Eötvös Loránd Tudományegyetem
Informatikai Kar

Sudoku megoldó algoritmusok

Készítette: Balázs Viktor (BAVIAAT.ELTE)
programtervező matematikus szak, nappali tagozat
Témavezető: Iványi Antal

Budapest, 2007. június

Tartalom

I. A feladat

1. A sudoku szabályai

A világszerte jelenleg legnépszerűbb számrejtvény, a sudoku szabályai: egy 9x9-es négyzetrács − amely 3x3-as kis négyzetekre van bontva − mezőiben az X = {0, 1, 2, …, 9} ábécé egy-egy betűje szerepel. A feladat az, hogy a nullákat úgy helyettesítsük az ábécé többi betűjével, hogy végül minden házban (azaz minden sorban, minden oszlopban, és minden 3x3-as kis négyzetben) minden szám pontosan egyszer szerepeljen.

A rejtvény általánosítása: egy NxN-es kis négyzetekből álló, N^2xN^2-es négyzetrács mezői az X = {0, 1, …, N^2} ábécé elemeivel vannak kitöltve. A feladat a nullák helyettesítése az X ábécé többi betűjével úgy, hogy végül minden házban az ábécé minden betűje pontosan egyszer forduljon elő.

2. A program célja

A program célja, hogy bemutassa az általánosított sudoku rejtvény megoldásához használható ötletek alkalmazhatóságát. Ehhez a működése során naplót készít az aktuális lépésekről.

II. Felhasználói dokumentáció

1. Hardver-szoftver követelmények

Szoftver:
A program Microsoft .NET Framework 2.0 környezetet igényel.

Hardver:
A futtatáshoz az operációs rendszer és a futó programok hardverigényén felül 32MB RAM-ra van szükség.

2. Telepítés

A program a Sudoku.exe fájlt elindítva közvetlenül futtatható, telepítést nem igényel.

3. A program részei

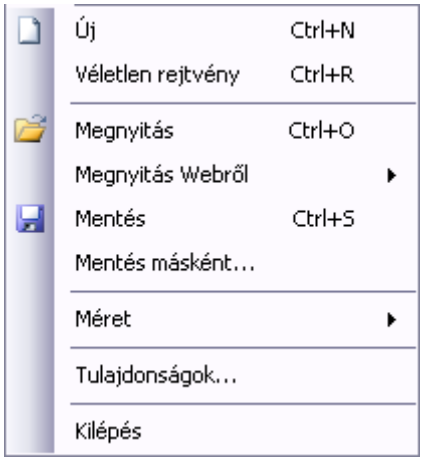
A program működéséhez a következő fájlok szükségesek:

- Sudoku.exe
- SudokuControls.dll
- SudokuTypes.dll
- sudokuhelp.chm

4. A program használata

a) Menük

Fájlmenü:



1. ábra: Fájlmenü

Új: Létrehoz egy üres sudokut (billentyűkombináció: CTRL + N).

Véletlen rejtvény: Véletlen sudoku rejtvényt generál alkalmazkodva az aktuális mérethez. (Billentyűkombináció: CTRL + R)

Megnyitás: Megnyitás ablak előhozása (billentyűkombináció: CTRL + O).

Megnyitás Webről: Az interneten számos oldal naponta új sudoku feladványokat tesz közzé, ebben a menüpontban a kiválasztott oldal napi feladványát lehet betölteni a programba.

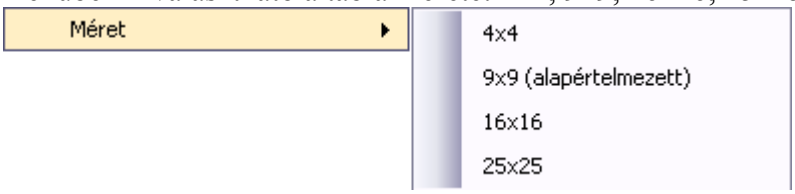
2. ábra: Megnyitás Webről

Az aktuális változat a http://www.dailysudoku.co.uk oldal napi feladványát tudja betölteni.

Mentés: Az aktuális rejtvényt menti az adott kitöltöttségi állapotban. Ha létező fájlt szerkesztettünk a fájlnevet és a formátumot nem változtatja, egyébként a fájlnév és a formátum szabadon állítható. Részletek a „Támogatott fájlformátumok” pontban (billentyűkombináció: CTRL + S).

Mentés másként: Az aktuális rejtvényt menti az adott kitöltöttségi állapotban, a fájlnév és a formátum szabadon állítható.

Méret: A kinyíló almenüben kiválasztható a tábla mérete: 4x4, 9x9, 16x16, 25x25

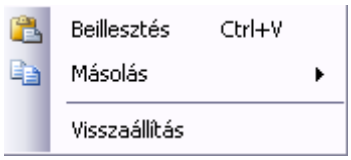


3. ábra: Táblaméret állítása

Tulajdonságok: Az SDK formátumú fájlokban különféle leíró adatok szerepelhetnek, ezek megjelenítését, szerkesztését lehet itt elvégezni.

Kilépés: A program bezárása.

Szerkesztés menü:



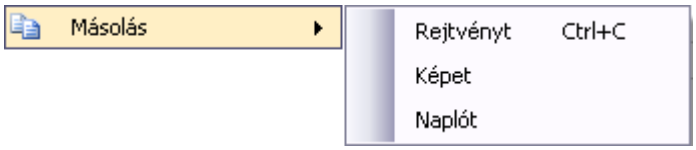
4. ábra: Szerkesztés menü

Beillesztés: A vágólapon található rejtvényeket olvassa be (billentyűkombináció: CTRL + V).
Az ismert formátumok 9x9-es sorfolytonos leírásúak. Példa:

```
009800210100000040060000005500002006000003000004080097000500000600720000092001500
..98..21.1......4..6......55....2..6.....3.....4.8..97...5.....6..72.....92..15..
__98__21_+1______4_+_6______5+5____2__6+_____3___+__4_8__97+___5_____+6__72____+_92__15__
```

Másolás: A vágólapra illeszti a rejtvényt, a képet (fekete-fehérben), illetve a Naplót.
Rejtvény vágólapra illesztésének billentyűkombinációja: CTRL + C. A sorfolytonos leírása (példa):

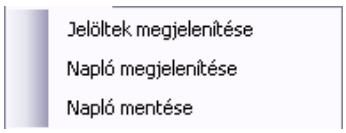
```
009800210100000040060000005500002006000003000004080097000500000600720000092001500
```



5. ábra: Vágólapra másolás

Visszaállítás: Visszaállítja a rejtvényt a megoldás előtti állapotába.

Eszközök menü:



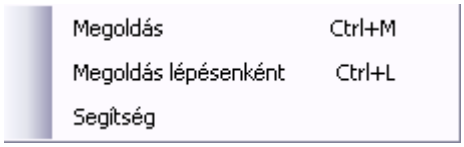
6. ábra: Eszközök menü

Jelöltek megjelenítése: Ki- és bekapcsolható a mezőkben megengedett jelölt megjelenítése.

Napló megjelenítése: Ki- és bekapcsolható az algoritmusok működését követő Napló megjelenítése.

Napló mentése: Napló mentése TXT fájlba.

Megoldás menü:



7. ábra: Megoldás menü

Megoldás: Megoldja a sudoku rejtvényt (billentyűkombináció: CTRL + M).

Megoldás lépésenként: Egyet lép a végső megoldás felé (billentyűkombináció: CTRL + L).

Segítség: Az állapotsoron megmutatja a következő lépést.

Súgó menü:

8. ábra: Súgó menü

Tartalom: Elindítja a súgót.

Névjegy: A program névjegye.

b) Rejtvény készítés

A program lehetőséget ad rejtvények készítésére. Az egérrel ki kell választani a táblán a kívánt pozíciót, majd a billentyűzettel lehet az adott mezőt kitölteni. 4x4-es és 9x9-es tábla esetén a számok használhatók (rendre 1-től 4-ig, illetve 1-től 9-ig), 16x16-os és 25x25-ös tábla esetén a betűk (A-tól P-ig, illetve A-tól Y-ig). A további mezők kitöltéséhez már a billentyűzet nyilait is fel lehet használni a navigációhoz.

Elírás esetén egy mezőben található adat átírható, illetve a Delete gombbal törölhető a tartalma.

