INSTITUTO POLITECNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

Tarea 3. Archivo de exposición

(Recorrido del caballo en ajedrez)

ALUMNO: Colín Ramiro Joel

GRUPO: 3CM3

PROFESORA: Sánchez García Luz María

MATERIA: Análisis y Diseño de Algoritmos

Descripción de la estrategia de diseño

Definición del problema

Se trata de colocar inicialmente el caballo en una casilla inicial cualquiera (Xo, Yo), de un tablero de ajedrez de tamaño (n x n) y moverlo hasta haber pisado todas las casillas sin duplicar ninguna.

Solución

Para su solución se utilizará un algoritmo recursivo de vuelta atrás (backtracking) basado en ir colocando, según un orden, un caballo detrás de otro evitando las casillas ya ocupadas. El tablero lo guardaremos como una matriz de NxN (siendo N el número de filas y columnas) de tal forma que inicialmente tendrá valores cero (casillas todavía no visitadas). El programa recibe como único parámetro la posición de partida del caballo, donde se ubicará el primer caballo marcándolo con un 1 en el tablero para indicar esta circunstancia.

El algoritmo consistirá en un procedimiento “saltoCaballo (int x, int y)” que se invoca con la posición inicial del caballo y que, a su vez, llamará al procedimiento recursivo saltoCaballoR con cada uno de los ocho movimientos posibles.

Pseudocódigo

```
CONST TAMMAX = ... (* dimension maxima del tablero *)
TYPE tablero = ARRAY[1..TAMMAX],[1..TAMMAX] OF INT
```

```
Algoritmo Caballo( VAR t:tablero; n:INT; x,y:INT):BOOLEAN

BEGIN
 InicTablero(t,n)    (* inicializa las casillas del tablero a 0 *)
 FOR i ← 1 TO n*n DO
       t[x,y]← i
       IF NOT NuevoMov(t,n,x,y) AND (i<n*n-1) THEN
             RETURN FALSE
       END
 END
RETURN TRUE    (* hemos recorrido las n*n casillas *)
END Caballo
```

```
algoritmo NuevoMov(VAR t:tablero; n:INT; VAR x,y:INT):BOOLEAN
(*Esta función es la que va a ir calculando la nueva casilla a la que salta el caballo
siguiendo la indicación del enunciado, devolviendo FALSE si no puede Moverse*)

INT accesibles,minaccesibles,i,solx,soly,nuevax,nuevay
BEGIN
  minaccesibles ← 9
  solx ← x
  soly ← y
  FOR i:=1 TO 8 DO
          IF Salto(t,n,i,x,y,nuevax,nuevay) THEN
                  accesibles:=Cuenta(t,n,nuevax,nuevay)
                  IF (accesibles>0) AND (accesibles<minaccesibles) THEN
                        minaccesibles ← accesibles
                        solx ← nuevax  soly ← nuevay
                  END
          END
  END
X ← solx
 y ← soly
RETURN (minaccesibles<9)
END NuevoMov
```

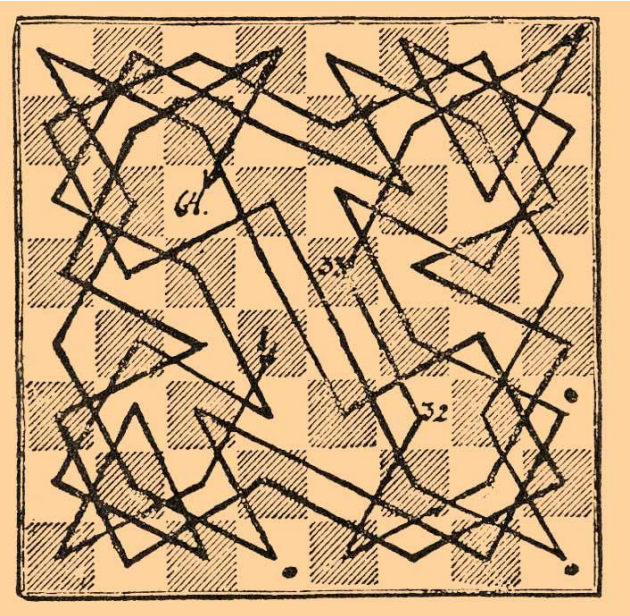
```
Algoritmo Salto(VAR t:tablero; n,i,x,y: INT;VAR nx,ny:INT ) :BOOL
(* i indica el numero del movimiento, (x,y) es la casilla actual, y (nx,ny) es la casilla a
donde salta. *)
BEGIN
        CASE i OF
                1: nx ← x-2     ny ← y+1
                2: nx ← x-1     ny ← y+2
                3: nx ← x+1     ny ← y+2
                4: nx ← x+2     ny ← y+1
                5: nx ← x+2     ny ← y-1
                6: nx ← x+1     ny ← y-2
                7: nx ← x-1     ny ← y-2
                8: nx ← x-2     ny ← y-1
END
RETURN((1<=nx) AND (nx<=n) AND (1<=ny) AND (ny<=n) AND (t[nx,ny]=0))
END Salto
```

Complejidad temporal

- $\boldsymbol{T(n) = T(2n + 3) + 5}$
- $\boldsymbol{O(n)}$

Ejemplo de funcionamiento

En la siguiente imagen, podemos observar un tablero de ajedrez tradicional, de 8x8 casillas, en el cual está trazado el recorrido del caballo, aplicando este algoritmo. El cual no repite ninguna casilla



Conclusiones

Podemos concluir que la salida del programa anterior nos permite inferir un patrón general para las soluciones del problema

- Para n = 4, el problema no tiene solución.
- Para n > 4, n par, el problema tiene solución para cualquier casilla inicial.
- Para n < 4, n impar, el problema tiene solución para aquellas casillas iniciales (Xo,Yo) que verifiquen que X0 + Yo sea par, es decir si la casilla inicial es blanca

Se observa que el algoritmo implementado, no ha encontrado solución en todas estas situaciones. Por ejemplo, para n = 5, Xo = 5, Yo = 3 el algoritmo dice que no la hay. Sin embargo, si la encuentra para n = 5, Xo = 1, Yo = 3, para n = 5, Xo = 3, Yo = 1 y para n = 5, Xo = 3, Yo = 5, los cuales son casos simétricos. De existir una solución para alguno de ellos, por simetría se obtiene para los otros. ¿Porqué no la encuentra este algoritmo?

La respuesta radica en cómo se busca la siguiente casilla a donde saltar. Por la forma en la que funciona el algoritmo, siempre se prueban las ocho casillas posibles en el sentido de las agujas del reloj, siguiendo el orden y la pauta de la función Salto. Esto hace que el algoritmo no sea simétrico. En resumen, este algoritmo si es considerado voraz el cuál no funciona para todos los casos.

Bibliografía

- http://www.lcc.uma.es/~av/Libro/CAP4.pdf
- https://es.slideshare.net/luzenith_g/algoritmos-voraces-greedy

- http://jorgep.blogspot.com/2010/11/el-problema-del-recorrido-del-caballo.html#:~:text=La%20peregrinaci%C3%B3n%20del%20caballo%20de,de%20una%20%22peregrinaci%C3%B3n%20cerrada%22.