

Phiếu chấm điểm

Sinh viên thực hiện:

Họ và tên	Chữ ký	Ghi chú
Hoàng Đức Long		Mã Sv: 18810310452

Giảng viên chấm:

Họ và tên	Chữ ký	Ghi chú
Giảng viên chấm 1:		
Giảng viên chấm 2:		

MỤC LỤC

MỤC LỤC.....	2
MỞ ĐẦU.....	3
CHƯƠNG 1: PHƯƠNG PHÁP QUAY LUI.....	5
1.1. KHÁI NIỆM QUAY LUI:.....	5
1.2. MÔ HÌNH CỦA BÀI TOÁN:.....	6
1.3. ỨNG DỤNG:	7
1.3.1 Phương pháp :	7
1.3.2 Giải thuật tổng quát :	8
CHƯƠNG 2: BÀI TOÁN GIẢI SUDOKU.....	10
2.1.GIỚI THIỆU BÀI TOÁN:.....	10
2.1.1. Lịch sử ra đời và luật chơi:	10
2.1.2.Phát biểu bài toán:	10
2.2. Thiết kế tối ưu thuật toán:.....	11
2.2.1 Biểu diễn sudoku.....	12
2.2.2 Các khả năng lựa chọn cho số nhập vào	12
2.2.3 Lựa chọn xi sao cho hiệu quả	13
2.2.4 Cách ghi nhận trạng thái mới. Trả về trạng thái cũ.....	13
2.2.5 Ghi nhận nhiệm vụ	13
2.2.6 Mô tả.....	14
2.2.7 Đánh giá thuật toán	14
2.2.8. Xây dựng các hàm:.....	15
2.3. GIỚI THIỆU CHƯƠNG TRÌNH GIẢI:.....	16
2.3.1. Tổng quan:	16
2.3.2. Cách thức chơi :	18
CHƯƠNG 3 :CÀI ĐẶT CHƯƠNG TRÌNH THỬ NGHIỆM :.....	19
3.1.Chạy chương trình:	19
3.2.Chạy chương trình kết thúc:	20
KẾT LUẬN.....	23
TÀI LIỆU THAM KHẢO	24

MỞ ĐẦU

LÝ DO CHỌN ĐỀ TÀI:

Ngày nay với sự phát triển không ngừng của khoa học kỹ thuật, đặc biệt trong các ngành mũi nhọn như: Điện Tử - Tin Học - Viễn Thông .v.v. Nếu chúng ta không thường xuyên cập nhật thông tin thì chúng ta không bắt kịp đà phát triển của thế giới. Đất nước ngày càng tiến bộ, khoa học kỹ thuật ngày càng phát triển đòi hỏi con người ngày nay phải có năng lực thật sự, phải có khả năng tư duy logic tốt...để tiếp cận với công nghệ hiện đại một cách nhanh chóng nhất. Vấn đề đặt ra là làm sao để phát triển khả năng tư duy của con người, đôi khi chúng không nhận ra rằng chính những trò chơi trí tuệ, trò chơi với những con số có thể giúp chúng ta rèn luyện trí não, tư duy logic.

Trong khi đó thuật toán quay lui là một thuật toán điển hình để giải lớp bài toán liệt kê hay bài toán tối ưu như: bài toán người giao hàng, bài toán giải sudoku, bài toán 8 hậu, bài toán tìm đường đi trong mê cung, giải sudoku ...Bằng việc liệt kê các tình huống, thử các khả năng có thể cho đến khi tìm thấy một lời giải đúng, thuật toán quay lui chia nhỏ bài toán, lời giải của bài toán lớn sẽ là kết quả của việc tìm kiếm theo chiều sâu của tập hợp các bài toán phần tử.

Nhận thấy tư tưởng của thuật toán rất phù hợp với cách giải của trò chơi, nên đề tài ***”Nghiên cứu phương pháp quay lui và ứng dụng giải bài toán giải SUDOKU”*** được thực hiện nhằm tìm hiểu về thuật toán quay lui và ứng dụng của nó.

MỤC TIÊU, NHIỆM VỤ:

Đề tài này được thực hiện nhằm đạt được mục tiêu là hiểu rõ, hiểu sâu sắc hơn về thuật toán quay lui. Xây dựng thành công chương trình giải sudoku với nhiều đáp án và với một thời gian nhanh nhất.

