SI – P1

Saúl Leyva Santarén - Joaquín Solla Vázquez

GRUPO 2.5

Ejercicio 1:

-Búsqueda avara

Paso	Frontera	Explorados
1	start(2)	-
2	A(6), B(4), C(6)	start(0)
3	A(6), E(2), C(6)	start(0), B(10)
4	A(6), C(6), D(2), end(0)	start(0), B(10), E(12)
5	A(6), C(6), D(2)	start(0), B(10), E(12), end(18)

-Camino solución: start -> B -> E -> end

-Coste camino: 18

-Búsqueda A*

Paso	Frontera	Explorados
1	start(0+2)	-
2	A(5+6), B(10+4), C(5+6)	start(0)
3	B(10+4), C(5+6), E(9+2)	start(0), A(5)
4	B(10+4), E(9+2), F(11+3)	start(0), A(5), C(5)
5	B(9+4), F(11+3), D(13+2), end(15)	start(0), A(5), C(5), E(9)
6	F(11+3), D(13+2), end(15)	start(0), A(5), C(5), E(9), B(9)
7	D(13+2), end(14)	start(0), A(5), C(5), E(9), B(9), F(11)
8	D(13+2)	start(0), A(5), C(5), E(9), B(9), F(11), end(14)

-Camino solución: start -> C -> F -> end

-Coste camino: 14

El algoritmo de búsqueda A* ha encontrado un camino con menor coste (14) que el algoritmo de búsqueda avara (18). Por otro lado, este último necesitó de 5 pasos para encontrar la solución, mientras que A* realizó 8 pasos.

Ejercicio 2:

1-Distancias

Inicio: 8 Uno: 0 Dos: 6 Tres: 8 Meta: 10	Inicio: 7 Uno: 1 Dos: 5 Tres: 7 Meta: 9	Inicio: 6 Uno: 2 Dos: 4 Tres: 6 Meta: 8		Inicio: 10 Uno: 10 Dos: 6 Tres: 2 Meta: 0
		Inicio: 5 Uno: 3 Dos: 3 Tres: 5 Meta: 7		Inicio: 9 Uno: 9 Dos: 5 Tres: 1 Meta: 1
Inicio: 2 Uno: 6 Dos: 4 Tres: 6 Meta: 8	Inicio: 3 Uno: 5 Dos: 3 Tres: 5 Meta: 7	Inicio: 4 Uno: 4 Dos: 2 Tres: 4 Meta: 6		Inicio: 8 Uno: 8 Dos: 4 Tres: 0 Meta: 2
Inicio: 1 Uno: 7 Dos: 5 Tres: 7 Meta: 9		Inicio: 5 Uno: 5 Dos: 1 Tres: 3 Meta: 5	Inicio: 6 Uno: 6 Dos: 2 Tres: 2 Meta: 4	Inicio: 7 Uno: 7 Dos: 3 Tres: 1 Meta: 3
Inicio: 0 Uno: 8 Dos: 6 Tres: 8 Meta: 10		Inicio: 6 Uno: 6 Dos: 0 Tres: 4 Meta: 6	Inicio: 7 Uno: 7 Dos: 1 Tres: 3 Meta: 5	Inicio: 8 Uno: 8 Dos: 2 Tres: 2 Meta: 4

	Inicio	**Uno**	**Dos**	**Tres**	**Meta**
Inicio	0	8	6	8	10
Uno	8	0	6	8	10
Dos	6	6	0	4	6
Tres	8	8	4	0	2
Meta	10	10	6	2	0

2-Estados

14 ESTADOS:

1. Se encuentra en "**inicio**" sin haber recogido ningún paquete.
2. Se encuentra en "**uno**" y viene de "inicio" (tiene 0 paquetes).
3. Se encuentra en "**dos**" y viene de "inicio" (tiene 0 paquetes).
4. Se encuentra en "**tres**" y viene de "inicio" (tiene 0 paquetes).
5. Se encuentra en "**dos**" y tiene el paquete "uno".
6. Se encuentra en "**tres**" y tiene el paquete "uno".
7. Se encuentra en "**uno**" y tiene el paquete "dos".

8. Se encuentra en “**tres**” y tiene el paquete “dos”.
9. Se encuentra en “**uno**” y tiene el paquete “tres”.
10. Se encuentra en “**dos**” y tiene el paquete “tres”.
11. Se encuentra en “**tres**” y tiene los paquetes “uno” y “dos”.
12. Se encuentra en “**dos**” y tiene los paquetes “uno” y “tres”.
13. Se encuentra en “**uno**” y tiene los paquetes “dos” y “tres”.
14. Se encuentra en “**meta**” con todos los paquetes (3).

3-Acciones

1. Ir a “uno”.
2. Ir a “dos”.
3. Ir a “tres”.
4. Ir a “meta” -> se entregan los paquetes.

*Se considera que cuando se va a una localidad con paquete, se coge su paquete.

4-Heurística

Heurística distancia Manhattan + coste de levantar paquete. Sumaremos la distancia mínima posible hasta la meta, sin tener en cuenta las casillas que no pueden ser visitadas. Cuando se levante un paquete, también se le sumará su peso.

