INSTITUTO POLITECNICO NACIONAL

UNIDAD PROFESIONAL INTERDISCIPLINARIA DE INGENIERIA CAMPUS ZACATECAS

Alumno:

Mariel López Beltrán

Docente:

Roberto Oswaldo Cruz Leija

Grupo:

3CM1

Asignatura:

Análisis de algoritmos

Tarea:

Caballo Ávido

Fecha de entrega:

28/10/2019

Introducción:

Actualmente el problema del caballo en uno de los algoritmos que se podría definir en simples palabras como complicado, esto se debe a que entre los problemas matemáticos inspirados en el ajedrez, uno de los más interesantes es el problema de la marcha del caballo. Consiste en recorrer las 64 casillas del tablero con un caballo, en 64 movimientos y sin pasar dos veces por la misma casilla, sin embargo, se han presentado diversas maneras de resolverlo una de ellas es mediante la utilización de una fórmula de Euler que nos permite llevar a cabo de una manera satisfactoria la solución de dicho problema en la programación

En el presente trabajo se llevara a cabo la programación del Caballo en java con el único objetivo de comprender su funcionamiento mediante el uso de la programación.

Marco Teórico:

La peregrinación del caballo de ajedrez consiste en su paseo por todas las casillas del tablero sin pasar dos veces por la misma, utilizando sus movimientos: dos casillas horizontales y una vertical o a la inversa. Cuando desde la última casilla podamos pasar a la primera se trata de una "peregrinación cerrada".

1	48	31	50	33	16	63	18
30	51	46	3	62	19	14	35
47	2	49	32	15	34	17	64
52	29	4	45	20	61	36	13
5	44	25	56	9	40	21	60
28	53	8	41	24	57	12	37
43	6	55	26	39	10	59	22
54	27	42	7	58	23	38	11

Para resolver este problema se puede utilizar la técnica de backtracking con un algoritmo recursivo relativamente simple: marcar la posición solicitada, hacer una lista de las siguientes posiciones vacías a las que se puede saltar desde la posición marcar, y tomar la primera de ellas, marcándola entonces en forma recursiva. Si en alguno de los pasos recursivos no se alcanza una solución, la casilla se desmarca y se pasa a la siguiente que estaba en la lista. Si se termina la lista y no se llega a una solución, entonces se devuelve a la casilla anterior desde la que se hizo el llamado.

El trabajo más importante en relación a este problema, se atribuye al genial Leonhard Euler, que destacó por sus ingeniosas y fantásticas soluciones

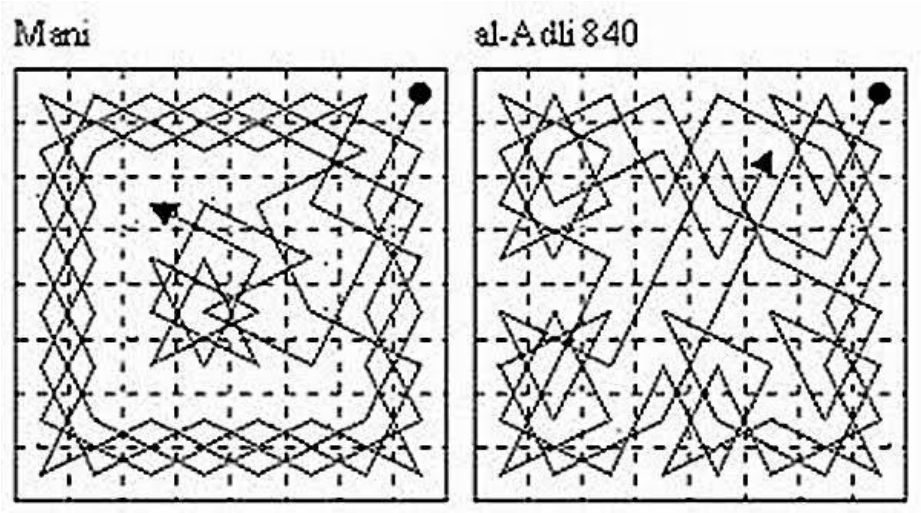
Restricciones:

- No salirse del tablero
- Deberá de tener 64 movimientos
- Movimiento en forma de L (respetando el movimiento que tiene en el ajedrez)

Consideraciones:

- Empezar y terminar en la misma casilla (circuito cerrado). Más complicado.
- Empezar en una casilla y terminar en otra (circuito abierto). Más «sencillo»

Una de las soluciones que dio este genio matemático asombró por su belleza. Euler construyó un cuadrado mágico donde las filas y las columnas sumaban 260. El caballo se desplaza desde la casilla 1 hasta la 64 en orden numérico.



