Departamento de Informática
Universidad Carlos III de Madrid

Ingeniería en Informática

Inteligencia Artificial

Octubre 2007

Hoja de Ejercicios 2:
Búsqueda no informada

Comentarios generales sobre los ejercicios

- Asumiendo que se conocen los contenidos teóricos, el tiempo estimado para realizar los ejercicios es de **2 horas**
- Describir las soluciones a los ejercicios de una manera lo más formal posible

Ejercicio 1

Como es sabido, el algoritmo de búsqueda bidireccional consiste en desarrollar, simultaneamente, dos búsquedas: una desde el nodo inicial, *s*, y otra desde el nodo final, *t*. Se pide determinar si el algoritmo bidireccional resultante será o no completo y/o admisible en cada uno de los siguientes supuestos:

1. Cada una de las búsquedas se implementa en amplitud y los costes son siempre iguales a la unidad
2. Ambas búsquedas se implementan en amplitud y los costes son cantidades arbitrarias estrictamente positivas
3. Ambas búsquedas se implementan en profundidad con costes idénticos y siempre iguales a la unidad
4. Una búsqueda es en amplitud y la otra con profundización iterativa, con costes idénticos y siempre iguales a la unidad

dado que, en todos los casos, las búsquedas se alternan inmediatamente. Esto es, después de una expansión en una de ellas, se cambia inmediatamente a la dirección opuesta y así sucesivamente.

Ejercicio 2

1. Resolver el problema de las garrafas mediante el método de búsqueda en amplitud, teniendo en cuenta que:
   - el orden de selección de los operadores es:
     - `llenar_grande`
     - `llenar_pequeña`
     - `vaciar_grande`
     - `vaciar_pequeña`
     - `traspasar_grande_pequeña`
     - `traspasar_pequeña_gande`
   - suponer que cada vez que se genera un nodo, se elimina si en ese momento está en el árbol de búsqueda, es decir, si se detecta un ciclo
2. ¿Puedes garantizar que la solución encontrada anteriormente es óptima?

3. ¿Podrías encontrar una solución con el algoritmo de búsqueda en profundidad, para una una profundidad máxima de 4? En caso afirmativo, ¿te garantizaría la búsqueda en profundidad que la solución sea óptima?

4. ¿Podrías encontrar una solución con el algoritmo de búsqueda en profundidad, para una una profundidad máxima de 6? En caso afirmativo, ¿te garantizaría la búsqueda en profundidad que la solución sea óptima?

5. ¿Podrías encontrar una solución con el algoritmo de búsqueda en profundidad, para una una profundidad máxima de 8? En caso afirmativo, ¿te garantizaría la búsqueda en profundidad que la solución sea óptima?

6. ¿Podrías encontrar una solución con el algoritmo de búsqueda en profundidad iterativa, para una una profundidad máxima inicial de 2, y un incremento de profundidad de 3? En caso afirmativo, ¿puedes garantizar que la solución sea óptima? En caso de no asegurar optimalidad, ¿puedes estimar en qué coste ha sido excedido el óptimo?

7. ¿Se puede resolver este problema con una búsqueda hacia atrás?

8. ¿Y con una búsqueda bidireccional?

Ejercicio 3

Se quiere determinar el número mínimo de movimientos que hacen falta para llegar desde cualquier estado del 15-Puzle hasta uno cualquiera de los estados finales mostrados en la figura 1. En la figura, el blanco está representado con una casilla sólida de color gris.

(a) Primer estado final (b) Segundo estado final

Figura 1: El 15-Puzle

1. Considerando que el coste de cada desplazamiento del blanco es siempre igual a la unidad, ¿cómo puede determinarse el número mínimo de movimientos para llegar desde cualquier estado hasta el estado final de la figura 1(a)?

2. Si intercambiar el blanco con una casilla "*rayada*" tiene un coste nulo, mientras que el coste de desplazar el blanco a una casilla numerada es igual a la unidad, ¿cómo puede determinarse ahora el número mínimo de movimientos para llegar desde cualquier permutación con los mismos símbolos que hay en la figura 1(b) hasta el estado final de la figura 1(b)?

Ejercicio 4

Dado el árbol de la figura 2 donde `B` y `L` son los 2 únicos nodos meta y `A` es el nodo inicial.

Indica en qué orden se visitarían los nodos, distinguiendo nodos generados de nodos expandidos, para los siguientes algoritmos:

1. Amplitud

2. Profundidad

3. Mejor primero, tomando como el mejor nodo en cada ramificación aquel con menor orden alfabético

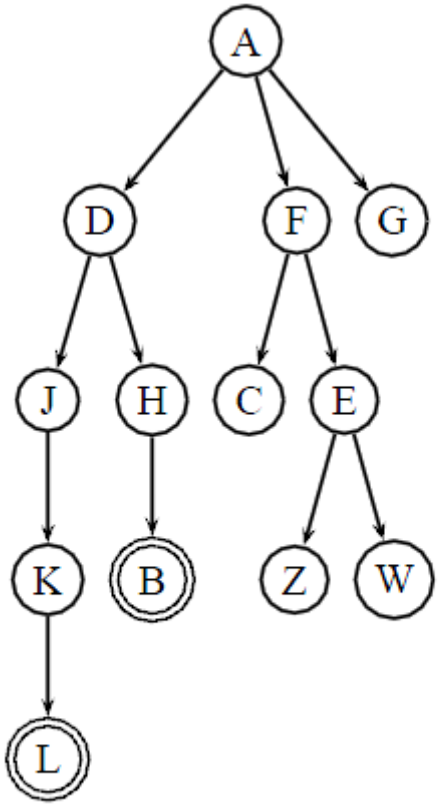

Figura 2: Árbol de búsqueda