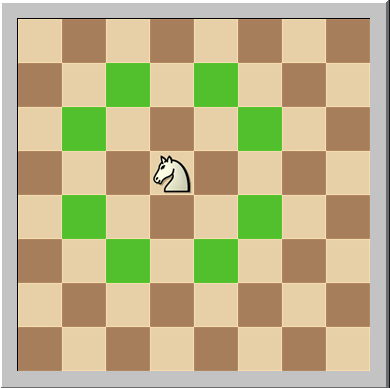
David Marquez Mínguez

47319570Z

Ejercicios TEMA 5

ALGORITMIA Y COMPLEJIDAD

# EJERCICIO 4

En este problema, disponemos de un tablero M de un tamaño determinado(filas x columnas). En una casilla determinada, ponemos un caballo de ajedrez(posX, posY). El objetivo es determinar si es posible recorrer el tablero con el caballo sabiendo que cada casilla solo puede recorrerse una sola vez.

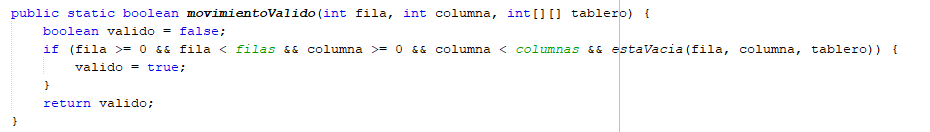
Por otro lado, el caballo puede terminar en cualquier posición del tablero. Antes de comenzar debemos tener en cuenta algunas condiciones. En primer lugar, el caballo no puede salirse del tablero, y, en segundo lugar, debemos tener en cuenta el movimiento de un caballo normal de ajedrez. Si seguimos con el ejemplo de la imagen las casillas marcadas en verde son los movimientos posibles que habría que estudiar.

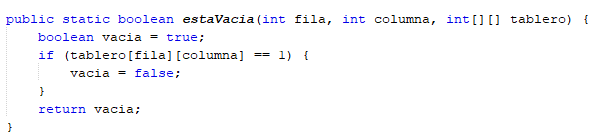
El algoritmo estudiara todas las posibilidades para cada posición, una vez conocidas las posiciones posibles, estudiara cada una de ellas de manera recursiva.

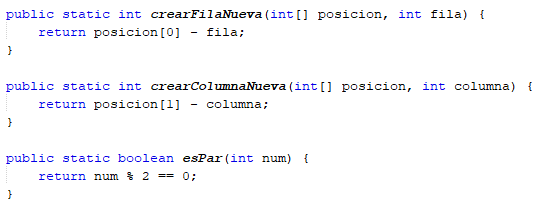
El algoritmo consistirá en lo siguiente, se dispondrá de un tablero, donde cada posición representará cas una de las celdas del mismo. Cada celda estará compuesta de un 0 o un 1, lo que representará si ya hemos pisado esa celda o no. Por otro lado, se disponen de dos bucles, un bucle exterior, y los bucles interiores. El bucle exterior se encarga de recorrer las filas, mientras que los interiores se encargan de recorrer las colunas. La idea es que cuando visite las filas 2 y -2 tenga que visitar las columnas 1 y -1. Y cuando visite las filas 1 y -1 tenga que visitar las columnas 2 y -2.

El algoritmo acaba cuando no existe solución posible para el problema, o se encuentra la solución. Una vez hecho esto y para cuando el algoritmo termine tendremos almacenado en nuestra variable recorrido todo el recorrido que hemos ido haciendo dentro del tablero siguiendo los movimientos del caballo del ajedrez de tal forma que todas las casillas del tablero han sido visitadas, pero solo una vez.

A continuación, se va a mostrar el algoritmo implementado.

El resto de funciones que hay en el programa, son funciones auxiliares que ayudan a que este se ejecute de manera correcta y nos proporcione la salida deseada. Las funciones auxiliares implementadas mas importantes son las siguientes:





A continuación, voy a ejecutar el programa. La salida del caballo será la posición {1, 1} del tablero. Lo que se va a mostrar es la lista del recorrido que se ha hecho hasta legar a la solución. Para poder ver mejor la solución he creado una función que determina la posición del tablero a partir de sus coordenadas.