- k = nº acciones no realizadas

- $(x, y)_0$ = coordenadas actuales

- $(x, y)_k$ = coordenadas meta

- w = peso del objeto (inicio y meta no tienen)

k=0 -> 0 (Está en la meta)

k>0 -> $h(n) = \min(|x_0 - x_1| + |y_0 - y_1| + w_1 + |x_1 - x_2| + |y_1 - y_2| + w_2 + \ldots + |x_{k-1} - x_k| + |y_{k-1} - y_k|,$
$|x_0 - x_2| + |y_0 - y_2| + w_2 + |x_2 - x_1| + |y_2 - y_1| + w_1 + \ldots + |x_{k-2} - x_k| + |y_{k-2} - y_k|, \ldots)$

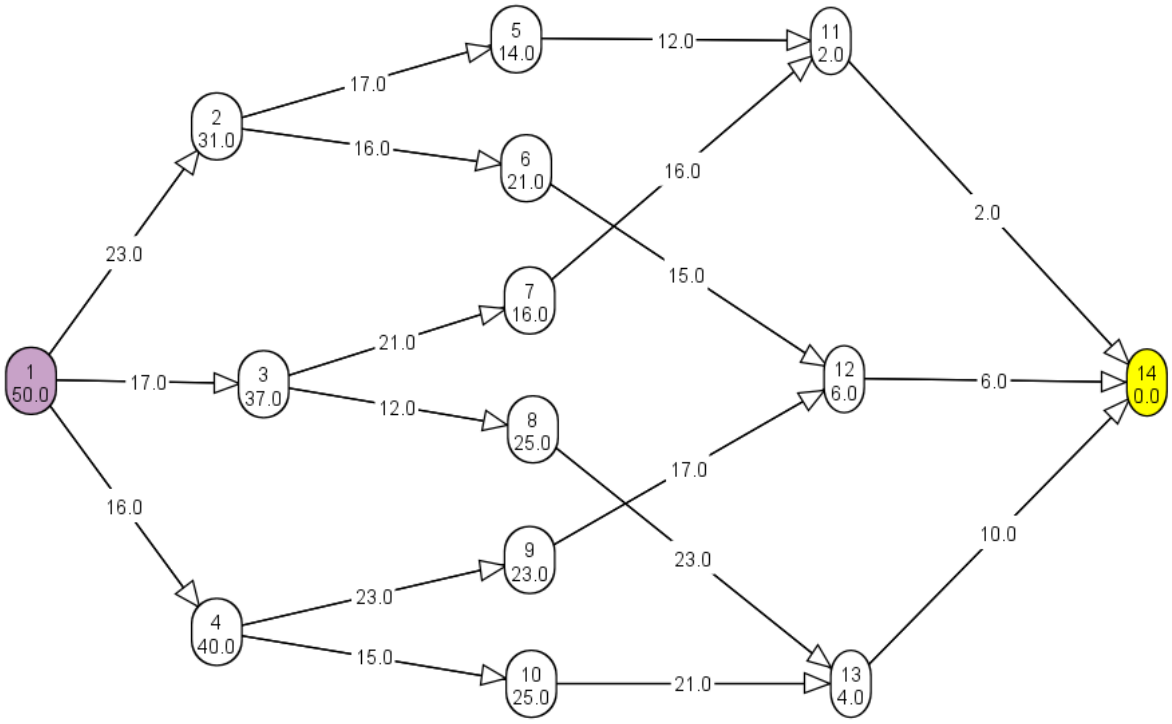
Se coge el valor mínimo de todas las posibles combinaciones diferentes de $(x, y)_1$ hasta $(x, y)_{k-1}$

Ej: $h(2) = |0 - 2| + |0 - 4| + 11 + |2 - 4| + |4 - 2| + 8 + |4 - 4| + |2 - 0| = 31 <= 31$ coste real

Uno -> (0, 0) Dos -> (2, 4) Tres -> (4, 2) Meta -> (4, 0) k=3

Esta es una buena heurística porque el coste estimado es siempre menor o igual que el coste mínimo de alcanzar el estado objetivo.

5-Grafo



6-Resultados

1) Depth First

- Camino: 1 -> 2 -> 5 -> 11 -> 14(Meta)
- Coste camino: 54
- Nodos expandidos: 5

2) Breadth First

- Camino: 1 -> 2 -> 5 -> 11 -> 14(Meta)
- Coste camino: 54
- Nodos expandidos: 17

3) Lowest Cost First (Coste Uniforme)

- Camino: 1 -> 2 -> 5 -> 11 -> 14(Meta)
- Coste camino: 54
- Nodos expandidos: 16

4) Best First (Avara)

- Camino: 1 -> 2 -> 5 -> 11 -> 14(Meta)

- Coste camino: 54

- Nodos expandidos: 5

5) A*

- Camino: 1 -> 2 -> 5 -> 11 -> 14(Meta)

- Coste camino: 54

- Nodos expandidos: 8

7-Conclusiones y discusión

Los 5 algoritmos empleados en el punto anterior obtienen el mismo camino (con el mismo peso). Aun así, se pueden sacar conclusiones del ejercicio:

A cada algoritmo le lleva una cantidad de pasos y nodos expandidos diferente llegar a la solución del problema. En este caso los mejores algoritmos para dar con el mejor camino serían Depth First y Best First, a los cuales sólo les toma 5 pasos llegar a la solución.

Los resultados de los algoritmos sí son óptimos en este caso puesto que, tras observar el grafo del apartado 5, el camino determinado por los algoritmos es el más corto de todos los posibles.

En este caso ocurre, pero no siempre van a dar la solución óptima todos estos algoritmos. Dependiendo del problema estos pueden dar con una solución válida simplemente o, como en este caso, encontrar la solución óptima.

Algunos algoritmos no darían el mismo resultado si hiciéramos alguna pequeña variación, por ejemplo, al disminuir el coste de levantar el paquete “uno”, algunos algoritmos seguirían dando el mismo camino pero algún otro como Best First, devolvería otra solución.

Los algoritmos que mejor funcionarían ante variaciones en el problema son Breadth First y Lowest Cost First, ya que son los que expanden la mayor cantidad de nodos para que, en caso de existir, encontrar una solución mejor que la de los algoritmos que expanden notablemente menos nodos (es decir, analizan muchos más casos y caminos que el resto).