|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | Báo cáoNiên luận cơ sở ngành Kỹ thuật phần mềm | | **XÂY DỰNG ỨNG DỤNG GIẢI SUDOKU BẬC 3 ĐẾN 16** 3x3 | 4x4 | 5x5 | 6x67x7 | 8x8 | 9x910x10 | 12x12 | 16x16 **ĐẠI HỌC CẦN THƠ** 01-11-2017 | | |  | | --- | | Họ và tên: Lê Minh LuânMã số: B1400704Khóa: 40Ngành: kỹ thuật phần mềmLớp: DI1496A1 | | |  |  | | --- | --- | |  |  | | luanb1400704@gmail.com | 0964054244 | |  |  | | | **Giảng viên hướng dẫn** Ths. Huỳnh Quang NghiBộ môn: Công nghệ phần mềmKhoa: Công nghệ thông tin và truyền thông | |

# mở đầu về sudoku

## Lịch sử

Sudoku được hình thành ở Nhật Bản vào năm 990 với tên gọi ban đầu là ô số kỳ ảo, nó phổ biến sang các quốc gia Ả Rập và Châu Âu trong thời gian sau đó.

Sudoku có thêm một bước tiến hóa mới vào năm 1776 khi một nhà toán học kiêm vật lý học người Thuỵ Sĩ tên Leonhard Euler bắt đầu nghiên cứu và phát triển các luật chơi mà ngày nay ta gọi là luật chơi Sudoku. Từ đó trò chơi này được gọi với tên Sudoku.

Năm 1970 lần đầu tiên Sudoku được xuất bản trong tạp chí và phổ biến ra công chúng ở New York (Hoa Kỳ).

Năm 2004, niềm đam mê Sudoku đã đưa Wayne Gould đến với London (Anh). Nhân một chuyến thăm ngẫu nhiên báo The Times, Gould đã thuyết phục tổng biên tập của báo này cho đăng Sudoku bên cạnh các ô chữ. Độc giả lập tức bị cuốn hút và yêu cầu đăng thêm nữa. Chỉ trong vài tuần lễ, các tờ báo trên khắp nước Anh đã thi nhau đăng Sudoku. Từ đó, Sudoku bắt đầu lan rộng sang Mỹ, Canada, Úc, Pháp, Nam Phi và nhiều quốc gia khác.

Sudoku là trò puzzle (đoán số hay chữ) phát triển nhanh nhất trên thế giới. Nó hiện có hàng triệu tín đồ và con nghiền. Nhiều nhân vật nổi tiếng ủng hộ nó. Và nó đã có được một nhà vô địch thế giới. Chính quyền nhiều nước đã khuyến cáo Sudoku như một công cụ rèn luyện trí lực và hạn chế sự phát triển của bệnh Alzheimer.

Từ đó đến nay Sudoku đã bắt đầu phát triển với tốc độ khá nhanh và nhiều dị bản cho nó. Sự phát triển của Sudoko cũng được so sánh gần giống với sự phát triển của môn thể thao huyền thoại Rubik.

## Các dị bảng

Nền tảng và phiên bảng phổ biến nhất của Sudoku chính là ma trận khối 3x3 (3 hàng và 3 cột), các khối 3x3 này lại ghép theo luật 3 hàng 3 cột thành ma trận Sudoku 9x9.

Ngoài phiên bảng phổ biến nhất này, Sudoku còn nhiều dị bảng và khả năng giải chúng cũng như khả năng phân khối là vô cùng đa dạng. Bao gồm:

- Kích thước 4x4 ô chia làm 2x2 vùng  
- Kích thước 6x6 ô chia làm 2x3 vùng  
- Kích thước 5x5 ô chia vùng theo [Pentomino](https://en.wikipedia.org/wiki/Pentomino) (được phát hành với tên gọi Logi-5)  
- Kích thước 7x7 ô chia vùng theo [Heptomino](https://en.wikipedia.org/wiki/Heptomino)  
- Kích thước 8x8 ô chia vùng theo quy tắc (4x2):(4x2). Đây là cách chia thành 4 vùng chính, mỗi vùng 16 ô. Trong mỗi vùng chính lại chia thành 2 vùng 8x8 dựa vào màu nền của từng ô. Tuy theo cách bố trí các ô khác màu này, sẽ phát sinh thêm một biến thể con khác. Cách bố trí đơn giản nhất là các ô khác màu nằm xen kẽ nhau – trông rất giống bàn cờ quốc tế.

Biến thể với kích thước lớn cũng khá phổ biến:

- Kích thước 16x16 ô (Monster SuDoku)  
- Kích thước 12x12 ô chia làm 4x3 vùng (Dodeka Sudoku)  
- Kích thước 25x25 ô (Giant Sudoku)  
- Biến thể có kích thước lớn nhất được phổ biến là 100x100 ô.

## Sudoku trong lĩnh vực Lập trình

Từ những ngày đầu hình thành lĩnh vực lập trình nói riêng và công nghệ thông tin nói chung, ma trận Sudoku luôn được thực hiện giải với các giải thuật mới và ngày càng phổ biến. Sudoku cũng là đề bài cho nhiều người hứng thú với lập trình.

Hiện nay ta dễ dàng tải về một ứng dụng trò chơi Sudoku trên máy tính hoặc điện thoại di động.

Sudoku được lập tình bởi hầu hết nhiều ngôn ngữ như: C, C++, Java, C#, Các ngôn ngữ web,…

Quá trình tìm ra giải thuật tối ưu nhất có thể để giải Sudoku cũng là một bài toán phổ biến trong ngành lập trình.

# mô tả bài toán sudoku

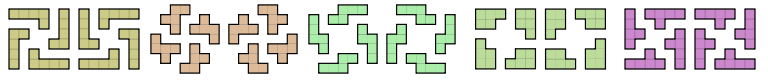
Sudoku có nhiều dị bản. Tuy nhiên, trong phần báo cáo này trình bài các dị bảng và luật chơi của chúng một cách khái quát nhất về luật chơi cho tất cả các dị bảng xoay quanh luật chơi 9x9 phổ biến nhất hiện nay.

## Cấu trúc một ma trận sudoku

Cấu trúc chung cho sudoku là một ma trận vuông cấp n (n hàng và n cột) với n là số nguyên lớn hơn 3 (Sudoku cấp 3 là nhỏ nhất).

Quy luật phân chia một Sudoku cũng có tính chất chung. Quy tắt này gọi là quy tắt độc nhất. Quy tắt này như sau:

* Quy tắc hàng ngang (Row): Một Sudoku cấp n thì trên mỗi hàng của nó chứa các số từ 1 đến n có thể không theo thứ tự, tức là trên một hàng không có 2 số trùng nhau.
* Quy tắc cột dọc (Colum): Một Sudoku cấp n thì trên mỗi cột của nó chứa các số từ 1 đến n có thể không theo thứ tự, tức là trên một cột không có 2 số trùng nhau.
* Quy tắc khối đơn (Block): Một Sudoku cấp n thì được chia làm n khối các nhau, một khối là tập hợp n ô, chúng có thể cùng hàng, hoặc cùng cột , hoặc khác cả hàng lẫn cột. Quy tắc là sau cho trong một khối n ô thì cũng có n số không trùng, tương tự như 2 quy tắc Row và Colum. Một Sudoku có thể có nhiều dạng chia khối khác nhau ví dụ như các Sudoku dưới đây (mỗi các ô cũng màu là cùng một khối).



