HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

**KHOA CÔNG NGHỆ THÔNG TIN**

Logo

Description automatically generatedBÁO CÁO BÀI TẬP LỚN

**NHẬP MÔN TRÍ TUỆ NHÂN TẠO**

**Đề tài: Game Sudoku sử dụng thuật toán quay lui**

|  |  |  |
| --- | --- | --- |
| **Giảng viên** | : | Đào Thị Thuý Quỳnh |
| **Nhóm môn học** | : | 12 |
| **Nhóm thực hiện** | : | 05 |

|  |  |
| --- | --- |
| Họ và Tên | Mã sinh viên |
| Đỗ Văn Quang | B19DCCN522 |
| Nguyễn Chi Nam | B19DCCN450 |

**HÀ NỘI - 2022**

# MỤC LỤC

[MỤC LỤC i](#_Toc104673590)

[DANH MỤC HÌNH ẢNH ii](#_Toc104673591)

[1 TÓM TẮT BÁO CÁO 1](#_Toc104673592)

[2 GIỚI THIỆU VỀ SUDOKU 1](#_Toc104673593)

[2.1 Lịch sử ra đời 1](#_Toc104673594)

[2.2 Hình thức cơ bản 1](#_Toc104673595)

[2.3 Luật chơi sudoku 2](#_Toc104673596)

[3 THUẬT TOÁN QUAY LUI 3](#_Toc104673597)

[3.1 Giới thiệu thuật toán 3](#_Toc104673598)

[3.2 Tư tưởng thuật toán 3](#_Toc104673599)

[3.3 Mô hình thuật toán 3](#_Toc104673600)

[3.4 Áp dụng thuật toán vào trò chơi Sudoku 4](#_Toc104673601)

[4 CÀI ĐẶT TRÒ CHƠI 6](#_Toc104673602)

[4.1 Các chức năng của trò chơi 6](#_Toc104673603)

[4.1.1 Tương tác với người dùng 6](#_Toc104673604)

[4.1.2 Thông báo của trò chơi 6](#_Toc104673605)

[4.1.3 Tạo câu đố random 6](#_Toc104673606)

[4.2 Giao diện trò chơi 8](#_Toc104673607)

[5 KẾT LUẬN 11](#_Toc104673608)

[5.1 Ưu điểm 11](#_Toc104673609)

[5.2 Nhược điểm 11](#_Toc104673610)

# DANH MỤC HÌNH ẢNH

[Hình 1 Câu đố sudoku 1](file:///C:\Users\dovan\Desktop\BaoCaoNMTTNT.docx#_Toc104671927)

[Hình 2 Câu đố Sudoku đã được giải 2](#_Toc104671928)

[Hình 4 Câu đố sudoku với các ô trống 4](#_Toc104671929)

[Hình 6 Câu đố sudoku được giải 5](#_Toc104671930)

[Hình 7 Giao diện chính của trò chơi 8](#_Toc104671931)

[Hình 8 Thông báo “Giá trị không chính xác!” 9](#_Toc104671932)

[Hình 9 Thông báo chiến thắng 9](#_Toc104671933)

[Hình 10 Thông báo “Đã giải xong câu đố!” 10](#_Toc104671934)

[Hình 11 Thông báo không thể tìm thấy lời giải!” 10](#_Toc104671935)

# TÓM TẮT BÁO CÁO

Trong bài báo cáo này, chúng em đưa ra thuật toán sau: quay lui có điều kiện. Thuật toán này được sử dụng để giải trò chơi Sudoku dựa theo một số các phần tử đã được xác định trước. Mặc dù tồn tại khá nhiều phương pháp để giải trò chơi Sudoku nhất định, nhưng với thuật toán này có thể giúp rút gọn thời gian chạy chương trình ngắn hơn so với một số thuật toán khác.

# GIỚI THIỆU VỀ SUDOKU

## Lịch sử ra đời

Mang tên gọi đầu tiên là “Number Place” hay “Đặt số vào vị trí đúng”, trò chơi xuất phát từ nước Mỹ xa xôi. Được đăng trong những cuốn sách câu đố vào cuối thập niên 70 của thế kỷ 19, trò chơi chỉ thực sự trở thành xu thế khi được xuất bản dưới cái tên “Sudoku” bởi nhà phát hành trò chơi Nikoli, Nhật Bản vào năm 1989.

