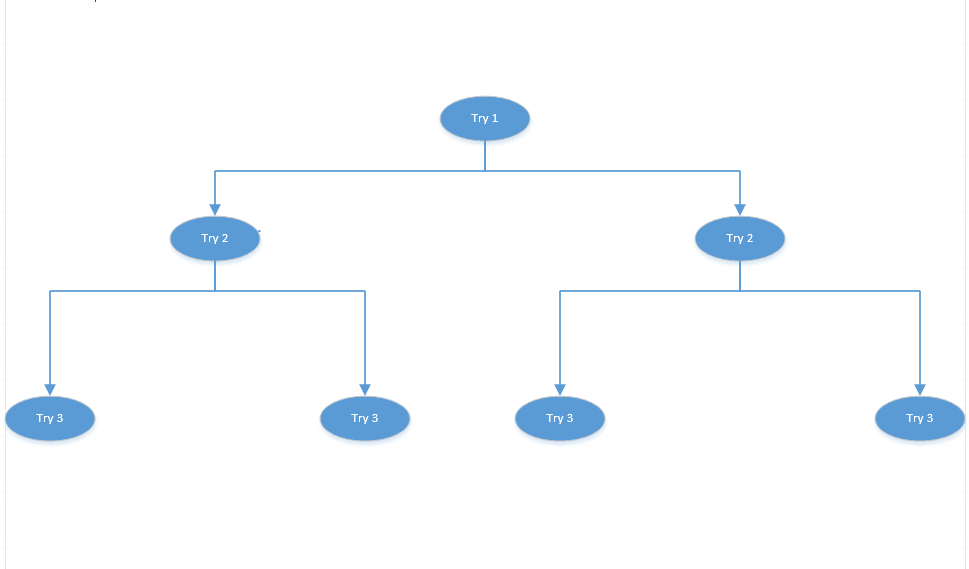
# Giới thiệu

# Lý do chọn đề tài

* Sudoku là một trò chơi từng gây nghiện của các quốc gia phát triển. Độ phức tạp của trò chơi tăng dần khi các số cho trước giảm dần. Vì vậy để chinh phục trò chơi này dễ dàng hơn, người ta đã tìm thuật toán(quay lui) và lập trình nó trên máy tính. Nhờ vào sự phát triển của công nghệ mà máy tính có thể giải một ma trận Sudoku trong thời gian ngắn.
* Từ đó ta cũng có thể viết hoàn thiện một game Sudoku và cho phép người dùng điền vào giá trị và máy tính liên tục kiểm tra các giá trị đó.
* Sở dĩ nhóm chọn đề tài về game Sudoku là vì phần thuật toán của game vô cùng hấp dẫn, cần tư duy logic cao. Ngoài ra, các thành viên trong nhóm cũng từng rất hứng thú với trò chơi này trên giấy. Trò chơi thật sự khó, phải tốn nhiều thời gian và công sức để giải. Vì vậy giải trong vòng vài giây, đối với nhóm mang lại sức hấp dẫn rất lớn.