Alapvetően hibás rejtvények készítését a program megakadályozza, azaz minden házban (sorban, oszlopban, blokkban) egy szám (illetve betű) csak egyszer szerepelhet.

c) Rejtvény megoldás

A Megoldás menü Megoldás, illetve Megoldás lépésenként menüpontjával lehet megoldani a feladványt.
A felhasznált algoritmusok:

- Naked Single
- Hidden Single
- Intersection
- Naked Pair
- Naked Triple
- Dancing Links.

d) Fájlformátumok

Támogatott be- és kimeneti fájlformátumok:

- SDK – 9x9-es táblákhoz, csak a kitöltött adatokról tárol információt, továbbá leíró adatokat tartalmazhat
- SDX – 9x9-es táblákhoz, a kitöltött adatokat és a jelölteket is tartalmazza
- SDXX – tetszőleges méretű táblákhoz, a kitöltött adatokat és a jelölteket is tartalmazza (saját formátum, más program egyelőre nem ismeri)

5. A program korlátozásai

A program az alábbi megkülönböztetéseket teszi az egyes rejtvény méretek között:

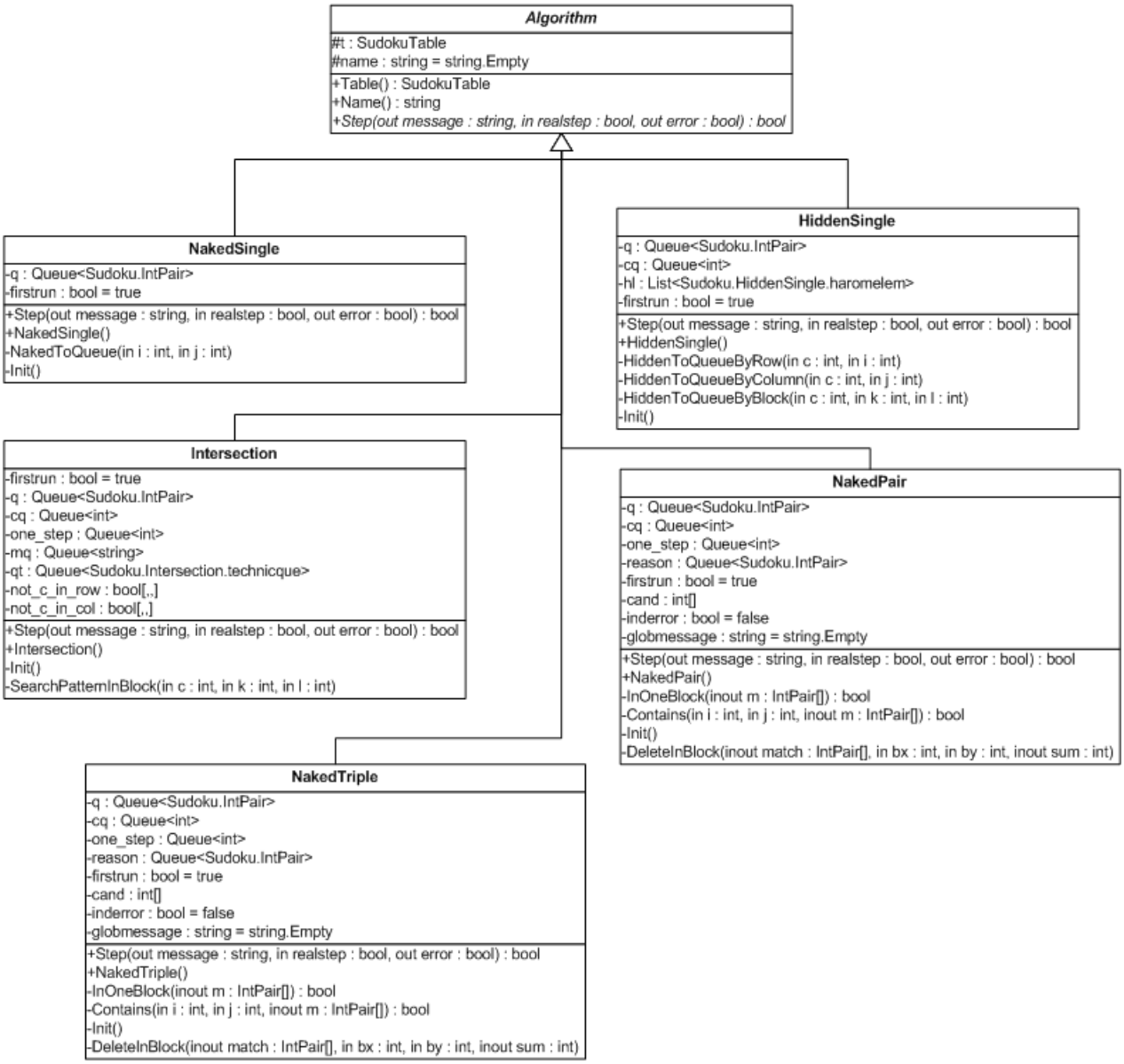
- Rejtvény tulajdonságainak állítása csak a 9x9-es méretben lehetséges, ott is csak sdk formátum esetén.
- Rejtvény vágólapra illesztése sorfolytonos formában csak 9x9-es méretben lehetséges.
- Véletlen rejtvény generálása csak 4x4, 9x9, 16x16-os esetben lehetséges.

III. Fejlesztői dokumentáció

1. Elnevezések

Mező: A rejtvényt alkotó kis négyzet.
Jelölt: Egy mezőben lehetséges érték.
N: Blokk oldalszélesség.
M: M=N*N, azaz a rejtvény oldalának mérete, a felhasznált ábécé számossága.
Blokk: N*N-es négyzet.
Ház: M mezőből álló sor, oszlop, blokk együttes neve.
Szomszéd: Egy mező egy másiknak szomszédja, ha vele azonos házban van.

2. Osztálydiagram



Az absztrakt Sudoku::Algorithm osztályból származnak a felhasznált polinomiális futásidejű algoritmusok. Ez az osztály tartalmaz egy `SudokuTable` típusú objektumot, és meghatározza az elvárt Step függvény paraméterlistáját.

3. Osztályok leírása

1. Adatszerkezetek

SudokuCell: (SudokuCell.cs)
`Icloneable` interfészt megvalósító, jelölteket tartalmazó adatszerkezet.
A jelölteket egy `bool[] candidate` tömbben tárolja, hogy egy jelölt meglétének lekérdezése gyorsan történhessen.

Függvényei:
`public void AddCandidate(int c)`
A `c` jelöltet hozzáadja a mezőhöz.

`public void RemoveCandidate(int c)`
A `c` jelöltet törli a mezőből.

`public bool IsCandidate(int c)`
Jelzi, hogy `c` jelöl-e a mezőben.

`public bool IsData()`
Jelzi, hogy van-e adat a mezőben.

`public void ResetCell()`
Visszaállítja a mezőt a kezdeti állapotba.

`public static bool IsLegalCharacter(char ch, int m)`
Jelzi, hogy `ch` érvényes jelölt-e, ha a mező mérete m.

`public static bool CompareByCandidates(SudokuCell cell1, SudokuCell cell2)`
Jelzi, hogy `cell2` a jelölteket tekintve megegyezik-e a `cell1` -el.

`public static bool Subset(SudokuCell cell1, SudokuCell cell2)`
Jelzi, hogy `cell2` a jelölteket tekintve részhalmaza-e a `cell1` -nek.

`public static bool Disjunct(SudokuCell cell1, SudokuCell cell2)`
Jelzi, hogy a két mező diszjunkt-e jelöltjeiket tekintve.

`public static string ConvertCellDataToString(int c, int m)`
Visszatér a rejtvény méretétől függően a `c` -nek megfelelő stringgel, azaz 9x9-es esetig magával a `c` -vel, felette a neki megfelelő betűvel.

`public static int ConvertCharToCellData(char ch)`
Egy karaktert alakít át a tárolási formának megfelelő integerré.

SudokuTable: (SudokuTable.cs)
`Icloneable` interfészt megvalósító, M*M-es `SudokuCell` tömböt tartalmazó adatszerkezet. A jelölttömbön felül nyilvántartja a meglévő adatok számát, továbbá egyéb leíró adatokat, melyek jellemzik a rejtvényt.

Függvényei:

```
public void setCellData(int i, int j, int c)
```
Beállítja i,j mezőben az adatot.

```
public int getCellData(int i, int j)
```
Visszaadja i,j mezőben található adatot.

```
public bool RemoveCandidate(int i, int j, int c)
```
Kitörli i,j mezőből a c jelöltet, és jelzi, hogy a törlés érvényes volt-e.

```
public bool DeleteCandidateFromRow(int x, int y, int candidate)
```
Kitörli a `candidate` jelöltet a `x, y` sorának mezőiből.

```
public bool DeleteCandidateFromColumn(int x, int y, int candidate)
```
Kitörli a `candidate` jelöltet a `x, y` oszlopának mezőiből.

```
public bool DeleteCandidateFromBlock(int x, int y, int candidate)
```
Kitörli a `candidate` jelöltet a `x, y` blokkjának mezőiből.

```
public bool DeleteCandidateFromThreeHouse(int x, int y, int
candidate)
```
Kitörli a `candidate` jelöltet a `x, y` szomszédaiból.

```
public void AddCandidateToRow(int x, int y, int candidate)
```
Hozzáadja a `candidate` jelöltet a `x, y` sorának minden mezőjéhez.

```
public void AddCandidateToColumn(int x, int y, int candidate)
```
Hozzáadja a `candidate` jelöltet a `x, y` oszlopának minden mezőjéhez.

```
public void AddCandidateToBlock(int x, int y, int candidate)
```
Hozzáadja a `candidate` jelöltet a `x, y` blokkjának minden mezőjéhez.

```
public void AddCandidateToThreeHouse(int x, int y, int candidate)
```
Hozzáadja a `candidate` jelöltet a `x, y` szomszédaihoz.

```
public bool ValidString(string str)
```
Ellenőrzi, hogy az adott string érvényes.

```
public bool LoadFromString(string str)
```
Feltölti a táblát az adott stringben található adatokkal.

```
public string SaveToString()
```
Előállítja az adott táblának adatait sorfolytonos leírásban.

`public string CandidateTable()`
Előállítja az adott táblának a jelölttábláját.

`public bool LoadFromSDXFile(string file)`
Feltölti a táblát a megadott elérésű sdx fájl adataival.

`public bool LoadFromSDXXFile(string file)`
Feltölti a táblát a megadott elérésű sdxx fájl adataival.

`public bool LoadFromSDKFile(string file)`
Feltölti a táblát a megadott elérésű sdk fájl adataival.

`public bool SaveToSDKFile(string file)`
Elmenti a táblát a megadott elérésű fájlba sdk formátumban.

`public bool SaveToSDXFile(string file)`
Elmenti a táblát a megadott elérésű fájlba sdx formátumban.

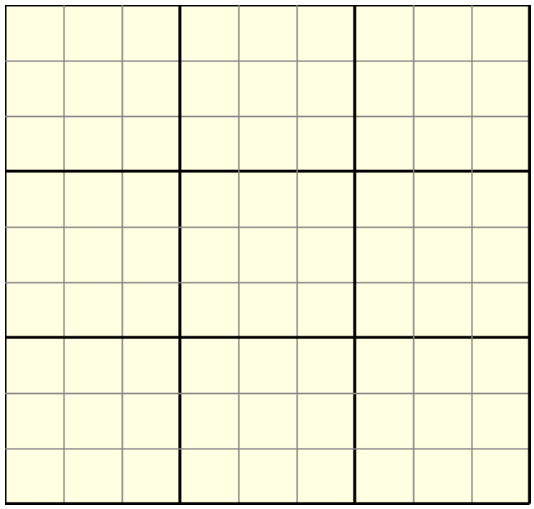
`public bool SaveToSDXXFile(string file)`
Elmenti a táblát a megadott elérésű fájlba sdxx formátumban.

IntPair: (IntPair.cs)
Két integer-t tartalmazó rekord.

```
namespace Sudoku
{
    struct IntPair
    {
        public IntPair(int i, int j)
        {
            this.i = i;
            this.j = j;
        }
        public int i;
        public int j;
    }
}
```

2. Vezérlők

SudokuControl: (SudokuControl.cs, SudokuControl.Designer.cs)
Egy `SudokuTable` típusú objektum szerkesztését teszi lehetővé. A mérete szabadon állítható.
Grafikus megjelenés:



Függvényei:

```
public void UpdateTable(SudokuTable t)
```
Új `SudokuTable` típusú objektumot fogad, melyet megjelenít.

```
public void Reset(int size)
```
Alapállapotra hozza a vezérlőt, a megadott méretben. (N értéket vár)

```
public void SelectCell(int i, int j)
```
Egy mezőt kijelölt.

```
public void UnSelectCell()
```
Leveszi egy mező kijelölését.

```
public void HighlightReset()
```
Kiüríti a kiemelt mezők listáját.

```
public void Highlight(int c, Color color)
```
Kiemeli a c -t jelöltként tartalmazó mezőket hátterük színének megváltoztatásával.

```
protected void OnTableChanged(object o, SudokuTableEventArgs e)
```
Kiváltja a `TableChanged` eseményt.

```
protected void OnWarningEvent(object o, string message)
```
Kiváltja a `WarningEvent` eseményt.

```
public void CopyImageToClipboard()
```
A tábla adott megjelenését fekete-fehér képként a vágólapra másolja.

```
public void DrawSudokuTable(Graphics g, Color color)
```
Megrajzolja a vezérlőt.

```
protected override void OnPaint(PaintEventArgs e)
```
Meghívja a `DrawSudokuTable` függvényt.

```
protected override void OnKeyPress(KeyPressEventArgs e)
```
A billentyűparancsokat dolgozza fel.

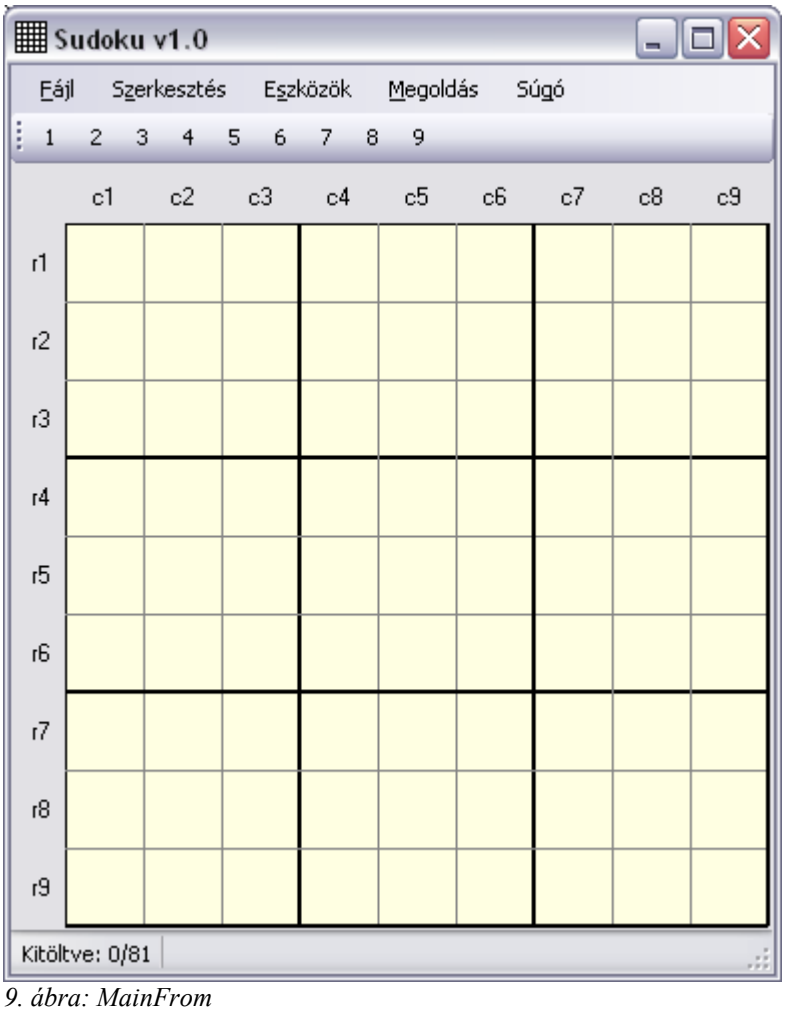
```
protected override bool ProcessDialogKey(Keys keyData)
```
A nyilak, illetve a `Delete` gomb lenyomását kezeli.

```
protected override void OnMouseDown(MouseEventArgs e)
```
Az egér kattintását kezeli.

3. Form-ok

MainForm: (MainForm.cs, MainForm.Designer.cs)
A főbb funkciók elérhetőségét biztosítja a felhasználónak.
Grafikus megjelenés:

MainForm vezérlői:

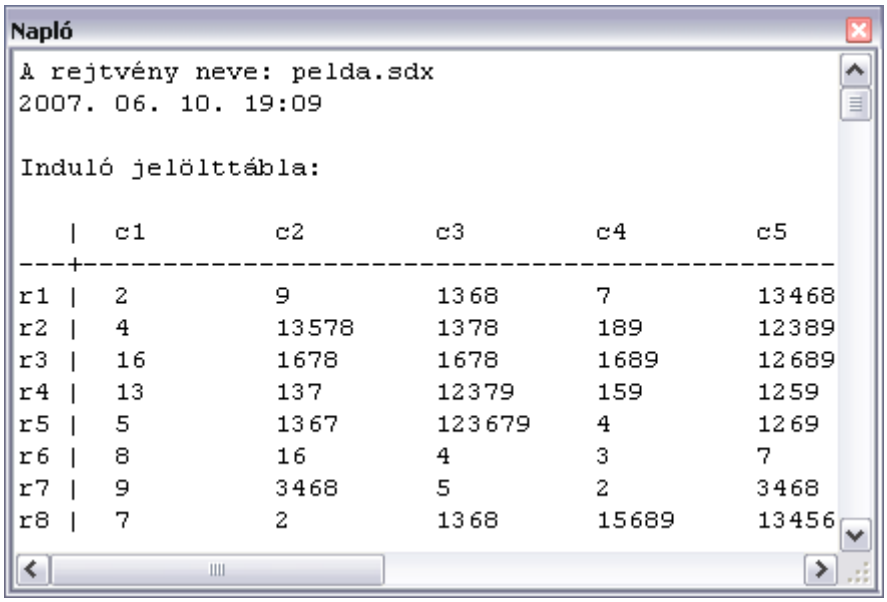


9. ábra: MainFrom

Vezérlő neve	*Típusa*	*Kezdeti értéke*
toolStrip1	ToolStrip	
mainmenu	MenuStrip	
statusStrip1	StatusStrip	
sudokuControl	SudokuControl	

A sudokuControl beállításakor a MainForm 2*M db Label típusú vezérlőt beállít és felcímkéz (r1,r2,...,c1,c2,...).
A statusStrip1 a sudokuControl kiemeléseit kezeli: az adott számra/betűre kattintva a sudokuControl kiemeli zöld háttérrel az adott jelöltet tartalmazó mezőket.

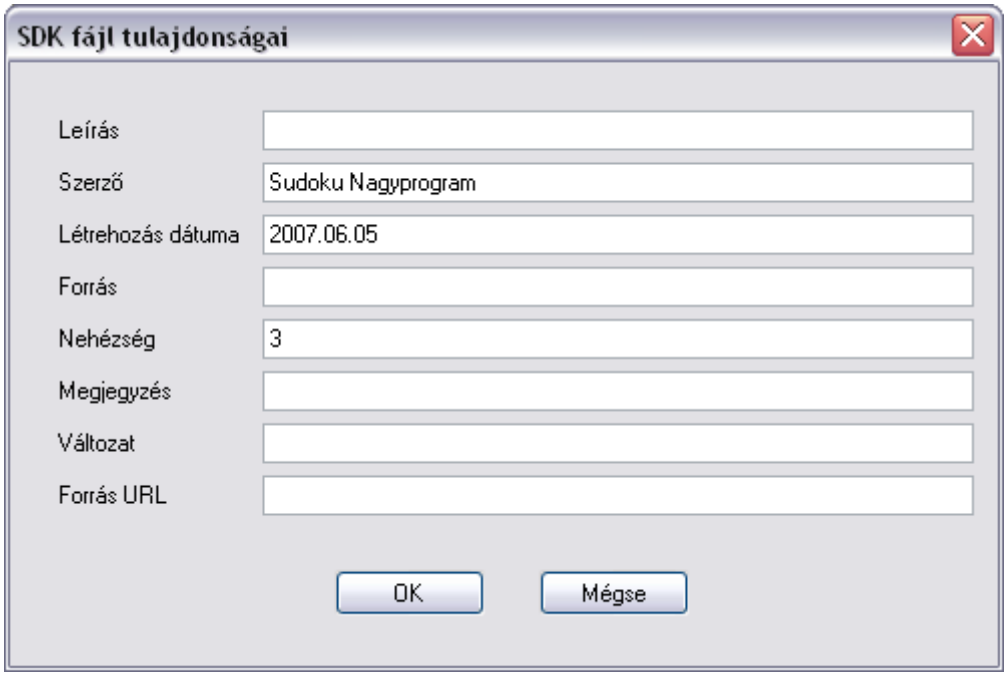
Report: (Report.cs, Report.Designer.cs)
Grafikus megjelenés:



10. ábra: Report

Vezérlő neve	*Típusa*	*Kezdeti értéke*
textBox1	TextBox	

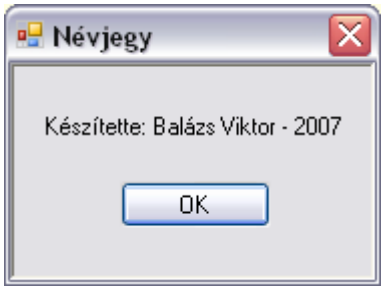
SetProperties: (SetProperties.cs, SetProperties.Designer.cs)
Grafikus megjelenés:



11. ábra: SetProperties

Vezérlő neve	*Típusa*	*Kezdeti értéke*
label1	Label	„Leírás”
label2	Label	„Szerző”
label3	Label	„Létrehozás dátuma”
label4	Label	„Forrás”
label5	Label	„Nehézség”
label6	Label	„Megjegyzés”
label7	Label	„Változat”
label8	Label	„Forrás URL”
desctb	TextBox	
authb	TextBox	
createtb	TextBox	
sourcetb	TextBox	
difftb	TextBox	
commtb	TextBox	
vartb	TextBox	
scurltb	TextBox	
okbtn	Button	„OK”
cancelbtn	Button	„Mégse”

about: (about.cs, about.Designer.cs)



12. ábra: about

Vezérlő neve	*Típusa*	*Kezdeti értéke*
label1	Label	„Készítette: Balázs Viktor - 2007”
button1	Button	„OK”

GeneratorForm: (GeneratorForm.cs, GeneratorForm.Designer.cs)
A SudokuGenerator osztály egy példányának állapotát mutatja, melyet külön szálon futtat egy BackgroundWorker objektum segítségével.

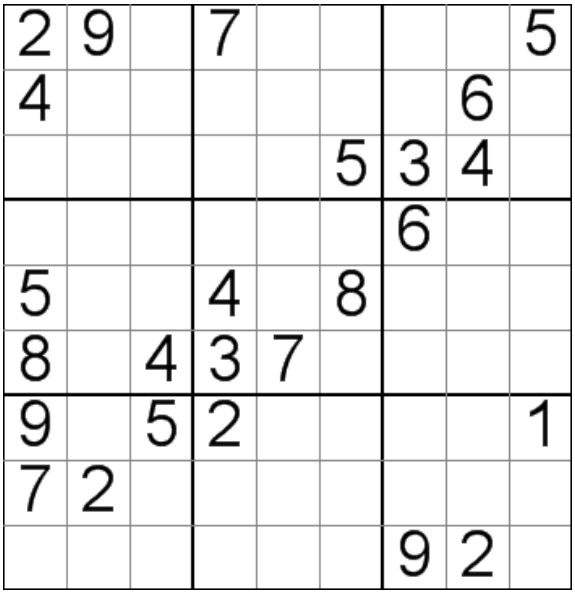
13. ábra: GeneratorForm: rejtvény generálása

Vezérlő neve	*Típusa*	*Kezdeti értéke*
label1	Label	„Kipróbált tábla:”
label2	Label	„0/0”
label3	Label	„Aktuális elemszám:”
label4	Label	„0”
progressBar1	ProgressBar	0
button1	Button	„Leállítás”

Ha egyértelműen megoldható rejtvényt talált a SudokuGenerator, akkor megpróbál egyszerűsíteni az egyes adatok egyenkénti elhagyásával.

14. ábra: Egyszerűsítés

4. **Fájlformátumok**



15. ábra: Példa

Az alábbi rejtvény tárolási formái:

SDK: 9x9-es tábla leírására alkalmas, a számok az adatok, a „.” a üres mező. Az SDK formátum tartalmazhat megjegyzéseket az adott feladványról.

```
#ASudoku Nagyprogram
#B2007.06.05
#L3
29.7....5
4......6.
.....534.
......6..
5..4.8...
8.437....
9.52....1
72.......
......92.
```

SDX: 9x9-es tábla leírására alkalmas, a kapcsos zárójelben álló számok a jelöltek, a zárójelen kívüli számok az adatok.

```
2 9 {1368} 7 {13468} {1346} {18} {18} 5
4 {13578} {1378} {189} {12389} {1239} {1278} 6 {2789}
{16} {1678} {1678} {1689} {12689} 5 3 4 {2789}
{13} {137} {12379} {159} {1259} {129} 6 {135789} {234789}
5 {1367} {123679} 4 {1269} 8 {127} {1379} {2379}
8 {16} 4 3 7 {1269} {125} {159} {29}
9 {3468} 5 2 {3468} {3467} {478} {378} 1
7 2 {1368} {15689} {1345689} {13469} {458} {358} {3468}
{136} {13468} {1368} {1568} {134568} {13467} 9 2 {34678}
```

