De Battista Agustin, LU: 121974.
Escobar Agustin, LU:121000.

Nonograma documentación 2

Descripción general

Para la segunda entrega del proyecto se nos pidió extender la funcionalidad del proyecto 1 para permitir al usuario obtener ayuda en el juego. Concretamente, extender la interfaz gráfica con:

- Un botón “revelar celda”, de tal manera que cuando el usuario cliquea este botón, se muestra el contenido de la celda de acuerdo a la solución.
- Un botón “mostrar solución” que permite mostrar/ocultar la solución completa del Nonograma.

Cambios realizados del primer proyecto

Implementamos la funcionalidad para chequear las pistas al inicio del juego.

React e interfaz

Para poder contemplar las dos grillas lo que hicimos fue lo siguiente:
Al iniciar el juego creamos una variable en la cual almacenamos la solución completa del juego. De esta manera tenemos en una variable la solución completa y en otra la solución parcial de la grilla.
Posteriormente, creamos un botón que funciona como “toggle” denominado **“Mostrar solución”** el cual tiene dos estados, que nos permite que en un estado la grilla obtenga el estado de la variable solución y en el siguiente estado cambie la grilla al estado de la solución parcial.
Esto posibilita que el usuario pueda alternar entre ver la solución del juego y seguir jugando su partida.

- Botón **Mostrar solución**:

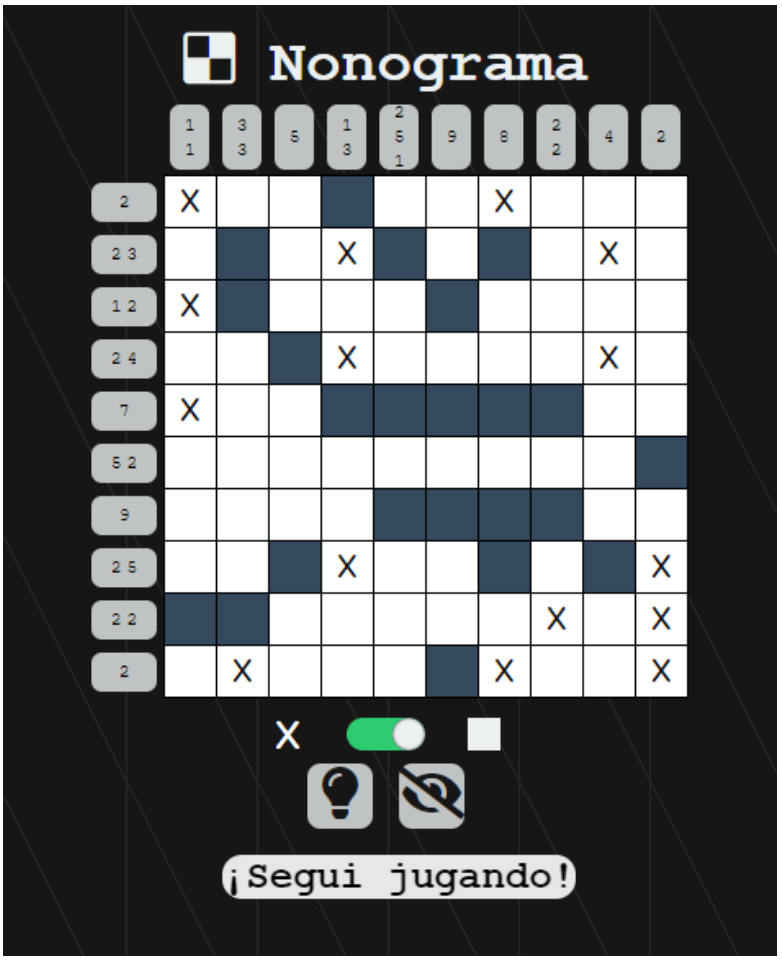
 **Desactivado**  **Activado**

Por otra parte, creamos el botón **“Revelar celda”**. Este, le permite al usuario obtener el valor correcto (de acuerdo a la solución) de la celda que clickeo. También es un botón que funciona como “toggle” como el mencionado anteriormente, lo que hace es, para una determinada celda en una determinada posición, obtener el contenido que tiene la variable solución en esa celda de la grilla.

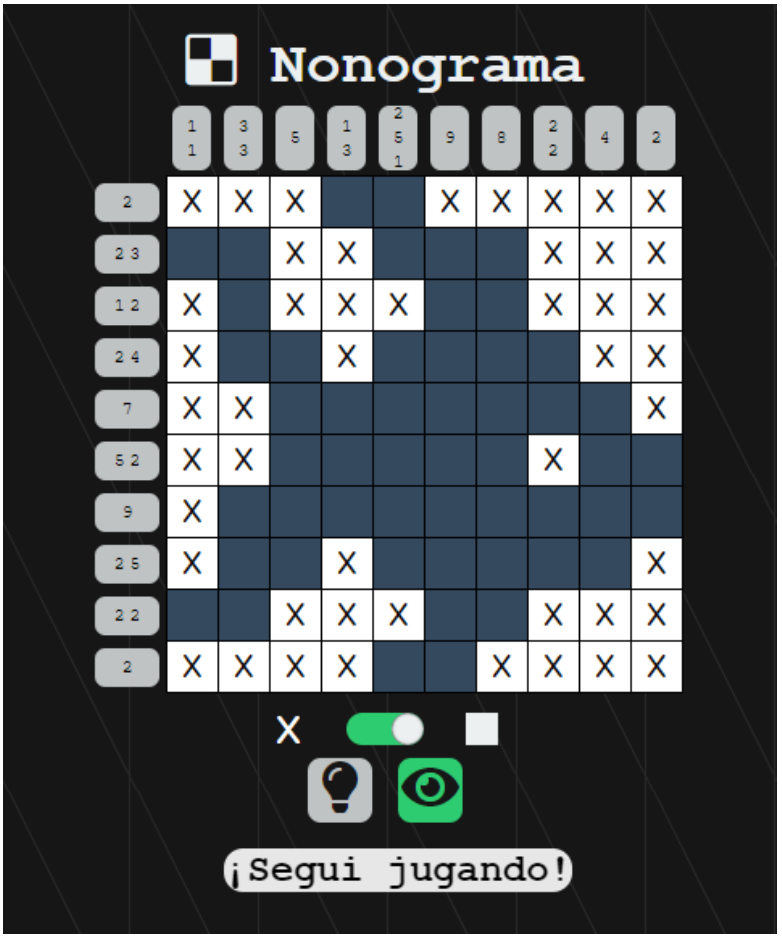
- Botón **Revelar celda**:

 **Desactivado**  **Activado**

A continuación, veremos en la siguiente imagen el juego como aparece inicialmente.



En la posterior imagen vamos a visualizar como queda la grilla al apretar el botón **mostrar solución.**



Implementación en Prolog

➔ Estrategia utilizada:

Se basa en ir haciendo pasadas cautas, que consta de intersectar todas las posibles soluciones para una fila y pista dada, la lista resultante de esta intersección es un paso “seguro”. Una pasada recorre primero todas las filas y luego las columnas.

➔ Predicados importantes:

`chequearInicio(+Grilla,+PistasF,+PistasC,-CumpleF,-CumpleC)`

- Dada una Grilla, una lista de pistas p/filas y otra p/columnas, para cada fila/columna, este predicado chequea si cumple con su pista correspondiente, en caso de que cumpla, ira un 1, caso contrario va un 0. Por lo que CumpleF y CumpleC van a ser listas conformadas por 0’s y 1’s. Donde va a tener un 1 en la posición de la fila(o columna) que ya este cumpliendo con la pista y 0’s donde no.

`solucion(+Grilla,+PistasF,+PistasC,-GrillaRes).`

- Dada una Grilla, una lista de Pistas p/filas y otra p/columnas, este predicado se encargará de que, luego de la llamada al predicado ‘`solucionAux`’, chequear si este retorno una grilla válida (i.e, el nonograma resuelto), si es asi, termina, caso contrario sigue iterando hasta encontrar una solución correcta.

`solucionAux(+Grilla,+PistasF,+PistasC,-GrillaRes).`

- Dada una Grilla, una listas de Pistas p/filas y otra p/columnas, lo que hace este predicado es que, a partir de la Grilla recibida, hace una pasada. Una pasada se conforma de: por cada fila en la grilla, se realiza un paso seguro para cada una de estas. Luego, para cada columna en la grilla resultante del paso anterior, se realiza el mismo procedimiento que para las filas.

`filaCauta(+Lista,+Pista,+LongitudLista,-FilaCautaRetornada).`

- A partir de una Lista, una lista de pista, y la longitud de la primera lista parametrizada, este predicado busca todas las posibles soluciones para la pista dada. Luego, se intersectan todas las posibles soluciones, y de ahí se obtiene un paso seguro. El paso seguro se constituye tanto de pintar como de marcar con una ‘`X`’ la fila.

`interseccion_aux(+SolucionesPosibles,+Index,+In,-Retorno).`

- Por cada lista en SolucionesPosibles, se intersectan todas las listas en el *Index* y se chequea que estas sean iguales, si estas son iguales, significa que va ese elemento y se agrega a la lista de retorno, caso contrario se sigue con la iteración.

`generarFilas(+Grilla,+PistasF,-GrillaOut).`

- Recibiendo una grilla y una lista de pistas, para cada fila en la grilla, se hace un paso “seguro”, tanto como para agregar una X como para pintar la celda correspondiente. Luego se retorna la grilla resultante de hacer una pasada de filas seguras.

`generarColumns(+Grilla,+PistasC,+Indice,-GrillaOut).`

- Recibiendo una grilla y una lista de pistas y un índice, para cada columna en la grilla, se hace un paso “seguro”, tanto como para agregar una X como para pintar la celda correspondiente. Luego se retorna la grilla resultante de hacer una pasada de columnas seguras.

Predicados predefinidos utilizados

- `length/2` para determinar el tamaño de una lista.
- `var/1` para chequear si una variable es instanciada o no.
- `member/2` para buscar una lista en la grilla.
- `forall/2` para chequear si en todas las listas de la grilla no hay variables instanciadas (Línea 164, proylcc.pl).
- `findall/3` para buscar todas las posibles soluciones de una pista en una lista (Línea 122, proylcc.pl).
- El operador de corte (!) se utilizó en el predicado `filaCauta/4` para tener éxito ya que descarta ciertas combinaciones que no son necesarias.

Casos de prueba

5x5

[[_,_,_,_,_],
[_,_,_,_,_],
[_,_,_,_,_],
[_,_,_,_,_],
[_,_,_,_,_]]

Caso:

- [[3],[1],[2],[1,3],[2,1]], %PistasFilas
- [[1,3],[3,1],[1,1],[2],[1]], %PistasCol

8x8

[[_,_,_,_,_,_,_,_],
[_,_,_,_,_,_,_,_],
[_,_,_,_,_,_,_,_],
[_,_,_,_,_,_,_,_],
[_,_,_,_,_,_,_,_],
[_,_,_,_,_,_,_,_],
[_,_,_,_,_,_,_,_],
[_,_,_,_,_,_,_,_]]

Casos:

- [[1,2,1],[1,1,1],[5],[1,3],[1,1,1],[1,1,1],[1,1,4],[1,2]], %PistasFilas
- [[7],[1,1],[7],[1,2,1],[1,5],[1],[2,1],[1]] %PistasColumnas

- [[2,4],[1,3],[4,2],[7],[6],[1,1],[1,1,1],[1,2,3]] %PistasFilas
- [[1,1,1],[1,3,1],[3,1],[8],[1,3],[2,2,2],[5,1],[3,1]] %PistasColumnas

10x10

[[_,_,_,_,_,_,_,_,_,_],
[_,_,_,_,_,_,_,_,_,_],
[_,_,_,_,_,_,_,_,_,_],
[_,_,_,_,_,_,_,_,_,_],
[_,_,_,_,_,_,_,_,_,_],
[_,_,_,_,_,_,_,_,_,_],
[_,_,_,_,_,_,_,_,_,_],
[_,_,_,_,_,_,_,_,_,_],
[_,_,_,_,_,_,_,_,_,_],
[_,_,_,_,_,_,_,_,_,_]]

Caso:

- [[1,3,4],[1,2],[4,1],[1,4],[5,4],[1,1,1],[2],[4],[3,1],[9]] %PistasFilas
- [[1,1,1,2],[4,2],[1,1,1,2],[1,1,1,1],[2,4,1],[2,1],[2,2,1,1],[5,1,1],[1,2,2],[1,3]] %PistasColumnas

15x15

[[_,_,_,_,_,_,_,_,_,_,_,_,_,_,_],
[_,_,_,_,_,_,_,_,_,_,_,_,_,_,_],
[_,_,_,_,_,_,_,_,_,_,_,_,_,_,_],
[_,_,_,_,_,_,_,_,_,_,_,_,_,_,_],
[_,_,_,_,_,_,_,_,_,_,_,_,_,_,_],
[_,_,_,_,_,_,_,_,_,_,_,_,_,_,_],
[_,_,_,_,_,_,_,_,_,_,_,_,_,_,_],
[_,_,_,_,_,_,_,_,_,_,_,_,_,_,_],
[_,_,_,_,_,_,_,_,_,_,_,_,_,_,_],
[_,_,_,_,_,_,_,_,_,_,_,_,_,_,_],
[_,_,_,_,_,_,_,_,_,_,_,_,_,_,_],
[_,_,_,_,_,_,_,_,_,_,_,_,_,_,_],
[_,_,_,_,_,_,_,_,_,_,_,_,_,_,_],
[_,_,_,_,_,_,_,_,_,_,_,_,_,_,_],
[_,_,_,_,_,_,_,_,_,_,_,_,_,_,_],]

Caso:

- [[9],[3,5],[6,3],[6,2],[2,1,2],[1,5,5],[8,2],[6,4],[1,5,4],[1,4,1],[1,2,6],[1,9],[4,3],[2,3],[6,3,3]] %pistasFilas
- [[2,1,2,4,1],[2,5,1,1],[2,2,1,1,1],[1,3,1,1,1],[2,4,1,1],[2,4,1,2],[1,4,1,1,1],[1,1,4,1,1],[1,1,3,2,1],[1,1,2,4,1],[3,3,4],[3,2,4],[3,2,1,2,2],[2,1,1,1,1,2],[2,1,1,3,2]] %PistasColumnas

4x9

[[_,_,_,_,_,_,_,_,_],
[_,_,_,_,_,_,_,_,_],
[_,_,_,_,_,_,_,_,_],
[_,_,_,_,_,_,_,_,_]]

Caso:

- [[1,1,1,1],[1,1],[3,1],[3]] %PistasFilas
- [[1,1],[1],[1],[2],[],[2],[1],[2],[1,1]] %PistasColumnas

Observación

- Probamos una grilla de 20x20 y no muestra nada en pantalla, por lo que no obtuvimos ningún resultado.