Departamento de Informática
Universidad Carlos III de Madrid

Grado en Ingeniería Informática

Inteligencia Artificial

Febrero 2015. Problemas 1. Grupo 81

Normas generales del examen

- El tiempo para realizar el examen es de **2 horas**
- Entregad cada problema en hojas separadas
- Solo se responderán preguntas sobre el examen los primeros 30 minutos
- Si se sale del aula, no se podrá volver a entrar durante el examen
- No se puede presentar el examen escrito a lápiz

Problema 1. (2 puntos)

En un sistema de producción se han introducido las siguientes reglas.

R1: IF elemento(1) THEN ∼elemento(1), `halt`

R2: IF elemento(X), elemento(Y), (X < Y) THEN elemento(X*Y), elemento(X-1), ∼elemento(X), ∼elemento(Y)

R3: IF elemento(X) THEN elemento(X-1)

El símbolo ∼ significa borrar de la base de hechos y `halt` parar el sistema de producción. Las reglas están ordenadas de mayor a menor prioridad, es decir, en caso de que en el conjunto conflicto hubiese una instancia de la R3 y otra de la R2 se ejecutará la instancia de la R2. Del mismo modo, si hay una instancia de la R1 y otra de la R2 se ejecutará la instancia de la R1.

La base de hechos o memoria de trabajo inicial contiene: elemento(5).

Se pide:

1. (1,5 puntos) Mostrar la secuencia de reglas ejecutadas detallando los hechos de la memoria de trabajo y el conjunto conflicto para cada ciclo de ejecución ordenado siguiendo una estrategia de resolución del conjunto conflicto LIFO (profundidad o primero en entrar último en salir).
2. (0,5 puntos) Si la base de hechos inicial contuviera el hecho elemento(15), ¿cual sería el contenido de la base de hechos al parar el sistema de producción?

Problema 2. (3 puntos)

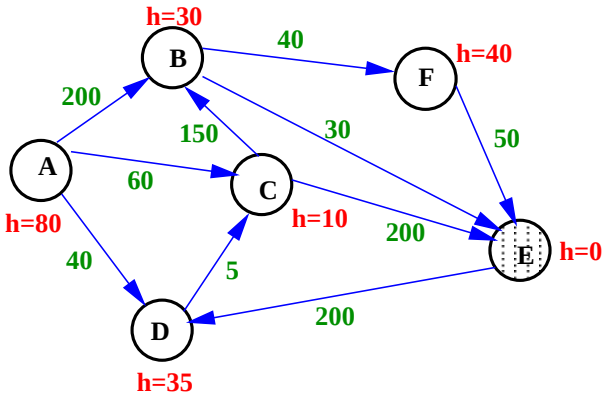
Utilizando sistemas de producción implementar un sistema que simule partidas del juego 3 en raya. El sistema tiene que ser capaz de ir marcando las casillas vacías alternando entre los dos jugadores y detectar cuándo un jugador consigue hacer 3 en raya. No es necesario implementar ningún mecanismo para que el marcado de casillas por cada jugador se haga de forma inteligente. Simplemente se pondría la estrategia *random* de resolución del conjunto conflicto y se dejaría que el sistema simule partidas.

Se pide:

1. (1 punto) Definir la base de hechos o memoria de trabajo necesaria para representar el problema.
2. (2 punto) Definir las reglas del sistema de producción pedido.

Problema 3. (3 puntos)

Dado el siguiente grafo donde se quiere ir desde el nodo A hasta el nodo E y se sigue un orden alfabético para generar sucesores.



Se pide:

1. (0,5 puntos) Pintar el árbol de búsqueda que generaría el algoritmo Dijkstra para resolver el problema, indicando el orden de los nodos generados y expandidos. Decir las características del algoritmo (completitud, admisibilidad y eficiencia).
2. (0,5 puntos) Pintar el árbol de búsqueda que generaría el algoritmo en escalada para resolver el problema, indicando el orden de los nodos generados y expandidos. Decir las características del algoritmo (completitud, admisibilidad y eficiencia).
3. (0,5 puntos) Decir cual es el factor de ramificación máximo y mínimo.
4. (1,5 puntos) Pintar el árbol de búsqueda que generaría el algoritmo A*, indicando el orden de los nodos generados y expandidos. Decir las características del algoritmo (completitud, admisibilidad y eficiencia).

