Universidad del valle

Inteligencia artificial

Entrega proyecto 1

Juan Camilo Ortiz Sanchez 1810223

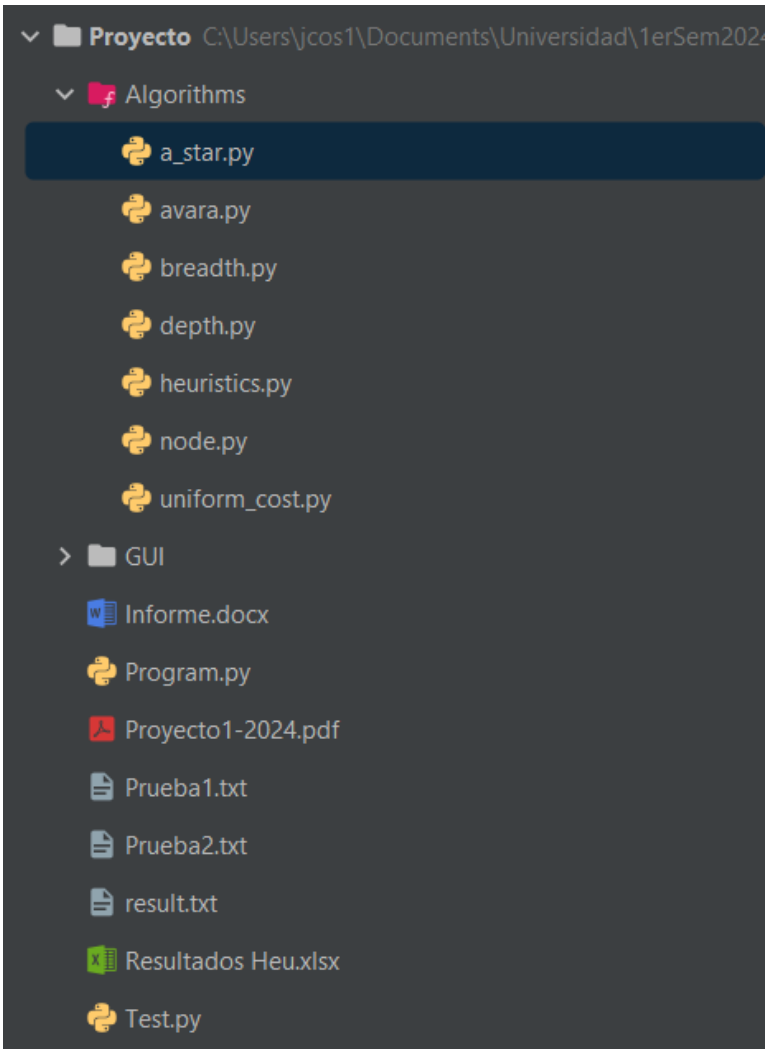
Fecha: 12 de abril del 2024

Código

El archivo principal es el llamado Program.py que se encuentra en la raíz, dentro del cual se debe especificar en la variable file_name el nombre del ambiente que debe estar al mismo nivel en la carpeta.

Se creó la carpeta Algorithms que contiene los diferentes tipos de búsqueda disponibles y la clase node que representa un nodo en el árbol.

Se creó la carpeta GUI que contiene lo necesario para crear la interfaz, ejecutar la búsqueda y mostrar los resultados.



El acceso al código se encuentra en: [Github]

https://github.com/cmlooz/MandoQuest.git

Función Heurística

Se definió la función heurística (Heuristic) como la distancia en L de la siguiente manera:

- Recibe los parámetros:
  - goal: meta (x,y)
  - curr_node: posición actual (x,y)
  - has_ship: tiene nave (acumulado de movimientos con la nave)
  - world: ambiente([][])
  - max_ship_fuel: cantidad máxima de movimientos con la nave
- Se aplica la función distancia en L entre curr_node y goal
- Si tiene la nave, reducimos el valor del movimiento a la mitad
- Si está en una casilla donde hay un enemigo, incrementamos el valor del movimiento

