PROGRAMACIÓN Y ESTRUCTURAS DE DATOS AVANZADAS

Septiembre 2014 (Original)

Normas de valoración del examen:

- La nota del examen representa el 80% de la valoración final de la asignatura (el 20% restante corresponde a las prácticas).
- Cada cuestión contestada correctamente vale 1 punto.
- Cada cuestión contestada incorrectamente baja la nota en 0.3 puntos.
- Debe obtenerse un mínimo de 3 puntos en las cuestiones para que el problema sea valorado (con 3 cuestiones correctas y alguna incorrecta el examen está suspenso).
- La nota total del examen debe ser al menos de 4.5 para aprobar.
- Las cuestiones se responden en una hoja de lectura óptica.

SOLUCION TEST:

Tipo A: 1A 2A 3D 4A 5C 6B
Tipo B: 1A 2A 3B 4C 5D 6A

Problema (4 puntos).

Desarrollar un programa que compruebe si es posible que un caballo de ajedrez, mediante una secuencia de sus movimientos permitidos, recorra todas las casillas de un tablero NxN a partir de una determinada casilla dada como entrada y sin repetir ninguna casilla.

La resolución del problema debe incluir, por este orden:

1. Elección del esquema más apropiado, el esquema general y explicación de su aplicación al problema (0,5 puntos)
2. Descripción de las estructuras de datos necesarias (0,5 puntos solo si el punto 1 es correcto)
3. Algoritmo completo a partir del refinamiento del esquema general (2,5 puntos solo si el punto 1 es correcto)
4. Estudio del coste del algoritmo desarrollado (0,5 puntos solo si el punto 1 es correcto)

SOLUCIÓN

El esquema correcto es el de vuelta atrás. No hay heurísticas ni criterios de búsqueda y el problema no puede descomponerse en subproblemas de su misma naturaleza, por lo que se utiliza la backtraking para la exploración del árbol de búsqueda.

Las estructuras de datos son el tablero de NxN que contiene en cada casilla valores que indiquen que el caballo no ha pasado (cero) o que ha pasado (m) donde m es un natural que

indica que en el movimiento *m-ésimo*, el caballo llega a dicha casilla. El nodo contaría con dicho tablero, con el número de caballos puestos en el tablero hasta el momento, y con la posición actual del caballo (el último movimiento efectuado)

El refinamiento del esquema precisa detallar cómo generar las compleciones y cómo realizar la recursión por el árbol implícito.

Las compleciones son las siguientes

```
Fun compleciones(ensayo)lista ← lista_ vacía

(i,j) → nodo.ultima_posición
n ← nodo.numero_movimientos
ultimo_tablero ← ensayo tablero

para i ← -2 hasta 2 hacer
     para j ← -2 hasta 2 hacer

          si |i|+|j|==3 AND nodo.tablero[i,j] == 0 entonces hacer

               nuevoTablero ← ultimoTablero
               nuevoTablero[i,j] ← n + 1
               nuevaPosicion ← (i,j)
               nuevoNodo ←  < nuevoTablero, nuevaPosición, n+1>
               lista.add(nuevoNodo)

          fsi
     fpara
fpara
dev lista
ffun
```

El algoritmo principal queda:

```
Fun saltoCaballo (nodo)

si solucion (nodo)       /* el numero de casillas libre es cero */
entonces escribe(nodo)    /* hemos terminado */
sino hacer

     lista ← compleciones (nodo) /* lista de los posibles movimiento
                                    válidos */
     mientras  ¬ lista.vacia() hacer
          w ← lista.primerElemento()
          saltoCaballo(w)
          lista ← lista.resto()
     fmientras
fsi
ffun
```

COSTE

Sin tener en cuenta las reglas de colocación del caballo, n^2 casillas pueden numerarse de $(n^2)!$ Maneras posibles, sin embargo sabemos que el numero máximo de ramificaciones del árbol es 8 por ser éstas las alternativas de movimiento de un caballo de ajedrez.

Considerando esto, el tamaño del árbol puede acotarse en $8^{n\hat{}2}$ pero se puede precisar que no hay 8 ramas en todos los casos. De cada n^2 casillas, solo desde n^2-8(n-2) es posible realizar 8 movimientos, y además los últimos movimientos no tendrán prácticamente ninguna alternativa.