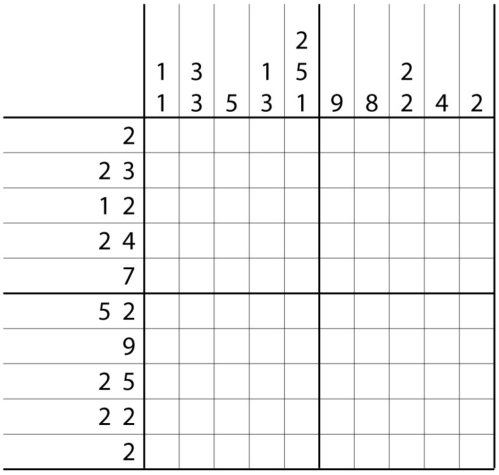
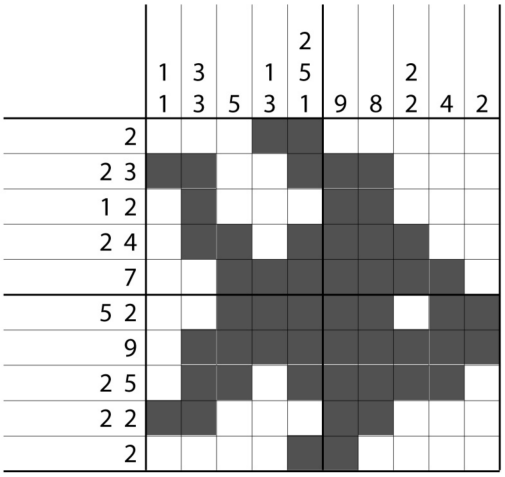
Depto. de Cs. e Ing. de la Computación
Universidad Nacional del Sur

Proyecto 1: Nonograma

¿En qué consiste?

El *Nonograma*[1], también conocido como *Pintar por números* o *Picross*, es un rompecabezas que consiste en colorear las celdas correctas de una cuadrícula, de acuerdo con los números especificados a los lados de la misma, llamados "pistas", con el fin de revelar una imagen oculta. En este tipo de rompecabezas, los números constituyen una especie de representación o codificación concisa o comprimida de la imagen oculta, esto es, los números especifican cuántos cuadros rellenos contiguos hay en una fila o columna dada da la grilla. Por ejemplo, una pista de "4 8 3" significa que hay grupos de cuatro, ocho, y tres cuadros rellenos, en ese orden, con al menos un espacio en blanco entre los grupos sucesivos.





Existe también una versión del Nonograma a color, pero para el proyecto vamos a restringirnos al blanco y negro.
En cuanto a las dimensiones de la grilla, no hay restricciones, pudiendo no ser cuadrada, con cualquier número de filas y columnas.

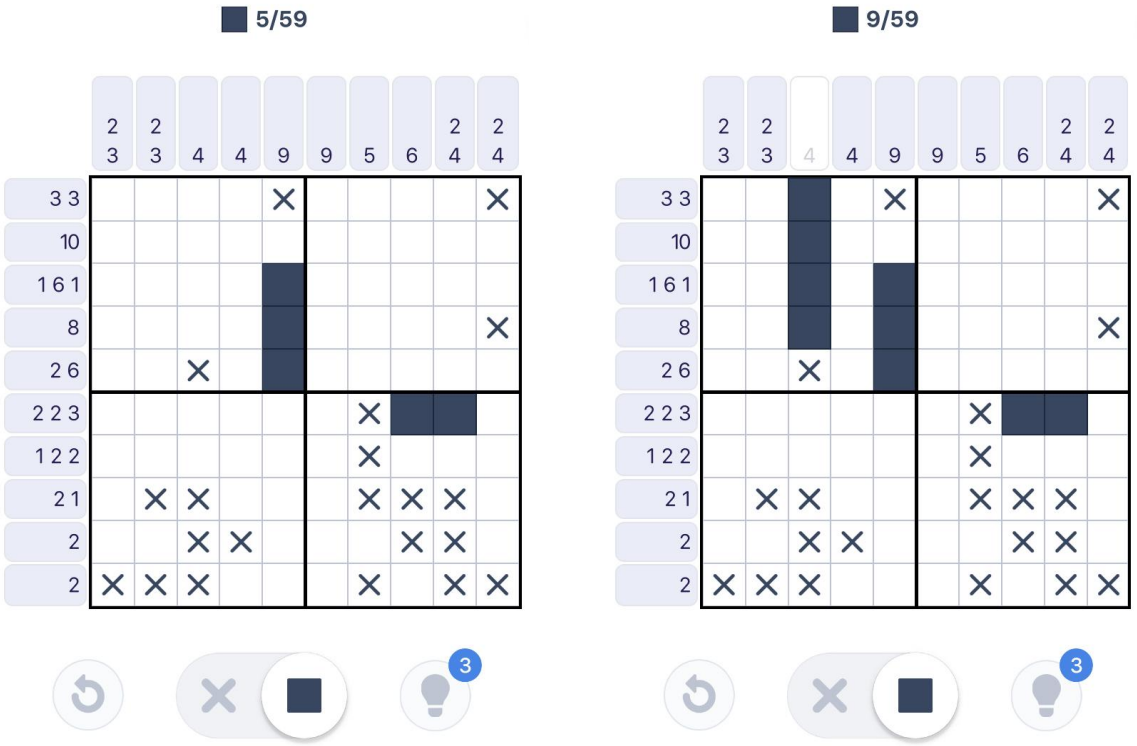
Les recomendamos que bajen la app https://nonogram.com, versión blanco y negro (disponible para Android y iOS), para familiarizarse con el juego. Tiene un tutorial interactivo muy claro, y además los requerimientos funcionales para el proyecto están basados en esta aplicación, en el modo de juego clásico (Settings > Game Mode > Classic).

[1] Esta explicación fue extraída de esta página en Wikipedia, con algunas modificaciones.

Requerimientos

Funcionalidad

Se debe implementar una aplicación que permita resolver interactivamente un Nonograma, al estilo de la app Nonogram.com, versión blanco y negro, y modo clásico (Settings > Game Mode > Classic).



A continuación se listan requerimientos más específicos:

- Presentar al jugador una grilla interactiva (con celdas clickeables) con las pistas a los costados, donde es posible que inicialmente ciertas celdas ya estén pintadas o marcadas como espacios. Esta grilla inicial se especificará en Prolog, como se explicará en la sección siguiente.
- Permitir elegir, en todo momento, el modo de marcado (checkbox / toggle), entre pintar o modo cruz. En modo pintar, al clickear una celda esta se pinta, excepto que ya esté pintada, y en ese caso se borra (queda vacía). En modo cruz, al clickear una celda esta se marca con una X, indicando explícitamente que la queremos dejar sin pintar (de ahora en adelante), excepto que ya tenga una X, y en ese caso se borra (queda vacía).

  A los efectos de determinar el cumplimiento de las pistas de una fila o columna, o la conclusión del rompecabezas, una X tiene exactamente el mismo efecto dejar la celda

vacía. Sin embargo es una herramienta muy útil para recordar que ya tomamos la decisión o concluímos que dicha celda debe quedar en blanco, y nos facilita inferir luego el contenido de otras celdas por descarte.

- En todo momento, y para cada fila o columna (diremos *línea* en general), indicar visualmente si esta satisface las pistas asociadas (por ejemplo, pintando de verde el panel con las pistas, o con un efecto de transparencia como en Nonogram.com). Observaciones:
    - una línea puede satisfacer las pistas y de todas maneras no ser correcta de acuerdo a la solución final.
    - una línea puede dejar de satisfacer las pistas si se pintan más celdas, o borran (o marcan con X) celdas pintadas.
- Detectar que se resolvió el nonograma: todas las líneas cumplen con las pistas.

Implementación

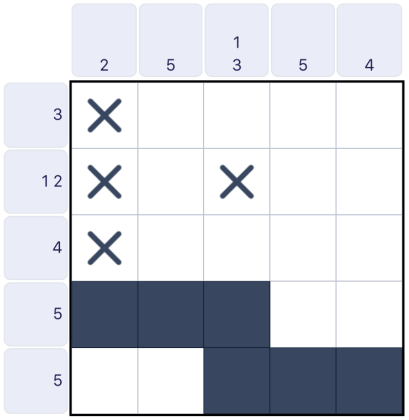
En nonograma a resolver se especificará en un archivo init.pl (módulo Prolog) en pengines_server/apps/proylcc, que será importado desde app.pl (entry point de la implementación Prolog), al igual que proylcc.pl, para poder ser consultado desde JS en el cliente. Se representará mediante un hecho `init/3` con la siguiente forma:

```
init(PistasFilas, PistasColumns, Grilla).
```

donde:

- `PistasFilas` es una lista de listas de números, representando las pistas de cada fila.
- `PistasColumnas` es una lista de listas de números, representando las pistas de cada columna.
- `Grilla` es una lista de filas, donde una fila es una lista de: `'#'` (celda pintada), `'X'` (celda marcada como no pintada), o variable sin instanciar (vacía, en blanco, sin valor).

