

Alejandro Enrique Vides Vivero – 202221712

William Anthony Pollock Ortega – 202221321

Andres Moncada Roldán - 202510444

Punto 1: DFS

Complejidad:

Tiempo: $O(b^m)$

Espacio: $O(bm)$

donde $b \leq 4$ (movimientos ortogonales) y m es la profundidad máxima del árbol.

Complejitud:

Sí, en este dominio finito y usando lista de visitados.

Optimalidad:

No es óptimo ni en pasos ni en costo energético.

Punto 2: BFS

Complejidad:

Tiempo: $O(b^d)$

Espacio: $O(b^d)$

donde $b \leq 4$ y d es la profundidad de la solución más superficial.

Complejitud:

Sí, en este dominio finito y usando lista de visitados.

Optimalidad:

Es óptimo en pasos mas no en costo energético.

Punto 3: UCS

Complejidad:

Tiempo (peor caso): $O(b^{(C^*/\epsilon)})$

Espacio: $O(b^{(C^*/\epsilon)})$

donde: $b \leq 4$

C^* = costo óptimo de la solución

ϵ = costo mínimo por paso (aquí $\epsilon = 1$)

Complejitud:

Sí es completo

Optimalidad:

Es óptimo en términos de costo.

Punto 4: AStar, Manhattan & Euclidian

Tiempo: $O(b^{(C^*/\epsilon)})$

Espacio: $O(b^{(C^*/\epsilon)})$

Complejitud:

Sí es completo en este dominio finito

Optimalidad:

Sí, siempre que la heurística sea admisible.

Heurísticas:

En este problema, tanto la heurística Manhattan como la heurística Euclidiana son consistentes. Esto se debe a que el robot se mueve únicamente en cuatro direcciones y cada movimiento tiene un costo mínimo de 1, por lo que la estimación del costo restante nunca disminuye más rápido de lo que realmente cuesta avanzar un paso.

Punto 5: Survivor

La heurística Survivor se puede considerar consistente ya que consiste en parte de la heurística Manhattan que ya es consistente con el MST que también es consistente porque solo varía una vez se rescata a un sobreviviente, y esta variación únicamente reduce el conjunto, no suma.

Uso de IA

La IA fue utilizada principalmente para correr de manera correcta el código dentro de la entrega, ya que inicialmente algunos datos no se estaban importando y otros mostraban errores. La IA agregó archivos ‘__pycache__’ dentro de los folders, y gracias a esto el código de prueba ya corría correctamente. También se usó para entender de manera más clara qué debía estar dentro de las fronteras en algunos de los algoritmos de búsqueda y también de qué forma se tenían que atacar las soluciones.