Name of Allah, Most Gracious, Most Merciful

E-Mail: Hossein Fereidooni

It was the best motivation for writing this article.

E-Mail: hosein.fereidooni@gmail.com

Hossein Fereidooni

Solving Sudoku Using Informed Search Algorithms

E-Mail: hosein.fereidooni@gmail.com

Introduction

In computer systems and in the third millennium , The speed is very important . Therefore , the algorithm attempts to design systems that will increase efficiency and The speed . In this important role in the computer games are entertaining people . The design of this game most of the algorithms used in artificial intelligence and the The speed mutual accountability between users and systems is important . Sudoku is one of the game now has its particular fans .

This game simply because of the repeated failure to produce high numbers of permutations and variations in the size of the place is special.

As an integral part of these games have been in news papers and journals.

Many algorithms have been developed for the production of Othello and Backtracking recursive types that can be pointed and informed search, and more. Implement these algorithms , each has its limits.

Program presented in this paper is to incomplete a Sudoku with a lack of resolve in the input table and provide a unique solution. The output of this program is one-hundredth The speed milliseconds. This time , the system varies with different structures. But total code and provided solutions to the competitive and has good The speed.

Examples of solutions implemented in different times and this is visible.

It speeds up the code using the search methods like binary search can be improved.

Aim

This article has tried using an array of three-dimensional array of values to help the candidate to be kept informed by the corresponding table, complete with its original element be in place. Original Sudoku solving the following three conditions:

1) The value of each column is unique.
2) The value of each row is unique.
3) the amount per square N * N is a unique internal.

Note: The uniqueness of the meaning of Sudoku irregular, but between one and N is non-repetitive . (In this code, the full two-dimensional array of N = 3, and Table 81 has been assumed).
Each search is evaluated. Using this technique ensures the uniqueness of each table.

Informed by what?

The use of algorithms for solving various problems to solve every problem is the hardest part. The algorithms informed use of different solutions to one of the best options for solving problems with limited visibility and space is finite . Provided that the appropriate selection method can have instant access to your choice. Sudoku short finite set of numbers is repeated with a combination of the above conditions are in place. To this

E-Mail: hosein.fereidooni@gmail.com

end the two-dimensional array can be used for display space. Due to the unique numbers to this point will be replaced Space needed in the algorithm is considerable space And even arrays with N = 4,5, 6 to easily use this method. Recommended to solve the N side of the table with an infinite desire to be avoided . Because there will be enough space for the auxiliary array. This algorithm is derived from the same characteristic The speed.

Inappropriate and unnecessary to repeat it in other programs and remove the The speed increases.

informed search algorithms is presented in the chart below:

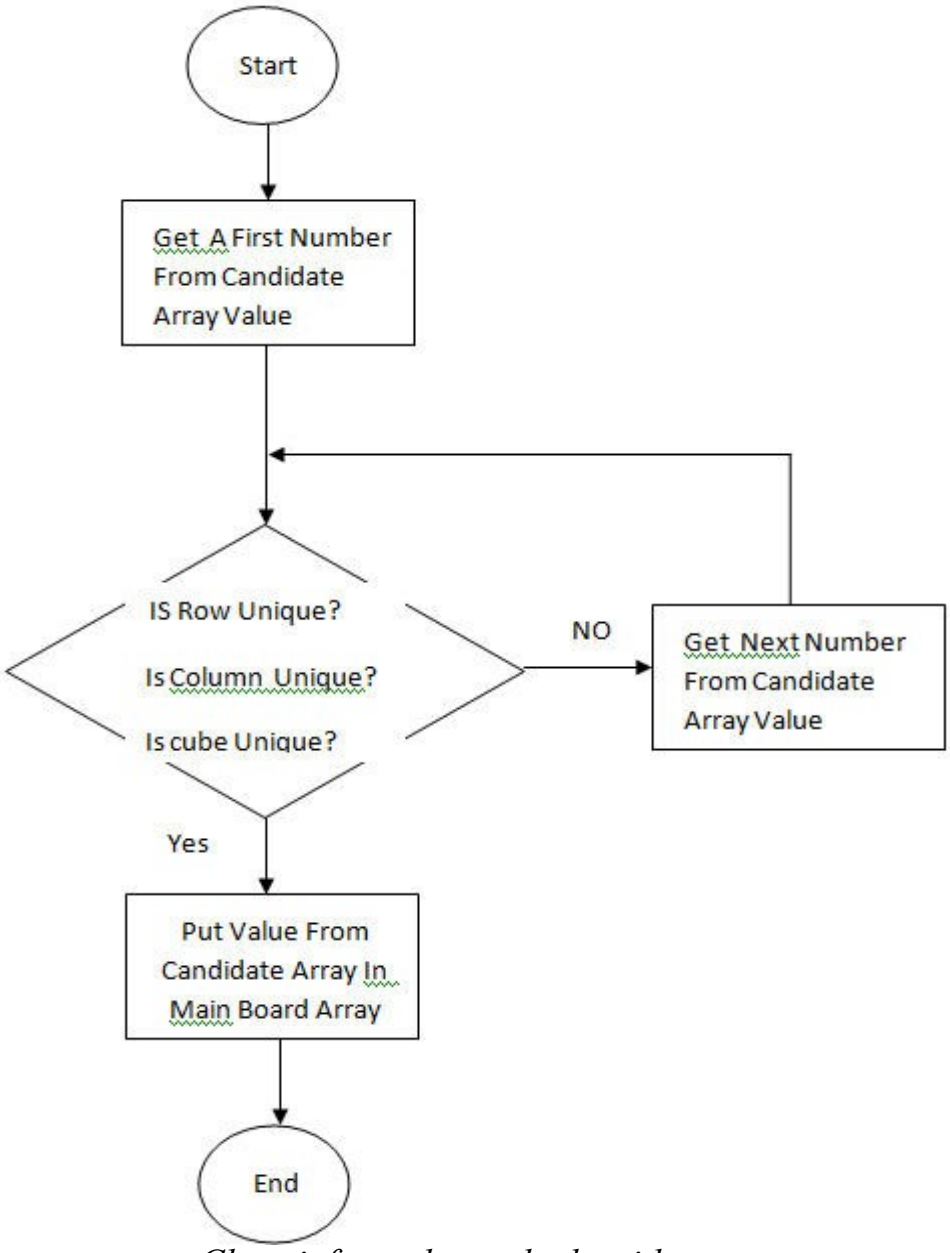


Chart informed search algorithms

E-Mail: hosein.fereidooni@gmail.com

Code Analysis:

It is composed of two layers:
1) Presentation Layer
2) Business Layer
Presentation layer in the array and to receive constructive input in the Business Class Sudoku sends.

This code will see the two classes. It is a two-dimensional array for storing the operating table and uses it on. Square value of the array's internal default is 3. The amount varies for different size tables. Also, an internal array of type *Bool* is used for storing the unique array. These variables are defined as global-level class with a different method of operation can be performed on these values.
The following code is inserted Sudoku incomplete. Zero points in the input string represented in the table is empty. Sudoku for the complete accuracy of the output is presented By comparing these values to the user able to see the correct answer. This code is part of a class of Sudoku and the Fill method to send the missing states. This method is used to fill the array of internal Sudoku.

Presentation layer code

```
Sudoku SUdok = new Sudoku();

            string[] SolvedSudoku = new string[81];

            string  temp1 =
"219876435843591276765243891394157628127368954658429317482735169536914782971682543";

            for (int i = 0; i < 81; i++)
            {
                SolvedSudoku[i] = Convert.ToString(temp1[i]);
            }

            SUdok.Fill(SolvedSudoku);

            lblCompleteSudoko.Text = SUdok.Show();

            for (int i = 0; i < 81; i++)
            {
                SolvedSudoku[i] = "0";
            }

            //entry sudoku
            // value 0 is null in entry sudoku
            string temp =
"219876435803591276065243891394150628127368954658429307482735169536914702970682543";

            for (int i = 0; i < 81; i++)
            {
                SolvedSudoku[i] = Convert.ToString(temp[i]);
            }

            SUdok.Fill(SolvedSudoku);
```


E-Mail: hosein.fereidooni@gmail.com

These variables have been created and initialized by the constructor of the variables are defined.

