PASATIEMPOS LÓGICOS

Cuestiones generales:

- Sobre la interfaz:
  - Opción de cargar/salvar partida en juego.
  - Opción de proponer juego nuevo (tanto definido por el usuario como generado por la propia máquina). Cuando sea por el usuario, hay que habilitar una opción para comprobar si el juego tiene o no solución lógica única. En cualquier caso, preguntar tamaño (caso de ser propuesto por la máquina, preguntar asimismo por el grado de dificultad deseado).
  - Botón de comprobación del juego en marcha (la salida es el número de errores detectados, y si así se requiere por el usuario, adicionalmente se señala uno; en cada caso, se resta puntuación, o bien se penaliza incrementando el tiempo transcurrido en la solución).
  - Botón de siguiente jugada (la máquina realiza la siguiente jugada para el usuario; también penaliza y/o suma tiempo).
  - Botón de resolución (paso a paso o completa; se cancela la puntuación).
  - Botón de añadir o marcar (las formas posibles de cada casilla, que cambien cliqueando sucesivamente sobre la propia casilla).
  - Botones de guardar estado (que no partida) y recuperar estado guardado (con idea de facilitar que el usuario, en su caso, pueda volver a un estado anterior, previamente guardado).
  - Incluir un cronómetro para controlar el tiempo invertido en la resolución del juego.
  - Incluir un marcador (debe valorar las jugadas correctas, distinguiendo entre normales, buenas y excelentes; así como penalizar errores o peticiones de ayuda; debe puntuar también el tiempo utilizado por cada jugada).
  - Incluir un tutorial con la historia y copyright del juego, enlace a la web correspondiente, las reglas del juego y algunos ejemplos ilustrativos de las diferentes situaciones posibles.
  - Incluir un cuadro de honor con los mejores jugadores/partidas jugadas (en función puntos conseguidos, dificultad del juego, tiempo utilizado, ...).
- Sobre el algoritmo para resolver:
  - Tratar de caracterizar la unicidad lógica de la solución.
  - Diseñar teóricamente el algoritmo, e implementarlo (primero, encontrar una batería de técnicas elementales de resolución: ejecutar un bucle mientras hasta que ninguna de estas técnicas provea información nueva; después llamar a un backtracking, y dentro de cada suposición de ese backtracking volver a ejecutar el bucle mientras de las técnicas elementales).
  - Estudiar la posibilidad de establecer puntuación en la resolución, así como medir el grado de dificultad del juego.
- Sobre el algoritmo para generar juegos nuevos:
  - Trabajar con unicidad lógica de solución.
  - Diseñar teóricamente el algoritmo, e implementarlo (se basará en generar información aleatoriamente, y mientras la llamada al resolutor devuelva más de una solución, añadir una pizca de información adicional que discrimine entre las dos primeras soluciones encontradas).
  - Establecer puntuación en la resolución, así como medir el grado de dificultad del juego.
  - Permitir elegir tamaño y dificultad (en función de las técnicas de resolución que requiera).
- Eventualmente, extender el juego a 3D.
- Eventualmente, extender el juego a otros tableros: cualquier tablero plano con/sin casillas prohibidas, tableros cilíndricos (los laterales izquierdo y derecho están conectados entre sí), etc.

El algoritmo que resuelve completo, incluido el backtracking, debería quedar algo así:

Dato de entrada:

Un pasatiempo.

Inicialización de variables:

Una matriz ***tablero*** cuyas entradas contienen el número correspondiente a la casilla en cuestión (nada ó 0 en caso de que no se conozca con seguridad).

Una lista ***desconocidas*** con las posiciones (i,j) de las celdas que no tienen determinado aún el número.

Una lista ***pila***, que controla el backtracking, cuyos elementos son listas del tipo

{x,y,estado,tablero,desconocidas}, donde (x,y) son las coordenadas de la celda cuyo estado se supone, "estado" refleja el estado que se supone y el resto de datos guarda la información provisional en el instante dado de cada una de las variables principales; de manera que si en el backtracking se tiene que volver a la situación inicial en que se hizo la suposición, se pueda recuperar la información que imperaba en aquel instante.

Una lista ***solución*** que albergará las soluciones al pasatiempo. Inicialmente, vacía.

Parte básica del algoritmo de resolución:

booleano=verdad.
Mientras booleano hacer
    booleano=falso.
    Si alguna técnica elemental produce algún cambio, hacer booleano=verdad. Fin si
Fin mientras.
Si ***desconocidas*** está vacía,
    añadir a ***soluciones*** la solución encontrada. FIN (se acaba el proceso, porque se ha resuelto el pasatiempo con las técnicas elementales, sin necesidad de entrar en backtracking),
fin si.
Comentario: Cuando se sale de este mientras, es porque las técnicas elementales no aportan información nueva. Entonces, si quedan celdas en la lista ***desconocidas***, hay que entrar en una búsqueda exhaustiva de posibilidades, mediante un backtracking. Como se desea encontrar todas las hipotéticas soluciones del pasatiempo, y no sólo una, hay que agotar el backtracking hasta el final.

Parte del backtracking del algoritmo de resolución:

Añadir a ***pila*** la primera celda de la lista ***desconocidas***, con estado =al primer valor posible (1, si no tiene una casilla adyacente con valor 1), y guardando los valores de cada una de las variables principales del programa.

Mientras que ***pila*** no esté vacía (i.e. su longitud sea mayor que cero), hacer

Booleano2=verdad.

cargar los valores de las variables **(x,y,estado,tablero,desconocidas)** que corresponden al último elemento de la lista ***pila***.

Pasar la parte básica de resolución del algoritmo, actualizando booleano2 a falso en caso de encontrar una contradicción..

Si booleano2 (no hemos llegado a contradicción) entonces hacer

si ***desconocidas*** está vacía,

añadir a ***soluciones*** la solución encontrada,

mientras que ***pila*** sea no vacía y el estado del último elemento de ***pila*** no pueda variar (porque coincida con el valor máximo posible para la celda),

eliminar de ***pila*** su último elemento,

fin mientras,

si ***pila*** es no vacía,

cambiar el estado de su último elemento al siguiente valor posible,

fin si.

En caso contrario,

añadir a ***pila*** la primera celda de la lista ***desconocidas***, con estado = al primer valor válido para la celda, y guardando los valores de cada una de las variables principales del programa,

fin si.

En caso contrario (hemos llegado a contradicción)

hacer mientras que ***pila*** sea no vacía y el estado del último elemento de ***pila*** sea el mayor posible,

eliminar de ***pila*** su último elemento,

fin mientras,

si ***pila*** es no vacía,

cambiar el estado de su último elemento de al siguiente valor posible,

fin si.

Fin si

Fin Mientras.

Salida: Lista ***soluciones*** con todas las soluciones.