*Biến thể block 5x5*

Thỏa mãn 3 quy tắc đã nêu thì ta có một ma trận Sudoku

Ngoài các Sudoku vuông n, còn tồn tại các biến thể puzzle theo ma trận **m** x **n** tuy nhiên tài liệu này chỉ tập trung vào các biến thể vuông n cạnh.

## bài toán ẩn số trong sudoku

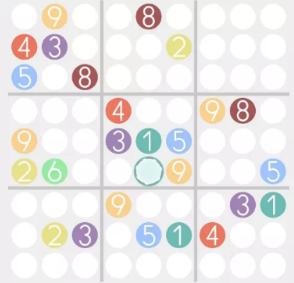
Để chơi một trò chơi trí tuệ như Sudoku người chơi cần phải điền đầy đủ tất cả các ô trong một ma trận Sudoku.

Để khởi động trò chơi người chơi sẽ được cung cấp một ma trận Sudoku trống hoặc có điền số nhưng không điền hết. Nhiệm vụ của người chơi là phải điền đủ tất cả các ô cho hợp theo 3 quy luật chơi Sudoku (Row, Colum, Block).

Độ phức tạp của Sudoku được xác định vào cấu trúc ma trận và các số đã điền sẵn trong đó:

* Sudoku cấp n, thì n càng lớp độ khó càng cao
* Sudoku cấp n, số ô điền sẵn càng ít thì độ khó càng cao
* Sudoku n là số lẽ và không không có căn bật 2 số học thì sẽ khó giải hơn các Sudoku số chẵn hoặc có căn bật 2 số học

Dưới đây là một bài toán Sudoku quen thuộc:



# Vấn đề cần giải quyết

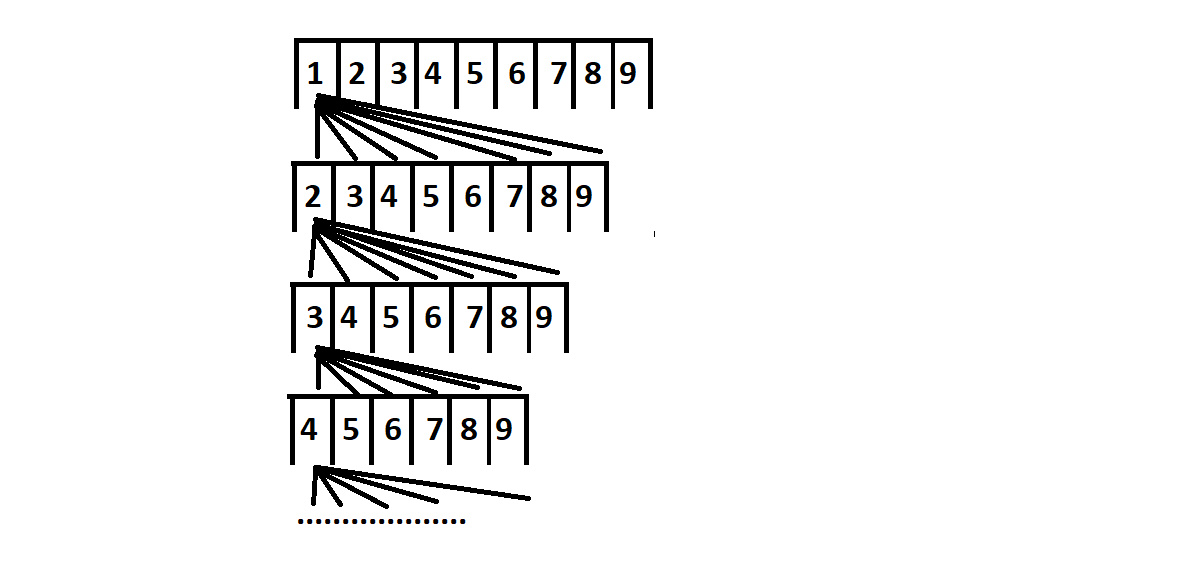
Một ma trận Sudoku nào đó có thể có từ 0 đến nhiều nghiệm, việc có bao nhiêu nghiệm thì phụ thuộc vào cách sắp xếp các ô số của đề bài. Tuy nhiên các ô cho sẵn càng ít thì số nghiệm còn lại sẽ nhiều hơn. Tuy nhiên vẫn có các Sudoku vô nghiệm, và việc giải một Sudoku bằng cách chọn số thường gặp 2 vấn đề sau.

## thời gian

Nếu giải một Sudoku một các không theo quy tắc, người chơi sẽ mất rất nhiều thời gian. Cụ thể một Sudoku 9x9 có 81 ô số, và để điền một hàng bất kỳ người chơi cần lựa chọn 1 trong 9 con số, và cứ tiếp tục như thế người đó sẽ chọn cách điền cho con số tiếp theo. Với một hàng hoặc cột sẽ có **9x8x76x5x4x3x2x1 = 362880** cách chọn, và theo hàng và khối cũng tương tự. Giả sử người chơi chọn 1 trong 3 cách là điền theo hàng, cột, hoặc khối thì họ phải thực hiện 9 lần như trên tức là: **362880 x 9 = 3265920** cách điền. Việc thực hiện theo hướng thủ công là vô cùng khó khăn và tốn thời gian. Trung bình với một người thường chơi Sudoku thì mất khoản 15-30 phút cho một biến thể 9x9 không điền trước. Nếu vậy để thực hiện **3265920** cách thì người đó tốn khoản **48988800 đến 97977600 phút** tương đương **từ 816480 giờ đến 1632960 giờ**. Điều này là quá sức so với một trò chơi ở mức độ giải trí

## Công sức

Về mặt công sức,xét ma trận 9x9 nếu quy đổi các giải thuần ra một cây các cách giải thì ta có một cây để điền cho cho 81 ô với các chữ số từ 1 đến 9 như sau:



Việc thực hiện tuần tự một phép tính tay là tốn quá nhiều công sức, và việc kiểm tra chúng cũng thế. Cứ mỗi lần điền như thế ta kiểm tra lại ma trận, nếu thỏa thì xem như giả thành công, trong trường hợp đáp án chỉ có 1 và là cách điền cuối cùng trong danh sách các cách thì sẽ vô cùng tốn thời gian.

# hướng tiếp cận

Để giải một Sudoku người ta chọn giải thuật Quay lui ( còn gọi là Backtracking) , với hướng dùng giải thuật này ,ta sẽ vừa thực hiện giải vừa kiểm tra, đảm bảo việc chuyển hướng giải đúng lúc, rút ngắn thời gian tối đa.

Về hoạt động: Quay lui là một kĩ thuật thiết kế giải thuật dựa trên đệ quy. Ý tưởng của quay lui là tìm lời giải từng bước, mỗi bước chọn một trong số các lựa chọn khả dĩ và đệ quy.

Giả thiết cấu hình cần liệt kê có dạng ( x 1 , x 2 , . . . , x n ). Khi đó thuật toán quay lui được thực hiện qua các bước sau:

* Xét tất cả các giá trị x 1 {\displaystyle x\_{1}} x1 có thể nhận, thử cho x 1 {\displaystyle x\_{1}} x1 nhận lần lượt các giá trị đó. Với mỗi giá trị thử cho x 1 {\displaystyle x\_{1}} x1 ta sẽ:
* Xét tất cả các giá trị x 2 {\displaystyle x\_{2}} x2 có thể nhận, lại thử cho x 2 {\displaystyle x\_{2}} x2 nhận lần lượt các giá trị đó. Với mỗi giá trị thử gán cho x 2 {\displaystyle x\_{2}} x2 lại xét tiếp các khả năng chọn x 3 {\displaystyle x\_{3}} x2... cứ tiếp tục như vậy.