Business Layer Code

```
public class Sudoku
    {
        //N*N  Board
        int N = 3;

        //Main Board
        int[,] board = new int[9, 9];

        //this arry used by methods that Checked the value is unique in row and col
and inner cubes
        bool[] checkValue = new bool[10];

        //constructore
        public Sudoku()
        {
            //Fill main Array board by 0 value
            for (int i = 0; i <= (N * N)-1; i++)
            {
                for (int j = 0; j <= (N * N)-1; j++)
                {
                    board[i, j] = 0;
                }
            }

            //fill check array with default value
            for (int i = 0; i <= 9; i++)
            {
                checkValue[i] = true;
            }
        }
```

This method of access to the *i, j* element two-dimensional array that provides the main table.

```
// set value in main board
        //i , j position of element array
        //value i,j element array
        public void setSquare(int i, int j, int value)
        {
            board[i, j] = value;
        }

        //get value for acrross row and col value
        // i = row
        // j = col
        public int getSquare(int i, int j)
        {
            return board[i, j];
        }
```

This method of display two-dimensional table with the three main characters + and -
and| are the numbers. Character -
a row of | | the intersection of the column and row and column is used.

E-Mail: hosein.fereidooni@gmail.com

```
//this method Add "|" and "-" and "+" to show result in output.%3 put "|" in each
ternary in row.
        //
        public string Show()
        {
            string Temp = string.Empty;

            for (int i = 0; i <= (N * N)-1; i++)
            {
                for (int j = 0; j <= (N * N)-1; j++)
                {
                    //this if dont allow to loop for input "|" at first
                   if (j % 3 == 0 && j != 0)
                    {
                        Temp += "  |  ";
                    }
                    else
                    {
                        //this if check for put "+" in out put at Confluence row and
col in each cubes number
                        if (i % 3 == 0 && i != 0 && j == 0)
                        {
                            for (int k = 0; k < 18; k++)
                            {
                                if (k % 6 == 0 && k != 0)
                                {
                                    Temp += "+";
                                }
                                else
                                {
                                    Temp += "---";
                                }
                            }
                            Temp += "\r\n";
                        }
                    }
                   // insert dot instead of zero
                    if (board[i, j] == 0)
                    {
                        Temp += "  .  ";
                    }
                    else
                    {
                        Temp += "  " + board[i, j] + "  ";
                    }
                }
                Temp += "\r\n";
            }
            return Temp;
        }
```

As above-mentioned method for inserting *FILL* incomplete array of input values used in the original array. FancyFill method for array values as input in excess of incomplete Tues. characters + and - and | are used for display.

```
  //fill main board by input string array
        public void Fill(string[] lines)
        {
            int k = 0;
```


E-Mail: hosein.fereidooni@gmail.com

```
        for (int i = 0; i <= (N * N)-1; i++)
        {
            for (int j = 0; j <= (N * N)-1; j++)
            {
                board[i, j] = int.Parse(lines[k]);
                k++;
            }
        }
    }
    //fill board by input string array and delete any + - | in input string
    // this method for check two condition use peterson solution
    public void FancyFill(string[] lines)
    {
        for (int i = 0; i <= (N * N)-1; i++)
        {
            for (int j = 0; j <= (N * N)-1; j++)
            {
                board[i, j] = 0;
            }
        }

        int k = 0;

        bool flag = true;

        for (int i = 0; i <= (N * N)-1; i++)
        {
            for (int j = 0; j <= (N * N)-1; j++)
            {
                flag = true;
                //peterson solution
                for (int l = k; l < lines.Length && flag; l++)
                {

                    if (char.IsDigit(char.Parse(lines[l])))
                    {
                        board[i, j] = int.Parse(lines[l]);
                        flag = false;
                    }
                    if (char.Parse(lines[l]) == '.')
                    {
                        board[i, j] = 0;
                        flag = false;
                    }

                    k++;
                }
            }

        }
    }
```

IsSolution() : Checks whether or not it is solving *sudoku*?

For this purpose, uses four auxiliary functions. The performance of four methods as follows:

1) *IsComplete():* This method is used for all array element.
2) *isRowCheckUnique()* : This method is unique for all rows of the original array is used.
3) *isColCheckUnique()* : This unique method for all columns used in the original array.
4) *isCubeUnique() :* This method is unique for each of the 3 × 3 cube is built.