PHƯƠNG PHÁP NGHIÊN CỨU:

- Tìm hiểu thông tin trên mạng internet, sách, báo, tạp chí...
- Thu thập ý kiến chuyên gia (giáo viên hướng dẫn, các giáo viên trong và ngoài khoa, ý kiến của bạn bè,...).
- Kết hợp nghiên cứu lý thuyết với thực hành, rèn luyện kỹ năng phân tích chương trình và kỹ năng lập trình.

BỐ CỤC CỦA ĐỀ TÀI:

Nội dung của đề tài được trình bày như sau:

MỞ ĐẦU.

Chương 1: PHƯƠNG PHÁP QUAY LUI.

Chương 2: BÀI TOÁN GIẢI SUDOKU.

Chương 3: CÀI ĐẶT CHƯƠNG TRÌNH THỬ NGHIỆM

KẾT LUẬN.

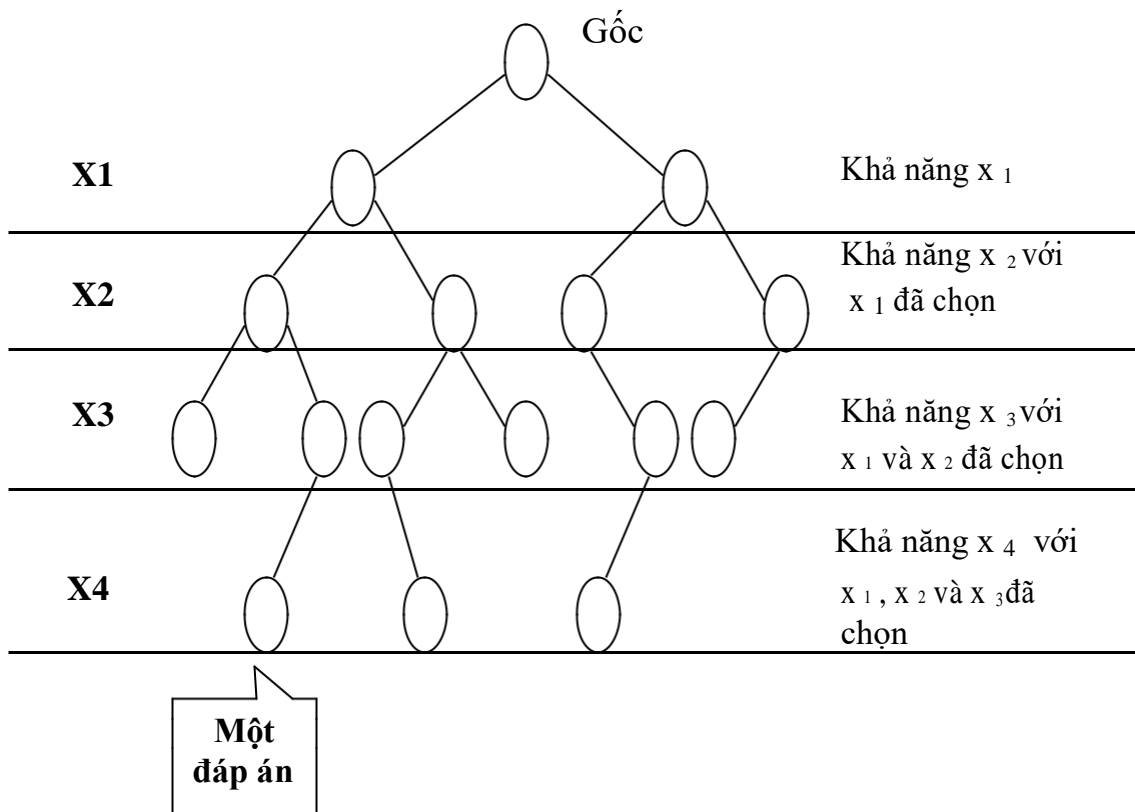
TÀI LIỆU THAM KHẢO.