Problema 4. (2 puntos)

Dado el problema de las torres de Hanoi representado en la Figura 1.

Figura 1: Torres de Hanoi.

Se pide:

1. (1 punto) Representar este problema como un problema de búsqueda.
2. (1 punto) Describir alguna heurística admisible e informada para este problema.

El problema de las torres de Hanoi consiste en que inicialmente todos los discos están colocados en el poste de un extremo y se quiere que acaben colocados en el poste del extremo opuesto. Un disco solo se puede colocar o bien en un poste vacío (sin ningún disco), o bien, encima de otro disco de tamaño mayor.

Solución al problema 1

1. La memoria de trabajo y el conjunto conflicto en cada ciclo son:

   MT_0={elemento(5)}
   CC_0={(R3,X=5)}
   MT_1={elemento(5), elemento(4)}
   CC_1={(R2, X=4, Y=5) (R3,X=4)}
   MT_2={elemento(3), elemento(20)}
   CC_2={ (R2, X=3, Y=20), (R3,X=3), (R3,X=20)}
   MT_3={elemento(60), elemento(2)}
   CC_3={(R2, X=2, Y=60),(R3,X=2), (R3,X=60)}
   MT_4={elemento(120), elemento(1)}
   CC_4={(R1),(R2, X=1, Y=120),(R3,X=1), (R3,X=120)}
   MT_5={elemento(120)}

2. El sistema calcula el factorial del elemento inicial en la base de hechos, por tanto, en caso de tener elemento(15), cuando pare el SP la MT contendrá elemento(15!)

Solución al problema 2

1. Se pueden utilizar los siguientes predicados:

   - casilla(X,Y,Z): casilla de la fila X y columna Y (empezamos esquina superior izquierda) tiene la marca Z (puede valer nil, x,o)
   - turno(Z): le toca el turno al jugador que marca con Z (x,o)
   - inversa(X,Y): después del jugador X debe marcar el jugador Y

   La MT_o={inversa(x,o), inversa(o,x), turno(x), casilla(1,1,nil), casilla(1,2,nil) .... , casilla(3,3,nil)}

2. Las reglas serán:

   R1: marcar una casilla en blanco
   SI turno(T), inversa(T,T2), casilla(X,Y,nil) ENTONCES casilla(X,Y,T),turno(T2), ∼turno(T), ∼casilla(X,Y,nil)

   R2: comprueba si un jugador ha hecho tres en raya en horizontal
   SI casilla(X,Y,Z), casilla(X,Y+1,Z), casilla(X,Y+2,Z) ENTONCES stop

   R3: comprueba si un jugador ha hecho tres en raya en vertical
   SI casilla(X,Y,Z), casilla(X+1,Y,Z), casilla(X+2,Y,Z) ENTONCES stop

   R4: comprueba si un jugador ha hecho tres en raya en diagonal
   SI casilla(1,1,Z), casilla(2,2,Z), casilla(3,3,Z) ENTONCES stop

   R5: comprueba si un jugador ha hecho tres en raya en la otra diagonal
   SI casilla(3,1,Z), casilla(2,2,Z), casilla(1,3,Z) ENTONCES stop

   La R1 tiene menos prioridad que las otras.

Solución al problema 3

1. El árbol se muestra en la figura 2. Dijkstra es completo, admisible y la complejidad espacial y temporal es exponencial.