………

* Xét tất cả các giá trị x n {\displaystyle x\_{n}} xn có thể nhận, thử cho x n {\displaystyle x\_{n}} xn nhận lần lượt các giá trị đó, thông báo cấu hình tìm được ( x1,x2,x3,…,xn ) ( x 1 , x 2 , . . . , x n ) {\displaystyle (x\_{1},x\_{2},...,x\_{n})} .

Thuật toán quay lui có thể được mô tả bằng đoạn mã giả sau:

{Thủ tục này thử cho x i {\displaystyle x\_{i}} x thứ i nhận lần lượt các giá trị mà nó có thể nhận}

procedure Try(i: Integer);

**begin**

for (mọi giá trị có thể gán cho x i {\displaystyle x\_{i}} x thứ i) do

**begin**

<Thử cho x i {\displaystyle x\_{i}} x thứ i:= V>;

**if** ( x i {\displaystyle x\_{i}} x thứ i là phần tử cuối cùng trong cấu hình) then <Thông báo cấu hình tìm được>

**else**

b**egin**

<Ghi nhận việc cho x i {\displaystyle x\_{i}} x thứ i nhận giá trị V (Nếu cần)>;

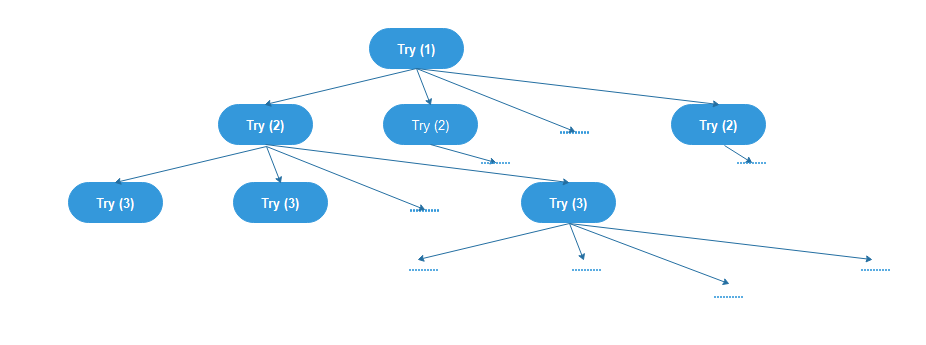
Try(i + 1); {Gọi đệ quy để chọn tiếp x i + 1 {\displaystyle x\_{i+1}} x thứ i+1}

**end;**

<Nếu cần, bỏ ghi nhận việc thử x i {\displaystyle x\_{i}} x thứ i:= V, để thử giá trị khác>;

**end;**

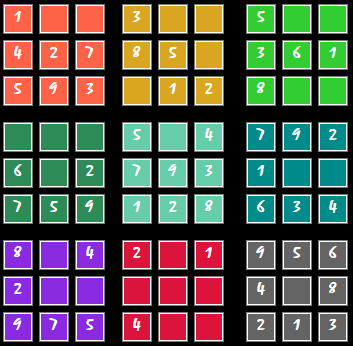
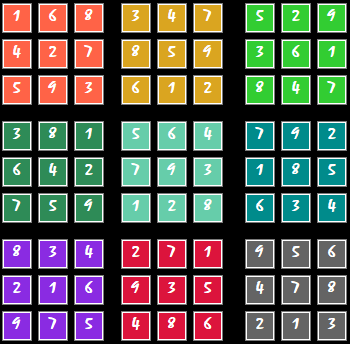
**end;**



# cách giải quyết

Để giải một Sudoku ta có thể tiếp cận nó bằng giải thuật quay lui như sau

Ta có đề bài và đáp án sau, hãy xem giải thuật hoạt động thế nào trên ngôn ngữ C++:

Mỗi bước tìm tập các giá trị khả dĩ để điền vào ô trống, và sau đó đệ quy để điền ô tiếp theo. Giả mã của thuật toán (ở đây chú ý mảng chỉ có kích thước 9×9). Thủ tục Feasible(*S*,*x*,*y*,*k*) kiểm trả xem giá trị *k* có khả dĩ với ô *S*[*x*][*y*] không.

**Sudoku(S[1,2,…,9][1,2,…,9],x,y):   
    if y=10  
        if x=9  
            print S  
       else  
            Sudoku(S[1,2,…,9][1,2,…,9],x+1,1)  
    else if S[x,y]=∅  
        for k←1 to 9  
            if Feasible(S,x,y,k)  
                S[x,y]←k  
                Sudoku(S[1,2,…,9][1,2,…,9],x,y+1)  
                S[x,y]←∅        ≪ for next branching ≫  
    else                                    ≪S[x,y] is given ≫  
        Sudoku(S[1,2,…,9][1,2,…,9],x,y+1)**

**Code trên C++:**

void solve\_sudoku(int S[][9], int x, int y){

    if(y == 9){

        if(x == 8){

            printSolution(S);

            exit(0);

        } else {

            solve\_sudoku(S, x+1,0);

        }

    } else if(S[x][y] == 0){

        for (int k = 1; k <=9; k++){

            if(feasible(S,x,y,k)){

                S[x][y] = k;

                solve\_sudoku(S, x, y+1);

                S[x][y] = 0;

            }

        }

    } else {

        solve\_sudoku(S,x,y+1);

    }

}

Giả mã của thủ tục Feasible(S,x,y,k) như sau:

**Feasible(S[1,2,…,9][1,2,…,9],x,y,k):   
    for i←1 to 9  
        if S[x,i]=k  
            return False  
    for i←1 to 9  
        if S[i,y]=k  
            return False  
    a←⌊(x−1)/3⌋,b←⌊(y−1)/3⌋  
    for i←3a+1 to 3a+3  
        for j←3b+1 to 3b+3  
           if S[i,j]=k  
                return False  
    return True**

**Code bằng C++:**

int feasible(int S[][9], int x, int y, int k){

    int i = 0, j = 0;

    for(i = 0; i <9 ; i++){

        if(S[x][i] == k) return 0;

    }

    for(j = 0; j <9 ; j++){

            if(S[j][y] == k) return 0;

        }

    int a = x/3, b = y/3;

    for(u = 3\*a; u < 3\*a+3; u++){

        for(v = 3\*b; v < 3\*b+3; v++){

            if(S[u][v] == k) return 0;

