Análisis de algoritmos
Tarea 5

Adaptar un algoritmo de Dijkstra para que trabaje en una rejilla de 40x40 donde se defina el punto inicial (ri, ci) y el punto final (rf, cf) y encuentre la ruta optima con las siguientes variantes:

1) Usar 4 vecinos con distancias unitarias.
2) Usar 8 vecinos.
3) Incluir posibilidades de definir obstáculos.

El lenguaje utilizado para el desarrollo del algoritmo es Python, utilizando el modulo Dijkstra del usuario crixodia en github, https://github.com/crixodia/python-dijkstra/blob/master/dijkstra.py, al cual solamente se le tiene que pasar la matriz de adyacencia, ademas se hace uno de la librería pygame para la creación del grid.

Para la creación de la matriz de adyacencia se tomo como referencia la matriz de un grid de 3x3 como se muestra a continuación:

0	1	2
3	4	5
6	7	8

0, 1, 2, 3, 4, 5, 6, 7, 8
0 [0. 1. 0. 1. 0. 0. 0. 0. 0.]
1 [1. 0. 1. 0. 1. 0. 0. 0. 0.]
2 [0. 1. 0. 0. 0. 1. 0. 0. 0.]
3 [1. 0. 0. 0. 1. 0. 1. 0. 0.]
4 [0. 1. 0. 1. 0. 1. 0. 1. 0.]
5 [0. 0. 1. 0. 1. 0. 0. 0. 1.]
6 [0. 0. 0. 1. 0. 0. 0. 1. 0.]
7 [0. 0. 0. 0. 1. 0. 1. 0. 1.]
8 [0. 0. 0. 0. 0. 1. 0. 1. 0.]

Se utiliza un ciclo for para crear los movimientos de cada celda, donde se crean 3 casos específicos:

1. Para las celdas del centro, estas se pueden mover hacia la izquierda, derecha.
2. Para las celdas del lado izquierdo, haciendo que solamente se puedan mover hacia la derecha.
3. Para las celdas del lado derecho haciendo que solo se puedan mover hacia la izquierda.

Para los movimientos de arriba y abajo se hace uso de una variable llamada tamtab, la cual tiene como valor el numero de celdas en el grid para cada lado, gracias a esto es posible cambiar el tamaño del grid sin modificar lo demás del código, ademas de permitir que dentro el for se agreguen estos movimientos a cada celda. Y ademas gracias a esta misma variable se agregan los movimientos para los diagonales en el caso de los 8 vecinos. En el caso de que el usuario meta algún obstáculos, los movimientos de estas celdas son eliminadas de la matriz, para que de esta manera sea imposible llegar a ellos.

El código tiene limitaciones para que el inicio y el fin sean coordenadas dentro del tamaño del grid, ademas que los obstáculos que el usuario meta no puedan ser ni el inicio ni el fin, sin tener limitaciones en el numero de obstáculos que meta.