SDXX: N^2xN^2-es tábla leírására alkalmas, a kapcsos zárójelben álló számok a jelöltek vesszővel elválasztva, a zárójelen kívüli számok az adatok. A vessző lényege, hogy 9x9-es tábla felett egyértelmű legyen a jelöltek listája.

```
2 9 {1,3,6,8} 7 {1,3,4,6,8} {1,3,4,6} {1,8} {1,8} 5
4 {1,3,5,7,8} {1,3,7,8} {1,8,9} {1,2,3,8,9} {1,2,3,9} {1,2,7,8} 6 {2,7,8,9}
{1,6} {1,6,7,8} {1,6,7,8} {1,6,8,9} {1,2,6,8,9} 5 3 4 {2,7,8,9}
{1,3} {1,3,7} {1,2,3,7,9} {1,5,9} {1,2,5,9} {1,2,9} 6 {1,3,5,7,8,9} {2,3,4,7,8,9}
5 {1,3,6,7} {1,2,3,6,7,9} 4 {1,2,6,9} 8 {1,2,7} {1,3,7,9} {2,3,7,9}
8 {1,6} 4 3 7 {1,2,6,9} {1,2,5} {1,5,9} {2,9}
9 {3,4,6,8} 5 2 {3,4,6,8} {3,4,6,7} {4,7,8} {3,7,8} 1
7 2 {1,3,6,8} {1,5,6,8,9} {1,3,4,5,6,8,9} {1,3,4,6,9} {4,5,8} {3,5,8} {3,4,6,8}
{1,3,6} {1,3,4,6,8} {1,3,6,8} {1,5,6,8} {1,3,4,5,6,8} {1,3,4,6,7} 9 2 {3,4,6,7,8}
```

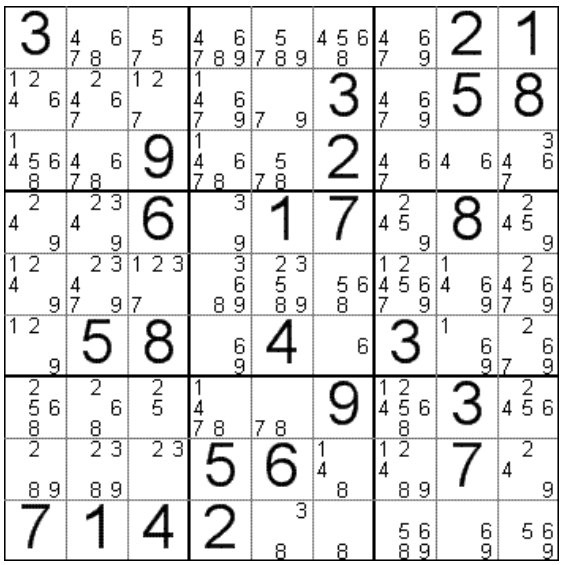
5. Algoritmusok

1. Polinomiális

Az alábbi polinomiális algoritmusok – az itteni megvalósításuk szerint - két fő részre bonthatók. Az első részben adatokat gyűjtenek a tábláról: meghatározzák, hol tudnak valami hasznosat csinálni, és ennek eredményét egy vagy több sorban tárolják. A második részben a sorból kiveszik az elemeket, és felhasználják azokat.
Az algoritmusok felbontásának lényege, hogy lépésenként lehessen vizsgálni a megoldást.
A közös szerkezet miatt egy közös Sudoku::Algorithm osztályból származnak.

NakedSingle

Az algoritmus megvizsgálja, hogy mely mezőkben lehetséges egyedül egy jelölt. Ha talál ilyet, akkor beírja az adott jelöltet a mezőbe adatként, és ezután kihúzza az adott mező $3*N^2-2*N-1$ szomszédjának jelöltjei közül az adott számot.
Példa:

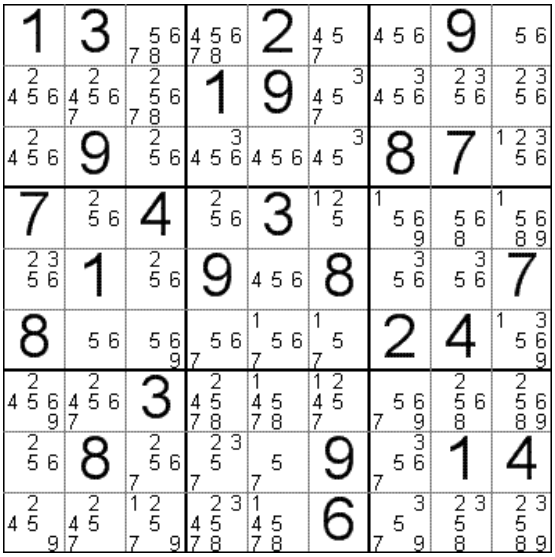


16. ábra: r9c6-ban a 8 Naked Single

HiddenSingle

Az algoritmus megvizsgálja, hogy mely jelölt szerepel pontosan egyszer egy házban (sorban, oszlopban,blokkban). Ha talál ilyet, akkor beírja az adott jelöltet a mezőbe adatként, és ezután kihúzza az adott mező $3*N^2-2*N-1$ szomszédjának jelöltjei közül az adott számot.

Példa:



17. ábra: r3c9-ben az 1 Hidden Single sor és blokk szerint

Intersection

Ha egy háznak csak egy másik házzal alkotott metszetében található meg egy adott jelölt, akkor a másik ház metszeten kívüli részéből törölhető a jelölt.

Két esetre felbontható **[7-Techniques]**:

Jelmagyarázat:

`X` **:** azon mezők, ahol a vizsgált jelölt megtalálható

`-` **:** azon mezők, ahol a vizsgált jelölt nem található meg

`*` **:** azon mezők, ahonnan megkísérelhetjük a vizsgált jelölt kihúzását

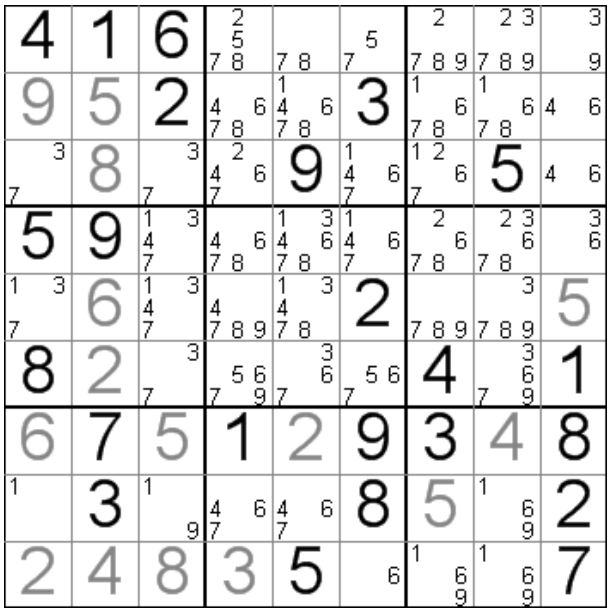
1. eset:

```
.-------.-------.-------.
| * * * | * * * | X X X |
|       |       | - - - |
|       |       | - - - |
'-------'-------'-------'
```

2. eset:

```
.-------.-------.-------.
| - - - | - - - | X X X |
|       |       | * * * |
|       |       | * * * |
'-------'-------'-------'
```

A megvalósított algoritmus ezt a két esetet vizsgálja külön-külön, jelöltekre lebontva.
1.eset: Végigmegy a táblán blokkokra lebontva, és megnézi, hogy az adott jelölt egy blokkban lévő összes előfordulása sorban vagy oszlopban található-e.
Példa:

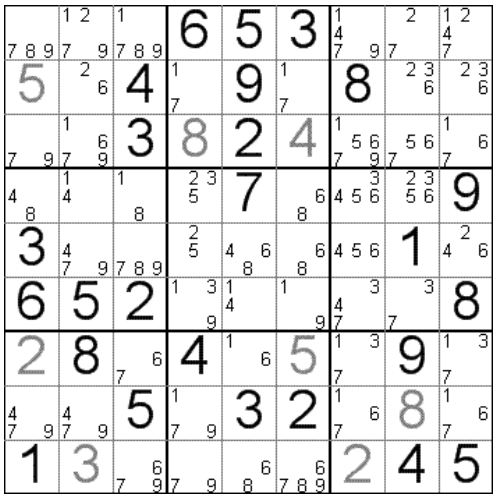


18. ábra: 7-es az [1,1] blokkban csak az r3 sorban fordul elő, tehát a sor többi részéből törölhető

2.eset: A táblát kis téglalapokra bontja vízszintesen és függőlegesen (1xN-es illetve Nx1-esekre). Ha a vizsgált jelölt szerepel egy mezőben, akkor a hozzá tartozó kis téglalapot megjelöli. A tábla végére érve ha létezik olyan azonos sorban (oszlopban) elhelyezkedő N téglalap, melyek közül csak egy van megjelölve, a megjelölt téglalap blokkjából törölhető a jelölt.

NakedPair

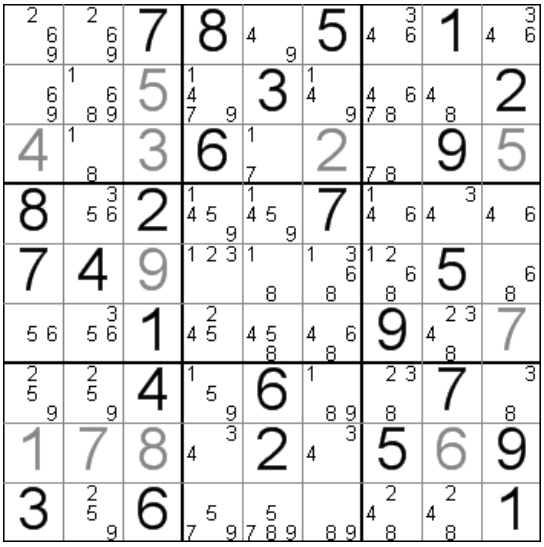
Van-e két olyan mező egy házban, melyekben csak két jelölt lehetséges, és ezek a jelöltek megegyeznek-e. Ha ilyen van, akkor az adott két jelölt csak ebben a két mezőben fordulhat elő, tehát a házban a többi helyről törölhető.
Az algoritmus végigmegy sorfolytonosan a táblán, ha két jelöltet tartalmazó mezőt talál, akkor szomszédjai közül keres olyat, amely ugyanazt a két jelöltet tartalmazza.
Példa:



19. ábra: Naked Pair: r4c6 és r5c6-ban megegyezik a két jelölt, ezért r5c5-ből a 6, 8 és r9c6-ból a 6, 8 kihúzható

NakedTriple

Ha egy házban három mezőben található jelöltek uniójának számossága három, akkor a három jelölt csak ezekben a mezőkben fordulhat elő, a ház többi mezőjéből törölhető.
Az algoritmus végigmegy sorfolytonosan a táblán, ha három jelöltet tartalmazó mezőt talál, akkor szomszédjai közül keres olyat, amely ugyanannak a három jelöltnek a részhalmazát tartalmazza.



20. ábra: r1c1, r1c2, r2c1 miatt a 6 és 9 törölhető r2c2 mezőből

2. Dancing Links

Donald E. Knuth által leírt fedési problémát megoldó Dancing Links algoritmus **[4-Knuth2000]** megvalósítása. A sudoku feladványt először átalakítjuk az **[2-Exact2007]** oldalon található módszer szerint fedési problémára.

Felhasznált adatszerkezet: négy irányba láncolt lista.
Egy csúcs felépítése:

```
class Node
{
    public Node left;
    public Node right;
    public Node up;
    public Node down;
    public Header head;
    public int row;

    public Node() : this(-1)
    { }

    public Node(int row)
    {
        left = this;
        right = this;
        up = this;
        down = this;
```

```
            this.row = row;
            head = null;
        }
}
```

Ezekből a csúcsokból épül fel a hiányos mátrix, melyben az oszlopok nyilvántartják, hogy hány elemük van. Erre egy, a Node osztályból származtatott Header osztály szolgál.

```
class Header : Node
{
    public int size;
    public int name;

    public Header(int name)
    {
        this.name = name;
        this.size = 0;
        this.head = this;
    }
}
```

Az algoritmus lépései:
BuildTable(): A hiányos mátrix keretének felépítése: oszlopokra, illetve sorokra mutató Header-ök és Node-ok.
SetStartingGrid(): A sudoku tábla adatai szerint feltölti a hiányos mátrixot (a jelölteket nem veszi figyelembe).**[2-Exact2007]**
A sudoku négy egyszerű szabálya szerint alakítjuk át a rejtvényt fedési problémára:

- Minden mező egy adatot tartalmazhat.
- Minden sorban minden szám csak egyszer fordulhat elő.
- Minden oszlopban minden szám csak egyszer fordulhat el.
- Minden blokkban minden szám csak egyszer fordulhat elő.

Minden szabályhoz egy M*M*M sorból (sorok*oszlopok*lehetséges értékek) és M*M oszlopból (mezők száma) álló mátrix feleltethető meg. A négy szabály így egy nagy M^3*(4*M^2)-es mátrixot határoz meg.
Egy 9x9-es sudokuban ez az átalakítás egy 9^3*(4*9^2)-es mátrixot jelent.
Egy ilyen mátrix megtalálható ezen a helyen:
http://www.stolaf.edu/people/hansonr/sudoku/exactcovermatrix.htm
Solve(): Megoldja a feladatot, és közben számolja, hogy hány megoldást talált. Ha egynél több van, akkor leáll.
WriteBackToTable(): Ha legalább egy találat van, visszaírja az adatokat a sudokutáblába.

6. Algoritmusok szervezése

A polinomiális algoritmusok egy `Algorithm` típusú tömbben helyezkednek el.

```
Algorithm[] algorithms = new Algorithm[5];
algorithms[0] = new NakedSingle();
algorithms[1] = new HiddenSingle();
algorithms[2] = new Intersection();
algorithms[3] = new NakedPair();
algorithms[4] = new NakedTriple();
```

A ciklus elindul ezen a tömbön, minden algoritmusnak adva egy esélyt. Egy algoritmus addig működik, amíg tud lépni. Ha valamelyik tudott lépni, és még nincs kitöltve a rejtvény, akkor visszaadja a vezérlést az első algoritmusnak.

Ha egyik polinomiális algoritmus sem tudott lépni, és a rejtvény még nincs kitöltve, akkor a DancingLinks kapja meg a feladatot, ami vagy megoldja a sudokut, vagy jelzi a hibát.

7. **Rejtvény generálása**

SudokuGenerator: (SudokuGenerator.cs)
A DancingLinks algoritmust felhasználva egyértelműen megoldható rejtvényt generál.
Ehhez véletlenszerűen hozzávesz egy elemet a táblához az AddNewRandomData() függvénnyel, és ellenőriz a Dancing Links algoritmussal:

- Ha a tábla egyértelműen megoldható akkor megpróbálja redukálni a Reduce() függvény segítségével.
- Ha a tábla ellentmondásos, eltávolít belőle egy elemet.
- Ha a táblának több megoldása van, akkor hozzávesz egy újabb elemet és újra ellenőriz.

Az algoritmus befejeződik, ha:

- ➔ Egyértelműen megoldható rejtvényt talál,
- ➔ vagy elér egy kritikus próbálkozásszámot,
- ➔ vagy a felhasználó leállítja.

Függvények:
`Reduce`():
Végigmegy a tábla elemein. Az elemeket egyesével elhagyja, miközben az így kialakult táblát ellenőrzi:

- Ha a tábla továbbra is egyértelműen megoldható, akkor veszi a következő elemet.
- Egyébként visszarakja a táblába a választott elemet, és csak ezután megy tovább a következő elemre.

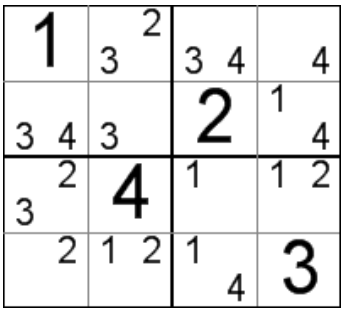
`AddNewRandomData`():
Hozzávesz egy elemet a táblához, és a szomszédaiból kihúzza az adott jelöltet. Ha eközben valamelyik szomszédban a jelöltek száma 0-ra ér, akkor törli az elemet a választott mezőből, és a szomszédokban visszaállítja a jelölteket, majd új elemet választ.

Megjegyzés:
Bár az algoritmus univerzális, a program mégsem támogatja a 25x25-ös rejtvények megoldását, mivel ezzel a módszerrel nagyon sokáig tartana.

8. **Tesztelési terv**

A programot különböző méretű, és különböző bonyolultságú érvényes, illetve megoldhatatlan feladványokkal teszteltem. A végeredményt - ahol lehetett - összehasonlítottam a SudoCue **[2]** program által előállított végeredménnyel (erre csak a 9x9-es esetben volt lehetőség).

4x4-es tábla jelöltekkel (2A1-1-Puzzle.sdxx) [Iványi2006]:



Induló jelölttábla:

```
    |  c1    c2    c3    c4
---+----------------------
r1  |  1     23    34    4
r2  |  34    3     2     14
r3  |  23    4     1     12
r4  |  2     12    14    3
```

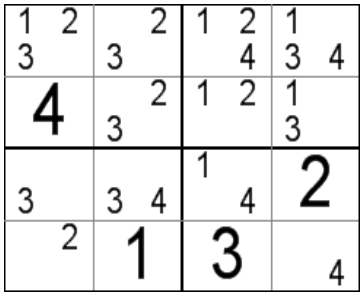
A program az alábbi naplót készítette:

```
1. lépés: r1c4 <-- 4 (Naked Single)
2. lépés: r2c2 <-- 3 (Naked Single)
3. lépés: r3c3 <-- 1 (Naked Single)
4. lépés: r4c1 <-- 2 (Naked Single)
5. lépés: r1c3 <-- 3 (Naked Single)
6. lépés: r2c4 <-- 1 (Naked Single)
7. lépés: r2c1 <-- 4 (Naked Single)
8. lépés: r1c2 <-- 2 (Naked Single)
9. lépés: r3c4 <-- 2 (Naked Single)
10. lépés: r4c3 <-- 4 (Naked Single)
11. lépés: r4c2 <-- 1 (Naked Single)
12. lépés: r3c1 <-- 3 (Naked Single)
A megoldás:

   |  c1   c2   c3   c4
---+---------------------
r1 |  1    2    3    4
r2 |  4    3    2    1
r3 |  3    4    1    2
r4 |  2    1    4    3


Futásidő: 10,0144ms
```

Véletlen rejtvény



Induló jelölttábla:

```
    |  c1    c2    c3    c4
---+----------------------
r1  |  123   23    124   134
r2  |  4     23    12    13
r3  |  3     34    14    2
r4  |  2     1     3     4
```

A program az alábbi naplót készítette:

```
1. lépés: r3c1 <-- 3 (Naked Single)
2. lépés: r4c1 <-- 2 (Naked Single)
3. lépés: r4c4 <-- 4 (Naked Single)
4. lépés: r3c2 <-- 4 (Naked Single)
5. lépés: r1c1 <-- 1 (Naked Single)
6. lépés: r3c3 <-- 1 (Naked Single)
7. lépés: r1c4 <-- 3 (Naked Single)
8. lépés: r2c3 <-- 2 (Naked Single)
9. lépés: r1c2 <-- 2 (Naked Single)
10. lépés: r2c4 <-- 1 (Naked Single)
11. lépés: r2c2 <-- 3 (Naked Single)
12. lépés: r1c3 <-- 4 (Naked Single)
A megoldás:

   |  c1   c2   c3   c4
---+---------------------
r1 |  1    2    4    3
r2 |  4    3    2    1
r3 |  3    4    1    2
r4 |  2    1    3    4


Futásidő: 0ms
```

Hibás tábla:

1	4	2 3 4	4
3 4	2	1	4
4	3	4	2
2 4	1 4	4	3

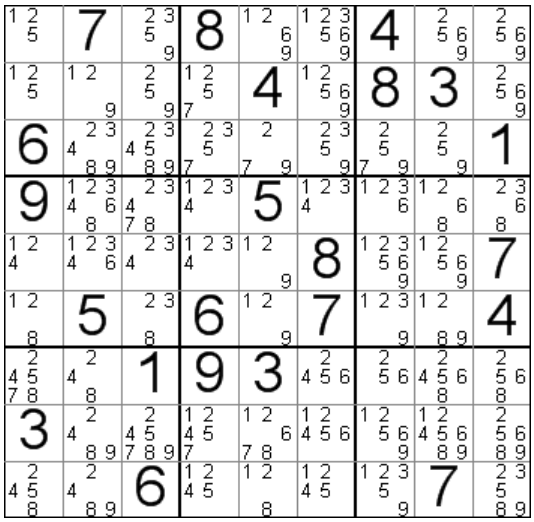
Induló jelölttábla:

```
   |  c1    c2    c3    c4
---+-----------------------
r1 |  1     4     234   4
r2 |  34    2     1     4
r3 |  4     3     4     2
r4 |  24    14    4     3
```

0. lépés: Hiba: r1c2 <-- 4 után r1c4 kihúzásakor (Naked Single)

Futásidő: 0ms

9x9-es tábla: 3A5-1.sdx (a megoldáshoz felhasznált algoritmusok: NakedSingle, HiddenSingle, Intersection, Naked Triple)



Induló jelölttábla:

	c1	c2	c3	c4	c5	c6	c7	c8	c9
r1	125	7	2359	8	1269	123569	4	2569	2569
r2	125	129	259	1257	4	12569	8	3	2569
r3	6	23489	234589	2357	279	2359	2579	259	1
r4	9	123468	23478	1234	5	1234	1236	1268	2368
r5	124	12346	234	1234	129	8	123569	12569	7
r6	128	5	238	6	129	7	1239	1289	4
r7	24578	248	1	9	3	2456	256	24568	2568
r8	3	2489	245789	12457	12678	12456	12569	1245689	25689
r9	2458	2489	6	1245	128	1245	12359	7	23589

A program az alábbi naplót készítette:

```
1. lépés: r2c4 <-- 7 sor szerint (Hidden Single)
2. lépés: r4c3 <-- 7 sor  blokk szerint (Hidden Single)
3. lépés: r7c1 <-- 7 sor  oszlop szerint (Hidden Single)
4. lépés: r3c7 <-- 7 sor  oszlop  blokk szerint (Hidden Single)
5. lépés: r8c5 <-- 7 sor  oszlop szerint (Hidden Single)
6. lépés: r1c5 <-- 6 oszlop szerint (Hidden Single)
7. lépés: r9c5 <-- 8 oszlop  blokk szerint (Hidden Single)
8. lépés: r2c9 <-- 6 sor  blokk szerint (Hidden Single)
9. lépés: r6c1 <-- 8 oszlop szerint (Hidden Single)
```

	c1	c2	c3	c4	c5	c6	c7	c8	c9
r1	125	7	2359	8	6	12359	4	259	259
r2	125	129	259	7	4	1259	8	3	6
r3	6	23489	234589	235	29	2359	7	259	1
r4	9	12346	7	1234	5	1234	1236	1268	238
r5	124	12346	234	1234	129	8	123569	12569	7
r6	8	5	23	6	129	7	1239	129	4
r7	7	248	1	9	3	2456	256	24568	258
r8	3	2489	24589	1245	7	12456	12569	1245689	2589
r9	245	249	6	1245	8	1245	12359	7	2359

```
10. lépés: A 1 a [1,4] blokkban csak a c6 oszlopban van: 1 kihúzása r4c6, r8c6, r9c6 helyekről
(Intersection)
11. lépés: A 1 a c5 oszlopnak csak a [4,4] blokkba eső részében lehet: 1 kihúzása r4c4, r5c4
helyekről (Intersection)
12. lépés: A 9 a [4,4] blokkban csak a c5 oszlopban van: 9 kihúzása r3c5 helyről (Intersection)
13. lépés: A 1 a [7,4] blokkban csak a c4 oszlopban van: 1 kihúzása r4c4, r5c4 helyekről
(Intersection)
```

	c1	c2	c3	c4	c5	c6	c7	c8	c9
r1	125	7	2359	8	6	12359	4	259	259
r2	125	129	259	7	4	1259	8	3	6
r3	6	23489	234589	235	2	2359	7	259	1
r4	9	12346	7	234	5	234	1236	1268	238
r5	124	12346	234	234	129	8	123569	12569	7
r6	8	5	23	6	129	7	1239	129	4
r7	7	248	1	9	3	2456	256	24568	258
r8	3	2489	24589	1245	7	2456	12569	1245689	2589
r9	245	249	6	1245	8	245	12359	7	2359

14. lépés: r3c5 <-- 2 (Naked Single)

	c1	c2	c3	c4	c5	c6	c7	c8	c9
r1	125	7	2359	8	6	1359	4	259	259
r2	125	129	259	7	4	159	8	3	6
r3	6	3489	34589	35	2	359	7	59	1
r4	9	12346	7	234	5	234	1236	1268	238
r5	124	12346	234	234	19	8	123569	12569	7
r6	8	5	23	6	19	7	1239	129	4
r7	7	248	1	9	3	2456	256	24568	258
r8	3	2489	24589	1245	7	2456	12569	1245689	2589
r9	245	249	6	1245	8	245	12359	7	2359

15. lépés: A 2 a [1,7] blokkban csak a r1 sorban van: 2 kihúzása r1c1, r1c3 helyekről (Intersection)

	c1	c2	c3	c4	c5	c6	c7	c8	c9
r1	15	7	359	8	6	1359	4	259	259
r2	125	129	259	7	4	159	8	3	6
r3	6	3489	34589	35	2	359	7	59	1
r4	9	12346	7	234	5	234	1236	1268	238
r5	124	12346	234	234	19	8	123569	12569	7
r6	8	5	23	6	19	7	1239	129	4
r7	7	248	1	9	3	2456	256	24568	258
r8	3	2489	24589	1245	7	2456	12569	1245689	2589
r9	245	249	6	1245	8	245	12359	7	2359