Pruebas de ejecución:

Posición (1,1):

```
1 | 60 | 39 | 34 | 31 | 18 | 9 | 64 |
38 | 35 | 32 | 61 | 10 | 63 | 30 | 17 |
59 | 2 | 37 | 40 | 33 | 28 | 19 | 8 |
36 | 49 | 42 | 27 | 62 | 11 | 16 | 29 |
43 | 58 | 3 | 50 | 41 | 24 | 7 | 20 |
48 | 51 | 46 | 55 | 26 | 21 | 12 | 15 |
57 | 44 | 53 | 4 | 23 | 14 | 25 | 6 |
52 | 47 | 56 | 45 | 54 | 5 | 22 | 13 |
BUILD SUCCESSFUL (total time: 0 seconds)
```

Posición (8,5):

```
46 | 61 | 28 | 51 | 12 | 63 | 26 | 5 |
29 | 50 | 45 | 62 | 27 | 6 | 11 | 64 |
60 | 47 | 54 | 23 | 52 | 13 | 4 | 25 |
49 | 30 | 59 | 44 | 55 | 24 | 7 | 10 |
58 | 43 | 48 | 53 | 22 | 9 | 14 | 3 |
37 | 40 | 31 | 56 | 33 | 16 | 19 | 8 |
42 | 57 | 38 | 35 | 18 | 21 | 2 | 15 |
39 | 36 | 41 | 32 | 1 | 34 | 17 | 20 |
BUILD SUCCESSFUL (total time: 1 second)
```

Posición (4,4):

```
33 | 58 | 39 | 62 | 31 | 16 | 7 | 64 |
40 | 61 | 32 | 27 | 8 | 63 | 30 | 15 |
57 | 34 | 59 | 38 | 43 | 28 | 17 | 6 |
60 | 41 | 44 | 1 | 26 | 9 | 14 | 29 |
45 | 56 | 35 | 42 | 37 | 22 | 5 | 18 |
50 | 53 | 48 | 25 | 2 | 19 | 10 | 13 |
55 | 46 | 51 | 36 | 21 | 12 | 23 | 4 |
52 | 49 | 54 | 47 | 24 | 3 | 20 | 11 |
BUILD SUCCESSFUL (total time: 5 minutes 33 seconds)
```

Posición (4,3):

```
38 | 53 | 30 | 57 | 14 | 63 | 28 | 7 |
31 | 58 | 39 | 62 | 29 | 8 | 13 | 64 |
52 | 37 | 54 | 25 | 56 | 15 | 6 | 27 |
59 | 32 | 1 | 40 | 61 | 26 | 9 | 12 |
36 | 51 | 60 | 55 | 24 | 11 | 16 | 5 |
45 | 48 | 33 | 2 | 41 | 18 | 21 | 10 |
50 | 35 | 46 | 43 | 20 | 23 | 4 | 17 |
47 | 44 | 49 | 34 | 3 | 42 | 19 | 22 |
BUILD SUCCESSFUL (total time: 0 seconds)
```

Posición (1,4):

```
56 | 61 | 28 | 1 | 16 | 63 | 26 | 7 |
49 | 32 | 57 | 62 | 27 | 8 | 15 | 64 |
60 | 55 | 50 | 29 | 2 | 17 | 6 | 25 |
33 | 48 | 31 | 58 | 51 | 24 | 9 | 14 |
54 | 59 | 52 | 23 | 30 | 3 | 18 | 5 |
47 | 34 | 41 | 38 | 43 | 20 | 13 | 10 |
40 | 53 | 36 | 45 | 22 | 11 | 4 | 19 |
35 | 46 | 39 | 42 | 37 | 44 | 21 | 12 |
BUILD SUCCESSFUL (total time: 5 seconds)
```

Conclusiones:

En el presente reporte se llevó a cabo la programación del problema del caballo, sin embargo, fue un poco complicado realizarlo principalmente cuando no encontraba una solución o simplemente no podía resolverlo dependiendo de los casos que se podían realizar, además de que el algoritmo muestra ciertas cuestiones que son curiosas en mi opinión.

Un ejemplo de ello es al momento de inicializar la posición del caballo en (1,1) puede llevar a cabo la solución es milisegundos y probándolo con otras coordenadas similares me pude dar cuenta que puede llevar a cabo una solución satisfactoria cuando la posición del caballo se encuentra en las orillas, mientras que si se agrega una posición en el medio del tablero tarda hasta 4 minutos para poder llevar a cabo una solución satisfactoria.

Aunque en determinados momentos no lograba llevar a cabo una determinado solución como el caso de (2,3) ya que obtuvo un tiempo de 30 minutos y no demostró que pueda haber una solución