INTRODUCCIÓN A LA PLANIFICACIÓN DE ROBOTS

Algoritmo de ruta mas corta tipo Greedy (Profundo o voraz)

- En base al algoritmo realizado en el repositorio de https://github.com/jgvictores/master-ipr y en los mapas ya creados se modificó el código, complementando la lógica del algoritmo A* para que sea más profundo.
- El código fue realizado en Python.
- Se compararon los resultados con el algoritmo base y el algoritmo modificado.

Código en Python Algoritmo Modificado

```
"""
Algoritmo de busqueda de ruta corta tipo Greedy basado en A*
By Kevin Medrano Ayala
Date: 24/10/2020
 1: Obstacle
 2: Start
 3: End
 8: Visited
"""
import csv
print("Start!")
# Clase Nodo
class Nodo:
    def __init__(self, x, y, distancia,myId,parentId):
        self.x = x
        self.y = y
        self.distancia = distancia
        self.myId = myId
        self.parentId = parentId
    def get_Posicion(self):
        return self.x, self.y
    def get_PosicionX(self):
        return self.x
    def get_PosicionY(self):
        return self.y
# Metodos
def getDistancia(x1,y1,x2,y2):
    return abs(x1 - x2) + abs(y1 - y2)

def mostrarMapa():
    for i in charMap:
```

```
        print(i)

def visitarNodos(posX,posY,id):
    end = False
    ID = -1
    if charMap[posX][posY]== '3':
        ID = Node.myId
        print("Nodo final encontrado!")
        end = True
    elif charMap[posX][posY]== '0':
        nodoAux = Nodo(posX,posY,getDistancia(posX,posY,END_X,END_Y),id,Node
.myId)
        listaAbierta.append(nodoAux)
        charMap[posX][posY] = '8'
    return end,ID
# Variables
FILE_NAME = "D:\\Maestria\\Master UC3M\\Planificacion de Robots\\master-
ipr\\map2\\map2.csv"
charMap = []
listaAbierta = []
listaCerrada = []
START_X = 1
START_Y = 2
END_X = 11
END_Y = 8
done = False
goalParentId = -1
# Cargar mapa del archivo .csv
with open(FILE_NAME) as f:
    line = f.readline()
    while line:
        charLine = line.strip().split(',')
        charMap.append(charLine)
        line = f.readline()

# Iniciar Origen en el mapa
charMap[START_X][START_Y] = '2'
charMap[END_X][END_Y] = '3'

# Inicializar valores
Node = Nodo(START_X,START_Y,0,0,-2)
listaCerrada.append(Node)
posicion_actual = listaCerrada[0].get_Posicion
mostrarMapa()
```

```
while not done:
    # Agregar Visita izquierda
    posX = Node.x - 1
    posY = Node.y
    done, pID = visitarNodos(posX,posY,len(listaAbierta)+len(listaCerrada))
    # Agregar Visita derecha
    posX = Node.x + 1
    posY = Node.y
    done, pID = visitarNodos(posX,posY,len(listaAbierta)+len(listaCerrada))
    # Agregar Visita abajo
    posX = Node.x
    posY = Node.y - 1
    done, pID = visitarNodos(posX,posY,len(listaAbierta)+len(listaCerrada))
    # Agregar Visita arriba
    posX = Node.x
    posY = Node.y + 1
    done, pID = visitarNodos(posX,posY,len(listaAbierta)+len(listaCerrada))
    if done:
        goalParentId = pID
        break
    menorD = 99
    pos = 0
    # Buscar nodo de menor distancia hacia la meta para agregar a la lista c
errada
    for i in range(len(listaAbierta)):
        if listaAbierta[i].distancia < menorD:
            menorD = listaAbierta[i].distancia
            posicion_actual = listaAbierta[i].get_Posicion()
            pos = i
    listaCerrada.append(listaAbierta[pos])
    Node = listaAbierta[pos]
    del listaAbierta[pos]
    mostrarMapa()
    print("Ciclo\n")
# Trazar la ruta mas corta final
ok = False
while not ok:
    for node in listaCerrada:
        if(node.myId == goalParentId):
            charMap[node.x][node.y]='X'
            goalParentId = node.parentId
            if(goalParentId == -2):
                ok = True
                charMap[node.x][node.y]='2'
mostrarMapa()
```

```
#Guardar Mapa
with open('Resultado_B.csv', 'w', newline='', encoding='utf-8') as csvfile:
    writer = csv.writer(csvfile)
    writer.writerows(charMap)
```

Comparaciones entre algoritmo base A y B

Algoritmo A

Para este caso:

- 1 representa el obstáculo
- 2 representa el nodo visitado
- 3 representa el nodo inicial (start)
- 4 representa nodo final (end)

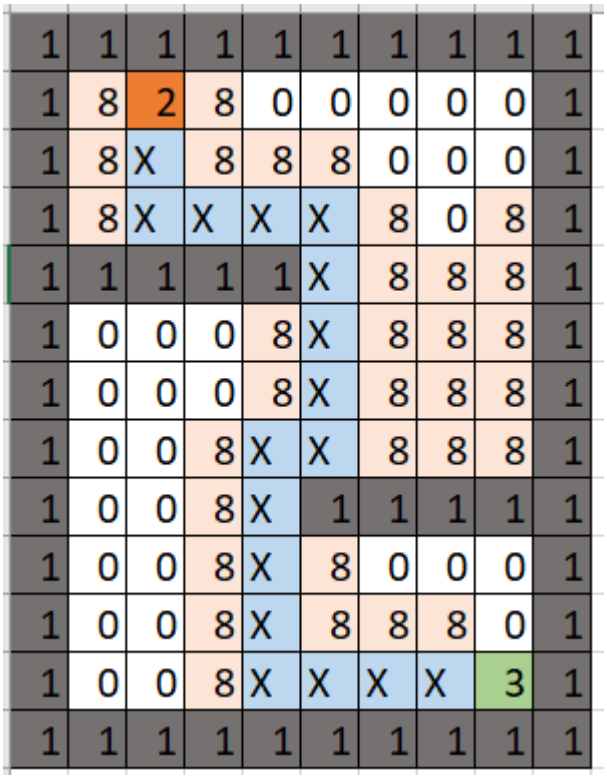
1	1	1	1	1	1	1	1	1	1
1	2	3	2	2	2	2	2	2	1
1	2	X	2	2	2	2	2	2	1
1	2	X	X	X	X	2	2	2	1
1	1	1	1	1	X	2	2	2	1
1	2	2	2	2	X	2	2	2	1
1	2	2	2	2	X	2	2	2	1
1	2	2	2	X	X	2	2	2	1
1	2	2	2	X	1	1	1	1	1
1	2	2	2	X	2	2	2	2	1
1	2	2	2	X	2	2	2	2	1
1	2	2	2	X	X	X	X	4	1
1	1	1	1	1	1	1	1	1	1

A simple vista se observa que el algoritmo A realiza muchos más ciclos de ejecución que A, por lo que se demora más tiempo en concentrar la ruta mas corta.

Algoritmo B

Para este caso:

- 1 representa el obstáculo
- 2 representa el nodo inicial (start)
- 3 representa el nodo final (end)
- 8 representa el nodo visitado



1	1	1	1	1	1	1	1	1	1
1	8	2	8	0	0	0	0	0	1
1	8	X	8	8	8	0	0	0	1
1	8	X	X	X	X	8	0	8	1
1	1	1	1	1	X	8	8	8	1
1	0	0	0	8	X	8	8	8	1
1	0	0	0	8	X	8	8	8	1
1	0	0	8	X	X	8	8	8	1
1	0	0	8	X	1	1	1	1	1
1	0	0	8	X	8	0	0	0	1
1	0	0	8	X	8	8	8	0	1
1	0	0	8	X	X	X	X	3	1
1	1	1	1	1	1	1	1	1	1

El algoritmo modificado A, mejora considerablemente el numero de ciclos de ejecución para encontrar la ruta mas corta, siendo en este caso mucho mas profundo en su búsqueda, la única diferencia es el tiempo siendo que ambos algoritmos encuentran la misma ruta.