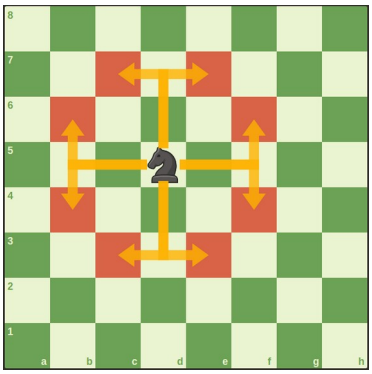
Laboratorio 2

Computación 1 - 2023

Descripción del problema



En el ajedrez, el caballo es una pieza que puede mover saltando dos casillas en un sentido (vertical, u horizontal) y una casilla en el otro. Debido a esto, como se muestra en la figura, en cada posición el caballo tiene como máximo 8 casillas posibles a las que mover (algunas pueden estar ocupadas o fuera del tablero).

En esta tarea, se representará el tablero como una matriz cuadrada de 8x8, donde un 1 en la posición `(i,j)` representa que la casilla está ocupada, y un 0 representa que está vacía. Definiremos además un "camino" de largo *n* dentro del tablero como una matriz `C` de tamaño nx2 tal que para cada índice `i`, `C(i,1)` es el índice de fila de una casilla por la que pasa dicho camino y `C(i,2)` es su respectivo índice de columna.

Desarrollar en Octave las siguientes funciones:

Funciones auxiliares (no recursivas)

- `posEsValida`, que recibe (en el orden de la lista):
  - **(1)** Un entero `i`.
  - **(2)** Un entero `j`.
  - **(3)** Una matriz `tablero` de 8x8 que puede tener casillas libres y ocupadas.

  La función `posEsValida` devuelve 1 si la casilla `(i,j)` está dentro de los límites del `tablero` y está vacía, devolviendo 0 en caso contrario.
- `esMovimientoCaballo`, que recibe (en el orden de la lista):
  - **(1)** Un entero `i_0`.
  - **(2)** Un entero `j_0`.
  - **(3)** Un entero `i_f`.
  - **(4)** Un entero `j_f`.

  La función `esMovimientoCaballo` devuelve 1 si es posible moverse desde `(i_0,j_0)` a `(i_f,j_f)` con un movimiento de caballo, devolviendo 0 en caso contrario.

Funciones recursivas

Desarrollar en Octave las siguientes funciones **recursivas**:

- `esCamino`, que recibe:

- **(1)** Una matriz de dos columnas que representa un camino `C`
- **(2)** Una matriz `tablero` de 8x8 que puede tener casillas libres y ocupadas.

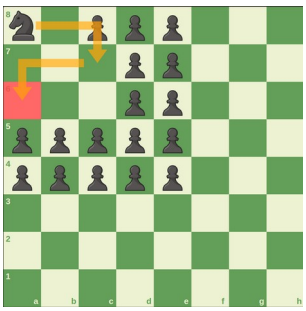
La función `esCamino` devuelve 1 si la matriz `C` es un camino válido en `tablero` (todas las casillas son posiciones válidas y es posible ir de una casilla a la siguiente con un movimiento de caballo), devolviendo 0 en caso contrario.

- `caminoPasaPor`, que recibe:
  - **(1)** Una matriz de dos columnas que representa un camino `C`
  - **(2)** Un entero `i`
  - **(3)** Un entero `j`

  En caso de que la casilla `(i,j)` se encuentre en `C`, la función devuelve el índice de la fila de `C` en la que se encuentra `(i,j)`. Si `(i,j)` no está en `C`, la función devuelve -1. Puede asumir que en `C` no hay casillas repetidas.
- `encontrarCamino`, que recibe (en el orden de la lista):
  - **(1)** Un entero `i_0`
  - **(2)** Un entero `j_0`
  - **(3)** Un entero `i_f`
  - **(4)** Un entero `j_f`
  - **(5)** Una matriz `tablero` de 8x8 que puede tener casillas libres y ocupadas
  - **(6)** Un entero `maxLargo` representando el largo máximo del camino

  La función devuelve una matriz `camino` de 2 columnas y hasta `maxLargo` filas que representa un camino posible para el caballo de ajedrez desde la casilla inicial `tablero(i_0,j_0)` a la final `tablero(i_f,j_f)`. En caso de que no exista un camino posible de largo menor o igual a `maxLargo` la función retorna la matriz vacía.

Consejo para encontrarCamino:



En la situación de la figura existe una sola forma de llegar desde la casilla inicial a la casilla final que corresponde con el siguiente camino:

```
camino = [1 1;
          2 3;
          3 1]
```

Sin embargo, si se permite que existan casillas repetidas en el camino, se forman ciclos que potencialmente son de largo infinito. En el ejemplo anterior, el caballo siempre podría retornar a la casilla inicial varias veces luego de avanzar un paso, produciendo el siguiente camino:

```
camino = [1 1; 2 3; 1 1; 2 3; 1 1; 2 3; 3 1]
```

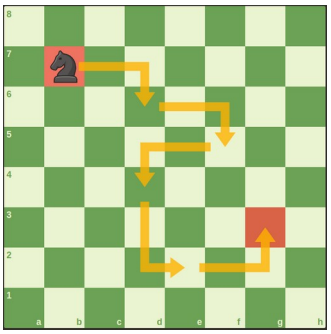
Por esta razón, conviene evitar que el caballo vuelva a una casilla ya incluida en el camino.