        }

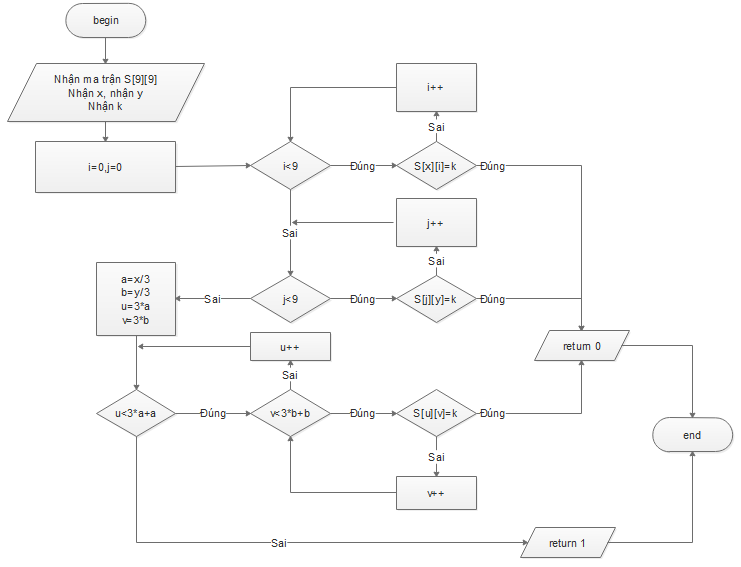
    }

    return 1;

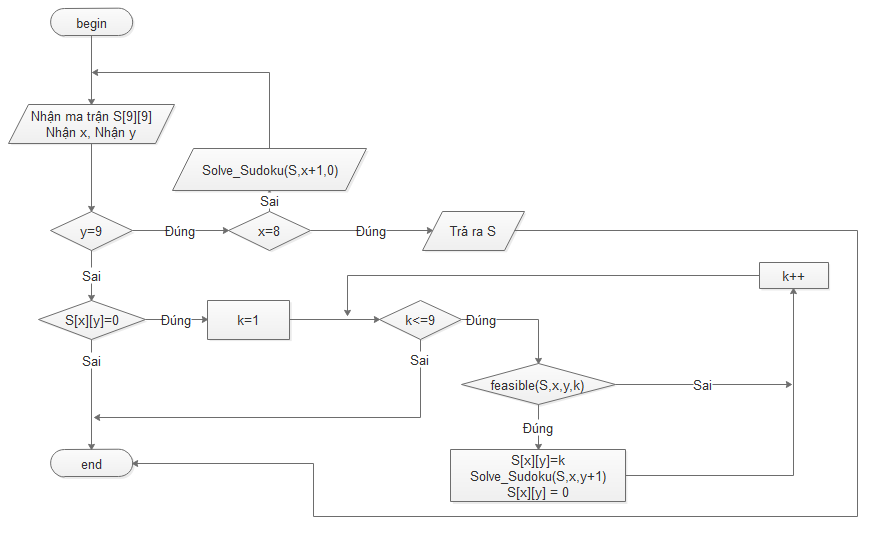
}

# lưu đồ giải thuật

Giải thuật giả Sudoku 9x9 (Solve\_Sudoku)



Giải thuật kiểm tra tính hợp lệ

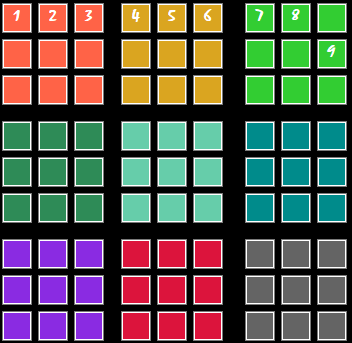


# cải tiến bài toán

## Khuyết điểm bài toán hiện tại

Từ phần trình bày ở hai phần 5 và 6 ta dễ dàng thấy bài toán giải Sudoku có một số khuyết điểm trong giải.

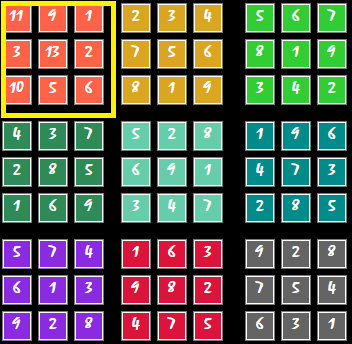
* **Vấn đề 1: Không giải quyết được vấn đề ma trận vô nghiệm:**



*Một ví dụ ma trận vô nghiệm ô 1-9 chỉ có thể điền 9, tuy nhiên ô 2-9 đã trùng*

Nếu ta đưa một ma trận vô nghiệm vào thuật toán trên thì nó sẽ vô tình trở thành một vòng lặp rất lớn chạy tất cả các trường hợp của phép giải ma trận. Quá trình sẽ thực hiện khá lâu và kết quả cuối cùng trong vòng lập trả ra sẽ là một ma trận mà trong đó không đảm bảo các tính chất của một Sudoku.

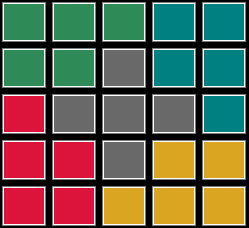
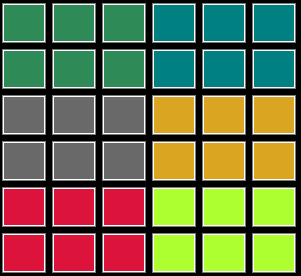
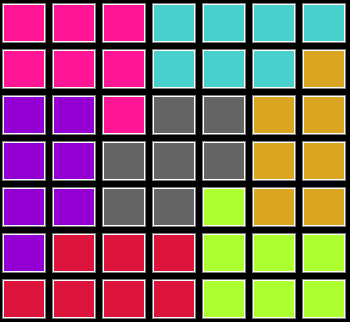
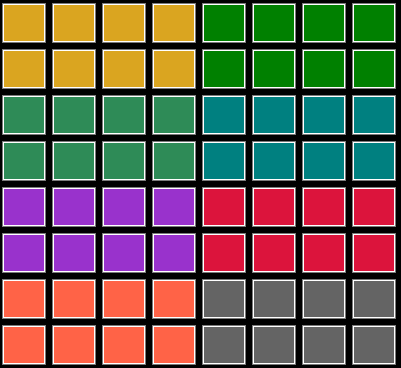
* **Vấn đề 2: Không kiểm tra được điểm biên và miền giá trị:**



*Ví dụ về miền giá trị bị vượt , ô 1-1, ô 2-2, ô 3-1*

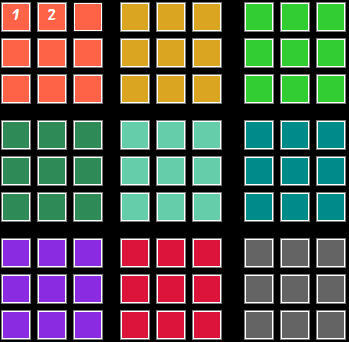
Một ma trận mặt định 9x9 sẽ được giải mà không thông qua việc kiểm tra miền giá trị. Thuật toán trên vẫn sẽ giải đồng thời cho ra một kết quả sai vì nó không xác định được điểm biên của một ma trận. Trong quá trình giải các giá trị vượt điểm biên sẽ được giữ lại và làm cho ma trận kết quả bị sai

* **Vấn đề 3: Thuật toán trên không thể giải được một Sudoku lớn hơn hoặc có phân vùng Block khác với hệ 9x9.**

*Cấu trúc 5x5, 6x6, 7x7, 8x8 khác với 9x9*

* **Vấn đề 4: Cách giải nghiệm kém đa dạng**



Ma trận có nhiều nghiệm (trái) và 2 trong các nghiệm của nó

Tính đa dạng trong việc thực hiện sẽ bị giới hạn, với một ma trận nhiền nghiệm thì thuật toán nào sẽ chỉ tìm được 1 nghiệm trong các lần thực hiện, thay vì ngẫu nhiên lấy 1 nghiệm trong danh sách các nghiệm.