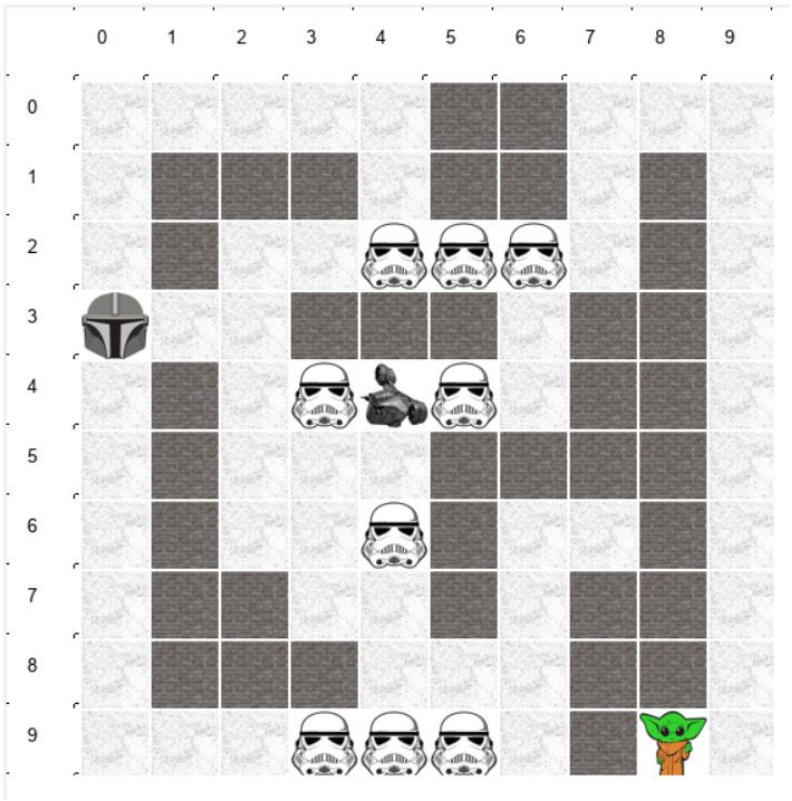
```python
def Heuristic(goal, curr_node, has_ship, world, max_ship_fuel):
    cell = world[curr_node[0]][curr_node[1]]
    #Distancia de Manhattan
    heu = abs(curr_node[0] - goal[0]) + abs(curr_node[1] - goal[1])
    # Si tiene la nave, reducimos el costo de movimiento a la mitad
    if 0 < has_ship <= max_ship_fuel:
        heu *= 0.5
    else:
        # Si estamos en una casilla donde hay un enemigo, incrementamos el costo
        if cell == 4:
            heu += 4
    #return 0 if cell == 5 else heu
    return heu
```

Se hizo la comparación con la función heurística usando la distancia en línea recta (Heuristic_altern) aplicando las mismas condiciones y se tomó la primera opción considerando que esta función de distancia nos da valores mas grandes siendo una heurística dominante sin dejar de ser optimista.

```python
def Heuristic_altern(goal, curr_node, has_ship, world, max_ship_fuel):
    cell = world[curr_node[0]][curr_node[1]]
    #Línea recta
    heu = math.sqrt(((goal[0] - curr_node[0])**2) + ((goal[1] - curr_node[1])**2))
    # Si tiene la nave, reducimos el costo de movimiento a la mitad
    if 0 < has_ship <= max_ship_fuel:
        heu *= 0.5
    else:
        # Si estamos en una casilla donde hay un enemigo, incrementamos el costo
        if cell == 4:
            heu += 4
    #return 0 if cell == 5 else heu
    return heu
```

Se creó el archivo Test.py (escribe los resultados en el archivo result.txt) para realizar las comparaciones entre las funciones heurísticas (distancia en L y distancia en línea recta) para cada caso en cada función se ejecutó la función con y sin nave con las siguientes condiciones:

- Ambiente:



- Nodos:
  - (0,3)
  - (2,6)
  - (2,9)
  - (8,9)
  - (9,8)
  - (9,9)

Se obtuvieron los siguientes resultados que ayudaron a la selección de la heurística

	A	B	C	D
	Posición	Lleva Nave	En L (Heuristic)	Línea Recta (Heuristic_altern)
	(0,3)	No	14	10,296
	(2,6)	No	13	11,280
	(2,9)	No	8	7,071
	(8,9)	No	2	1,414
	(9,9)	No	1	1,000
	(8,9)	No	0	0,000
	(0,3)	Sí	7	5,148
	(2,6)	Sí	4,5	3,640
0	(2,9)	Sí	4	3,536
1	(8,9)	Sí	1	0,707
2	(9,9)	Sí	0,5	0,500
3	(8,9)	Sí	0	0,000