MEMORIA INTELIGENCIA ARTIFICIAL

PRÁCTICA 1

871627 Irene Pascual Albericio

Resolución del Problema EightPuzzleDemo:

Este proyecto tiene como objetivo resolver el problema del 8-Puzzle mediante diversos algoritmos de búsqueda no informada, como: BFS (BreadthFirstSearch), DFS (DepthFirstSearch), DLS (DepthLimitedSearch), IDS (IterativeDeepeningSearch) y UCS (UniformCostSearch), todos implementados tanto en árbol como en grafo (excepto el DLS Y el IDS).

Datos obtenidos:

He creado una función “printPractica1”, para poder mostrar la línea de cada algoritmo con sus datos correspondientes. Muchos datos los obtuve directamente mediante las propiedades almacenadas en **aima.core.search.framework.nodeExpander**. No obstante, hay ciertos datos que he tenido que crear, como es el caso de los nodos expandidos, la complejidad temporal y la complejidad espacial.

Para poder obtener el número de **nodos expandidos** por cada algoritmo, he realizado modificaciones en el archivo aima.core.search.framework.nodeExpander. Se ha añadido la variable “nodesExpanded”, que cuenta los nodos expandidos durante el proceso de búsqueda. En la parte del bucle del final del archivo, he añadido al final del bucle “for” la suma en 1 de la variable “nodesExpanded” (de tal manera que cada vez que se finaliza el recorrido de un nodo, se suma un nodo a los expandidos).

Los algoritmos implementados se encargan de explorar diferentes estrategias de búsqueda para encontrar la solución al problema:

1. **BFS (BreadthFirstSearch):** Explora el árbol por niveles, expandiendo primero los nodos cercanos a la raíz. En cuanto a la complejidad temporal y espacial, esta es **O(b^(d+1))**, donde b es el factor de ramificación y d es la profundidad de la solución.

2. **DFS (DepthFirstSearch):** Prioriza la búsqueda en profundidad, explorando caminos completos antes de retroceder. La complejidad temporal es **O(b^m)** y la espacial es **O(b*m)**, siendo m la máxima profundidad.

3. **DLS (DepthLimitedSearch):** DFS limitado por una profundidad máxima l. La complejidad temporal es **O(b^l)** y la espacial es **O(b*l)**.

4. **IDS (IterativeDeepeningSearch):** Combina las ventajas de BFS y DFS, realizando DFS repetidamente con profundidades crecientes. Tiene una complejidad temporal de **O(b^m)** y espacial de **O(b*m)**.

5. **UCS (UniformCostSearch):** Utiliza el costo acumulado de los caminos para expandir los nodos de menor costo primero. Su complejidad depende del costo óptimo C* y la distancia óptima e, con una complejidad temporal y espacial de **O(b^(C*/e))**.

ANEXO 1. FOTO RESULTADOS EIGHT PUZZLE DEMO

Resolución del Problema de los Misioneros y Caníbales

Este proyecto implementa la resolución del problema clásico de los Misioneros y Caníbales utilizando los algoritmos de búsqueda BFS (BreadthFirstSearch), DLS (DepthLimitedSearch) e IDLS (IterativeDeepeningSearch).

Definición del Problema

El problema consiste en llevar a tres misioneros y tres caníbales desde un lado del río hasta el otro usando una barca. La barca tiene capacidad limitada, pudiendo transportar un máximo de dos personas en cada viaje. Las restricciones principales son que nunca supere el número de caníbales al de los misioneros.
El estado inicial se define como {3,3,0,0,0}, donde los tres primeros números representan la cantidad de caníbales, misioneros y la barca en la parte del río a la izquierda, y los otros dos la cantidad de caníbales y misioneros en la derecha.

Implementación de Acciones y Estados

Las acciones disponibles para el problema en el archivo aima.core.environment.canibales. Las acciones posibles son:

1. Mover 1 Caníbal (C1)
2. Mover 2 Caníbales (C2)
3. Mover 1 Misionero (M1)
4. Mover 2 Misioneros (M2)
5. Mover 1 Caníbal y 1 Misionero (M1C1)

Cada acción tiene una correspondiente operación que se encarga de actualizar los valores del estado mediante la función setValue (que suma o resta al estado actual el número que se desee para cada valor).

Validación de Estados y Comprobaciones

La función “sePuedeMover” comprueba si una acción es válida antes de ejecutarse (es decir, que no viole las restricciones). Una vez ejecutada la acción, la función “esEstadoValido” verifica si el estado alcanzado no tiene por ejemplo algún número negativo en alguna de sus componentes entre otras cosas.

Implementación de los Algoritmos de Búsqueda

Se implementaron tres algoritmos de búsqueda para resolver el problema:

6. **BFS (BreadthFirstSearch):** Se explora el árbol de búsqueda por niveles. En este caso, se implementa una búsqueda en grafo para evitar estados repetidos.

7. **DLS (DepthLimitedSearch):** Este algoritmo realiza una búsqueda en profundidad limitada por un máximo de 11 niveles, evitando un recorrido excesivamente profundo.

8. **IDLS (IterativeDeepeningSearch):** Combina las ventajas de BFS y DFS, realizando búsquedas en profundidad de forma iterativa con profundidades crecientes.

Resultados y Comparación

Cada algoritmo imprime los estados intermedios y las acciones realizadas hasta llegar al estado objetivo, además de los datos del algoritmo en cuestión.

ANEXO 2. FOTO RESULTADOS CANÍBALES Y MISIONEROS

ANEXO 1. FOTO RESULTADOS EIGHT PUZZLE DEMO

Problema	Profundidad	Expand	Q.Size	MaxQS	tiempo	Generados	Ritmo	Complejidad temp	Complejidad esp
BFS-G-3	3	5	4	5	2	14	7000,00	O(b^(d) ->	256	O(b^(d) ->	256
BFS-T-3	3	6	9	10	0	17	Infinity	O(b^(d) ->	256	O(b^(d) ->	256
DFS-G-3	59123	120491	39830	42913	489	321072	656588,96	O(b^m) ->	64	O(b*m) ->	12
DFS-T-3	---	---	---	---	(1)	---	---		---		---
DLS-9-3	9	10	0	0	1	26	26000,00	O(b^m) ->	262144	O(b*m) ->	36
DLS-3-3	3	4	0	0	0	11	Infinity	O(b^m) ->	64	O(b*m) ->	12
IDS-3	3	9	0	0	0	0	NaN	O(b^m) ->	262144	O(b*m) ->	36
UCS-G-3	3	16	9	10	2	40	20000,00	O(b^(1+C*/e)) ->	256	O(b^(1+C*/e)) ->	256
UCS-T-3	3	32	57	58	0	89	Infinity	O(b^(1+C*/e)) ->	256	O(b^(1+C*/e)) ->	256
BFS-G-9	9	288	198	199	2	801	400500,00	O(b^(d) ->	4611686018427387904	O(b^(d) ->	4611686018427387904
BFS-T-9	9	5821	11055	11056	7	16878	2411142,86	O(b^(d) ->	4611686018427387904	O(b^(d) ->	4611686018427387904
DFS-G-9	44665	141452	32012	42967	288	374614	1300743,06	O(b^m) ->	262144	O(b*m) ->	12
DFS-T-9	---	---	---	---	(1)	---	---		---		---
DLS-9-9	9	5474	0	0	5	15924	3184800,00	O(b^m) ->	262144	O(b*m) ->	36
DLS-3-9	0	12	0	0	0	35	Infinity	O(b^m) ->	64	O(b*m) ->	12
IDS-9	9	9063	0	0	20	0	0,00	O(b^m) ->	262144	O(b*m) ->	36
UCS-G-9	9	385	235	239	2	1040	520000,00	O(b^(1+C*/e)) ->	1048576	O(b^(1+C*/e)) ->	1048576
UCS-T-9	9	18070	31593	31594	21	49663	2364904,76	O(b^(1+C*/e)) ->	1048576	O(b^(1+C*/e)) ->	1048576
BFS-G-30	30	181058	365	24048	364	482871	1326568,68	O(b^(d) ->	4611686018427387904	O(b^(d) ->	4611686018427387904
BFS-T-30	---	---	---	---	(2)	---	---		---		---
DFS-G-30	62856	80569	41533	41534	225	219540	975733,33	O(b^m) ->	115292150460684697 6	O(b*m) ->	12
DFS-T-30	---	---	---	---	(1)	---	---		---		---