* **Vấn đề 5: Phương pháp giải thuật theo hướng lập trình cấu trúc, gây khó khăn cho việc hình dung giải thuật và tối ưu hóa cách giải.**

## cải tiến bài toán

Để giải quyết được những nhược điểm trên ta thực hiện cải tiến thuật toán giải Sudoku và xây dựng thêm các giải thuật bên trong để hỗ trợ quá trình giải sudoku nxn bất kỳ. Tiến hành cải thiện theo hướng giao diện và sử dụng ngôn ngữ lập trình C#:

* **Đề xuất chuyển từ hướng lập trình cấu trúc sang lập trình hướng đối tượng. Hình thành đối tượng ô số Sudoku với cấu trúc sau (giải quyết vấn đề 5):**

class Number

{

private int row; //hàng của ô

private int column; //cột của ô

private int status;

/\*\*\* trạng thái ô

\* -1 là trạng thái mới khởi tạo

\* 0 là trạng thái mặc định bị che đi, click vào để chỉnh sửa số cho nó

\* 1 là trạng thái ô cho sẵn không thể sửa

\* 2 là ô đã điền rồi, nhưng có thể chỉnh sửa lại

\*/

private int value; // Chứa giá trị cho một ô

}

Với một đối tượng ô số ta dễ dàng xác định vị trí của nó trong ma trận, cũng như xác định giá trị lẫn mối quan hệ của nó với các ô số khác.

* **Giải quyết vấn đề ma trận nxn và kiểm tra phân vùng (vấn đề 3):**

Sử dụng mảng để lưu và khởi tạo các đối tượng ô:

//Tạo ra một giá trị cho một block

public Number newBlockValue(int row, int colum, int status, int value)

{

Number num = new Number();

num.Row = row;

num.Column = colum;

num.Status = status;

num.Value = value;

return num;

}

//Hàm tạo ma trận rỗng x\*y

public Number[,] newMatrix(int number)

{

Number[,] num = new Number[number, number];

for (int i = 0; i < number; i++)

{

for (int j = 0; j < number; j++)

{

//Tạo ô i cột j với ô này false (bị khóa lại) và giá trị ban đầu bị trống

num[i, j] = this.newBlockValue(i, j, 0, 0);

}

}

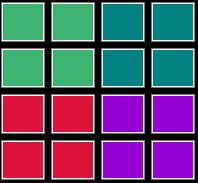
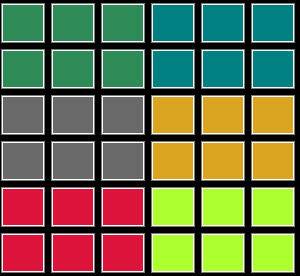
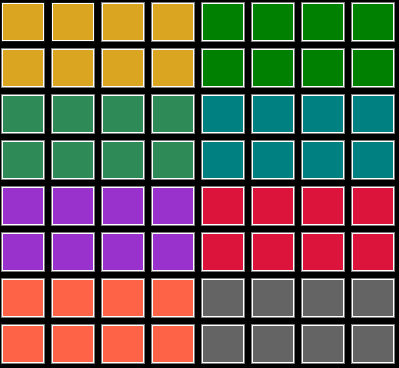
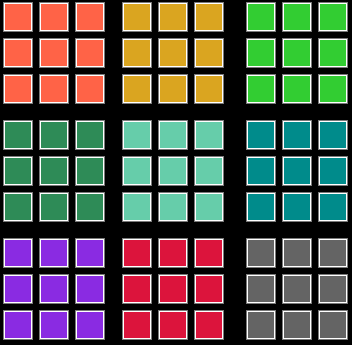
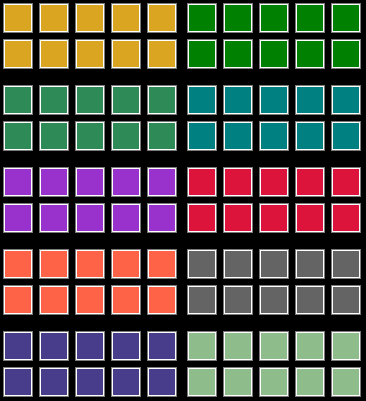
return num;

}

Cải tiến các phân vùng:

Có 2 cách phân vùng cho một ma trận Sudoku:

* Phân các ma trận có phân dùng theo kiểu **m x n** (ô vuông hoặc chữ nhật)

*Các dạng tiêu biểu cho Block* ***m x n (4x4, 6x6, 8x8, 9x9, 10x10)***

Ta dùng thuật thuật toán để kiểm tra cho dạng phân vùng này

public bool feasibleBlock(Number[,] Sudoku, int x, int y, int number, int value,int m, int n)

{

//Kiểm tra xe có bị trùng với ô nào trong khối của nó không //Dạng phân vùng m x n, kiểm tra ô [x,y] trong phân vùng

int a = x / m, b = y / n;

for (int i = 3 \* a; i < 3 \* a + 3; i++)

{

for (int j = 4 \* b; j < 4 \* b + 4; j++)

{

if (Sudoku[i, j].Value == value && i != x && j != y) return false;

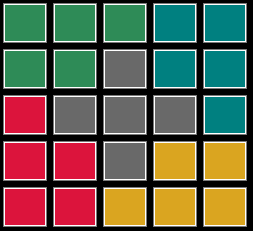
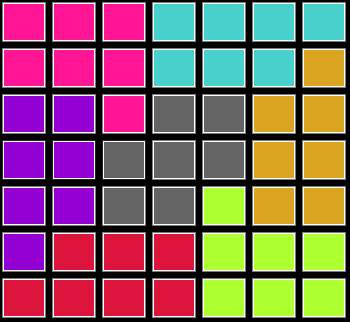
}

}

return true;

}

* Phân các ma trận có phân dùng theo kiểu **bất quy tắc** n ô cho ma trận **n x n** (ô vuông hoặc chữ nhật):

*Tiêu biểu cho phân vùng bất quy tắc là dạng ma trận lẻ và không có căn bật 2 số học như* ***5x5*** *và* ***7x7***

Để giải dạng này ta tiến hành thay đổi cách phân vùng cho một ma trận bằng các sử dụng mảng chứa các chỉ số cũng phân vùng

//Hàm kiểm tra tính hợp lệ trong một ô 7x7

public override bool feasibleBlock(Number[,] nb, int x, int y, int number, int value)

{

//Xác định các tập hợp

Number[] group1 = { nb[0, 0], nb[0, 1], nb[0, 2], nb[1, 0], nb[1, 1], nb[1, 2], nb[2, 2] };

Number[] group2 = { nb[0, 3], nb[0, 4], nb[0, 5], nb[0, 5], nb[1, 4], nb[1, 5], nb[1, 6] };

Number[] group3 = { nb[2, 0], nb[2, 1], nb[3, 0], nb[3, 1], nb[4, 0], nb[4, 1], nb[5, 0] };

Number[] group4 = { nb[2, 3], nb[2, 4], nb[3, 2], nb[3, 3], nb[3, 4], nb[4, 2], nb[4, 3] };

Number[] group5 = { nb[1, 7], nb[2, 5], nb[2, 6], nb[3, 5], nb[3, 6], nb[4, 5], nb[4, 6] };

