Universidad Nacional de Jujuy

Facultad de Ingeniería

Programación II

AÑO 2013

Trabajo Final

Sudoku

INTEGRANTES:

Rearte, David Eduardo Lic. en Sistemas LU: 1059

INDICE

Introducción

Este informe tiene el objetivo de explicar el proceso de realización del llamado juego Sudoku. Cuyo módulo de generación adquiere importancia por su gran dificultad y por la aplicación de la teoría y práctica aprendidas en diferentes materias.

Conceptos previos

Sudoku

Es un pasatiempos matemático consistente en completar con números del 1 al n una tabla de n x n celdas (generalmente 9 x 9) de tal modo que en el resultado final ninguna fila, columna o subregión dada contenga más de una vez alguno de los números indicados. Todo sudoku debe presentar una única solución.

Datos técnicos

Se utilizaron las librerías de GTK y el programa se desarrolló en C, dadas las librerías gráficas GTK, el programa sólo correrá en sistemas operativos basados en el kenel GNU/Linux.

Aclaraciones

El desarrollo de este juego está basado en la interfaz de la mayoría de las aplicaciones ya existentes observadas y se consideró lo más correcto y tradicional en cuanto a modo de juego, el no mostrar la solución, ni los errores a medida de que el usuario ingrese los números. Justamente el objetivo es que el usuario “suponga” que su entrada total es correcta y no “soplarle” la solución a medida que ingresa los números. Presionando el botón “Listo”, se considera que se ha rendido y se le indican los errores en el caso de haberlos.

Instrucciones

Al iniciar el juego se abre una ventana inicial, donde se elige la dificultad y se genera un sudoku a partir de lo seleccionado.

Pasamos a otra pantalla donde se deberán introducir los números que se consideren correctos en las casillas que lo permitan.

Puede presionar el botón “Listo B|” al terminar de ingresar los números, ya sea porque se ha completado la grilla o porque se dá por vencido. Esto controlará los datos ingresados y marcará en rojo los números incorrectos y las casillas vacías. Si es correcto, se considera que ha ganado y se le permite volver a jugar o salir.

Puede presionar el botón “Atrás :[“ para volver a seleccionar la dificultad.

Puede presionar el botón “Salir :(“ para terminar la aplicación.

Desarrollo

Se dividió el desarrollo del juego en 2 fases, la primera con dos módulos (investigación y planeación) y la segunda con 4 módulos (Principal, Generación, Preparación y Solución). Se verán a detalle a continuación.

Fase 1: De la investigación y planteamiento de la solución.

Investigación

- Se recopiló información de distintas páginas web, entre ellas:

http://en.wikipedia.org/wiki/Mathematics_of_Sudoku

http://www.cristalab.com/tutoriales/algoritmo-para-generar-y-resolver-sudokus-con-codigo-actionscript-2-c234l/

http://www.wikihow.com/Create-a-Sudoku

http://es.wikipedia.org/wiki/Sudoku

- También se usaron herramientas para pruebas, y verificación de resultados:

http://www.sudokuwiki.org/sudoku.htm

- Resultado de la investigación: El problema detrás de lo evidente

A pesar de parecer algo sencillo, el sudoku es clasificado como problema NP-Completo (NPC). Así es llamado por la rama de la ciencia Complejidad Computacional.

Un problema es NPC cuando es NP y todos los demás problemas NP del mismo tipo se pueden reducir a él, es decir, solucionando el primero se solucionan los demás también. Se dice "solución" cunado el problema tiene una solución y dicha solución se ejecuta en un tiempo polinómico. La razón de la importancia del tiempo es que: si la solución puede ejecutarse en un tiempo, por ejemplo, exponencial, para cierto número de entradas puede llegar tardar años en encontrar la solución y eso no sirve.

Un problema es NP cuando puede ser resuelto por un algoritmo no determinista en un tiempo polinómico.

Entonces, el sudoku es un problema que puede ser resuelto en por un algoritmo no determinista en un tiempo polinómico, y que además, dandole una solución, otros problemas del mismo tipo se resuelven también, por ejemplo el SAT.

Un circuito SAT es un problema NP que consiste en: dado un circuito booleano y ciertas entradas, ¿se puede modificar las demás entradas de forma tal que la salida sea afirmativa? El sudoku es análogo a esta situación, los circuitos serían las reglas del sudoku, y las entradas los números del uno al nueve. Es por eso que resolviendo uno se resuelve el otro.

- Conclusión de la investigación

Investigados los diferentes métodos de generación y resolución del sudoku llegamos a la conclusión de que no existe un eficiente generador de sudoku's, pero sí varios métodos solucionadores. Los métodos para generar un sudoku son muy ineficientes y por demás indeterministas, no es factible su modelado matemático y su programación. Por lo que se creó un generador propio; y en cuanto al solucionador, se utilizó los métodos existentes.

Planteamiento de la solución

Se crean las variables necesarias, entre ellas la matriz que representa el sudoku completo y otra matriz que representa el sudoku a resolver. Se levanta una pantalla inicial donde se selecciona la dificultad. Luego se llama al módulo para que genere el sudoku, éste módulo le devuelve el sudoku completo, luego se lo copia en la segunda matriz y ésta se agujerea (borrar números) en función de la dificultad, para generar el problema a resolver. Por último pintar en pantalla el sudoku agujereado.

Fase 2: De la implementación de la solución.

La codificación se divide en módulos para que sea más fácil su mantenimiento, desarrollo, y explicación.

- Módulo: Principal

Representado en el archivo “main.c”, el cual llamará a los demás módulos cuando corresponda, crea las variables y pinta la ventana en la cual se desenvolverá el juego y donde permite elegir la dificultad. Esta dificultad se representa como la cantidad de números iniciales para el sudoku a resolver. Se usaron, según las fuentes, las siguientes cantidades:

Fácil 51, Medio 34 y Difícil 17.

Crea una variable matriz, para representar el sudoku, la cual envía al módulo de generación para que sea cargada con un sudoku correcto, y luego lo manda al módulo de producción para que quite elementos de forma tal de que la unicidad de la solución se mantenga, luego pinta ese sudoku “agujereado” para que el usuario lo resuelva.

Cuando el usuario presiona en “Listo”, es decir, considera su jugada correcta. Para corroborar que la entrada sea correcta, sólo se debe comparar la matriz ingresada por el usuario con la matriz solución.

Si el usuario desea volver a jugar o finalizar puede presionar los botones corrientes.

- Módulo: Generación

Consiste en el archivo “generarSudoku.h”. La solución planteada para el generador estará representada por una matriz de 9x9 la cual es provista por el módulo Principal, y el procedimiento es el siguiente:

Se recorre la matriz celda por celda recorriendo primero las filas, es decir, (0,0), (0,1)…(1,0), (1,1)…(9,9); y se generan números al azar del 1 al 9 para cada celda. Por cada número generado se controla si cumple con las reglas del sudoku antes de agregarlo a la matriz en la posición correspondiente, si no cumple con alguna se lo descarta y se genera otro, así hasta que se encuentra uno que cumpla. Si se han intentado con los 9 números y ninguno cumple, entonces estamos en lo que decimos un “callejón sin salida”, donde ningún número del 1 al 9 satisface esa celda. En este caso se procede a buscar el número del 1 al 9 que sólo tenga un conflicto en su columna, si existe, se coloca ese nuevo número, se borra el conflicto y se vuelve a pasar por la misma columna para rellenar ese hueco creado; si no existe un número con un solo conflicto en su columna, se busca el que tenga menor cantidad de conflictos. Si tiene conflicto en su columna se borra el número con el que tiene el conflicto y si tiene conflicto en su fila se empieza un proceso recursivo de intercambio del número con el que tiene conflicto y su celda superior o inferior según se esté más lejos del margen superior o inferior. Si ese intercambio de números logró satisfacer las reglas se lo deja así, sino se vuelve la matriz al estado seguro anterior y se busca el siguiente número con menor cantidad de conflictos y se repite el proceso anterior hasta encontrar uno que sí cumpla. En el peor de los casos, ningún número en esa celda cumplirá con las reglas, en este caso ese sudoku se da como sin solución y se reinicia el sudoku desde cero.

- Módulo: Preparación

Representado por el archivo “agujerearSudoku.h”. Una vez generado el sudoku por el módulo de generación, el módulo principal manda la matriz a este módulo, el cual tiene como función preparar la matriz a solucionar. Se sigue el siguiente procedimiento:

Se crea una copia de la matriz original, a esta copia se la trabaja escogiendo un número de celda al azar de 1 a 81 de forma que se recorren las 81 celdas, sólo que en des-orden y se detiene cuando ha borrado una cantidad x de números de forma tal que $x = 81 – p$, donde p es la

cantidad de celdas que representa la dificultad, esto se elige en la interfaz antes de comenzar el juego. Al seleccionar una celda a borrar, se analiza si el borrara producirá que el sudoku deje de tener solución y que ésta sea única. Para esto se utiliza el módulo de solucionador, es decir, se borra el número correspondiente en la matriz de prueba y se manda la matriz al módulo solucionador, si éste no lo puede resolver, entonces se revierte ese paso y se intenta con otra celda. Cada número borrado incrementa un contador de números que se van borrando, hasta que sea igual que x.

- Módulo: Solución

Representado por el archivo “resolverSudoku.h”, este módulo es utilizado por el módulo de Preparación para saber si el sudoku provisto tiene solución. Aquí se implementa un solucionador común y corriente. No se puede usar el módulo de generación para esto porque éste implica la alteración de los números de la matriz para lograr un sudoku correcto. Por lo que el solucionador debe ser creado por separado. Para esto se siguen las siguientes técnicas de

1. Dado un número, averiguar si en una fila, o en una columna, o en un caja sólo cabe en una posición.

2. Dada una posición, averiguar si en ella sólo es posible ubicar un dígito.

Si ninguna de las anteriores funciona, se da como sin solución y se devuelve un cero.

- Utilidades

También hay un archivo de utilidades “sudokuUtils.h”, cuyo rol es el encapsulamiento de las funciones más comunes y más independientes de la lógica implementada, por ejemplo: el saber si un elemento se encuentra en un array.

No llega a conformar un módulo por no ser parte intrínseca de la lógica implementada.

Conclusión

Se ha visto cómo se puede generar un sudoku a pesar de su situación de complejidad. También se justificó esta complejidad con investigaciones sobre el mismo. Ha resultado increíblemente enriquecedor a nivel intelectual, de técnicas de programación y teorías de la complejidad en la computación.