E-Mail: hosein.fereidooni@gmail.com

The four were Sudoku solving method if the answer is True.

```
// check sudoku is solved? if true return true
        public bool isSolution()
        {
            if (isComplete() && isRowCheckUnique() && isColCheckUnique() &&
isCubeUnique())
            {
                return true;
            }
            return false;
        }

        //check board array is complete by number?
        public bool isComplete()
        {
            for (int i = 0; i <= (N * N)-1; i++)
            {
                for (int j = 0; j <= (N * N)-1; j++)
                {
                    if (!(board[i, j] <= 9 && board[i, j] >= 1))
                    {
                        return false;
                    }
                }
            }


            return true;
        }

        //values in rows is unique? use check value array for true or false
        public bool isRowCheckUnique()
        {
            for (int i = 0; i <= (N * N)-1; i++)
            {
                for (int j = 0; j <= (N * N)-1; j++)
                {
                    //checkValue[0] = true;

                    if (checkValue[board[i,j]] == true)
                    {
                        checkValue[board[i,j]] = false;
                    }
                    else
                    {
                        return false;
                    }
                }
                for (int k = 0; k <= 9; k++)
                {
                    checkValue[k] = true;
                }
            }


            return true;
        }
```


E-Mail: hosein.fereidooni@gmail.com

```
//values in columns is unique? use check value array for true or false
      public bool isColCheckUnique()
      {
          for (int i = 0; i <= (N * N)-1; i++)
          {
              for (int j = 0; j <= (N * N)-1; j++)
              {
                  if (checkValue[board[i,j]] == true)
                  {
                      checkValue[board[i,j]] = false;
                  }
                  else
                  {
                      return false;
                  }
              }
              for (int k = 0; k <= 9; k++)
              {
                  checkValue[k] = true;
              }
          }

          return true;
      }

//this method check evry 3*3 is 1..9 value?
      //if is unique return true.
      public bool isCubeUnique()
      {

          #region iscubeunique
          //1,1
          for (int i = 0; i <= 2; i++)
          {
              for (int j = 0; j <= 2; j++)
              {
                  checkValue[0] = true;

                  if (checkValue[board[i, j]] == true)
                  {
                      checkValue[board[i, j]] = false;
                  }
                  else
                  {
                      return false;
                  }
              }


          }
          for (int k = 0; k <= 9; k++)
          {
              checkValue[k] = true;
          }
          //1,2
          for (int i = 0; i <= 2; i++)
          {
              for (int j = 3; j <= 5; j++)
              {
                  checkValue[0] = true;
```


E-Mail: hosein.fereidooni@gmail.com

```
            if (checkValue[board[i, j]] == true)
            {
                checkValue[board[i, j]] = false;
            }
            else
            {
                return false;
            }
        }

    }
    for (int k = 0; k <= 9; k++)
    {
        checkValue[k] = true;
    }

    //1,3
    for (int i = 0; i <= 2; i++)
    {
        for (int j = 6; j <= 8; j++)
        {
            checkValue[0] = true;

            if (checkValue[board[i, j]] == true)
            {
                checkValue[board[i, j]] = false;
            }
            else
            {
                return false;
            }
        }

    }
    for (int k = 0; k <= 9; k++)
    {
        checkValue[k] = true;
    }
    //2,1
    for (int i = 3; i <= 5; i++)
    {
        for (int j = 0; j <= 2; j++)
        {
            checkValue[0] = true;

            if (checkValue[board[i, j]] == true)
            {
                checkValue[board[i, j]] = false;
            }
            else
            {
                return false;
            }
        }

    }
    for (int k = 0; k <= 9; k++)
    {
        checkValue[k] = true;
    }
    //2,2
```

```
for (int i = 3; i <= 5; i++)
{
    for (int j = 3; j <= 5; j++)
    {
        checkValue[0] = true;

        if (checkValue[board[i, j]] == true)
        {
            checkValue[board[i, j]] = false;
        }
        else
        {
            return false;
        }
    }

}
for (int k = 0; k <= 9; k++)
{
    checkValue[k] = true;
}
//2,3
for (int i = 3; i <= 5; i++)
{
    for (int j = 6; j <= 8; j++)
    {
        checkValue[0] = true;

        if (checkValue[board[i, j]] == true)
        {
            checkValue[board[i, j]] = false;
        }
        else
        {
            return false;
        }
    }

}
for (int k = 0; k <= 9; k++)
{
    checkValue[k] = true;
}
//3,1
for (int i = 6; i <= 8; i++)
{
    for (int j = 0; j <= 2; j++)
    {
        checkValue[0] = true;

        if (checkValue[board[i, j]] == true)
        {
            checkValue[board[i, j]] = false;
        }
        else
        {
            return false;
        }
    }

}
```