# Thuật toán quay lui

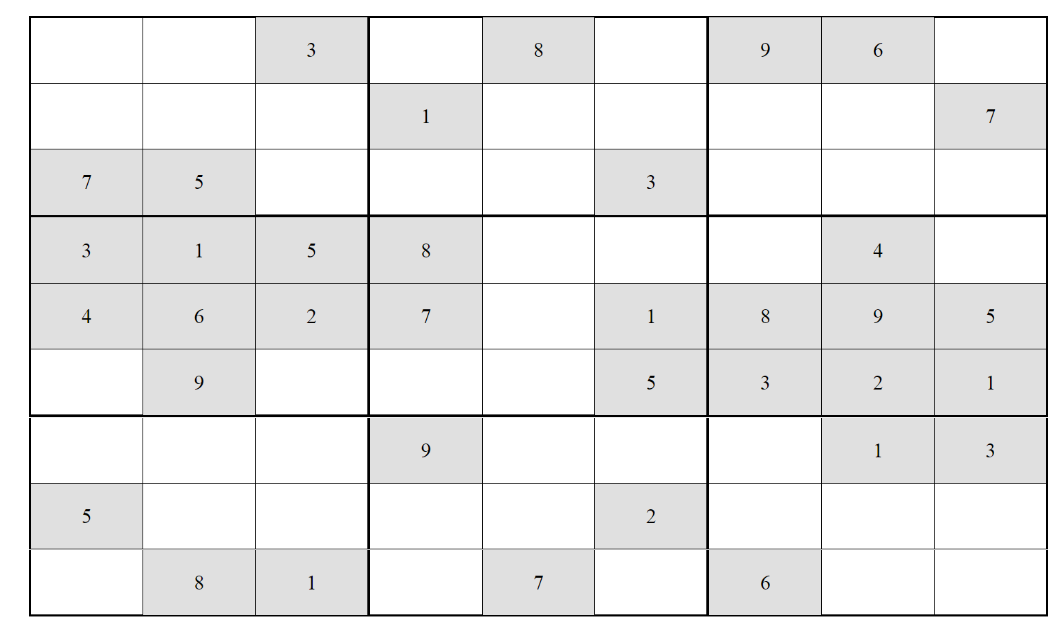
* Thuật toán quay lui là một thuật toán điển hình để giải các bài toán ứng dụng trong tin học. Bằng việc liệt kê các tình huống, thử các khả năng có thể cho đến khi tìm thấy một lời giải đúng, thuật toán quay lui chia nhỏ bài toán, lời giải của bài toán lớn sẻ là kết quả của việc tìm kiếm theo chiều sâu của tập hợp 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 kiếm đó. Trong trường hợp không còn một hướng nào khác nửa thì thuật toán kết thúc.
* Khác với thuật toán tham lam (cũng là điểm mạnh), thuật toán quay lui có điểm khác là nó không cần phải duyệt hết tất cả các khả năng, nhờ đó tránh được các khả năng không đúng nên có thể giảm được thời gian giải. Thuật toán quay lui thường được cài đặt theo lối đệ quy, hàm đệ quy được thực hiện để giải quyết các bài toán con để trả về kết quả của bài toán lớn. 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.



