = plt.subplots(figsize=(20,20))

ax.imshow(grid, cmap=plt.cm.Dark2)

ax.scatter(start[1],start[0], marker = "\*", color = "yellow", s = 200)

ax.scatter(goal[1],goal[0], marker = "\*", color = "red", s = 200)

ax.plot(y\_coords,x\_coords, color = "black")

plt.show()