Ejemplos

A continuación se presentan una serie de ejemplos relativos a las funciones que se pide implementar.

Ejemplo 1

En el tablero de la figura existe al menos un camino posible entre el caballo negro y la casilla final (casilla roja).



En este caso el tablero se representa de la siguiente forma

```
tablero = [0 0 0 0 0 0 0 0;
           0 0 0 0 0 0 0 0;
           0 0 0 0 0 0 0 0;
           0 0 0 0 0 0 0 0;
           0 0 0 0 0 0 0 0;
           0 0 0 0 0 0 0 0;
           0 0 0 0 0 0 0 0;
           0 0 0 0 0 0 0 0]
```

La casilla inicial es la (2,2) y la final es la (6,7). Con estos datos de entrada `encontrarCamino` puede devolver cualquier camino válido entre (2,2) y (6,7) de largo menor o igual al parámentro (6). En el caso de la imagen el camino se representa por la matriz

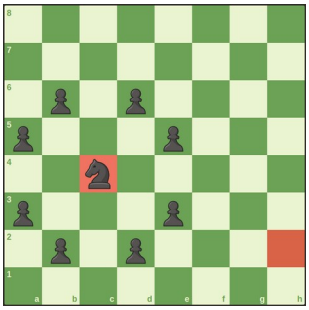
```
camino = [2 2; 3 4; 4 6; 5 4; 7 5; 6 7]
```

Para esta matriz `camino` y esta matriz `tablero` `esCamino(camino,tablero)` devuelve 1.

Para esta esta matriz `tablero`, `esCamino([2,2;3,3;4,6],tablero)` devuelve 0 porque el camino contiene movimientos de caballo inválidos.

Ejemplo 2

En este otro ejemplo no existe un camino posible entre las casillas inicial y final, ya que todas las casillas a las que el caballo puede mover están ocupadas.



En este caso el tablero se representa de la siguiente forma

```
tablero = [0 0 0 0 0 0 0 0;
           0 0 0 0 0 0 0 0;
           0 1 0 1 0 0 0 0;
           1 0 0 0 1 0 0 0;
           0 0 0 0 0 0 0 0;
           1 0 0 0 1 0 0 0;
           0 1 0 1 0 0 0 0;
           0 0 0 0 0 0 0 0]
```

La casilla inicial es la (5,3) y la final es la (7,8). Con estos datos de entrada `encontrarCamino` debe devolver `[]`.

Para esta matriz `tablero`, `esCamino([5,3;6,5;5,7;7,8],tablero)` devuelve 0 porque la casilla (6,5) está ocupada.

Ejemplo 3

Finalmente dado el camino:

```
camino = [2 2;
          3 4;
          4 6;
          5 4;
          7 5;
          6 7]
```

`caminoPasaPor(camino, 4, 6)` devuelve 3 (ya que la fila 3 es [4 6])

`caminoPasaPor(camino, 8, 8)` devuelve -1 (ya que la casilla [8 8] no está en `camino`)

Formato de entrega

Entregar los 5 archivos .m proporcionados por separado (no en un .zip o archivo comprimido) modificando únicamente el contenido de las funciones (no escribir código antes de function o luego del último end y no modificar las líneas de código proporcionadas). Esto es imprescindible para el proceso de corrección y no respetarlo puede derivar en pérdida de puntos. No se permite definir otras funciones además de las 5 mencionadas. Está permitido invocar cualquiera de las 5 funciones dentro de otra. **Las funciones no deben desplegar texto en pantalla ni recibir entrada directa del usuario**.

No está permitido utilizar facilidades de Octave que permitan resolver los problemas de forma trivial (funciones max, min, sort, etc.). En esta tarea **NO SE PERMITE USAR FOR NI WHILE**.

Los valores válidos de entrada son matrices de 8x8 con 1s y 0s en el caso de los tableros, y matrices de nx2 en el caso de los caminos. No se evaluará el comportamiento de la entrega ante entradas inválidas (matrices de otros tamaños, matrices tablero vacías o con valores distintos de 0/1, etc.).

La entrega debe producir la salida correcta al menos para los casos que figuran en la letra, pero podrá ser evaluada con casos de prueba adicionales en la etapa de corrección.

La ejecución se realizará desde la línea de comandos (sin interfaz gráfica).

En esta tarea, como en todos los problemas de este curso, se valorará, además de la lógica correcta, la utilización de un buen estilo de programación de acuerdo a los criterios impartidos en el curso. De esta manera, se hará énfasis en buenas prácticas de programación que lleven a un código legible y bien documentado, tales como:

•indentación adecuada

•utilización correcta y apropiada de las estructuras de control

•código claro y legible

•algoritmos razonablemente eficientes

•utilización de comentarios que documenten y complementen el código

•nombres mnemotécnicos para variables, constantes, etc.