Number[] group6 = { nb[5, 1], nb[5, 2], nb[5, 3], nb[6, 0], nb[6, 1], nb[6, 2], nb[6, 3] };

Number[] group7 = { nb[4, 4], nb[5, 4], nb[5, 5], nb[5, 6], nb[6, 4], nb[6, 5], nb[6, 6] };

if (findGroup(group1, x, y, number, value))

if (findGroup(group2, x, y, number, value))

if (findGroup(group3, x, y, number, value))

if (findGroup(group4, x, y, number, value))

if (findGroup(group5, x, y, number, value))

if (findGroup(group6, x, y, number, value))

if (findGroup(group7, x, y, number, value)) return true;

return false;

}

* **Xử lý vô nghiệm cho ma trận n x n bằng các tạo hàm (vấn đề 1):**

//Hàm kiểm tra tính khả thi của ma trận khi đem nó đi giải

public bool checkFeasible(Number[,] Sudoku, int number)

{

for (int i = 0; i < number; i++)

{

for (int j = 0; j < number; j++)

{

//Xác định tính hợp lệ thông qua hàm feasibleAll kiểm tra hợp lệ hàng, cột và khối cho một ô

if (!feasibleAll(Sudoku, number,i,j, Sudoku[i, j].Value) && Sudoku[i, j].Value != 0 )

return false;

if (Convert.ToInt32(Sudoku[i, j].Value) < 0 || Convert.ToInt32(Sudoku[i, j].Value) > number)

return false;

}

}

//Nếu hợp lệ thì giải ma trận đó

this.solveSudoku(Sudoku, 0, 0, number);

for (int i = 0; i < number; i++)

{

for (int j = 0; j < number; j++)

{

if (Sudoku[i, j].Value == 0) return false;

}

}

return true;

}

Hàm kiểm tra như sau:

public bool feasibleAll(Number[,] Sudoku, int number, int x, int y, int value)

{

//Kiểm tra tính hợp lệ trên hàng x xem có ô nào trùng không, trùng thì báo sai

if (!feasibleRow(Sudoku, number, x,y ,value))

return false;

//Kiểm tra tính hợp lệ trên hàng y xem có ô nào trùng không, trùng thì báo sai

else if (!feasibleColum(Sudoku, number, x,y, value))

return false;

//Kiểm tra xe có bị trùng với ô nào trong khối của nó không

else if (!feasibleBlock(Sudoku, x, y, number, value))

return false;

//Qua các trường hợp trên là hợp lệ

return true;

}

* **Tạo hàm kiểm tra điểm biên và miền giá trị (giải quyết vấn đề 2):**

//Hàm kiểm tra tính khả thi của ma trận n = number khi đem nó đi giải

public bool checkFeasible(Number[,] Sudoku, int number)

{

for (int i = 0; i < number; i++)

{

for (int j = 0; j < number; j++)

{

//Hàm kiểm tra phân vùng ,hàng, cột

if (!feasibleAll(Sudoku, number,i,j,

Sudoku[i, j].Value) && Sudoku[i, j].Value != 0 ) return false;

//Hàm kiểm tra miền giá trị

if (Convert.ToInt32(Sudoku[i, j].Value) < 0 || Convert.ToInt32(Sudoku[i, j].Value) > number) return false;

}

}

this.solveSudoku(Sudoku, 0, 0, number);

for (int i = 0; i < number; i++)

{

for (int j = 0; j < number; j++)

{

if (Sudoku[i, j].Value == 0) return false;

}

}

return true;

}

* **Tạo hàm giải ngẫu nhiên tăng tính đa dạng bằng cách tạo mảng và trộn giá trị mảng, và xuất ngẫu nhiên nghiệm trong các nghiệm của ma trận Sudoku nhiều nghiệm (giải quyết vấn đề 4):**

//Hàm trộn chuỗi để tự động giải sudoku một cách ngẫu nhiên

public int[] autoConvertArray(int number)

{

int[] arrayNum = new int[number];

//Nạp các giá trị có thể vào một mảng

for (int a = 0; a < number; a++)

{

arrayNum[a] = a + 1;

}

//Tạo ra giá trị tự động tên ran

Random ran = new Random();

int x, y, temp;

//Trộn 50 lần

for (int i = 1; i < 50; i++)

{

x = ran.Next(0, number - 1);

y = ran.Next(0, number - 1);

temp = arrayNum[x];

arrayNum[x] = arrayNum[y];

arrayNum[y] = temp;

}

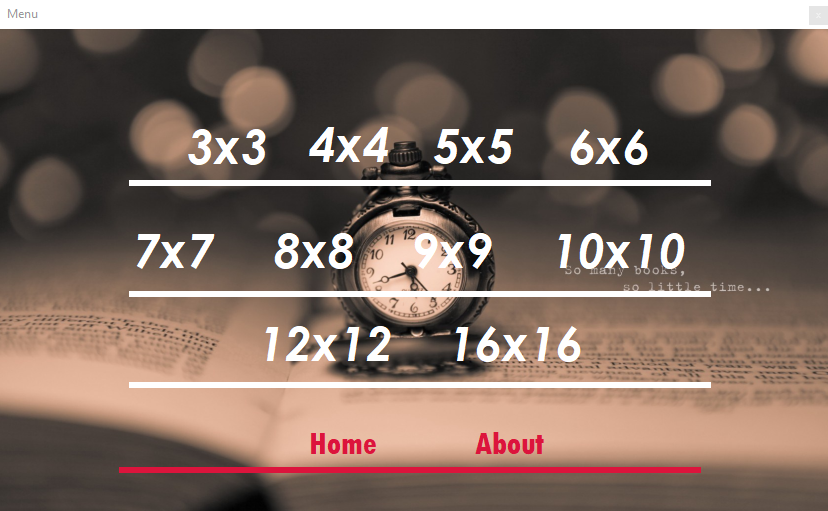
return arrayNum;

}

# kết quả

Từ quá trình cải biến thuật toán ta có phần mềm máy giải tự động một ma trận Sudoku trên giao diện ứng dụng Desktop

**Mở đầu chế độ chơi và Bảng chọn :** Ta có 10 chế độ chơi

**Chế độ giải Sudoku 7x7**



Nút **Check** cho phép kiểm tra tính hợp lệ của ma trận trước khi giải

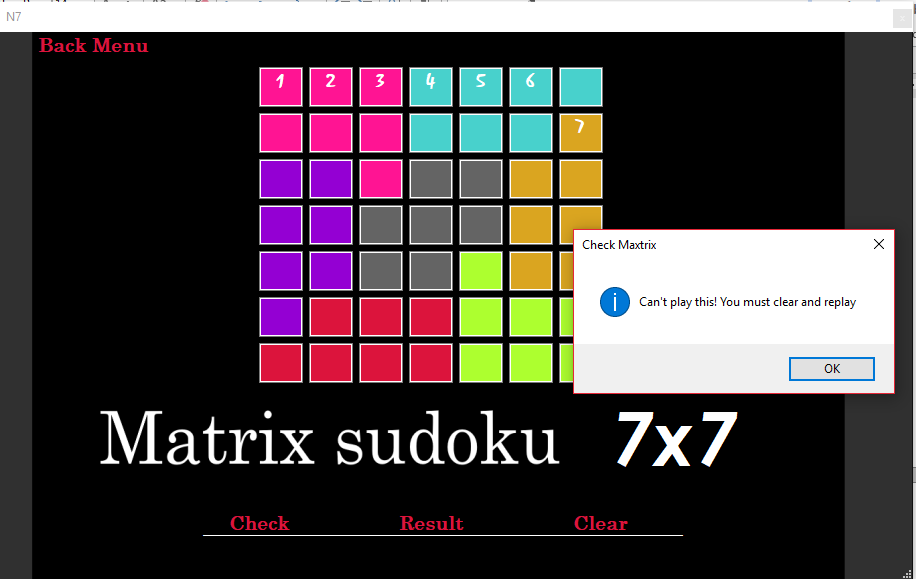
Nút **Result** cho ra một nghiệm ma trận sau khi giải