CHƯƠNG 1: PHƯƠNG PHÁP QUAY LUI

1.1. KHÁI NIỆM QUAY LUI:

Kỹ thuật quay lui (backtracking) như tên gọi của nó, là một quá trình phân tích đi xuống và quay lui trở lại theo con đường đã đi qua. Bằng việc liệt kê các tình huống, thử các khả năng có thể có cho đến khi tìm thấy một lời giải đúng, thuật toán quay lui chia nhỏ bài toán, lời giải của bài toán lớn sẽ là kết quả của việc tìm kiếm theo chiều sâu của các bài toán phần tử. Trong suốt quá trình tìm kiếm nếu gặp phải một hướng nào đó mà biết chắc không thể tìm thấy đáp án thì quay lại bước trước đó và tìm hướng khác kế tiếp hướng vừa tìm đó. Trong trường hợp không còn một hướng nào khác nữa thì thuật toán kết thúc.

Thuật toán quay lui có thể được thể hiện theo sơ đồ cây tìm kiếm theo chiều sâu như hình dưới:



Từ hình vẽ, dễ dàng nhận thấy:

- Ở một bài toán hiện tại (mỗi nốt), ta đi tìm lời giải cho bài toán đó. Ứng với lời giải, ta đi giải bài toán kế tiếp cho đến khi bài toán gốc trở nên đầy đủ.
- Lời giải của bài toán gốc thường là một lối đi từ gốc đến nốt cuối cùng (không có nốt con).

*** Tư tưởng của thuật toán:**

- Nét đặc trưng của kỹ thuật quay lui là các bước hướng tới lời giải cuối cùng của bài toán hoàn toàn được làm thử.
- Tại mỗi bước, nếu có một lựa chọn được chấp nhận thì ghi nhận lại lựa chọn này và tiến hành các bước thử tiếp theo. Còn ngược lại không có lựa chọn nào thích hợp thì làm lại bước trước, xóa bỏ sự ghi nhận và quay về chu trình thử các lựa chọn còn lại.
- Điểm quan trọng của thuật toán là phải ghi nhớ tại mỗi bước đi qua để tránh trùng lặp khi quay lui. Các thông tin này cần được lưu trữ vào một ngăn xếp-Stack (vào sau ra trước), nên thuật toán thể hiện ý thiết kế một cách đệ quy.
- Thuật toán quay lui thường được cài đặt theo lối đệ quy, mỗi lần gọi hàm đệ quy, hàm đệ quy được truyền một tham số (trong các tham số) là chỉ số của bài toán con, trong hàm sẽ cố gắng tìm lời giải cho bài toán con đó, nếu tìm thấy thì gọi hàm đệ quy để giải bài toán con tiếp theo hoặc là đưa ra đáp án bài toán lớn nếu đã đầy đủ lời giải, nếu không tìm thấy thì chương trình sẽ trở về điểm gọi hàm đó. Mục đích của việc sử dụng hàm đệ quy là để thuật toán được rõ ràng, dễ viết, dễ hiểu hơn và cũng để bảo toàn các biến, các trạng thái lúc giải bài toán con.

1.2. MÔ HÌNH CỦA BÀI TOÁN:

Lời giải của bài toán thường biểu diễn bằng một véc tơ gồm n thành phần phải thỏa mãn các điều kiện nào đó. Để chỉ ra lời giải x , ta phải xây dựng dần các thành phần lời giải $x=(x_1, x_2, \dots, x_n)$

Tại mỗi bước i :

- Đã xây dựng xong các thành phần $x=(x_1, x_2, \dots, x_n)$
- Xây dựng thành phần x_i bằng cách lần lượt thử tất cả các khả năng mà x_i có thể chọn:
 - + Nếu một khả năng j nào đó phù hợp cho x_i thì xác định x_i theo khả năng j . Thường phải có thêm thao tác ghi nhận trạng thái mới của bài toán để hỗ trợ cho bước quay lui. Nếu $i = n$ thì ta có được một lời giải, ngược lại thì tiến hành bước $i+1$ để xác định x_{i+1} .
 - + Nếu không có một khả năng nào chấp nhận được cho x_i thì ta lùi lại bước trước (bước $i-1$) để xác định lại thành phần x_{i-1}

1.3. ỨNG DỤNG:

Sử dụng thuật toán quay lui dùng để giải bài toán liệt kê các cấu hình. Mỗi cấu hình được xây dựng bằng cách xây dựng từng phần tử, mỗi phần tử được chọn bằng cách thử tất cả các khả năng.