Código:

```
from cmath import inf
from re import S
import pygame
import dijkstra
import numpy as np

#DEFINICION DE VARIABLES A USAR
AZUL = (0, 0, 255)
BLACK = (0, 0, 0)
RED = (255, 0, 0)
GREEN= (0, 255, 0)
tamtab=3 #MODIFICAR DEPENDIENDO EL NUMERO DE CELDAS QUE SE QUIERAN EN EL CUADRO
tamCuadro = 20
inicio=-1
fin=-1
xinicio=-1
yinicio=-1
xfin=-1
yfin=-1
obsop='s'
obs=-1
raiz=(2**0.5)
dia='a'

#AUXILIARES PARA GRAFICAR GRID
x=[]
for i in range((tamtab**2)):
        if i == 0:
```

```
                x=np.append(x,1)
        elif i % tamtab == 0:
                x=np.append(x,1)

        elif i > 0:
                aux=x[i-1]
        x=np.append(x,aux+tamCuadro+1)

y=[]
for i in range((tamtab**2)):
        if i == 0:
                y=np.append(y,1)

        elif i % tamtab == 0:
                aux=y[i-1]
                y=np.append(y,aux+tamCuadro+1)

        elif i > 0:
                aux=y[i-1]
                y=np.append(y,aux)

#CREACION DE GRID
pygame.init()
size = (tamtab+(tamtab*tamCuadro),tamtab+(tamtab*tamCuadro))
screen = pygame.display.set_mode(size)
pygame.display.set_caption("Grid on PYGAME")
clock = pygame.time.Clock()
gameOver = False

#INICIO DE GRID
while not gameOver:
        for event in pygame.event.get():
                if event.type == pygame.QUIT:
                        gameOver = True
        screen.fill(BLACK)
        for i in range(1, size[0], tamCuadro + 1):
                for j in range(1, size[1], tamCuadro + 1):
                        pygame.draw.rect(screen, AZUL, [i, j, tamCuadro, tamCuadro], 0)
        pygame.display.flip()

        #INGRESO DE COORDENAS DE PUNTO DE INICIO POR EL USUARIO
        while inicio < 0 or inicio >= tamtab**2:
```

```
        while xinicio <= 0 or xinicio > tamtab:
                xinicio=int(input('Ingrese la coordenada X de inicio:'))
                if xinicio <= 0 or xinicio > tamtab:
                        print('La coordenada de inicio en X esta fuera del rango.')

        while yinicio <= 0 or yinicio >tamtab:
                yinicio=int(input('Ingrese la coordenada Y de inicio:'))
                if yinicio <= 0 or yinicio > tamtab:
                        print('La coordenada de inicio en Y esta fuera del rango.')

        inicio=(tamtab*(yinicio-1))+xinicio-1
        print('inicio:')
        print(inicio)

        if inicio < 0 or inicio >= tamtab**2:
                print('El inicio esta fuera del rango.')

#GRAFICADO DE PUNTO DE INICIO
pygame.draw.rect(screen, RED, [x[inicio], y[inicio], tamCuadro, tamCuadro], 0)
pygame.display.flip()

#INGRESO DE COORDENADAS DE PUNTO DE FIN POR EL USUARIO
while fin < 0 or fin >= tamtab**2 or inicio == fin:

        while xfin <= 0 or xfin > tamtab:
                xfin=int(input('Ingrese la coordenada x de fin:'))
                if xfin <= 0 or xfin > tamtab:
                        print('La coordenada de fin en X esta fuera del rango.')

        while yfin <= 0 or yfin >tamtab:
                yfin=int(input('Ingrese la coordenada y de fin:'))
                if yfin <= 0 or yfin > tamtab:
                        print('La coordenada de fin en Y esta fuera del rango.')

        fin=(tamtab*(yfin-1))+xfin-1
        print('fin:')
        print(fin)

        if fin < 0 or fin >= tamtab**2:
                print('El fin esta fuera del rango.')

        if inicio == fin:
                print('El inicio y el fin son iguales:')
```

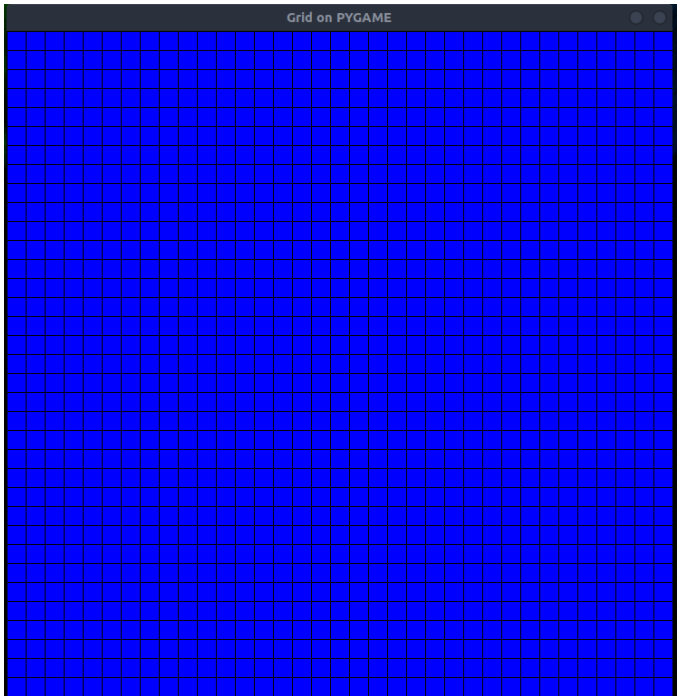
```
                xfin=-1
                yfin=-1

        #GRAFICADO DE PUNTO DE FIN
        pygame.draw.rect(screen, RED, [x[fin], y[fin], tamCuadro, tamCuadro], 0)
        pygame.display.flip()

        #OPCION DE MOVIMIENTOS EN DIAGONAL
        while dia != 'n' and dia != 's':
                dia=str(input('Desea que se pueda mover en diagonal(s/n):'))

        if dia == 'n':
                #CREACION DE MATRIZ DE MOVIMIENTOS RECTOS
                matrix=np.zeros(((tamtab**2),(tamtab**2)+tamtab))
                for i in range(tamtab**2):
                        #Para los de la orilla izquierda de la cuadricula
                        if i % tamtab == 0 or i ==0:
                                matrix[i][i]=0
                                matrix[i][i+1]=1
                                matrix[i][i+tamtab]=1
                                matrix[i][i-tamtab]=1

                        #para los de la orilla derecha de la cuadricula
                        elif (i+1) % tamtab == 0:
                                matrix[i][i]=0
                                matrix[i][i-1]=1
                                matrix[i][i+tamtab]=1
                                matrix[i][i-tamtab]=1

                        #para todos los cuadros restantes
                        else:
                                #Todos los valores en la diagonal principal son 0
                                matrix[i][i]=0
                                matrix[i][i+1]=1
                                matrix[i][i-1]=1
                                matrix[i][i+tamtab]=1
                                matrix[i][i-tamtab]=1
                                matriz2=matrix

                for i in range(tamtab):
                        matriz2=np.delete(matriz2,((tamtab**2)),axis=1)

        elif dia == 's':
```

```
        #CREACION DE MATRIZ DE MOVIMIENTOS
        matrix=np.zeros(((tamtab**2),(tamtab**2)+tamtab))
        for i in range(tamtab**2):
                #Para los de la orilla izquierda de la cuadricula
                if i % tamtab == 0 or i ==0:
                        matrix[i][i]=0
                        matrix[i][i+1]=1
                        matrix[i][i+tamtab]=1
                        matrix[i][i-tamtab]=1
                        matrix[i][i+tamtab+1]=raiz
                        matrix[i][i-tamtab+1]=raiz

                #para los de la orilla derecha de la cuadricula
                elif (i+1) % tamtab == 0:
                        matrix[i][i]=0
                        matrix[i][i-1]=1
                        matrix[i][i+tamtab]=1
                        matrix[i][i-tamtab]=1
                        matrix[i][i+tamtab-1]=raiz
                        matrix[i][i-tamtab-1]=raiz

                #para todos los cuadros restantes
                else:
                        #Todos los valores en la diagonal principal son 0
                        matrix[i][i]=0
                        matrix[i][i+1]=1
                        matrix[i][i-1]=1
                        matrix[i][i+tamtab]=1
                        matrix[i][i-tamtab]=1
                        matrix[i][i+tamtab-1]=raiz
                        matrix[i][i+tamtab+1]=raiz
                        matrix[i][i-tamtab-1]=raiz
                        matrix[i][i-tamtab+1]=raiz

        matriz2=matrix

        for i in range(tamtab):
                matriz2=np.delete(matriz2,((tamtab**2)),axis=1)

#AGREGADO DE OBSTACULOS
while obsop != 'n':
        obsop=str(input('Desea agregar un obstaculo(s/n):'))
        obsaux='n'
```