Ejemplo:



```
init(
      [[3], [1,2], [4], [5], [5]],     % PistasFilas

      [[2], [5], [1,3], [5], [4]],     % PistasColumnas

      [["X", _ , _ , _ , _ ],
       ["X", _ ,"X", _ , _ ],
       ["X", _ , _ , _ , _ ],          % Grilla
       ["#","#","#", _ , _ ],
       [ _ , _ ,"#","#","#"]
      ]
).
```

Implementar un predicado `put/8`:

```
put(+Contenido, +Pos, +PistasFilas, +PistasColumnas, +Grilla,
-GrillaRes, -FilaSat, -ColSat)
```

donde:

- `Contenido` es `"#"`, `"X"` o `_` (variable sin instanciar),
- `Pos` es una lista `[Fila, Columna]`, indicando la posición donde se desea colocar `Contenido`,
- `PistasFilas`, `PistasColumnas`, `Grilla`, con la misma representación que para `init/3`,
- `GrillaRes`, la grilla resultante, con la misma representación que para Grilla,
- `FilaSat` es `1` si las fila de `Pos` satisface las pistas asociadas, y `0` en caso contrario.
- `ColSat` es `1` si las columna de `Pos` satisface las pistas asociadas, y `0` en caso contrario.

Observaciones:

- Alternativamente, `Contenido` podría tomar sólo valores `"#"` y `"X"`, y en caso de que la celda en `Pos` tenga el mismo valor, entonces se borra, esto es, se cambia por `_` (variable sin instanciar). En este caso la lógica de 'borrar cuando se marca con lo que ya había' queda del lado de Prolog.
- Luego de una movida únicamente puede haber cambiado el estado (satisfecha / no satisfecha) de la fila y la columna de la celda que se cambió, y es por esto que con `FilaSat` y `ColSat` retornados por `put/7` es suficiente para actualizar la interfaz, incluyendo determinar la conclusión del rompecabezas.

Documentación

Se deberá realizar un informe que explique claramente la **implementación en Prolog** realizada.
Para la **implementación en React** simplemente pedimos que especifique los valores mantenidos en el estado del componente principal y su propósito, a alto nivel, además de las consultas realizadas a Prolog, y cómo se actualiza el estado, también a alto nivel, a partir de las respuestas obtenidas. Además, deberá escribirse una sección que explique brevemente los **pasos** requeridos **para interactuar con la interfaz**.

Se recomienda estructurar el informe de manera top-down, comenzando con una descripción a alto nivel de la implementación. Puede aprovechar el informe para destacar características positivas de la resolución, y documentar cualquier otra observación que considere pertinente.

Importante: en el desarrollo de software, la documentación de la implementación constituye un elemento fundamental. Es por esto que, para la evaluación del presente proyecto, se dará suma importancia a la calidad (claridad y completitud) del informe entregado. Aproveche las consultas para preguntar acerca del desarrollo del informe.

Comisiones y Entrega

1. Las comisiones pueden estar conformadas por hasta 2 integrantes, y deben ser registradas en la página de la materia. A cada comisión se le asignará un docente de la práctica, quien hará el seguimiento y corregirá el proyecto de la comisión.
2. La fecha límite de entrega del presente proyecto se encuentra publicada en la página de la materia. Los proyectos entregados fuera de término recibirán una penalización en su calificación, la cual será proporcional al retraso incurrido.
3. La entrega del proyecto consiste del envío por mail de la resolución del proyecto y versión electrónica del informe.
    a. Enviar por mail directamente al integrante de cátedra asignado a la comisión, con copia al asistente (en caso de no ser el asignado). Mails:
        - Federico: fedeschmidt.10+LCC@gmail.com
        - Diego: dieorbe96+LCC@gmail.com
        - Germán: germang04+LCC@gmail.com
        - Mauro (asistente): mgomezlucero+LCC@gmail.com
    b. Asunto del mail: “Proyecto 1 LCC - Comisión <Ap.y Nom. Integrantes>”
    c. Adjunto un.zip conteniendo las **carpetas `public`, `src`,** (implementación React) **y `pengines-master/apps/proylcc`** (implementación Prolog) y un **pdf con el informe**. Alternativamente a enviar un adjunto puede compartir alguna carpeta en la nube (ejemplo: dropbox, google drive, etc.). Esta opción evita posibles 'rebotes' del mail por cuestiones de seguridad. Importante: en cualquier caso verifique que el mail no fue 'rebotado'.