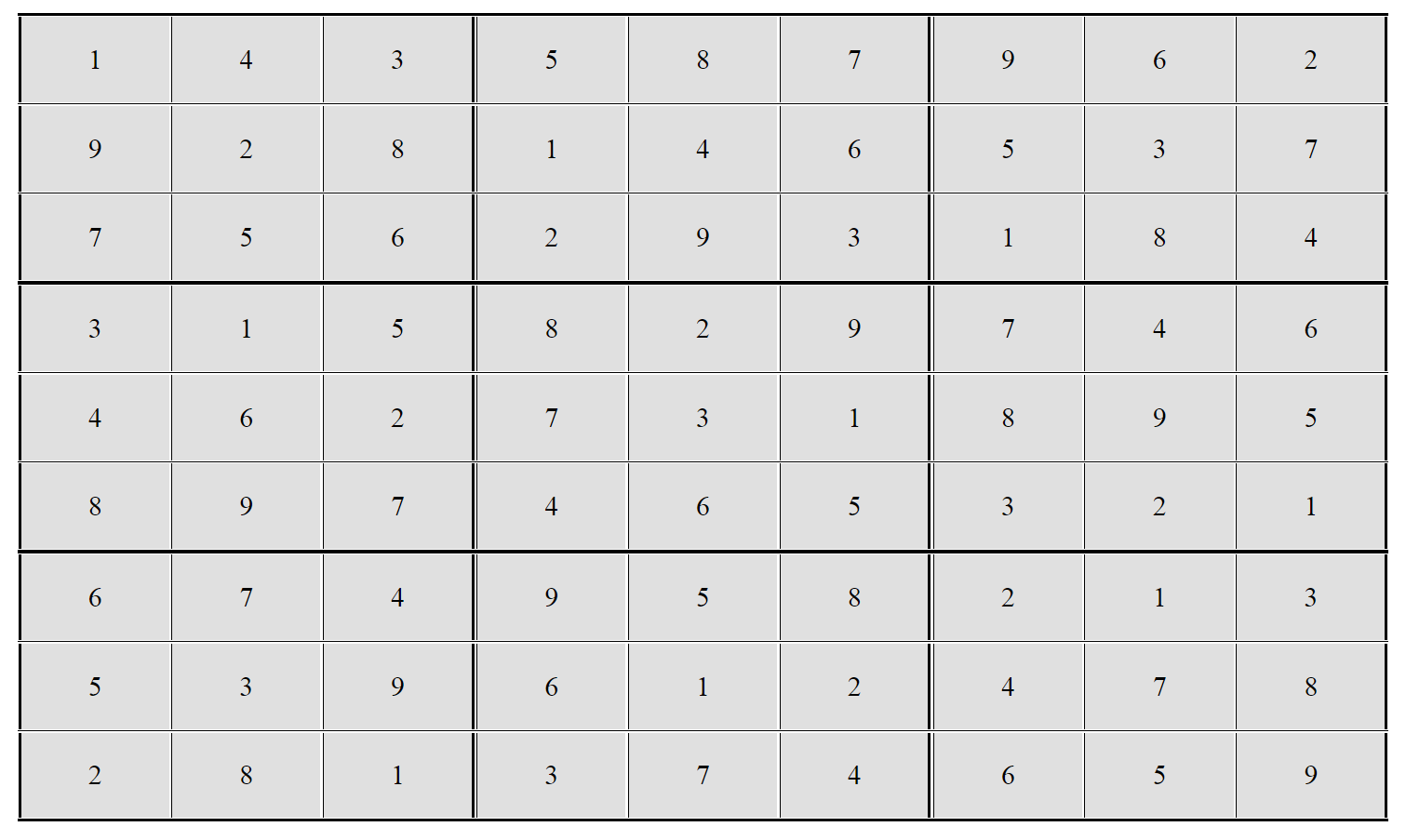
* 1. Sơ đồ thuật toán quay lui.
* Thuật toán quay lui có thể được thể hiện theo sơ đồ cây tìm kiếm theo chiều sâu như hình trên. Từ hình vẽ, ta dễ dàng nhận thấy rằng:
  + - Ở 1 bài toán hiện tại (mỗi nốt), ta đi tìm lời giải cho bài toán đó. Ứng với lời giải, ta đi giải bài toán kế tiếp cho đến lúc 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)

# Giới thiệu bài toán ứng dụng

* Sudoku là một trò chơi trí tuệ nổi tiếng, thu hút nhiều người tham gia đặc biệt là giới trẻ. Ra đời ở Nhật và không lâu sau đã trở nên cực kỳ phổ biến trên thế giới. Quy luật của trò chơi tương đối đơn giản, cho một bàn hình vuông được chia thành một lưới 81 ô nhỏ trên 9 hàng và 9 cột. 81 ô nhỏ đó lại được chia thành 9 vùng, mỗi vùng có 9 ô. Đề bài Sudoku là một bàn hình vuông như thế, trên đó tại một số ô, người ta đã điền sẳn một số giá trị.
* VD: Đây là một ma trận sudoku chưa được giải, với các dòng, các cột, các vùng 3x3 chưa được điền đầy đủ.



* 1. Một ma trận Sudoku chưa được giải.
* Sao khi được tiền đầy đủ thì các dòng, các cột, các vùng ô 3x3 sẽ được lắp đầy với các số từ 1 đến 9 ngẫu nhiên không trùng nhau.
* Hình bên dưới là VD về một ma trận Sudoku được giải. Đáp án được máy tính giải được bằng thuật toán quay lui.

