Inteligencia Artificial I **(14–Abril–2000)**

Ejercicio 1 [2.5 puntos]

El orden de los operadores en un problema de búsqueda puede afectar a la resolución de dicho problema con los algoritmos estudiados en la asignatura. Responder razonadamente a las siguientes cuestiones:

1. Es posible encontrar un problema de búsqueda con ramificación 3 y solución minimal a profundidad 3 tal que el orden de los operadores haga que, al utilizar el algoritmo de búsqueda en anchura, el número de nodos analizados sea inferior a 9.

2. En un problema de búsqueda con ramificación r y solución minimal a profundidad p ($p > 0$), ¿cúal es el número mínimo de nodos que tiene que analizar el algoritmo de búsqueda en anchura hasta llegar a la solución?.

3. Dado cualquier problema de búsqueda con ramificación r y solución minimal a profundidad p ($p > 0$), ¿es posible ordenar sus operadores de manera que el algoritmo de búsqueda en anchura sólo tenga que analizar la cantidad mínima de nodos?.

4. Es posible encontrar un problema de búsqueda con ramificación 3 y solución minimal a profundidad 3 tal que el orden de los operadores haga que, al utilizar el algoritmo de búsqueda en profundidad, el número de nodos analizados sea inferior a 9.

5. En un problema de búsqueda con ramificación r y solución minimal a profundidad p ($p > 0$), ¿cúal es el número mínimo de nodos que tiene que analizar el algoritmo de búsqueda en profundidad hasta llegar a la solución?.

6. Dado cualquier problema de búsqueda con ramificación r y solución minimal a profundidad p ($p > 0$), ¿es posible ordenar sus operadores de manera que el algoritmo de búsqueda en profundidad sólo tenga que analizar la cantidad mínima de nodos?.

7. Es posible que el orden de los operadores afecte al número de nodos analizados en el algoritmo primero el mejor.

Ejercicio 2 [2.5 puntos]

Se considera la siguiente definición del procedimiento de búsqueda A*

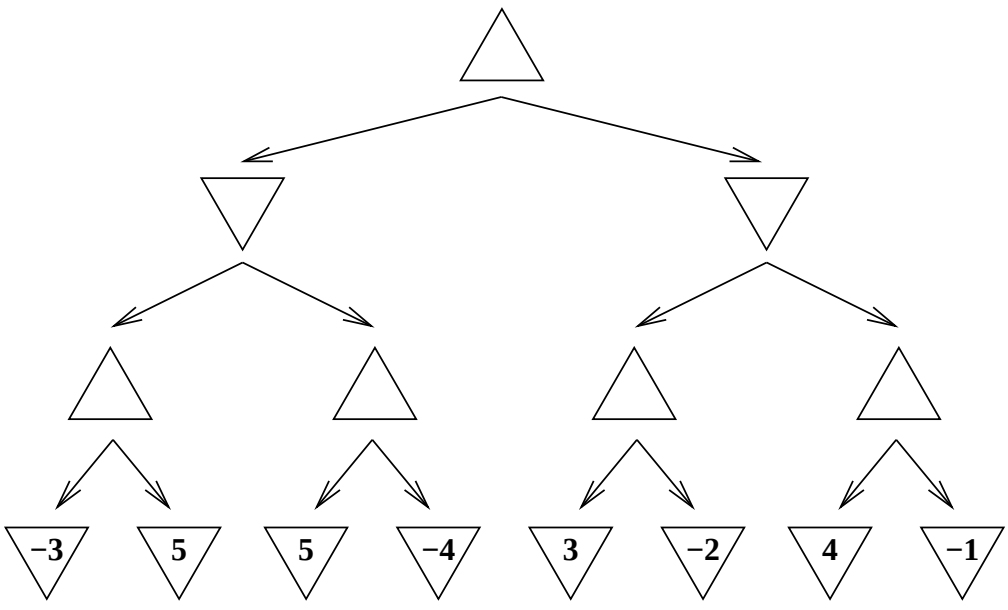
Apellidos: .....
Nombre: .....

```
(defun busqueda-a-estrella-1 ()
  (let ((abiertos (list (crea-nodo-hc
                         :estado *estado-inicial*
                         :camino nil
                         :coste-camino 0
                         :coste-mas-heuristica
                         (heuristica *estado-inicial*))))
        (cerrados nil)
        (actual nil)
        (sucesores nil))
    (loop until (null abiertos) do
      (setf actual   (first abiertos)
            cerrados (cons actual cerrados)
            abiertos (rest abiertos))
      (cond ((es-estado-final actual)
             (return actual))
            (t (setf sucesores (sucesores actual)
                     abiertos (append abiertos sucesores)))))))
```

Explicar si la definición anterior es correcta o incorrecta y, en el caso de ser incorrecta, explicar cuales son los errores y cómo corregirlos para que la definición sea correcta.

Ejercicio 3 [2.5 puntos]
Realizar los siguientes apartados:

1. Aplicar la estrategia alfa-beta con cotas iniciales -5 (α) y 5 (β) al siguiente árbol:



2. En qué condiciones se puede realizar una poda en el primer nivel del árbol, cuando las cotas iniciales son números reales M (α) y N (β), con $M < N$.

3. Aplicar la estrategia alfa-beta con cotas iniciales $-\infty$ (α) y $+\infty$ (β) al árbol del apartado 1.

4. En qué condiciones se puede realizar una poda en el primer nivel del árbol, cuando las cotas iniciales son $-\infty$ (α) y $+\infty$ (β).

Ejercicio 4 [2.5 puntos]
Se considera el siguiente programa

```
long(nil,0).
long(cons(X,A),s(N)) :-
   long(A,N).

conc(nil,A,A).
conc(cons(X,A),B,cons(X,C)) :-
   conc(A,B,C).
```

Construir el árbol de resolución SLD correspondiente a dicho programa y a la pregunta

```
?- conc(cons(a,nil),cons(b,nil),A), long(A,N).
```

indicando las respuestas obtenidas.