1.3.1 Phương pháp :

Giả thiết cấu hình cần liệt kê có dạng $(x_1 \ x_2 \ \dots \ x_n)$. Khi đó thuật toán quay lui được thực hiện qua các bước sau :

Bước 1: Xét tất cả các giá trị x_1 có thể nhận, thử cho x_1 nhận lần lượt các giá trị đó. Với mỗi giá trị thử cho x_1 ta sẽ làm tiếp *Bước 2*.

Bước 2: Xét tất cả các giá trị x_2 có thể nhận, lại thử cho x_2 nhận lần lượt các giá trị đó. Với mỗi giá trị thử gán cho x_2 lại xét tiếp các khả năng chọn $x_3 \ \dots$ cứ tiếp tục như vậy đến bước n .

Bước n : Xét tất cả các giá trị x_n có thể nhận, thử cho x_n nhận lần lượt các giá trị đó, thông báo cấu hình tìm được $(x_1 \ x_2 \ \dots \ x_n)$.

1.3.2 Giải thuật tổng quát :

Thuật toán quay lui có thể được mô tả bằng đoạn mã sau (thủ tục này thử cho x_i nhận lần lượt các giá trị mà nó có thể nhận) :

```
void Try (int i)
{
    for <mọi giá trị v có thể gán cho x[i]>
    {
        <Thử cho x[i] bằng giá trị v>

        if <x[i] là phần tử cuối cùng trong cấu hình hoặc i=n>
            <Thông báo cấu hình tìm được>

        else
        {
            <Ghi nhận việc cho x[i] nhận giá trị v (nếu cần thiết)>

            Try(i+1);    // gọi đệ quy chọn tiếp x[i+1]

            <Nếu cần thì bỏ ghi nhận việc thử x[i]=v để thử giá trị khác>
        }
    }
}
```

(Thuật toán quay lui sẽ bắt đầu bằng lời gọi **Try(1)**)

+ Đầu tiên, Try(int i) là hàm đệ quy quay lui trong đó i là biến chạy từ 1 \square n tương ứng với số thứ tự của x1, x2, x3,, xn theo phương pháp đệ quy

+ Tiếp theo vòng lặp for cho các giá trị của từng phần tử x1, x2, x3,..... Tất nhiên, đối với mỗi bài toán riêng thì những giá trị này là khác nhau.

+ Trạng thái cuối tức là điểm dừng của thuật toán này. Thực chất thuật toán quay lui cũng là đệ quy nên việc xác định điểm dừng cực kỳ quan trọng và luôn luôn phải có.

+ Tiếp theo là phần xuất ra, vì như đã thấy, khi if (phần tử cuối cùng trong cấu hình) thì đã xuất ra màn hình và không chạy bất kỳ hàm đệ quy nào nữa. Tuy nhiên, khi xuất ra nó lại xuất tới nhiều dãy số (tùy từng bài tập). Lý do, nếu để ý thì chúng ta sẽ thấy phía trên có 1 vòng for, nó sẽ hoạt động giống như kiểu nhiều vòng lặp for lồng nhau.

for(i = 1; i <= n; i++)

for(j = 1; j < n; j++)

.....

for(z = 1; z < n; z++)

xuat()

Với m vòng for lồng nhau như vậy sẽ xuất ra n^m lần.

Như vậy nếu không đặt điều kiện nào thì vòng for đầu tiên sẽ là m lần, tiếp theo trong mỗi lần đó lại có m lần for khác

Tóm lại, trong thủ tục mô tả trên, điều quan trọng nhất là đưa ra được một danh sách các khả năng đề cử và xác định được giá trị của biểu thức logic [chấp nhận x[i]]. Dễ thấy rằng bài toán vô nghiệm khi ta đã duyệt hết mọi khả năng mà không có khả năng nào thoả mãn yêu cầu. Chú ý rằng là đến một lúc nào đó thuật toán phải lùi liên tiếp nhiều lần. Từ đó suy ra rằng, thông thường bài toán vô nghiệm khi không thể lùi được nữa.

