

```

import numpy as np
import pygame as pg

class MapaNodo:
    def __init__(self, position, cost, parent=None):
        self.position = position
        self.cost = cost
        self.parent = parent

    def __eq__(self, other):
        return self.position[0] == other.position[0] and self.position[1] == other.position[1]

class Dijkstra(object):
    def run(self, mapa, start, end):
        mapa = mapa.astype(np.float)

        unique, counts = np.unique(mapa, return_counts=True)
        nodosEnUno = counts[1]
        path = []
        vectorOfVisited = []
        vectorOfLabeled = []
        mapaRows, mapaCols = np.shape(mapa)
        visited = np.zeros(mapa.shape)
        costs = np.zeros(mapa.shape)
        vectorOfLabeled.append(MapaNodo(start[:-1], 0))
        endNode = MapaNodo(end[:-1], 0)

        while(len(vectorOfVisited) != nodosEnUno):
            currentNode = vectorOfLabeled.pop(0)

            movements = [[-1, -1, 1.4],
                        [0, -1, 1],
                        [1, -1, 1.4],
                        [-1, 0, 1],
                        [1, 0, 1],
                        [-1, 1, 1.4],
                        [0, 1, 1],
                        [1, 1, 1.4]]

            for movement in movements:
                newPosition = [currentNode.position[0] +
                              movement[0], currentNode.position[1]+movement[1]]
                adjacentNode = MapaNodo(
                    newPosition, currentNode.cost+movement[2], currentNode)
                if newPosition[0] < 0 or newPosition[1] < 0 or newPosition[0] >= mapaRows or newPo:
                    continue
                elif mapa[newPosition[0]][newPosition[1]] == 0:
                    continue
                elif visited[newPosition[0]][newPosition[1]] == 1:
                    continue
                else:
                    encontrado = False
                    for labeled in vectorOfLabeled:
                        if (labeled == adjacentNode):
                            encontrado = True

```

```

        if labeled.cost > adjacentNode.cost:
            labeled.cost = adjacentNode.cost
            costs[newPosition[0]][newPosition[1]] = adjacentNode.cost
            labeled.parent = currentNode
    if not encontrado:
        vectorOfLabeled.append(adjacentNode)
        costs[newPosition[0]][newPosition[1]] = adjacentNode.cost

    vectorOfVisited.append(currentNode)
    if currentNode == endNode:
        break
    visited[currentNode.position[0]][currentNode.position[1]] = 1
    vectorOfLabeled = sorted(vectorOfLabeled, key=lambda x: x.cost)

for visitedNode in vectorOfVisited:
    if visitedNode == endNode:
        endNode = visitedNode
        break

while endNode is not None:
    path.append(endNode.position)
    endNode = endNode.parent
return path, visited, costs

pg.init()
#mapaAlg = np.load('mapaProfundidad2.npy')
mapaAlg = np.load('mapaProfundidad.npy')
width, height = mapaAlg.shape
BLACK = pg.Color('black')
WHITE = pg.Color('white')
GREEN = pg.Color('green')
RED = pg.Color('red')
BLUE = pg.Color('blue')
color_light = (170, 170, 170)

color_dark = (100, 100, 100)
smallfont = pg.font.SysFont('arial', 30)
text = smallfont.render('Dijkstra', True, RED)
tile_size = 10
start = [10, 3]
goal = [43, 43]
topPadding = 50

search = Dijkstra()

screen = pg.display.set_mode(
    (width * tile_size, height * tile_size + topPadding))
clock = pg.time.Clock()

background = pg.Surface((width * tile_size, height * tile_size))
buttons = pg.Surface((width * tile_size, 50))

for y in range(0, height):
    for x in range(0, width):

```

```

rect = (x * tile_size, y * tile_size, tile_size, tile_size)
if (mapaAlg[y, x] == 0):
    color = BLACK
else:
    color = WHITE
if x == start[0] and y == start[1]:
    color = GREEN
if x == goal[0] and y == goal[1]:
    color = RED
pg.draw.rect(background, color, rect)

game_exit = False
while not game_exit:
    mouse = pg.mouse.get_pos()
    for event in pg.event.get():
        if event.type == pg.QUIT:
            game_exit = True
        if event.type == pg.MOUSEBUTTONDOWN:

            if 10 <= mouse[0] <= 150 and 10 <= mouse[1] <= 40:
                camino, mapavisited, costos = search.run(mapaAlg, start, goal)
                for point in camino:
                    rect = (point[1] * tile_size, point[0] * tile_size,
                           tile_size, tile_size)
                    pg.draw.rect(background, BLUE, rect)

            if 0 <= mouse[0] <= 140 and 10 <= mouse[1] <= 40:
                pg.draw.rect(buttons, color_light, [10, 10, 140, 30])

        else:
            pg.draw.rect(buttons, color_dark, [10, 10, 140, 30])

    screen.fill((0, 0, 0))

    screen.blit(buttons, (0, 0))
    screen.blit(background, (0, 50))
    screen.blit(text, (10, 10))
    pg.display.flip()
    clock.tick(30)
pg.display.quit()

```