16. lépés: r3c6, r3c4, r3c8 miatt: r3c2 -> 3, r3c2 -> 9, r3c3 -> 3, r3c3 -> 5, r3c3 -> 9 kihúzható (Naked Triple)

	c1	c2	c3	c4	c5	c6	c7	c8	c9
r1	15	7	359	8	6	1359	4	259	259
r2	125	129	259	7	4	159	8	3	6
r3	6	48	48	35	2	359	7	59	1
r4	9	12346	7	234	5	234	1236	1268	238
r5	124	12346	234	234	19	8	123569	12569	7
r6	8	5	23	6	19	7	1239	129	4
r7	7	248	1	9	3	2456	256	24568	258
r8	3	2489	24589	1245	7	2456	12569	1245689	2589
r9	245	249	6	1245	8	245	12359	7	2359

17. lépés: r1c3 <-- 3 blokk szerint (Hidden Single)
18. lépés: r6c7 <-- 3 sor szerint (Hidden Single)
19. lépés: r9c9 <-- 3 sor oszlop blokk szerint (Hidden Single)

	c1	c2	c3	c4	c5	c6	c7	c8	c9
r1	15	7	3	8	6	159	4	259	259
r2	125	129	259	7	4	159	8	3	6
r3	6	48	48	35	2	359	7	59	1
r4	9	12346	7	234	5	234	126	1268	28
r5	124	12346	24	234	19	8	12569	12569	7
r6	8	5	2	6	19	7	3	129	4
r7	7	248	1	9	3	2456	256	24568	258
r8	3	2489	24589	1245	7	2456	12569	1245689	2589
r9	245	249	6	1245	8	245	1259	7	3

20. lépés: r6c3 <-- 2 (Naked Single)
21. lépés: r5c3 <-- 4 (Naked Single)
22. lépés: r5c1 <-- 1 (Naked Single)
23. lépés: r3c3 <-- 8 (Naked Single)
24. lépés: r5c5 <-- 9 (Naked Single)
25. lépés: r1c1 <-- 5 (Naked Single)

```
26. lépés: r3c2 <-- 4 (Naked Single)
27. lépés: r6c5 <-- 1 (Naked Single)
28. lépés: r2c1 <-- 2 (Naked Single)
29. lépés: r2c3 <-- 9 (Naked Single)
30. lépés: r6c8 <-- 9 (Naked Single)
31. lépés: r9c1 <-- 4 (Naked Single)
32. lépés: r2c2 <-- 1 (Naked Single)
33. lépés: r8c3 <-- 5 (Naked Single)
34. lépés: r1c8 <-- 2 (Naked Single)
35. lépés: r3c8 <-- 5 (Naked Single)
36. lépés: r2c6 <-- 5 (Naked Single)
37. lépés: r1c9 <-- 9 (Naked Single)
38. lépés: r3c4 <-- 3 (Naked Single)
39. lépés: r5c8 <-- 6 (Naked Single)
40. lépés: r9c6 <-- 2 (Naked Single)
41. lépés: r1c6 <-- 1 (Naked Single)
42. lépés: r3c6 <-- 9 (Naked Single)
43. lépés: r5c4 <-- 2 (Naked Single)
44. lépés: r5c2 <-- 3 (Naked Single)
45. lépés: r9c2 <-- 9 (Naked Single)
46. lépés: r5c7 <-- 5 (Naked Single)
47. lépés: r4c4 <-- 4 (Naked Single)
48. lépés: r4c2 <-- 6 (Naked Single)
49. lépés: r9c7 <-- 1 (Naked Single)
50. lépés: r4c6 <-- 3 (Naked Single)
51. lépés: r8c4 <-- 1 (Naked Single)
52. lépés: r9c4 <-- 5 (Naked Single)
53. lépés: r4c7 <-- 2 (Naked Single)
54. lépés: r4c9 <-- 8 (Naked Single)
55. lépés: r7c7 <-- 6 (Naked Single)
56. lépés: r4c8 <-- 1 (Naked Single)
57. lépés: r8c9 <-- 2 (Naked Single)
58. lépés: r7c6 <-- 4 (Naked Single)
59. lépés: r8c7 <-- 9 (Naked Single)
60. lépés: r8c2 <-- 8 (Naked Single)
61. lépés: r7c9 <-- 5 (Naked Single)
62. lépés: r7c8 <-- 8 (Naked Single)
63. lépés: r8c6 <-- 6 (Naked Single)
64. lépés: r8c8 <-- 4 (Naked Single)
65. lépés: r7c2 <-- 2 (Naked Single)
A megoldás:
```

	c1	c2	c3	c4	c5	c6	c7	c8	c9
r1	5	7	3	8	6	1	4	2	9
r2	2	1	9	7	4	5	8	3	6
r3	6	4	8	3	2	9	7	5	1
r4	9	6	7	4	5	3	2	1	8
r5	1	3	4	2	9	8	5	6	7
r6	8	5	2	6	1	7	3	9	4
r7	7	2	1	9	3	4	6	8	5
r8	3	8	5	1	7	6	9	4	2
r9	4	9	6	5	8	2	1	7	3

```
Futásidő: 30,0432ms
```

További tesztelési eredmények a mellékelt CD-n találhatóak a „Teszteles” könyvtárban.

Megjegyzés a futásidő méréséről:

A programot egy 1.6GHz-es Pentium M processzoron futtattam Windows XP Professional operációs rendszerrel. A futásidők rendre 0ms (főleg kis méretű rejtvényeknél), illetve 10,0144ms többszörösei voltak. Próbáltam a program prioritását változtatni, hátha a Windows ütemezése okozza ezt a jelenséget, de az érték nem változott.
További megfigyelés: a program indításakor és az első megoldás futtatásakor megfigyelhető a JIT-fordító (just-in-time) működése: az első rejtvény első megoldása lassabb, mintha ugyanazt a rejtvényt később oldanám meg.

9. **Forráskód**

A program Microsoft Visual Studio 2005 alatt készült C# nyelven. A teljes forráskód a mellékelt CD-n található meg.

10. **Fejlesztési lehetőségek**

A program úgy lett elkészítve, hogy a továbbiakban könnyen bővíthető legyen új algoritmusokkal. Ehhez csak az algrithm tömbhöz kell hozzáadni az Algorithm osztályból származó új algoritmust. A modulokra (dll-ekre) bontás elősegíti más programokban az adattípusok (SudokuTable, SudokuCell) és a vezérlő osztály egyszerű felhasználását (SudokuControl) a forráskód ismerete nélkül.

11. **Verziójegyzék**

A következő iterációs lépéseken keresztül érte el a program a jelenlegi formáját.

1. SudokuCell és SudokuTable oszály létrehozása.
2. SudokuControl vezérlő létrehozása.
3. Absztrakt Algorithm osztály és NakedSingle algoritmus megvalósítása.
4. Algoritmusszervezés kialakítása.
5. HiddenSingle, Intersection algoritmusok megvalósítása, SDK, SDX fájlformátum bevezetése.
6. Dancing links algoritmus és rejtvénygenerálás.
7. NakedPair, NakedTriple algoritmus megvalósítása, SDXX univerzális formátum bevezetése.
8. A következő lépés meghatározása a lépés tényleges megtétele előtt.
9. Hibák jelzése.

IV. Irodalomjegyzék

[1-Albert2004] *Albert István*: A .NET Framework és programozása. Szak Kiadó Kft., Bicske, 2004, ISBN 963 9131 62 8.

[2-Exact2007] Exact cover (Sudoku átalakítása fedési problémára). http://en.wikipedia.org/wiki/Exact_cover. Letöltve: 2007. június 12.

[3-Iványi2007] Iványi Antal: *Módszeres sudoku*. Kézirat. Budapest, 2007.

[4-Knuth2000] Donald Ervin Knuth: Dancing links. In: *Millenial Perspectives in Computer Science* (edited by Jim Davies, Bill Roscoe and Jim Woodcock). Palgrave, 2000, 187−214. http://www-cs-faculty.stanford.edu/~knuth/preprints.html

[5-Logic] Sudoku Solver by Logic (sudoku megoldó program). http://www.sudokusolver.co.uk/step.html/ .

[6-SudoCue] Ruud van der Werl: Sudoku file formats (.sdk, .sdx), SudoCue program. http://www.sudocue.net/

[7-Techniques] Ruud van der Werl: Solving techniques. http://www.sudopedia.org.