E-Mail: hosein.fereidooni@gmail.com

```
            for (int k = 0; k <= 9; k++)
            {
                checkValue[k] = true;
            }
            //3,2
            for (int i = 6; i <= 8; i++)
            {
                for (int j = 3; j <= 5; j++)
                {
                    checkValue[0] = true;

                    if (checkValue[board[i, j]] == true)
                    {
                        checkValue[board[i, j]] = false;
                    }
                    else
                    {
                        return false;
                    }
                }

            }
            for (int k = 0; k <= 9; k++)
            {
                checkValue[k] = true;
            }
            //3,3,
            for (int i = 6; i <= 8; i++)
            {
                for (int j = 6; j <= 8; j++)
                {
                    checkValue[0] = true;

                    if (checkValue[board[i, j]] == true)
                    {
                        checkValue[board[i, j]] = false;
                    }
                    else
                    {
                        return false;
                    }
                }

            }
            for (int k = 0; k <= 9; k++)
            {
                checkValue[k] = true;
            }

            return true;
            #endregion
        }
```

Three-dimensional array ***InnerCandidateValue***

This array is used to hold element values for each candidate. Considering that the original 81 houses, each array must have some unique value in athree-dimensional array with the values and placed in the correct position is his.

```
//candidate value global list
     int[, ,] InnerCandidateValue = new int[10, 10, 11];
```

FillcandidateValue():

This method is used to fill the array of candidates for conservative values.

```
//inner candidate value fill
     public void FillcandidateValue()
     {
         //fill candidate value in innner list
         for (int i = 0; i < 10; i++)
         {
             for (int j = 0; j < 10; j++)
             {
                 for (int k = 0; k < 11; k++)
                 {
                     InnerCandidateValue[i, j, k] = k;
                 }
             }
         }
     }
```

solvedsudoku() :

This method is used to solve Sudoku incomplete. This program is part of the method. In this method, the number of incomplete element main array of the array is a candidate. If the three conditions was the main Sudoku array of candidates for the correct amount is placed on the next blank. If the array of candidates to replace the new value should be checked again. Finally, the correct amount in the blank element and informed seek to solve Sudoku.

```
//this method solve sudoku by Informed solution
     public void SolvedSudoku()
     {
         FillcandidateValue();
         //solve
         for (int i = 0; i <= (N * N) - 1; i++)
         {
             for (int j = 0; j <= (N * N) - 1; j++)
             {
                 if (board[i , j] == 0)
                 {
                     for (int l = 0; l <= 8; l++)
                     {
                         board[i, j] = InnerCandidateValue[i, j, l+1];

       if ((isOneRowCheckUniqe(i) && isOneColCheckUniqe(j) && isCubeUnique()))
                         {
                             l = 10;
                         }
                     }
                 }
             }
         }
     }
```