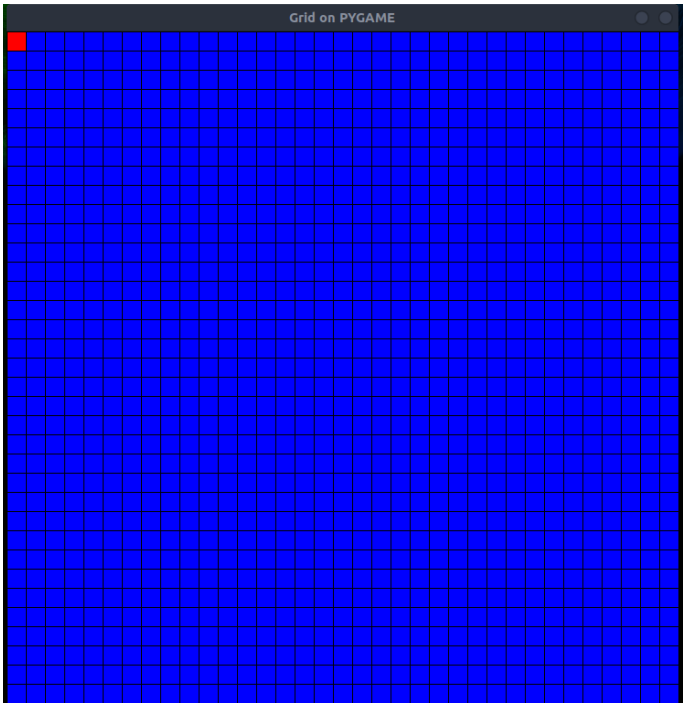
```
while obsop == 's':
        #INGRESO DE COORDENADA X DEL OBSTACULO
        xobs=int(input('Ingrese la coordenada X del obstaculo:'))
        if xobs <= 0 or xobs > tamtab:
                print('La coordenada del obstaculo en X esta fuera del rango.')
        else:
                obsop='n'
                obsaux='s'

while obsaux == 's':

        #INGRESO DE COORDENADA Y DEL OBSTACULO
        yobs=int(input('Ingrese la coordenada Y del obstaculo:'))
        if yobs <= 0 or yobs > tamtab:
                print('La coordenada del obstaculo en Y esta fuera del rango.')
        else:
                obsaux='n'
                obsop='s'
                obs=(tamtab*(yobs-1))+xobs-1

if obs == inicio:
        print('El obstaculo es:')
        print(obs)
        print('El obstaculo y el inicio son iguales:')
        obsop='s'

elif obs == fin:
        print('El obstaculo es:')
        print(obs)
        print('El obstaculo y el fin son iguales:')
        obsop='s'

elif obs != inicio and obs != fin and obsop == 's':
        print('El obstaculo es:')
        print(obs)
        for i in range(tamtab**2):
                matriz2[i][obs]=0
                matriz2[obs][i]=0

        pygame.draw.rect(screen, BLACK, [x[obs], y[obs], tamCuadro,
tamCuadro], 0)
        pygame.display.flip()
```

```
        #CALCULO DE CAMINO MAS CORTO Y DISTANCIA CON ALGORITMO DE DIJKSTRA
        camino=(dijkstra.find_shortest_path(matriz2,inicio,fin))
        print('El camino es:')
        print(camino)
        distancia=(dijkstra.find_shortest_distance(matriz2,inicio,fin))
        print('La distancia es:')
        print(distancia)
        if distancia == inf:
                print('No se puede llegar a ese punto.')

        #movimiento en celdas por el camino
        for i in camino:
                pygame.draw.rect(screen, GREEN, [x[i], y[i], tamCuadro, tamCuadro], 0)
                pygame.display.flip()
                pygame.time.delay(200)
        gameOver = True
pygame.quit()
```

