Departamento de Computación
FCEFQyN, Universidad Nacional de Río Cuarto
Asignaturas: Diseño de Algoritmos - Algoritmos II
Primer Cuatrimestre de 2019

Trabajo Práctico No. 2

La resolución de este trabajo práctico debe ser enviada a través del moodle de la materia `http://dc.exa.unrc.edu.ar/moodle`, antes de las **23:55 del domingo 16 de Junio de 2019**.
El código provisto como parte de la solución a los ejercicios deberá estar documentado apropiadamente (por ejemplo, con comentarios en el código). Aquellas soluciones que no requieran programación, como así también la documentación adicional de código que se desee proveer, debe entregarse en **archivos de texto convencionales o archivos en formato PDF** unicamente, con nombres que permitan identificar facilmente su contenido.
Tanto la calidad de la solución, como el código y su documentación serán considerados en la calificación. Recuerde además que los trabajos prácticos **no tienen recuperación**, y que el trabajo en la resolución de los ejercicios debe realizarse en grupos de 3 personas.

Ej. 1. *Light Up* es un interesante juego de ingenio, que consiste en posicionar, en un tablero de 7×7 celdas, un número de lámparas, de acuerdo a restricciones indicadas en el propio tablero, de manera de iluminar todo el tablero. El tablero está inicialmente ocupado por celdas blancas y negras. Las lámparas pueden ser ubicadas sólo en celdas blancas, y las celdas negras pueden contener restricciones numéricas: valores numéricos alojados en las celdas que indican cuántas celdas vecinas (horizontales y verticales) deben contener lámparas.

La luz de una lámpara se propaga horizontal y verticalmente, hasta el límite del tablero, o hasta chocar con una celda negra. Finalmente, no pueden alojarse dos lámparas de manera tal que sus respectivos haces de luz se choquen.

Como referencia, puede analizar una versión online del juego, en `https://www.puzzle-light-up.com`.

Se requiere implementar un programa que, dado un tablero de *Light Up*, busque automáticamente una solución al mismo mediante un algoritmo genético. Su solución debe implementarse utilizando la biblioteca JGAP, y debe incluir una descripción de la representación elegida para el problema, e instrucciones de uso de la aplicación. La aplicación debe permitir setear un tablero específico, a través de alguna API a diseñar como parte de la solución. No es necesario soportar una interfaz gráfica, es suficiente con una interfaz de consola.

Tanto la calidad de diseño de la solución como la capacidad del programa para resolver tableros serán analizados como parte de la evaluación.

Ej. 2. El *Duidoku* es una versión de Sudoku de dos jugadores. En este juego, los jugadores alternan movimientos, que consisten en posicionar valores en el tablero de manera tal de no generar conflictos en el mismo. Para un tablero clásico de 9×9, los conflictos son los usuales de Sudoku: no puede haber valores repetidos en una misma fila, columna, o región de 3×3; y los valores permitidos en el tablero son enteros entre 1 y 9. Gana el juego aquel jugador que es capaz de realizar el último movimiento en el mismo, es decir, aquel que fuerza que su adversario no pueda colocar ningún valor en el tablero.

Como referencia, puede considerar la versión online del juego en `http://www.duidoku.com`.

Se requiere implementar un programa que juegue automáticamente a *Duidoku*, contra un jugador humano, en un tablero clásico de 9×9. La solución debe implementarse utilizando la técnica *Minimax con poda alfa-beta*. No es necesario soportar una interfaz gráfica, es suficiente con una interfaz de consola. Tanto la calidad de diseño de la solución como la capacidad del programa para jugar a *Duidoku* serán analizados como parte de la evaluación. Su solución debe incluir descripción de la representación elegida para el problema, e instrucciones de uso de la aplicación.