CHƯƠNG 2: BÀI TOÁN GIẢI SUDOKU

2.1 GIỚI THIỆU BÀI TOÁN:

2.1.1. Lịch sử ra đời và luật chơi:

Theo các nguồn tư liệu, lịch sử ra đời của **sudoku** hơi phức tạp. Trò chơi này được kiến trúc sư Howard Garns ở New York thiết kế và được giới thiệu lần đầu tiên vào năm 1979 trên tạp chí Dell (Mỹ) với tên gọi là "Number Place". Tháng 4.1984, Number Place lần đầu tiên được giới thiệu tại Nhật trên báo Monthly Nikolist với tên gọi "Suuji wa dokushin ni kagiru", dịch sang tiếng Anh có nghĩa là "những con số phải độc nhất" hoặc "những con số tìm thấy chỉ một lần" và được "rút gọn" thành sudoku (su = number, doku = single).

Luật chơi của Sudoku khá là đơn giản đó là phải điền kín những ô còn lại với điều kiện:

1. **Các hàng ngang:** Phải có đủ các số từ 1 đến 9, không cần đúng thứ tự. Không trùng số nào ở hàng ngang.
2. **Các hàng dọc:** Đảm bảo có đủ các số từ 1-9, không cần theo thứ tự. Không trùng số nào ở hàng dọc.
3. **Mỗi vùng 3 x 3:** cũng phải có đủ các số từ 1-9. Không trùng số nào trong cùng 1 vùng 3 x3.

2.1.2. Phát biểu bài toán:

Đầu vào : Cho một ma trận 9x9 và một số các số gợi ý ngẫu nhiên.

Yêu cầu : Hãy tìm ra các chữ số còn lại sao cho thỏa mãn yêu cầu.

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

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

2.2. Thiết kế tối ưu thuật toán:

Phát thảo ý tưởng của bài toán như sau:

Try(i) \equiv

for (j = 1 \rightarrow k)

If (xi chấp nhận khả năng k)

{

Xác định xi theo khả năng k;

Ghi nhận trạng thái mới

If($i < n^2$)

Try(i+1);

Else

Ghi nhận nghiệm

Trả lại trạng thái cũ cho bài toán }

► vấn đề cần giải quyết cho bài toán

- Biểu diễn sudoku như thế nào?
- Các khả năng cho lựa chọn xi ?
- Chọn xi như thế nào cho hiệu quả?
- Cách ghi nhận trạng thái mới? trả về trạng thái cũ ?
- Ghi nhận nghiệm ?

2.2.1 Biểu diễn sudoku

Ta biểu diễn bàn cờ vua bằng 1 ma trận vuông cấp 9. `int ip[9][9]`

Quy ước :

- `ip[pos[0]][i] == num and pos[1] != i` : Kiểm tra số nhập vào theo hàng
- `ip[i][pos[1]] == num and pos[0] != i` : Kiểm tra số nhập vào theo cột
- `ip[i][j] == num and (i, j) != pos` : Kiểm tra số nhập vào theo ma trận 3x3

2.2.2 Các khả năng lựa chọn cho số nhập vào

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

Mỗi **NUM** ở đây sẽ có 9 cách chọn tổng sẽ $9*9*9*....$. Vậy tối đa 1 lần giải sudoku mình sẽ có tối $54*9$ trường hợp có thể xảy ra

2.2.3 Lựa chọn xi sao cho hiệu quả

Khi duyệt từng ô của ma trận mình sẽ so sánh số vừa điền vào với cột, hàng và ô box của số đó(sẽ là số nhỏ nhất từ 1 đến 9) rồi duyệt theo chiều ngang

2.2.4 Cách ghi nhận trạng thái mới. Trả về trạng thái cũ.

Để ghi nhận bước đi hợp lệ i ta gán $ip[row][col] = i$

Để hủy 1 bước đi ta gán $ip[row][col] = 0$

2.2.5 Ghi nhận nhiệm vụ

Ma trận ip ghi nhận lại kết quả nghiệm . Nếu tồn tại $h[x][y] = 0$; thì không phải kết quả của bài toán . ngược lại h chứa đường đi cần tìm của nghiệm.

