PROGRAMACIÓN FUNCIONAL

Lights out

Programación Declarativa

Pedro Álvarez Piedehierro

INTRODUCCION AL PROYECTO

El juego Lights Out tiene muchas versiones distintas, aunque siempre basadas en el mismo principio.

Se trata de un juego individual. Normalmente se juega en un tablero cuadrado. En cada posición se encuentra una bombilla que puede estar apagada o encendida.

Cada vez que se marca una posición, las bombillas que ocupan esa posición y las casillas adyacentes cambian de estado (de encendida a apagada, o viceversa). Generalmente, el patrón de activación de las casillas adyacentes suele cambiar el estado de las casillas que se encuentran arriba, abajo, a la derecha y a la izquierda de la marcada.

Partiendo de una situación inicial de bombillas encendidas, el objetivo es ir seleccionado casillas hasta conseguir que se apaguen todas las bombillas.

El **objetivo de este proyecto** es escribir una función en **HASKELL** que, dada una configuración inicial de luces encendidas en un tablero cuadrado, devuelva una lista con las coordenadas de las casillas que hay que marcar para conseguir que todas las luces del tablero se apaguen.

Lights n encendidas → lmarcadas

N representa el número de filas y columnas del tablero cuadrado en el que se juega. Supondremos que es un número entero mayor o igual a 1.

Encendidas es una lista de tuplas (F, C), donde se indican las coordenadas que están inicialmente encendidas.

Lmarcadas será una lista de listas de tuplas (F, C) que indican las distintas posibilidades para apagar el tablero.

En cada lista de tuplas aparecerán las casillas del tablero que hay que ir marcando para conseguir que, finalmente, todas estén apagadas.

FUNCIONES DEL PROYECTO

Aquí podremos ver todas las funciones de los que se compone el proyecto, dando a conocer que hace cada una, y para que sirven los argumentos de cada función.

lights n encendidas → lmarcadas

Esta función devuelve una lista con todas las posibles combinaciones de casillas que hay que pulsar para apagar la combinación de luces encendidas pasadas por el parámetro "encendidas". Devuelve dentro de esta lista todas las posibilidades, sin permutaciones de éstas.

Argumentos:

- **N:** Entero que indica el lado de el tablero
- **Encendidas:** Lista de tuplas (F, C) que indica las casillas que están encendidas.

Devuelve:

Lista de tuplas (F, C) que indica que casillas hay que pulsar para apagar la combinación LEncendidas.

insertar lista dato → lista

Esta función añade un elemento a una lista y devuelve la nueva lista.

Argumentos:

- **lista:** Lista en la que insertar el elemento.
- **dato:** Elemento a insertar en la lista.

Devuelve:

Lista en la que se ha insertado el dato.

borrar lista dato → lista

Esta función borra un elemento de una lista y devuelve la nueva lista.

Argumentos:

- **lista:** Lista en la que borrar el elemento.
- **dato:** Elemento a borrar de la lista.

Devuelve:

Lista en la que se ha borrado el dato.

buscar lista dato -> resultado

Busca el dato “dato” en la lista “lista”. Devuelve Verdadero si está en la lista, y Falso si no lo está.

Argumentos:

- **dato:** Elemento a buscar en la lista.
- **lista:** Lista en la que buscar el dato.

Devuelve:

Valor Booleano de resultado.

iguales casillaA casillaB → resultado

Esta función nos indica si la tupla (F,C) pasada por el parámetro “casillaA” es igual a la tupla (F,C) pasada por el parametro “casillaB”. Devolverá Verdadero si son iguales, y Falso en caso contrario.

Argumentos:

- **casillaA:** Tupla (F,C) que se quiere comparar.
- **casillaB:** Tupla (F,C) que se quiere comparar.

Devuelve:

Valor Booleano de resultado.

casillaValida n casilla → resultado

Esta función nos indica si la tupla (F,C) pasada por el parámetro “casilla” esta dentro de un tablero de juego de lado “n”. Devolverá Verdadero si pertenece al tablero, y Falso en caso contrario.

Argumentos:

- **N:** Entero que indica el lado del tablero.
- **casilla:** Tupla (F,C) que se quiere comprobar si es válida.

Devuelve:

Valor Booleano de resultado.

modificarCasilla n lencendidas casilla → lencendidas

Esta función nos modifica el estado de la casilla (F,C) pasada por el parámetro "casilla", es decir, si estaba la casilla encendida en el tablero la apaga, y si estaba apagada la enciende. Para representar esto utilizamos una lista en la cual guardamos las casillas encendidas "lencendidas". Por tanto si esta en la lista la borramos, y si no esta la insertamos, devolviendo como resultado la lista "lencendidas" tras la modificación.

Argumentos:

- **N:** Entero que indica el lado del tablero.
- **lencendidas:** Lista de tuplas (F,C) que indica las casillas encendidas.
- **casilla:** Tupla (F,C) que se quiere modificar el estado.

Devuelve:

Lista de tuplas (F,C) que representa las casillas encendidas tras la modificación.

pulsarCasilla n lencendidas casilla → lencendidas

Esta función nos modifica el estado de la casilla (F,C) pasada por el parámetro "casilla" y también el estado de las casillas adyacentes según el patrón lights out. Devuelve como resultado la lista "lencendidas" tras la modificación.

Argumentos:

- **N:** Entero que indica el lado del tablero.
- **lencendidas:** Lista de tuplas (F,C) que indica las casillas encendidas.
- **casilla:** Tupla (F,C) que se quiere pulsar.

Devuelve:

Lista de tuplas (F,C) que representa las casillas encendidas tras la modificación.

probarCombinacion n lencendidas lsolucion → resultado

Esta función comprueba si al pulsar la combinación de casillas "lsolucion" se apagan todas las luces que se encuentran en "lencendidas". Devuelve resultado Verdadero si ha apagado todas las casillas, y Falso si la solucion no es válida.

Argumentos:

- **N:** Entero que indica el lado del tablero.
- **lencendidas:** Lista de tuplas (F,C) que indica las casillas encendidas.
- **lsolucion:** Lista de tuplas (F,C) que se quiere comprobar si es solución.

Devuelve:

Valor Booleano de resultado.

generarPosibles n → lposibles

Esta función genera todas las posibles combinaciones de tuplas (F,C) (casillas) que se pueden pulsar en un tablero de lado "n" para solucionarlo. Devuelve una lista la cual contiene listas de tuplas (F,C).

Argumentos:

- **N:** Entero que indica el lado del tablero.

Devuelve:

Lista de listas de tuplas (F,C) con todas las combinaciones posibles.

crearCasillas n → lcasillas

Esta función genera todas las casillas de un tablero representándolas como tuplas (F,C). Devuelve una lista con todas estas tuplas (F,C).

Argumentos:

- **N:** Entero que indica el lado del tablero.

Devuelve:

Lista de tuplas (F,C) con todas las casillas del tablero.

EJEMPLOS DE FUNCIONAMIENTO

En este apartado podemos ver 3 ejemplos de la ejecución del predicado principal ‘ligths’, hay un ejemplo de un tablero de 2x2, otro de 3x3 y otro de 4x4.

Como podemos ver en la ejecución del código, en el ejemplo de 4x4 nos da las 16 alternativas diferentes para resolver el juego.

```
*Main> lights 2 [(1,2),(2,1),(2,2)]
[[(2,2)]]
it :: [[(Int, Int)]]
(0.02 secs, 522000 bytes)
*Main> lights 3 [(1,2),(2,1),(2,2)]
[[(1,3),(2,2),(1,2),(2,1),(3,1)]]
it :: [[(Int, Int)]]
(0.36 secs, 19662876 bytes)
*Main> lights 4 [(2,1),(2,2),(3,1),(3,3),(3,4),(4,3),(4,4)]
[[(2,4),(3,4),(4,4),(1,4),(2,3),(4,3),(4,2),(3,1)],[(2,4),(3,4),(4,4),(4,3),(1,3),(3,2),(4,2
),(1,2),(2,1),(4,1)],[(2,4),(3,4),(1,4),(2,3),(3,3),(2,2),(2,1),(1,1)],[(2,4),(3,4),(3,3),(1
,3),(2,2),(3,2),(1,2),(3,1),(4,1),(1,1)],[(2,4),(4,4),(1,4),(3,3),(4,2),(1,2),(4,1),(1,1)],[
(2,4),(4,4),(2,3),(3,3),(1,3),(3,2),(4,2),(2,1),(3,1),(1,1)],[(2,4),(1,4),(4,3),(2,2),(1,2),
(2,1),(3,1),(4,1)],[(2,4),(2,3),(4,3),(1,3),(2,2),(3,2)],[(3,4),(4,4),(1,4),(3,3),(4,3),(1,3
),(2,1),(3,1),(4,1),(1,1)],[(3,4),(4,4),(2,3),(3,3),(4,3),(3,2),(1,2),(1,1)],[(3,4),(1,4),(1
,3),(2,2),(4,2),(4,1)],[(3,4),(2,3),(2,2),(3,2),(4,2),(1,2),(2,1),(3,1)],[(4,4),(1,4),(2,3),
(1,3),(1,2),(2,1)],[(4,4),(3,2),(3,1),(4,1)],[(1,4),(2,3),(3,3),(4,3),(1,3),(2,2),(4,2),(1,2
),(3,1),(1,1)],[(3,3),(4,3),(2,2),(3,2),(4,2),(2,1),(4,1),(1,1)]]
it :: [[(Int, Int)]]
(86.36 secs, 6518410544 bytes)
*Main>
```

NOTA: Estas pruebas están incluidas en el fichero Haskell, se podrán ejecutar llamando a la función prueba:

Ejemplos:

- prueba 2
- prueba 3
- prueba 4
- prueba 5.

REFERENCIAS

- http://www.anarkasis.com/rafa/homees.htm. Esta Web es la que hemos usado para probar el predicado, comparando los resultados que nos dan los algoritmos que usa con los resultados que nos da nuestro predicado.

- http://en.wikipedia.org/wiki/Lights_Out_%28game%29. En esta Web nos hemos documentado acerca de como funciona el juego y de posibles algoritmos para resolverlo.