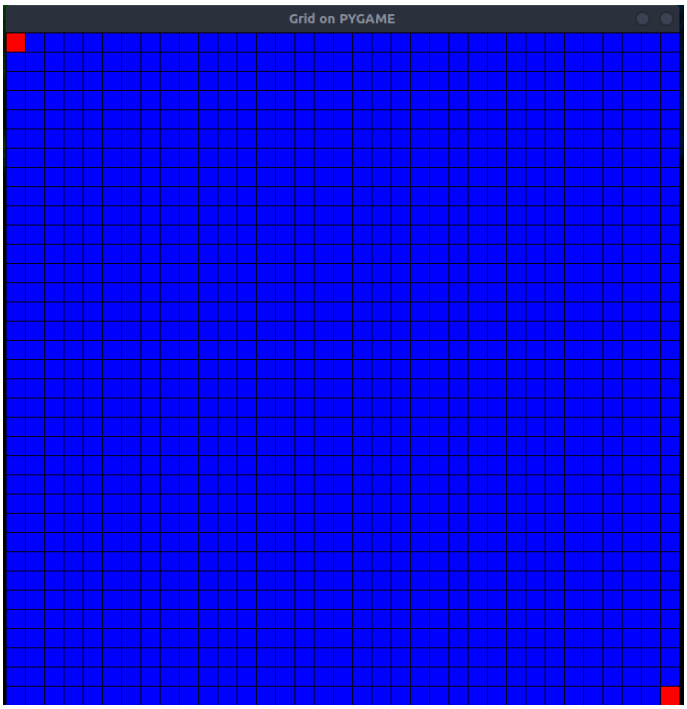
Al correr el código primero obtenemos un grid con todas las celdas en color azul:



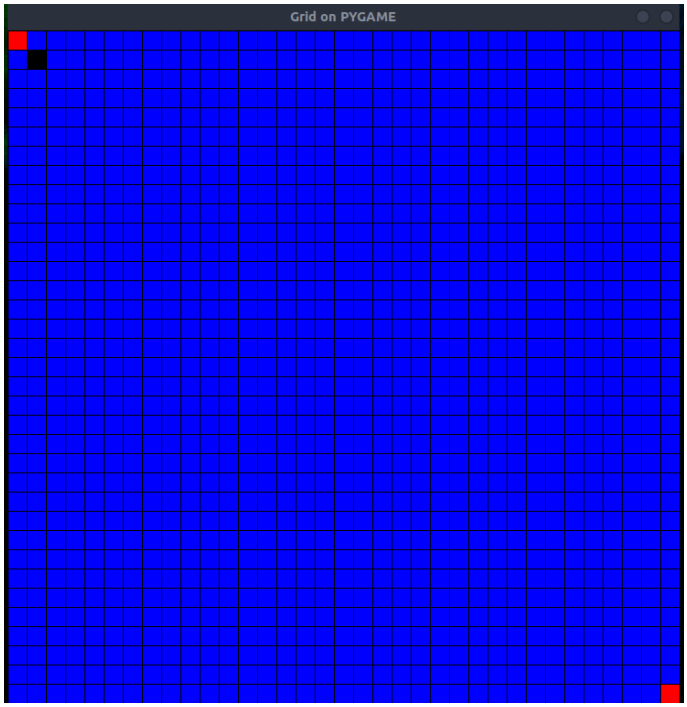
Después el usuario agrega el inicio, poniéndose la celda en color rojo:

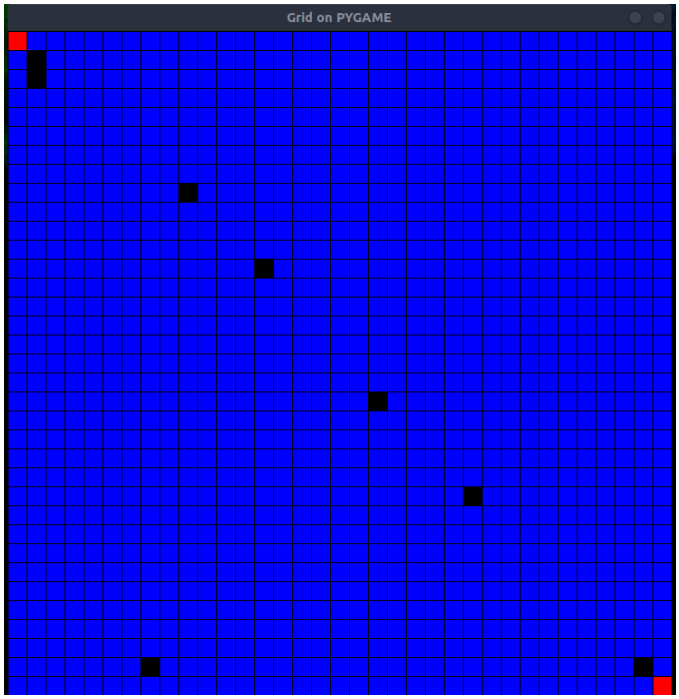


A continuación, el usuario agrega el final marcándose en color rojo:

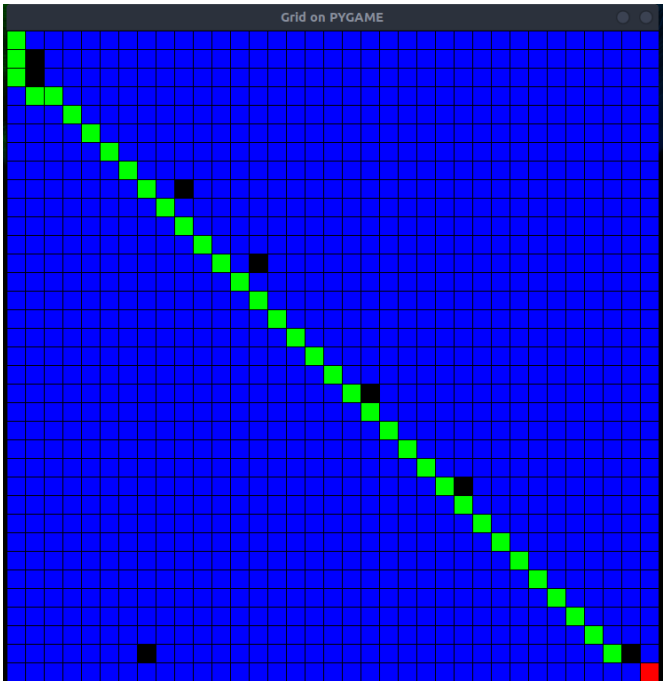


A continuación cada celda que el usuario ingrese como obstáculo se pondrá de color negro:

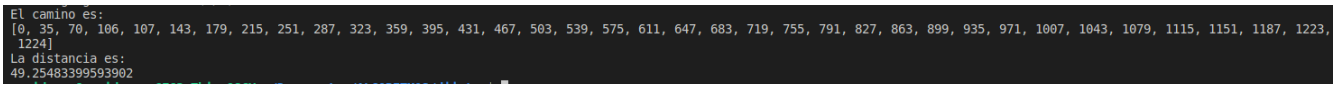




Finalmente, cuando el usuario decida no meter mas obstáculos, el recorrido empezara a verse, cambiando las celdas por las que pasa a color verde:



Mientras que en la Terminal se imprime cual es el camino elegido y la distancia del recorrido:



Conclusión:

Como se puede observar se llega al resultado esperado, obteniendo el camino mas corto para llegar al final, esto logrado gracias al algoritmo de Dijsktra, que a partir de la matriz de adyacencia calcula ese camino y la distancia, ademas gracias a la librería pygame es posible simular el recorrido para así ver de manera grafica como es que si obtiene ese camino y como se recorre.