2.2.6 Mô tả

```
Backtracking(k) {  
    for([Mỗi phương án chọn i(thuộc tập D)]) {  
        if ([Chấp nhận i]) {  
            [Chọn i cho X[k]];  
            if ([Thành công]) {  
                [Đưa ra kết quả];  
            } else {  
                Backtracking(k+1);  
                [Bỏ chọn i cho X[k]];  
            }  
        }  
    }  
}
```

2.2.7 Đánh giá thuật toán

- **Ưu điểm** : Việc quay lui là thử tất cả các tổ hợp để tìm được một lời giải. Thế mạnh của phương pháp này là nhiều cài đặt tránh được việc phải thử nhiều trường hợp chưa hoàn chỉnh, nhờ đó giảm thời gian chạy.
- **Nhược điểm** : Trong trường hợp xấu nhất độ phức tạp của quay lui vẫn là cấp số mũ. Vì nó mắc phải các nhược điểm sau:
 - Rơi vào tình trạng “thrashing”: quá trình tìm kiếm cứ gặp phải bế tắc với cùng một nguyên nhân.
 - Thực hiện các công việc dư thừa: Mỗi lần chúng ta quay lui, chúng ta cần phải đánh giá lại lời giải trong khi đôi lúc điều đó không cần thiết.
 - Không sớm phát hiện được các khả năng bị bế tắc trong tương lai. Quay lui chuẩn, không có cơ chế nhìn về tương lai để nhận biết đc nhánh tìm kiếm sẽ đi vào bế tắc.

2.2.8. Xây dựng các hàm:

- Hàm giải chương trình bằng phương pháp quay lui:

```
def solve(ip):
    find = find_empty(ip)
    if not find:
        return True
    else:
        row, col = find

    for i in range(1, 10):
        if valid(ip, i, (row, col)):
            ip[row][col] = i

            if solve(ip):
                return True

            ip[row][col] = 0

    return False
```

- Hàm kiểm tra:

```
def valid(ip, num, pos):

    # Check row
    for i in range(len(ip[0])):
        if ip[pos[0]][i] == num and pos[1] != i:
            return False

    # Check col
    for i in range(len(ip[0])):
        if ip[i][pos[1]] == num and pos[0] != i:
            return False


    # Check box
    box_x = pos[1] // 3
    box_y = pos[0] // 3
    for i in range(box_y * 3, box_y * 3 + 3):
        for j in range(box_x * 3, box_x * 3 + 3):
            if ip[i][j] == num and (i, j) != pos:
                return False

    return True
```

2.3. GIỚI THIỆU CHƯƠNG TRÌNH GIẢI:


2.3.1. Tổng quan:

Giao diện chương trình

 Sudoku — □ ×								
7	8		4			1	2	
6				7	5			9
			6		1		7	8
		7		4		2	6	
		1		5		9	3	
9		4		6				5
	7		3				1	2
1	2				7	4		
	4	9	2		6			7

Màn hình này dùng để nhập vào các thông tin cơ bản là :

- Kích thước SUDOKU
- Các ô còn trống

 Sudoku — □ ×

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

2.3.2. Cách thức chơi :

Đây là màn hình chính dùng chơi trò sudoku. Trên màn hình này ta có thể thực hiện các thao tác:

- Nhập 1 số mình nghĩ là đúng vào ô còn trống
- Kiểm tra xem có số đó có đúng hay không bằng cách ấn phím enter