Nút **Clear** xóa màn hình để tiến hành chơi lại

Nút **Back Menu** trở về màn hình chình

**Demo các trường hợp khi giải**

* Ma trận vô nghiệm



* **Ma trận không hợp lệ về miền giá trị** **tại ô [3-6]**



* **Ma trận hợp lệ và có thể giải**



* **Kết quả ma trận sau khi thực hiện giải**

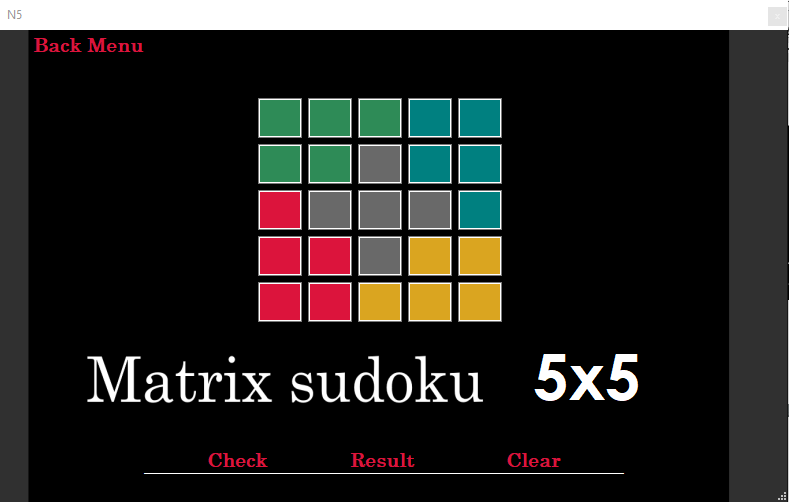
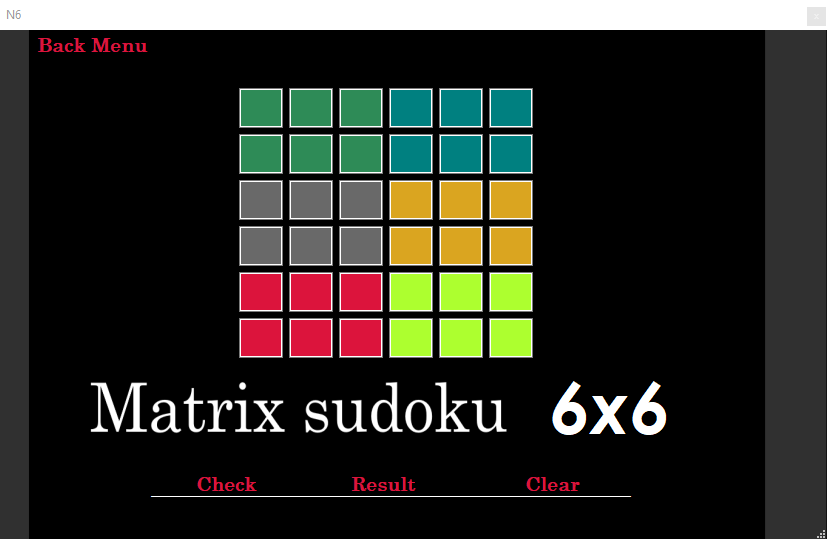


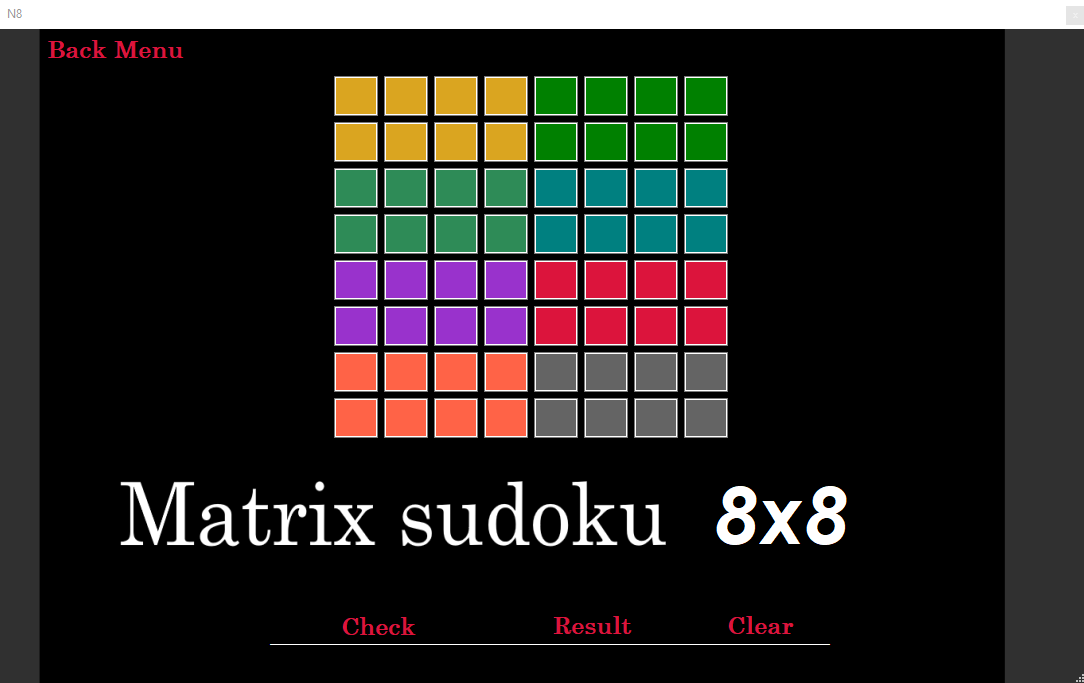
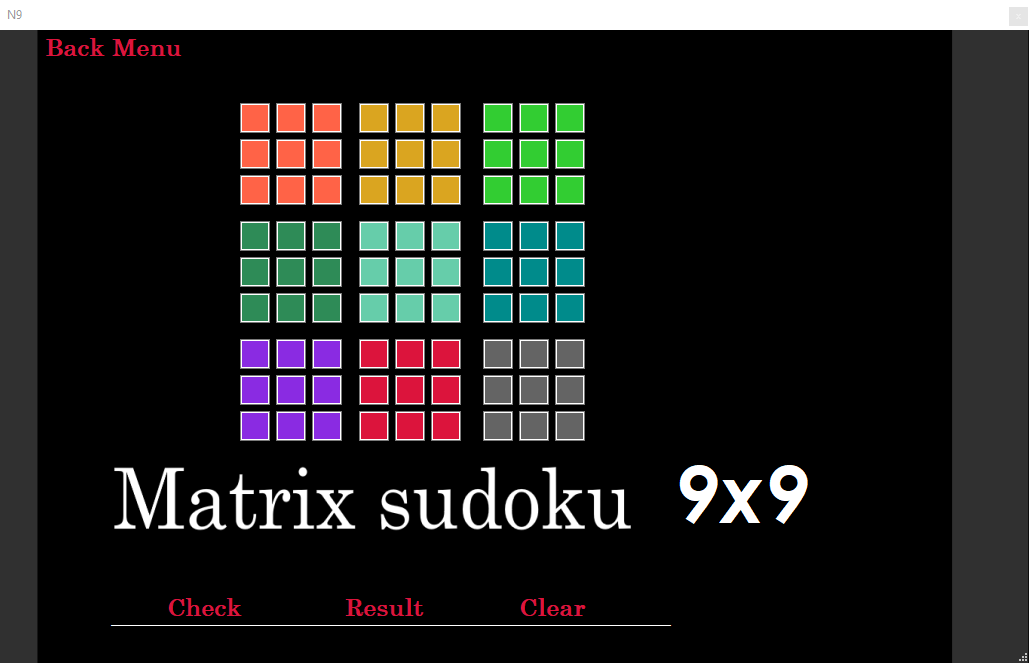
**Với mỗi lần giải người chơi thế thể có mỗi nghiệm khác nhau nếu đề bài đa nghiệm**. Ví dụ cho ô 1-1 có giá trị 1 và ô 1-2 có giá trị 2 ta có nhiều nghiệm sau:

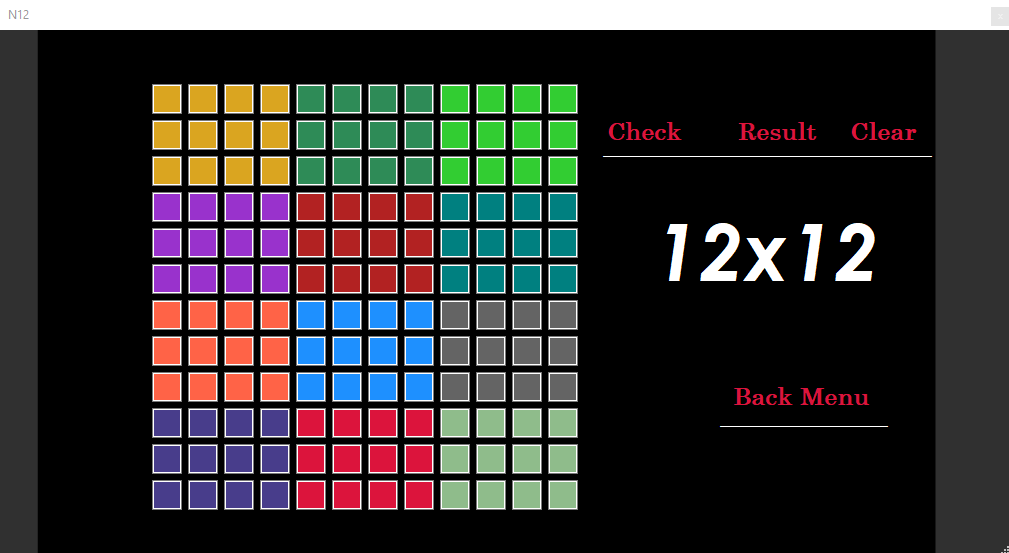
 

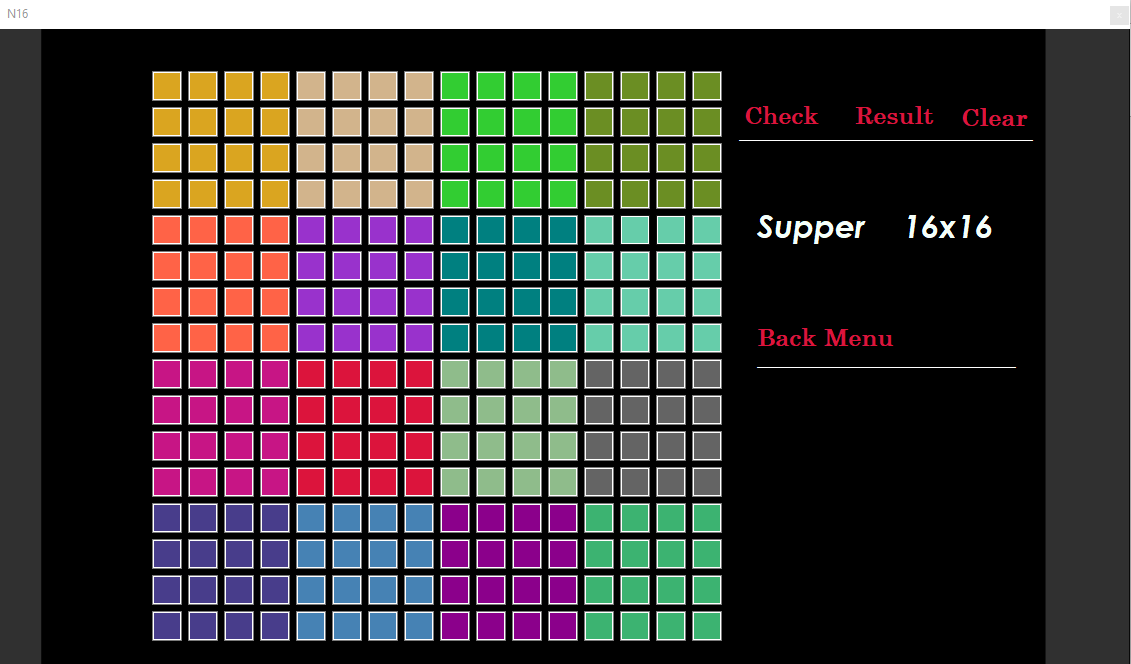
Cách chế độ chơi khách cũng tương tự như trên



* Thông tin phần mềm



# Phụ lục

Hiện tại vẫn tồn tại nhiều phương pháp giải Sudoku được cho là rút ngắn thời gian giải một cách nhanh nhất như các phương pháp:

* **Ô đơn hiện**
* **Ô đơn ẩn**
* **Những sự tương tác giữa khối và cột / khối và hàng.**
* **Các tương tác giữa các khối.**
* **Tập hợp con “hiện”**
* **Tập hợp con “ẩn”**
* **Cánh bướm (Nâng cao)**
* **Chuỗi bắt buộc (nâng cao)**
* **Nishio**
* **Thử và Sai**

**Trên thực tế thì các phương pháp này được thực hiện xoay quanh phương pháp quay lui truyền thống, tuy nhiên chúng có thể giúp con người tận dụng cảm quan logic chọn các hướng đi nhanh hơn, tuy nhiên đa số các phương pháp đều không đem lại một thuật toán đúng cho mọi trường hợp và không có hệ thống thuật toán rõ ràng. Vì lý do này đa số các phương pháp không thể ứng dụng trong lập trình.**

**Nguồn tài liệu thực hiện báo cáo này được lấy từ các trang web:**

<http://noibai.forumotion.com/t199-topic>

<http://www.giaithuatlaptrinh.com/?tag=sudoku>

<https://vi.wikipedia.org/wiki/Sudoku>

<https://text.123doc.org/document/1765986-do-an-cau-truc-du-lieu-va-giai-thuat-giai-sudoku-docx.htm>

**Hết**