ANEXO 2. FOTO RESULTADOS CANIBALES Y MISIONEROS

```
Misioneros y canibales BFS->
queueSize : 1
maxQueueSize : 3
nodesGenerated : 28
pathCost : 11.0
nodesExpanded : 13
Tiempo : 3mls

SOLUCIÓN:
   GOAL STATE
   RIBERA-IZQ           --RIO-- BOTE  M M M C C C    RIBERA-DCH
   CAMINO ENCONTRADO:
          ESTADO INICIAL:    RIBERA-IZQ M M M C C C BOTE --RIO-- RIBERA-DCH

          Action[name==C2]   RIBERA-IZQ M C C C --RIO-- BOTE M M RIBERA-DCH
          Action[name==C1]   RIBERA-IZQ M M C C C --RIO-- BOTE M RIBERA-DCH
          Action[name==C2]   RIBERA-IZQ M C C C --RIO-- BOTE M M RIBERA-DCH
          Action[name==C1]   RIBERA-IZQ M M C C C --RIO-- BOTE M RIBERA-DCH
          Action[name==M2]   RIBERA-IZQ M M M C C C BOTE --RIO-- RIBERA-DCH
        Action[name==M1C1]   RIBERA-IZQ M M C C --RIO-- BOTE M C RIBERA-DCH
          Action[name==M2]   RIBERA-IZQ M M M C C C BOTE --RIO-- RIBERA-DCH
          Action[name==C1]   RIBERA-IZQ M M C C C --RIO-- BOTE M RIBERA-DCH
          Action[name==C2]   RIBERA-IZQ M C C C --RIO-- BOTE M M RIBERA-DCH
          Action[name==C1]   RIBERA-IZQ M M C C C --RIO-- BOTE M RIBERA-DCH
          Action[name==C2]   RIBERA-IZQ M C C C --RIO-- BOTE M M RIBERA-DCH
--------------------------------------------------------------------------------
```

```
Misioneros y canibales DLS(11) ->
nodesGenerated : 5519
pathCost : 11.0
nodesExpanded : 2225
Tiempo : 15mls

SOLUCIÓN:
   GOAL STATE
   RIBERA-IZQ           --RIO-- BOTE  M M M C C C    RIBERA-DCH
   CAMINO ENCONTRADO:
          ESTADO INICIAL:    RIBERA-IZQ M M M C C C BOTE --RIO-- RIBERA-DCH

          Action[name==C2]   RIBERA-IZQ M C C C --RIO-- BOTE M M RIBERA-DCH
          Action[name==C1]   RIBERA-IZQ M M C C C --RIO-- BOTE M RIBERA-DCH
          Action[name==C2]   RIBERA-IZQ M C C C --RIO-- BOTE M M RIBERA-DCH
          Action[name==C1]   RIBERA-IZQ M M C C C --RIO-- BOTE M RIBERA-DCH
          Action[name==M2]   RIBERA-IZQ M M M C C C BOTE --RIO-- RIBERA-DCH
        Action[name==M1C1]   RIBERA-IZQ M M C C --RIO-- BOTE M C RIBERA-DCH
          Action[name==M2]   RIBERA-IZQ M M M C C C BOTE --RIO-- RIBERA-DCH
          Action[name==C1]   RIBERA-IZQ M M C C C --RIO-- BOTE M RIBERA-DCH
          Action[name==C2]   RIBERA-IZQ M C C C --RIO-- BOTE M M RIBERA-DCH
          Action[name==C1]   RIBERA-IZQ M M C C C --RIO-- BOTE M RIBERA-DCH
          Action[name==C2]   RIBERA-IZQ M C C C --RIO-- BOTE M M RIBERA-DCH
--------------------------------------------------------------------------------
```

```
Misioneros y canibales IDLS->
pathCost : 11.0
nodesExpanded : 8542
Tiempo : 11mls

SOLUCIÓN:
   GOAL STATE
   RIBERA-IZQ           --RIO-- BOTE  M M M C C C    RIBERA-DCH
   CAMINO ENCONTRADO:
          ESTADO INICIAL:    RIBERA-IZQ M M M C C C BOTE --RIO-- RIBERA-DCH

          Action[name==C2]   RIBERA-IZQ M C C C --RIO-- BOTE M M RIBERA-DCH
          Action[name==C1]   RIBERA-IZQ M M C C C --RIO-- BOTE M RIBERA-DCH
          Action[name==C2]   RIBERA-IZQ M C C C --RIO-- BOTE M M RIBERA-DCH
          Action[name==C1]   RIBERA-IZQ M M C C C --RIO-- BOTE M RIBERA-DCH
          Action[name==M2]   RIBERA-IZQ M M M C C C BOTE --RIO-- RIBERA-DCH
        Action[name==M1C1]   RIBERA-IZQ M M C C --RIO-- BOTE M C RIBERA-DCH
          Action[name==M2]   RIBERA-IZQ M M M C C C BOTE --RIO-- RIBERA-DCH
          Action[name==C1]   RIBERA-IZQ M M C C C --RIO-- BOTE M RIBERA-DCH
          Action[name==C2]   RIBERA-IZQ M C C C --RIO-- BOTE M M RIBERA-DCH
          Action[name==C1]   RIBERA-IZQ M M C C C --RIO-- BOTE M RIBERA-DCH
          Action[name==C2]   RIBERA-IZQ M C C C --RIO-- BOTE M M RIBERA-DCH
-------------------------------------------------------------------------------
```