He dividido la solución en varias partes puesto que sino no me cabía en una sola línea. Como se puede observar, se muestran todas las posiciones del tablero por las cuales el caballo ha tenido que pasar hasta llegar a la solución.

# EJERCICIO 6

En este problema disponemos de una matriz, a la que llamaremos matriz de sustitución. En dicha matriz tendremos una serie de letras tanto en las columnas como en las filas que nos indicaran los caracteres de la cadena a sustituir. La matriz es la siguiente:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | a | b | C | d |
| a | b | b | a | d |
| b | c | a | d | a |
| c | b | a | c | c |
| d | d | c | d | b |

A continuación, para entender mejor el problema se va a poner un ejemplo. Imaginemos que tenemos la siguiente secuencia de letras: “acabada”. El objetivo es reducir todas estas letras a una única siguiendo un proceso como el que sigue.

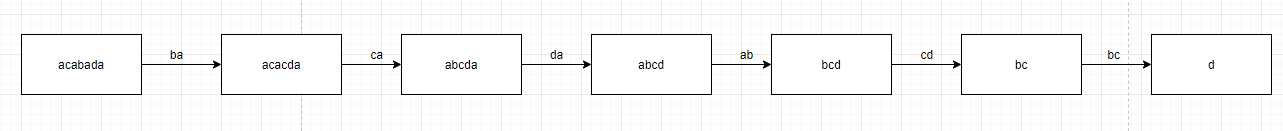
En primer lugar, disponemos de la secuencia acabada, y buscamos una secuencia de dos palabras para sustituir, por ejemplo, “ba”. Ahora buscamos en la tabla que letra se asigna a la combinación ba y la modificamos en nuestra secuencia. El resultado es la nueva palabra acacda.

Como todavía nuestra palabra no se ha reducido a una sola letra, seguimos iterando y seguimos el mismo proceso que antes. Buscamos nuevamente en nuestra nueva palabra una secuencia de caracteres a sustituir, en este caso, “ca”. Miramos en la tabla y sustituimos dichos caracteres por la letra correspondiente. Una vez realizado el proceso obtenemos la nueva palabra: abcda.

Todavía nuestra palabra no está compuesta por una única letra, por ello, seguimos iterando. Buscamos nuevamente una secuencia de la palabra que este en nuestra matriz, por ejemplo, “da”. Sustituimos en nuestra palabra la letra correspondiente. La nueva palabra es abcd.

Aún no hemos reducido la secuencia de letras a una única letra, por ello seguimos el mismo proceso que antes. Disponemos de la secuencia abcd, buscamos en el tablero una secuencia, en este caso “ab”, y modificamos nuestra secuencia a la secuencia correspondiente, bcd.

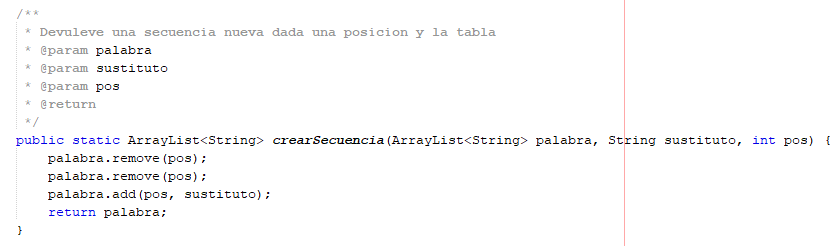
Seguimos iterando en este caso con la secuencia “cd”, y la sustituimos en nuestra secuencia por el carácter que nos marca el tablero, formando así la nueva secuencia bc.

Finalmente, las dos última letras las sustituimos en el tablero y obtenemos la secuencia reducida a una única letra, d.

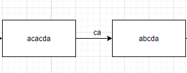
Obviamente, hay muchas más posibilidades de las que se han propuesto en el ejemplo. El objetivo es realizar un algoritmo que a partir de una cadena de texto como la mostrada en el ejemplo, sea capaz de encontrar la forma de realizar las sustituciones que permite reducir la cadena texto a un carácter final, si es posible.

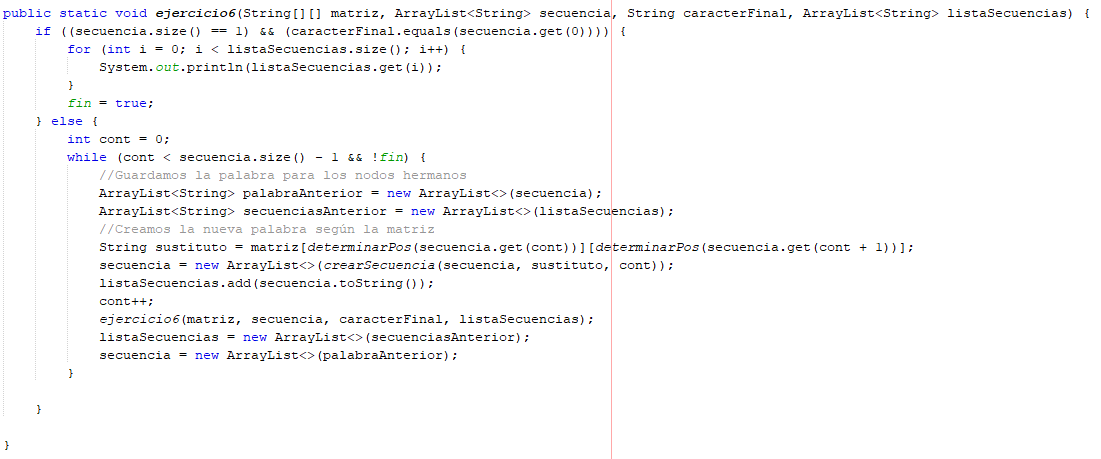
El procedimiento del algoritmo será el siguiente. Para cada palabra obtengo n-1 posibles sustituciones (siendo n el número de letras por las que está compuesta dicha palabra). Por ejemplo, para la palabra abcd puedo sustituir en la primera y segunda letra, en la segunda y tercera o en la tercera y cuarta para obtener bdc, add y abc como resultado de cada una de las sustituciones.

Una vez que ya conocemos el funcionamiento, voy a implementarlo en Java.

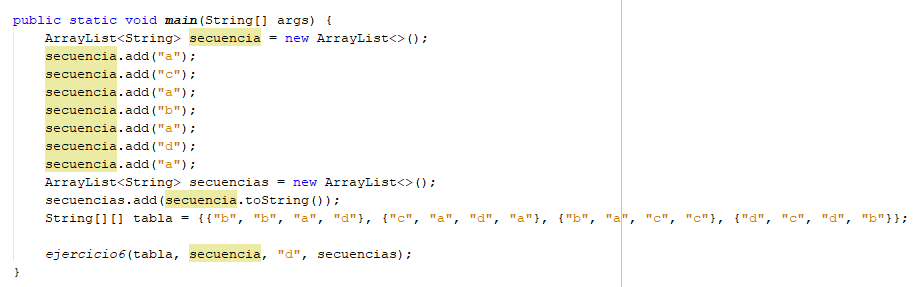
En primer lugar, creo una función que se encarga de modificar una secuencia dada a partir de una posición y de los valores de la tabla.

Esta función se encarga de transformar una determinada secuencia en otra. Dada una determinada posición y la tabla, devuelve otra secuencia. Del conjunto de nodos que hemos obtenido antes, esta función se encargaría de pasar de un nodo a otro, algo como esto.



Ahora que sabemos cómo transformar una secuencia en otra, solo debemos hacer las transformaciones necesarias para pasar de una determinada secuencia en una secuencia con un solo carácter. Para ello programo una función recursiva.

Esta función se encarga de calcular todas las secuencias posibles hasta llegar a la secuencia determinada que queremos, en este caso [“d”]. La función se ejecutará recursivamente hasta llegar al caso base, en el que la secuencia tenga solo un carácter y dicho carácter sea el que queremos.

A continuación, voy a ejecutar el programa con los mismos datos que en el ejemplo expuesto anteriormente para comprobar que el algoritmo funciona de manera correcta.