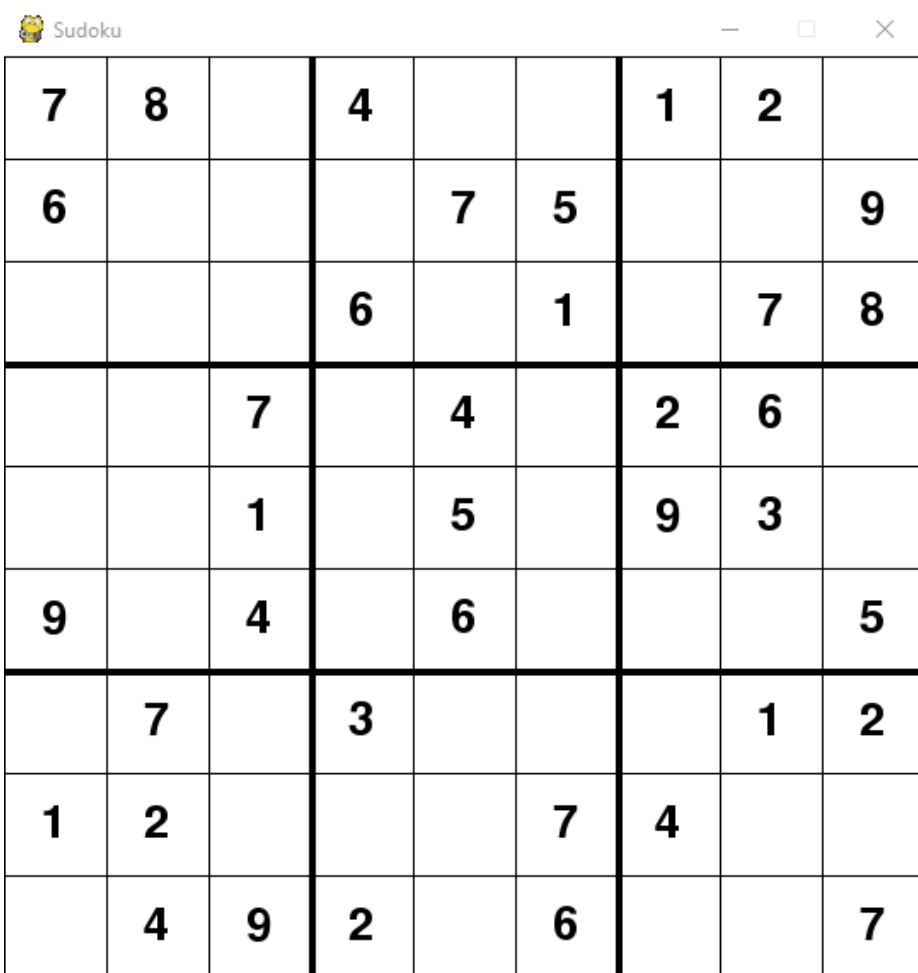
7	8		4			1	2	
6				7	5			9
			6		1		7	8
		7		4		2	6	
		1		5		9	3	
9		4		6				5
	7		3				1	2
1	2				7	4		
	4	9	2		6			7

X

Time: 24:3

CHƯƠNG 3 : CÀI ĐẶT CHƯƠNG TRÌNH THỬ NGHIỆM :

3.1.Chạy chương trình:



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

3.2.Chạy chương trình kết thúc:

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

Khi điền đủ các số thích hợp vào SUDOKU thì chương trình kết thúc ở CMD sẽ hiện lên game over:

```
Success
Success
Success
Success
Success
Success
Success
Success
Success
Success
Success
Success
Success
Success
Success
Success
Success
Success
Success
Success
Success
Success
Success
Success
Success
Success
Success
Success
Game over
```


KẾT LUẬN

I. NHỮNG KẾT QUẢ ĐẠT ĐƯỢC:

Do thời gian thực hiện đề tài luận văn là có hạn và đề tài cũng tương đối phức tạp. Tuy nhiên, với sự nỗ lực của bản thân đồng thời nhờ sự hướng dẫn tận tình của cô Nguyễn Thị Thanh Tân, cùng các thầy cô trong khoa Tin, và bạn bè nên đề tài đã hoàn thành đúng thời hạn quy định và thu được một số kết quả sau:

- Đã xây dựng thành công chương trình giải SUDOKU bằng thuật toán quay lui.
- Tìm hiểu khá kỹ về nội dung và kiến thức của thuật toán quay lui.
- Một kết quả quan trọng nữa là đã thu về cho bản thân những kinh nghiệm, kỹ năng lập trình, nghiên cứu tài liệu và nhiều kiến thức liên quan khác.

II. NHỮNG KẾT QUẢ CHƯA ĐẠT ĐƯỢC:

Bên cạnh những kết quả thu được đã nêu trên thì vẫn còn những kết quả chưa đạt được, đó là:

- Phần lý thuyết về kỹ thuật quay lui còn chưa nghiên cứu được nhiều và nghiên cứu sâu.
- Chương trình giải SUDOKU còn đơn giản, chưa thật sự đẹp.
- Do thời gian thực hiện đề tài còn hạn chế và phần kiểm thử chương trình chưa được kỹ nên chương trình có thể còn có nhiều lỗi và chưa được tối ưu phần mã lệnh.

TÀI LIỆU THAM KHẢO

[1]. Đinh Mạnh Tường, Cấu trúc dữ liệu và giải thuật, NXB khoa học kỹ thuật, 2001

[2] Python cơ bản - Võ Duy Tuấn

[3] Các giáo trình khác về ngôn ngữ lập trình Python.

* Một số Website:

- www.diendantinhoc.com
- www.thuvientinhoc.com
- www.slideshare.net
- <http://congdongcviet.com/>