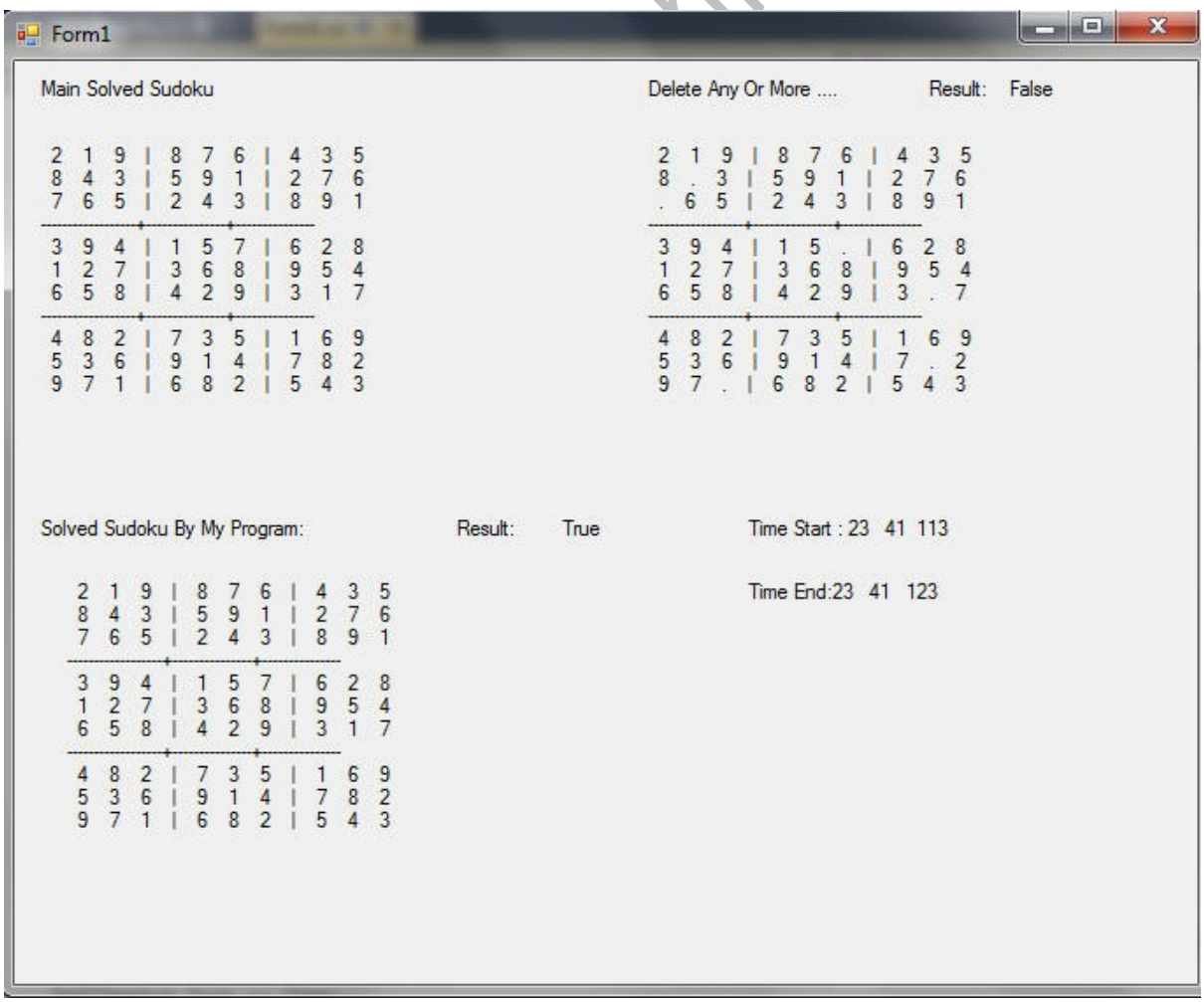
Finally, the presentation layer, it will display the full amount. This is shown next to an empty array. Finally, by solving an array of programs is displayed. The function of each table

E-Mail: hosein.fereidooni@gmail.com

Issolution () to check and ensure the correctness of the program is placed. Also at the beginning and end time in milliseconds is displayed on the system until the problem is presented.

```
OutPut.Text = SUdok.Show();

            lblissolution.Text = Convert.ToString(SUdok.isSolution());

            SUdok.SolvedSudoku();

            lblissolution2.Text = Convert.ToString(SUdok.isSolution());

            lblOutput2.Text = SUdok.Show();

            Time = string.Empty;

            Time = DateTime.Now.Minute.ToString() + "    " +
DateTime.Now.Second.ToString() + "    " + DateTime.Now.Millisecond.ToString();

            lblTimeEnd.Text += Time;
```

Using The Code:
To use the code to create a Windows Forms project and add to the Sudoku class. An input value of the class and put in the variable Temp. For example, with a truncated version of Sudoku has been placed in this variable.
The output format is shown below:



E-Mail: hosein.fereidooni@gmail.com