



Práctica 2 :

Resolución de problemas y búsqueda.

Búsqueda informada

1. Objetivo de la práctica

Familiarizarse con el código en java para resolver problemas de búsqueda informada y diferenciar sus características. Volveremos a utilizar el código en <http://code.google.com/p/aima-java/>.

Tu trabajo consistirá básicamente en realizar experimentos y recopilar información relevante de la búsqueda relacionada con aspectos de eficiencia. Realizaremos con el 8-puzzle primero búsquedas ciegas (BFS e IDS) y búsquedas informadas A* con las heurísticas de fichas descolocadas y Manhattan. En esta práctica vamos a utilizar la búsqueda A*, y a comparar la eficiencia de los algoritmos mostrando el número de nodos generados y el factor de ramificación efectivo b^* para distintas profundidades.

2. Tareas

Se deben entregar en un zip solos las clases escritas o modificadas, y una memoria con la tabla(s) obtenidas y los comentarios que estimes oportunos.

Notas de las tareas a realizar:

1. Factor de ramificación efectivo con heurísticas del problema del 8 puzle (**8/10**. Obligatorio). Se mostrará una tabla con los resultados tal como se muestra en el ejemplo:
 - Necesitarás implementar una clase Biseccion en `aima.core.util.math` que permite obtener los ceros de la función $N = b^* (b^{*d} - 1) / (b^* - 1)$ por aproximaciones sucesivas, donde b^* es el factor de ramificación efectivo, N el número de nodos generados y d la profundidad de la solución.
 - Debes generar 100 experimentos aleatorios de la profundidad deseada y calcular la media de los nodos generados. Puedes utilizar la clase `GenerateInitialEightPuzzleBoard` suministrada para que genere aleatoriamente estados iniciales, y estados finales de la profundidad deseada. El método `random` ejecuta acciones aleatorias desde el estado inicial, donde d es la profundidad deseada sin generar estados repetidos. Aún así, el que se hayan dado d pasos al estado final, no garantiza que no haya caminos más cortos al

estado final. Por lo que en los experimentos, o al generar los estados inicial y final, deberás comprobar que la solución óptima es del coste deseado.

- Tendrás que reescribir las clases `ManhattanHeuristicFunction` y `MisplacedTilleHeuristicFunction` para que sean útiles para cualquier estado final.

Nodos Generados					b*				
d	BFS	IDS	A*h(1)	A*h(2)	BFS	IDS	A*h(1)	A*h(2)	
2	8	11	6	6	2,37	2,85	2,00	2,00	
3	27	42	15	14	2,60	3,09	2,06	2,00	
4	64	138	28	25	2,51	3,12	1,96	1,89	
5	132	412	45	40	2,39	3,09	1,85	1,80	
6	251	1165	71	59	2,28	3,04	1,78	1,72	
7	471	3353	106	81	2,21	3,01	1,73	1,65	
8	838	9407	158	110	2,15	2,98	1,69	1,60	
9	1476	26710	234	145	2,09	2,96	1,66	1,55	
10	2488	76594	348	192	2,04	2,95	1,63	1,52	
11	4147	---	520	255	2,00	---	1,62	1,50	
12	6827	---	790	345	1,97	---	1,61	1,48	
13	11266	---	1204	461	1,94	---	1,60	1,47	
14	18275	---	1845	623	1,91	---	1,59	1,46	
15	29726	---	2820	827	1,89	---	1,59	1,45	
16	47026	---	4327	1153	1,87	---	1,59	1,44	
17	74590	---	6757	1567	1,85	---	1,58	1,44	
18	116378	---	10300	2033	1,83	---	1,58	1,43	
19	180446	---	15982	2735	1,81	---	1,58	1,42	
20	270434	---	24965	3586	1,79	---	1,58	1,42	
21	397135	---	38002	4862	1,78	---	1,57	1,41	
22	571906	---	59236	6609	1,76	---	1,57	1,41	
23	801706	---	89655	8912	1,74	---	1,57	1,41	
24	1086476	---	136721	12109	1,72	---	1,57	1,40	

1. Factor de ramificación con heurísticas del problema de las Fichas (2/10. Opcional):

- Plantea un par de heurísticas para el **problema de las Fichas** en el que se permite desplazar una ficha al hueco saltando **como máximo sobre una ficha** (Problema de las ranas saltarinas¹) y **construye de nuevo una tabla** con los nodos generados y el factor de ramificación efectivo con los algoritmos BFS, IDS, y A* con las heurísticas que has propuesto. Escribe la tabla hasta la profundidad que estimes oportuno.

¹ <https://www.youtube.com/watch?v=HD0NV5lYkS0>

```

+---+---+---+---+---+---+---+
| V | V | V |   | B | B | B |
+---+---+---+---+---+---+

```