2. El árbol se muestra en la figura 3. No es completo ni admisible y la complejidad espacial y temporal es lineal.

3. El factor de ramificación mínimo es 1 y el máximo 3.

4. El árbol se muestra en la figura 4. Es completo, admisible y la complejidad espacial y temporal es exponencial.

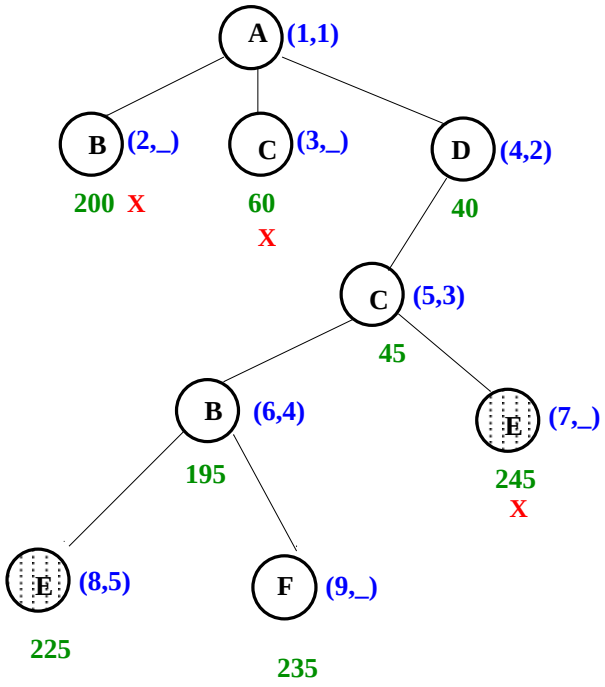


Figura 2: Árbol de búsqueda generado por el algoritmo Dijkstra

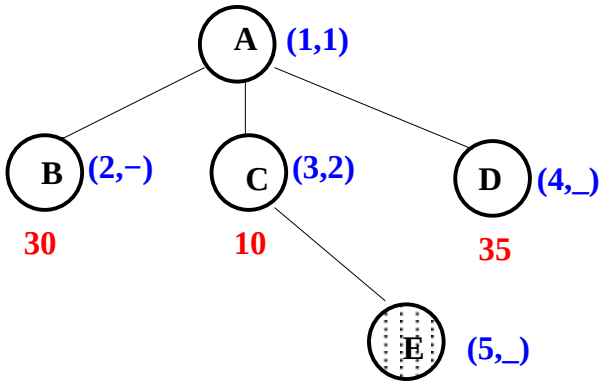


Figura 3: Árbol de búsqueda generado por el algoritmo Escalada

Solución al problema 4

1. Hay que definir los estados, operadores, estado inicial y final.
   - Estados: cada palo se representa por un par (X,Y), donde X representa el número de discos que contiene e Y el tamaño del último disco. Numeramos los discos de menor (1) a mayor tamaño (8)
   - El estado inicial será: ((8,1),(0,0), (0,0))
   - El estado final será: ((0,0),(0,0), (8,1))
2. Operadores: hay un único operador Mover(X,Y): mueve un disco del palo X al palo Y
3. Heurística admisible: número de discos que faltan por colocar en su sitio + número de discos mal colocados en palo 3.

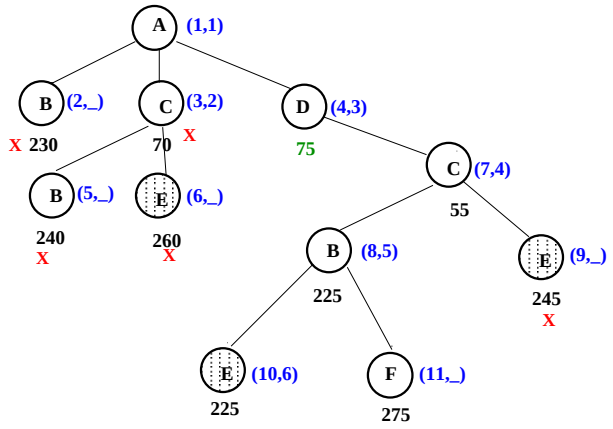


Figura 4: Árbol de búsqueda generado por el algoritmo A*