

# UNIVERSIDAD DE GUADALAJARA



## CENTRO UNIVERSITARIO DE CIENCIAS EXACTAS E INGENIERÍAS

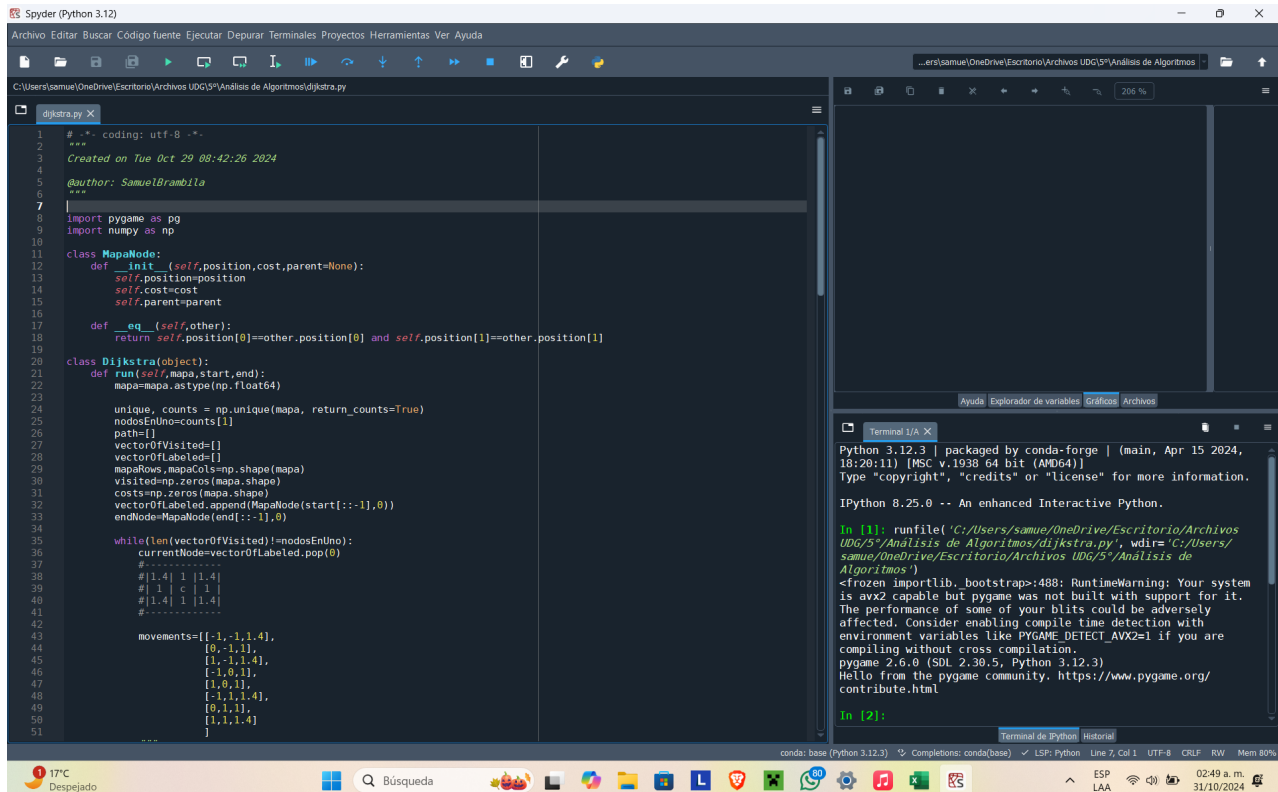
### Análisis de Algoritmos

#### Reporte de práctica

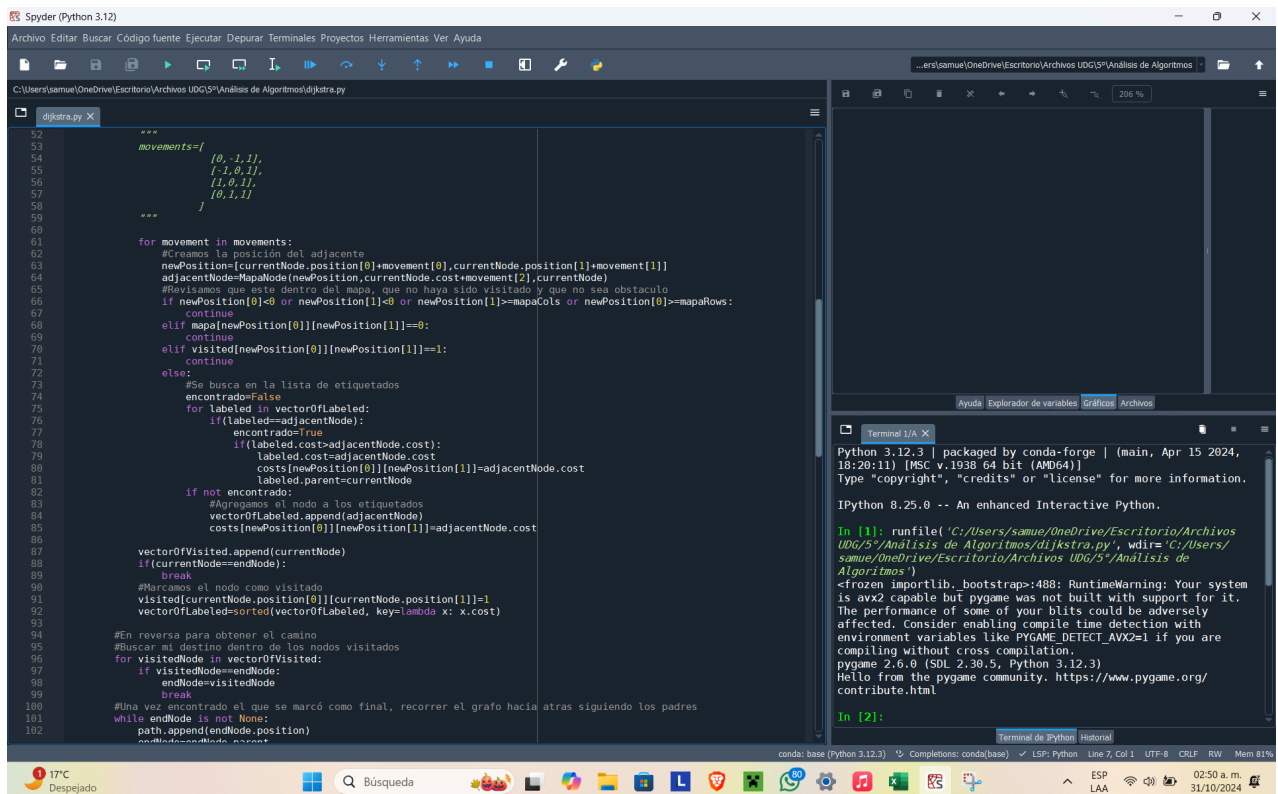
|                        |                                  |
|------------------------|----------------------------------|
| Nombre del alumno:     | Samuel David Pérez Brambila      |
| Código del alumno:     | 222966286                        |
| Profesor:              | Erasmus Gabriel Martínez Soltero |
| Título de la práctica: | "Dijkstra"                       |
| Fecha:                 | 05 de Noviembre de 2024          |