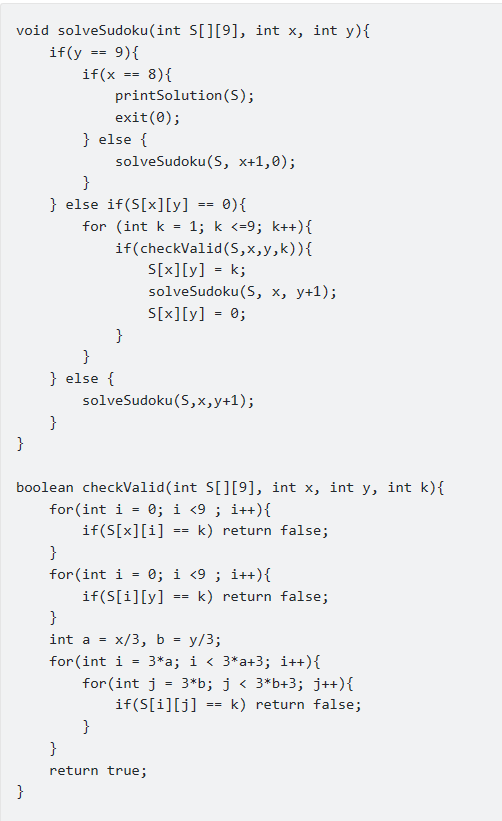


* 1. Ma trận Sudoku đã được giải.

# Giải quyết vấn đề với thuật toán

1. Xây dựng thuật toán giải bằng phương pháp đệ quy

* Như đã nói ở trên, thuật toán Sudoku căn bản là sử dụng phương pháp quay lui. Và điển hình của quay lui và sử dụng đệ quy để giải quyết vòng lặp. Ưu điểm của đệ quy giúp cho việc code trở nên đơn giản hơn rất nhiều, đở tốn công sức để code.
* Ví dụ dưới đây là code thuật toán giải Sudoku bằng đệ quy được viết bằng C++ với :
  + - printSolution (S): In ma trận ra màn hình.
    - solveSudoku(): Hàm giải Sudoku.
    - checkValid(): Hàm kiểm tra.



* Giải thích thuật toán:

1. Hàm kiểm tra ( boollean checkValid ( int S[][9], int x, int y, int k) )

* Các biến được truyền vào bao gồm:
  + - int S[][9] : Ma trận suduku.
    - int x : vị trí dòng
    - int y : vị trí cột
    - int k: số được người chơi điền vào.
* Ta kiểm tra đồng thời 3 đặt điểm của một ma trận suduku:
  + - Kiểm tra dòng (mỗi dòng gồm các chữ số từ 1 đến 9 không trùng nhau): Khởi tạo biến i chạy từ 0 đến 8 để kiểm tra số k vừa điền vào có trùng với giá trị vị trí dòng x cột i(S[x][i]) hay không. Nếu trùng thì trả về false.
    - Kiểm tra cột (mỗi dòng gồm các chữ số từ 1 đến 9 không trùng nhau): Khởi tạo biến i chạy từ 0 đến 8 để kiểm tra số k vừa điền vào có trùng với với giá trị vị trí dòng i cột y (S[x][i]) hay không . Nếu trùng thì trả về false.
    - Kiểm tra vùng 3x3: Kiểm tra xem trong vùng 3x3 với giá trị được thêm vào có trùng với các số có sẵn hoặc đã điền trước đó hay không. Nếu trùng thì trả về false.
    - Nếu cả 3 đều kiện trên đều không trả về false thì trả về true (giá trị k được chấp nhận điền vào).
    - Lưu ý: Giá trị k được điền vào chỉ đúng với trong thời điểm nhất định, có thể không đúng với toàn ma trận.
  1. Hàm giải ( solveSudoku( int S[][9], int x, int y) )
* Bước 1: Nếu x và y đi tới ô cuối cùng ( S[8][8] ), nghĩa là toàn bộ ma trận đã giải xong thì in ra màn hình kết quả của ma trận được giải. Nếu chưa thì chuyển sang bước 2.
* Bước 2: Bước 2 thực chất là gọi lại hàm đệ quy để giải lại bài toán cho đến khi hoàn thành.
  + - Bước 2.1: