PROYECTO PARTE 3. RESOLVER LABERINTOS O MAPAS USANDO MÉTODOS DE BÚSQUEDA

En esta parte del proyecto se deben resolver automáticamente laberintos o mapas usando algoritmos de búsqueda.

Integrar a su sistema lo siguiente:

1. (10) El usuario puede establecer la prioridad direccional para el algoritmo de búsqueda. En caso de búsqueda heurística esta prioridad determina la elección de nodos empatados en costo. Note que este desempate se da en el mismo nivel, es decir, si hay nodos del nivel 3 con mismo costo que nivel 2, los primeros en entrar en la lista (nivel2) son los primeros en salir. En otras palabras, la prioridad direccional me desempata nodos con mismo costo en el mismo nivel.
   Para el caso de búsqueda ciega, esta prioridad determina el orden de visita de los nodos hijos.
2. Permita elegir la medida de distancia para sus algoritmos de búsqueda heurística:
   a. (7.5) Implemente la medida Manhattan
   b. (7.5) Implemente la medida Euclidiana
3. Permita elegir el algoritmo de búsqueda con el que se resolverá el problema:
   a. (20) Resuelva automáticamente el mapa usando el algoritmo de coste uniforme.
   b. (20) Resuelva automáticamente el mapa usando el algoritmo de búsqueda voraz primero el mejor.
   c. (15) Resuelva automáticamente el mapa usando el algoritmo A*.

      Tome en cuenta que estos algoritmos evalúan todas las casillas(nodos) alcanzables y en caso de encontrar un nodo repetido que tiene mejor f(n), se quedará con este nuevo nodo y eliminará el nodo repetido anterior. No se evaluarán nodos que ya hayan sido cerrados/visitados.

4. (5) Al llegar al estado final debe mostrar en el mapa la ruta de solución encontrada (note que la solución no es lo mismo que el recorrido llevado a cabo para encontrarla).
5. (15) Al finalizar la búsqueda, se debe mostrar el árbol de búsqueda generado en donde se pueda observar el nivel de los nodos (con el fin de identificar nodos padres e hijos), nombre de los nodos, órden de visita de los nodos, estado del nodo (visitado o no), costo del nodo f(n) (para el caso de A* separar por g(n) y h(n)). Tome en cuenta que este árbol se puede ver como los que hacemos en pizarrón cuando resolvemos un problema con cualquier algoritmo de búsqueda, independientemente de si lo está graficando o mostrando como carpetas y archivos.

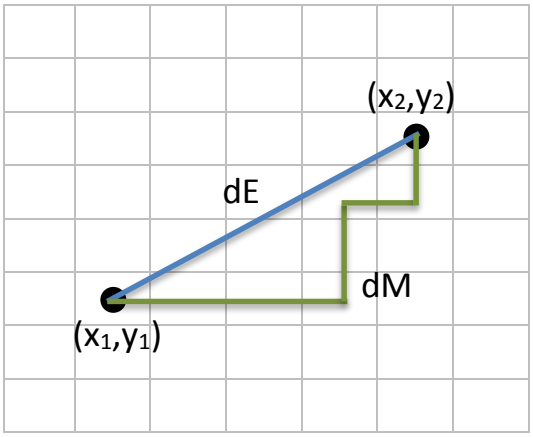
Ejemplo de búsqueda heurística

Distancia Euclideana

$$d_E(P_1, P_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Distancia Manhattan

$$d_M(P_1, P_2) = |x_1 - x_2| + |y_1 - y_2|$$



Considere el siguiente ejemplo en donde ponemos un mono en b,4 y debe llegar a g,4. El costo de movimiento para el terreno amarillo es de 1 mientras que para el terreno azul es de 4. Utilizando el algoritmo A*, estableciendo el órden abajo, izquierda, arriba, derecha y utilizando la medida Manhattan como h(n), el segundo movimiento del ser se vería como sigue:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1															
2															
3		(1+6=7)													
4	(1+6=7)	I,1, (0+5=5)	(4+4=8)				F								
5	(2+7=9)	X,2,(1+6=7)	(5+5=10)												
6		(2+7=9)													
7															
8															
9															
10															
11															
12															
13															
14															
15															

En el ejemplo, los nodos expandidos se encuentran desenmascarados, los nodos cerrados-visitados contienen el número de visita y cada nodo tiene entre paréntesis el costo total o f(n). La “X” indica la posición actual del ser en el mapa, la “I” indica la casilla inicial y “F” la casilla final.