# Metodología

## Código:



```
1  # -*- coding: utf-8 -*-
2  """
3  Created on Tue Oct 29 08:42:26 2024
4
5  @author: SamuelBrambila
6  """
7
8  import pygame as pg
9  import numpy as np
10
11  class MapaNode:
12      def __init__(self, position, cost, parent=None):
13          self.position = position
14          self.cost = cost
15          self.parent = parent
16
17      def __eq__(self, other):
18          return self.position[0] == other.position[0] and self.position[1] == other.position[1]
19
20  class Dijkstra(object):
21      def run(self, mapa, start, end):
22          mapa = mapa.astype(np.float64)
23
24          unique, counts = np.unique(mapa, return_counts=True)
25          nodosEnUno = counts[1]
26          path = []
27          vectorOfVisited = []
28          vectorOfLabeled = []
29          mapaflores, mapaflorescols = np.shape(mapa)
30          visited = np.zeros(mapa.shape)
31          costs = np.zeros(mapa.shape)
32          vectorOfLabeled.append(MapaNode(start[::1], 0))
33          endNode = MapaNode(end[::1], 0)
34
35          while (len(vectorOfVisited) != nodosEnUno):
36              currentNode = vectorOfLabeled.pop(0)
37
38              # [1,4] 1 [1,4]
39              # 1 | c | 1
40              # [1,4] 1 [1,4]
41              # -----
42
43              movements = [[-1, -1, 1, 4],
44                          [0, -1, 1, 1],
45                          [1, -1, 1, 4],
46                          [-1, 0, 1, 1],
47                          [1, 0, 1, 1],
48                          [-1, -1, 1, 4],
49                          [0, 1, 1, 1],
50                          [1, 1, 1, 4]]
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
```



```
52
53
54     movements = [
55         [0, -1, 1, 1],
56         [-1, 0, 1, 1],
57         [1, 0, 1, 1],
58         [0, 1, 1, 1]
59     ]
60
61     for movement in movements:
62         # Creamos la posición del adjacente
63         newPosition = [currentNode.position[0] + movement[0], currentNode.position[1] + movement[1]]
64         adjacentNode = MapaNode(newPosition, currentNode.cost + movement[2], currentNode)
65         # Revisamos que este dentro del mapa, que no haya sido visitado y que no sea obstáculo
66         if newPosition[0] < 0 or newPosition[1] < 0 or newPosition[1] > mapaflorescols or newPosition[0] > mapaflores:
67             continue
68         elif mapa[newPosition[0]][newPosition[1]] == 0:
69             continue
70         elif visited[newPosition[0]][newPosition[1]] == 1:
71             continue
72         else:
73             # Se busca en la lista de etiquetados
74             encontrado = False
75             for labeled in vectorOfLabeled:
76                 if (labeled == adjacentNode):
77                     encontrado = True
78                     if (labeled.cost > adjacentNode.cost):
79                         labeled.cost = adjacentNode.cost
80                         costs[newPosition[0]][newPosition[1]] = adjacentNode.cost
81                         labeled.parent = currentNode
82             if not encontrado:
83                 # Agregamos el nodo a los etiquetados
84                 vectorOfLabeled.append(adjacentNode)
85                 costs[newPosition[0]][newPosition[1]] = adjacentNode.cost
86
87             vectorOfVisited.append(currentNode)
88             if (currentNode == endNode):
89                 break
90             # Marcamos el nodo como visitado
91             visited[currentNode.position[0]][currentNode.position[1]] = 1
92             vectorOfLabeled = sorted(vectorOfLabeled, key=lambda x: x.cost)
93
94     # En reversa para obtener el camino
95     # Buscar el destino dentro de los nodos visitados
96     for visitedNode in vectorOfVisited:
97         if visitedNode == endNode:
98             endNode = visitedNode
99             break
100     # Una vez encontrado el que se marcó como final, recorrer el grafo hacia atrás siguiendo los padres
101     while endNode is not None:
102         path.append(endNode.position)
103         endNode = endNode.parent
```

Spyder (Python 3.12)

Archivo Editar Buscar Código fuente Ejecutar Depurar Terminales Proyectos Herramientas Ver Ayuda

C:\Users\samuel\OneDrive\Escritorio\Archivos UDG5º Análisis de Algoritmos\dijkstra.py

```
dijkstra.py
103     endNode=endNode.parent
104     return path, visited, costs
105
106
107
108 pg.init()
109 #Cargamos el archivo de numpy que contiene el mapa
110 mapaAlg=np.load('mapaProfundidad.npy')
111 #Checamos el tamaño del mapa
112 width,height = mapaAlg.shape
113 #Definimos los colores
114 BLACK = pg.Color('black')
115 WHITE = pg.Color('white')
116 GREEN = pg.Color('green')
117 RED = pg.Color('red')
118 BLUE = pg.Color('blue')
119 # Light shade of the button
120 color_light = (170,170,170)
121 # Dark shade of the button
122 color_dark = (100,100,100)
123 smallfont = pg.font.SysFont('comicsans', 30)
124 text = smallfont.render('Search', True, RED)
125
126 # Tamaño en pixeles de la celda o el cuadro
127 tile_size = 10
128 # Punto inicial en formato columna, fila (x,y)
129 start = [10,3]
130 # Punto final en formato columna, fila (x,y)
131 goal = [43,43]
132 # Tamaño para el espacio para el boton
133 topPadding = 50
134 # Creamos el objeto para la búsqueda con Dijkstra
135 search = Dijkstra()
136
137 # El tamaño del mapa debe tener la ventana por eso es el tamaño del mapa por el tamaño de los cuadros
138 screen = pg.display.set_mode((width*tile_size, height*tile_size+topPadding))
139
140 # Espacio para el mapa
141 background = pg.Surface((width*tile_size, height*tile_size))
142 # Espacio para el boton
143 buttons = pg.Surface((width*tile_size, 50))
144
145 # Dibujamos los cuadros del mapa
146 for y in range(0, height):
147     for x in range(0, width):
148         rect = (x*tile_size, y*tile_size, tile_size, tile_size)
149         if (mapaAlg[y,x]==0):
150             color=BLACK
151         else:
152             color=WHITE
153         if x==start[0] and y==start[1]:
154             color=GREEN
155         if x==goal[0] and y==goal[1]:
156             color=RED
157         pg.draw.rect(background, color, rect)
158
159 #Aquí es la ejecución del "Juego"
160 game_exit = False
161 while not game_exit:
162     mouse = pg.mouse.get_pos()
163     for event in pg.event.get():
164         if event.type == pg.QUIT:
165             game_exit = True
166         if event.type == pg.MOUSEBUTTONDOWN:
167             if 10 <= mouse[0] <= 150 and 10 <= mouse[1] <= 40:
168                 camino,mapavisited,costos=search.run(mapaAlg,start,goal)
169                 for point in camino:
170                     rect = (point[1]*tile_size, point[0]*tile_size, tile_size, tile_size)
171                     pg.draw.rect(background, BLUE, rect)
172
173 #Cuando el mouse esta sobre las coordenadas del boton le cambiamos el color a unos gris bajito
174 if 0 <= mouse[0] <= 140 and 10 <= mouse[1] <= 40:
175     pg.draw.rect(buttons,color_light,[10,10,140,30])
176 else:
177     pg.draw.rect(buttons,color_dark,[10,10,140,30])
178
179 screen.fill((0,0,0))
180 screen.blit(buttons, (0,0))
181 screen.blit(background, (0,50))
182 screen.blit(text, (10,10))
183 pg.display.flip()
184 pg.display.quit()
```

Terminal 1/A X

Python 3.12.3 | packaged by conda-forge | (main, Apr 15 2024, 18:20:11) [MSC v.1938 64 bit (AMD64)]  
Type "copyright", "credits" or "license" for more information.

IPython 8.25.0 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/samuel/OneDrive/Escritorio/Archivos UDG5º Análisis de Algoritmos/dijkstra.py', wdir='C:/Users/samuel/OneDrive/Escritorio/Archivos UDG5º Análisis de Algoritmos')

<Frozen Importlib.\_bootstrap:488> RuntimeWarning: Your system is avx2 capable but pygame was not built with support for it. The performance of some of your blits could be adversely affected. Consider enabling compile time detection with environment variables like PYGAME\_DETECT\_AVX2=1 if you are compiling without cross compilation.  
pygame 2.6.0 (SDL 2.30.5, Python 3.12.3)  
Hello from the pygame community. https://www.pygame.org/contribute.html

In [2]:

Spyder (Python 3.12)

Archivo Editar Buscar Código fuente Ejecutar Depurar Terminales Proyectos Herramientas Ver Ayuda

C:\Users\samuel\OneDrive\Escritorio\Archivos UDG5º Análisis de Algoritmos\dijkstra.py

```
dijkstra.py
140 # Espacio para el mapa
141 background = pg.Surface((width*tile_size, height*tile_size))
142 # Espacio para el boton
143 buttons = pg.Surface((width*tile_size, 50))
144
145 # Dibujamos los cuadros del mapa
146 for y in range(0, height):
147     for x in range(0, width):
148         rect = (x*tile_size, y*tile_size, tile_size, tile_size)
149         if (mapaAlg[y,x]==0):
150             color=BLACK
151         else:
152             color=WHITE
153         if x==start[0] and y==start[1]:
154             color=GREEN
155         if x==goal[0] and y==goal[1]:
156             color=RED
157         pg.draw.rect(background, color, rect)
158
159 #Aquí es la ejecución del "Juego"
160 game_exit = False
161 while not game_exit:
162     mouse = pg.mouse.get_pos()
163     for event in pg.event.get():
164         if event.type == pg.QUIT:
165             game_exit = True
166         if event.type == pg.MOUSEBUTTONDOWN:
167             if 10 <= mouse[0] <= 150 and 10 <= mouse[1] <= 40:
168                 camino,mapavisited,costos=search.run(mapaAlg,start,goal)
169                 for point in camino:
170                     rect = (point[1]*tile_size, point[0]*tile_size, tile_size, tile_size)
171                     pg.draw.rect(background, BLUE, rect)
172
173 #Cuando el mouse esta sobre las coordenadas del boton le cambiamos el color a unos gris bajito
174 if 0 <= mouse[0] <= 140 and 10 <= mouse[1] <= 40:
175     pg.draw.rect(buttons,color_light,[10,10,140,30])
176 else:
177     pg.draw.rect(buttons,color_dark,[10,10,140,30])
178
179 screen.fill((0,0,0))
180 screen.blit(buttons, (0,0))
181 screen.blit(background, (0,50))
182 screen.blit(text, (10,10))
183 pg.display.flip()
184 pg.display.quit()
```

Terminal 1/A X

Python 3.12.3 | packaged by conda-forge | (main, Apr 15 2024, 18:20:11) [MSC v.1938 64 bit (AMD64)]  
Type "copyright", "credits" or "license" for more information.

IPython 8.25.0 -- An enhanced Interactive Python.

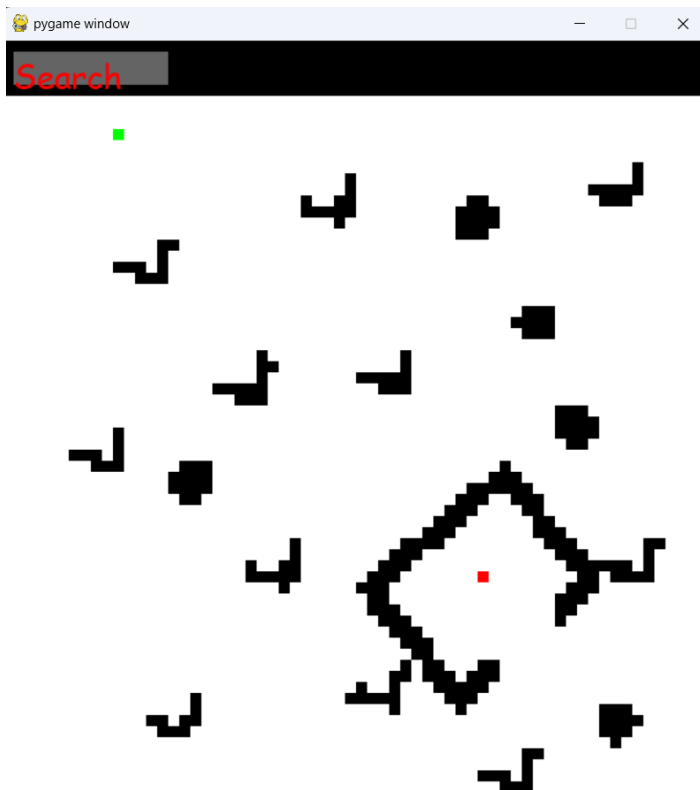
In [1]: runfile('C:/Users/samuel/OneDrive/Escritorio/Archivos UDG5º Análisis de Algoritmos/dijkstra.py', wdir='C:/Users/samuel/OneDrive/Escritorio/Archivos UDG5º Análisis de Algoritmos')

<Frozen Importlib.\_bootstrap:488> RuntimeWarning: Your system is avx2 capable but pygame was not built with support for it. The performance of some of your blits could be adversely affected. Consider enabling compile time detection with environment variables like PYGAME\_DETECT\_AVX2=1 if you are compiling without cross compilation.  
pygame 2.6.0 (SDL 2.30.5, Python 3.12.3)  
Hello from the pygame community. https://www.pygame.org/contribute.html

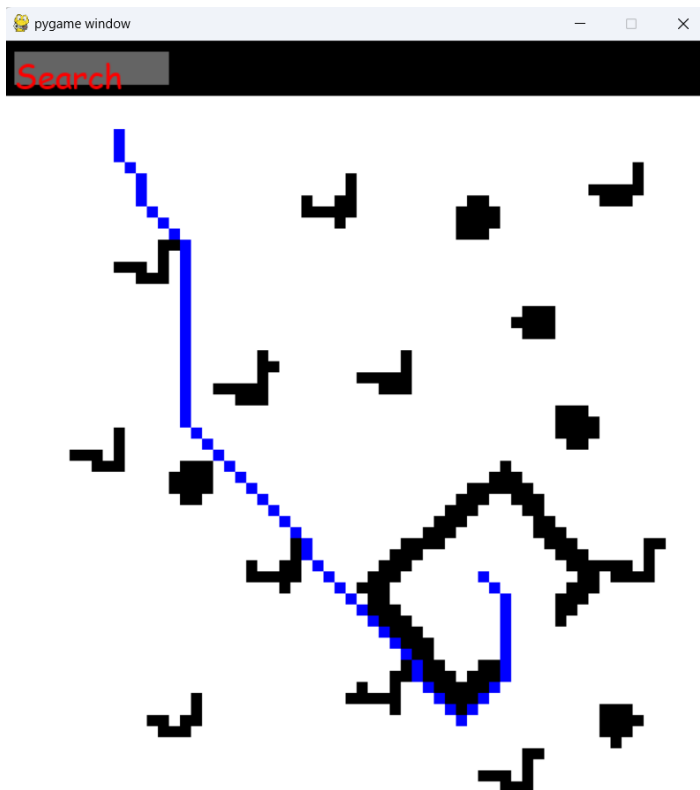
In [2]:

## Resultados

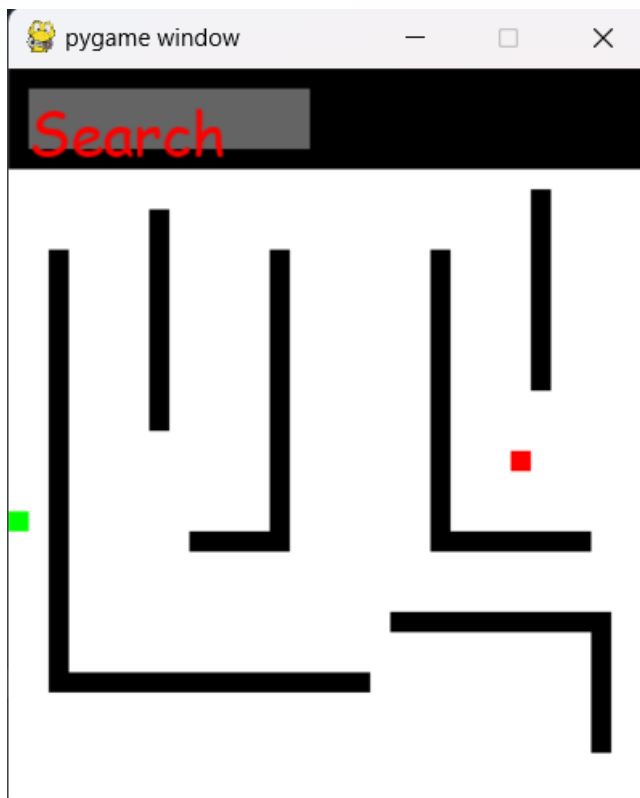
Mapa grande inicial:



Mapa grande con el camino más corto por el algoritmo Dijkstra:



Mapa chico inicial:



Mapa chico con el camino más corto por el algoritmo Dijkstra:

