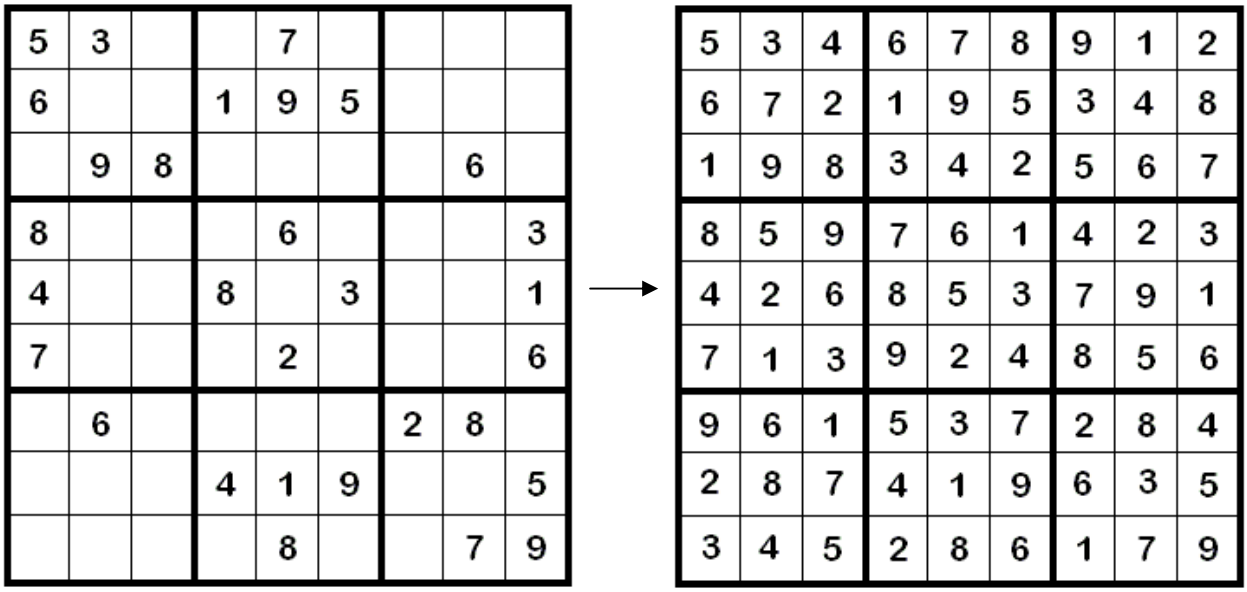
Estructuras de Datos (Gestión), curso 2005/06

Primera Práctica - Sudoku

Introducción

Sudoku es un rompecabezas matemático de colocación que se popularizó en Japón en 1986 y se dio a conocer en el ámbito internacional en 2005. El objetivo es rellenar una cuadrícula de 9×9 celdas dividida en subcuadrículas (también llamadas "cajas" o "regiones") de 3×3, con las cifras del 1 al 9 partiendo de algunos números ya dispuestos en algunas de las celdas. No se debe repetir ninguna cifra en una misma fila, columna o subcuadrícula. La resolución manual del problema requiere paciencia y ciertas dotes lógicas.

En la siguiente figura se puede ver un ejemplo de Sudoku y su solución:



Podéis encontrar más información sobre éste pasatiempo (y ejemplos para probar el programa) en www.sudoku.com y miles de sitios relacionados. Se debe tener en cuenta, sin embargo, que no es necesario (de hecho puede ser contraproducente) el saber resolver sudokus manualmente para poder desarrollar un programa que los resuelva.

Objetivo de la práctica

El objetivo de la práctica es construir un programa que sea capaz de resolver un sudoku de forma eficiente **encontrando todas las soluciones**. Para ello se debe utilizar alguna de las estrategias de diseño contempladas en la asignatura: divide y vencerás, fuerza bruta, backtracking, programación dinámica o algoritmo voraz.

En los sudokus normales sólo existe una solución para el problema planteado. En ésta práctica, sin embargo, se puede proporcionar unos datos de entrada para los cuales existan mas de una (o ninguna) solución posible.

Requisitos de la práctica

El programa que resuelva la práctica debe leer el rompecabezas de la entrada estándar siguiendo estrictamente el siguiente formato: Cada una de las filas del rompecabezas estará en una línea; cada línea está formada por 9 números del 0 al 9 separados por un espacio; el 0 indica una casilla vacía y el resto son números presentes en el sudoku al comenzar.

El ejemplo anterior se representará de la siguiente manera:

```
5 3 0 0 7 0 0 0 0
6 0 0 1 9 5 0 0 0
0 9 8 0 0 0 0 6 0
8 0 0 0 6 0 0 0 3
4 0 0 8 0 3 0 0 1
7 0 0 0 2 0 0 0 6
0 6 0 0 0 0 2 8 0
0 0 0 4 1 9 0 0 5
0 0 0 0 8 0 0 7 9
```

Una vez resuelto el sudoku, se debe mostrar por pantalla las soluciones, siguiendo estrictamente el siguiente formato: Primero aparece una linea indicando el número de soluciones encontradas, con el formato "*n* solucion(es) encontrada(s)", donde *n* es el número de soluciones, que puede ser 0 si el sudoku no tiene solución. A continuación se deben listar las soluciones (si hay alguna) con el siguiente formato: Cada fila del sudoku se imprimirá en una línea; cada línea estará formada por 9 números del 1 al 9 separados por un espacio.

Cada solución se separa de la siguiente por una linea en blanco, y no importa el orden en que se escriban las distintas soluciones.

La salida del programa en el ejemplo anterior debería ser:

```
1 solucion(es)encontrada(s)
5 3 4 6 7 8 9 1 2
6 7 2 1 9 5 3 4 8
1 9 8 3 4 2 5 6 7
8 5 9 7 6 1 4 2 3
4 2 6 8 5 3 7 9 1
7 1 3 9 2 4 8 5 6
9 6 1 5 3 7 2 8 4
2 8 7 4 1 9 6 3 5
3 4 5 2 8 6 1 7 9
```

Nota: Si existieran más de 1000 soluciones se permite que el programa calcule y presente solamente las 1000 primeras. En ese caso la primera linea de la salida sería "Más de 1000 soluciones encontradas".

La práctica se probará de forma automática, por este motivo debe respetarse tanto el formato de entrada como el de salida. El programa se probará mediante redirección de la entrada y de la salida:

```
sudoku < entrada.txt > salida.txt
```

Documentación que se debe entregar

Se deberá entregar el código fuente del programa que resuelve la práctica (se recomienda el uso del lenguaje Eiffel, aunque alternativamente se permite realizarlo en Java, C, C++ o Delphi/Kylix) y un documento (en formato Word, Latex (DVI) o Acrobat PDF) donde se indique lo siguiente:

- El tipo de estrategia(s) de diseño (divide y vencerás, fuerza bruta, backtracking, programación dinámica o algoritmo voraz) utilizada para resolver el problema planteado. Si se utiliza un algoritmo voraz debe incluirse una demostración de que con él se obtiene la solución óptima.
- Una breve descripción (1 o 2 páginas) del modo en que se ha aplicado la estrategia al problema.
- Una evaluación en notación asintótica de la eficiencia del programa (no es imprescindible que sea exacta: se pueden utilizar cotas inferiores y superiores). Para la evaluación se considerará que el tamaño de la entrada es igual al de casillas no asignadas de la entrada.

En la evaluación de la práctica se tendra en cuenta (entre otros factores) la eficiencia de la solución planteada.

Los ficheros de las prácticas se deben depositar en el directorio `$HOME/ed05_06/p1` de la cuenta del usuario en `jair` y se recogerán automáticamente el día **16 de diciembre** a las 10:00.

Comprobación de los programas

Los ejemplos que aparecen en periodicos, páginas web, etc. suelen tener una única solución. Para que podais probar el programa cuando no hay soluciones o existen múltiples soluciones se muestran a continuación dos ejemplos de ese tipo de situaciones:

Entrada (tomado del Norte de Castilla, 31/07/2005):

```
0 0 1 3 2 0 0 7 0
0 2 0 0 0 6 8 0 0
5 0 0 0 0 8 4 0 0
0 8 4 6 0 0 7 0 0
7 1 0 0 0 9 0 0 6
0 0 9 0 8 1 3 0 0
3 0 0 0 9 5 0 0 7
0 6 5 8 0 0 0 3 0
0 0 0 0 0 0 0 0 0
```

Salida del programa:

```
0 solucion(es)encontrada(s)
```

Entrada:

```
0 0 9 0 0 0 0 0 6
0 0 0 5 8 0 0 2 3
6 0 0 0 2 7 0 0 4
0 8 5 2 0 0 0 0 7
0 9 0 1 0 0 0 0 0
4 1 0 0 9 0 0 8 0
8 0 7 0 0 5 0 4 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 2 0 0
```

Salida del programa: (**Nota**: Las distintas soluciones se pueden presentar en cualquier orden)

```
3 solucion(es)encontrada(s)
2 5 9 3 4 1 8 7 6
1 7 4 5 8 6 9 2 3
6 3 8 9 2 7 5 1 4
3 8 5 2 6 4 1 9 7
7 9 6 1 5 8 4 3 2
4 1 2 7 9 3 6 8 5
8 2 7 6 1 5 3 4 9
9 6 1 4 3 2 7 5 8
5 4 3 8 7 9 2 6 1

2 5 9 4 3 1 8 7 6
1 7 4 5 8 6 9 2 3
6 3 8 9 2 7 5 1 4
3 8 5 2 6 4 1 9 7
7 9 6 1 5 8 4 3 2
4 1 2 7 9 3 6 8 5
8 2 7 6 1 5 3 4 9
9 6 1 3 4 2 7 5 8
5 4 3 8 7 9 2 6 1

2 5 9 4 3 1 8 7 6
1 7 4 5 8 6 9 2 3
6 3 8 9 2 7 5 1 4
3 8 5 2 6 4 1 9 7
7 9 6 1 5 8 4 3 2
4 1 2 7 9 3 6 8 5
8 2 7 6 1 5 3 4 9
9 6 3 8 4 2 7 5 1
5 4 1 3 7 9 2 6 8
```