Từ Sudoku (数独すうどく) xuất phát từ Nhật Bản và bao gồm các ký tự tiếng Nhật, trong đó “su” (có nghĩa là số) và “Doku” (có nghĩa là 'đơn/ một mình'). Ngày nay, nó là một trong những câu đố phổ biến nhất thu hút người già lẫn trẻ. Do tính chất gây nghiện và đầy thử thách, nó đã lan rộng như một trận cháy rừng trên toàn cầu và thu hút sự chú ý của nhiều nhà nghiên cứu, những người đang cố gắng thiết kế các thuật toán để giải quyết nó bằng cách áp dụng các cách tiếp cận khác nhau.

Không chỉ các kỹ thuật xác định, mà còn nỗ lực thiết kế các phương pháp heuristics như imulated annealing (Lewis, 2007) và Genetic Algorithms (Mantere và Koljonen, 2006) để giải các câu đố Sudoku. Cơ sở của Sudoku được cho là xuất hiện từ thế kỷ 18 khi nhà toán học vĩ đại người Thụy Sĩ Leonard Euler đưa ra ý tưởng về Hình vuông Latinh vào năm 1783.

## Hình thức cơ bản

Text, table

Description automatically generated Sudoku là câu đố trí tuệ có hình dạng lưới 9x9, gồm những ô vuông nhỏ tạo thành một ô vuông lớn, mỗi 9 ô vuông nhỏ sẽ tạo thành một khối vuông 3x3, lần lượt ta có 9 khối vuông tạo thành một ô vuông lớn hoàn chỉnh. Nhiệm vụ của người chơi vô cùng đơn giản, điền các con số có một chữ số vào vị trí còn trống trên lưới.

Hình 1 Câu đố sudoku

Một lưới Sudoku sẽ được điền trước những con số ban đầu, là chìa khóa để người chơi điền thêm những chữ số tiếp theo. Khi đã hoàn thành lời giải, lưới Sudoku sẽ trở thành một ma trận hình vuông Latin, tuân theo quy luật: mỗi số nguyên duy nhất sẽ không xuất hiện hai lần trong cùng một hàng, cột hoặc bất kỳ một trong chín khối vuông 3×3 nào của bảng trò chơi 9x9.

Text

Description automatically generated

Hình 2 Câu đố Sudoku đã được giải

Các số tĩnh được đưa ra trong Hình 1 vẫn giữ nguyên vị trí và giá trị của chúng trong hình 2. Trong đáp án, mỗi hàng, cột và bình phương con của giải pháp chứa các số nguyên từ 1 đến 9 chính xác một lần. Ngoài ra, người ta đã báo cáo rằng tổng số câu đố Sudoku 9 × 9 duy nhất có thể được tạo ra là 6.670.903.752.021.072.936.960 (~ 6,67 × 10 21) (Felgenhauer và cộng sự, 2006) mỗi câu đố có một giải pháp duy nhất.

## Luật chơi sudoku

Luật chơi vô cùng đơn giản, tất cả những gì bạn cần làm là điền những chữ số từ 1 tới 9 vào những vị trí phù hợp, tuân theo những quy luật đã được định sẵn. Mỗi một ô vuông nhỏ trong lưới sẽ thuộc một hàng, một cột và một khối 3x3 tương ứng. Do vậy, chữ số được điền vào ô trống phải đảm bảo đủ ba điều kiện: là chữ số duy nhất trong hàng, trong cột và trong khối đó. Ví dụ khi bạn điền số 7 vào ô trống, trong hàng dọc sẽ chỉ được có một chữ số 7, tương tự với trong cột và trong khối vuông 3x3.

Một lời giải hoàn chỉnh sẽ bao gồm 3 yếu tố:

* Mỗi hàng ngang đều phải có đủ 9 chữ số (từ 1 tới 9), không cần theo đúng thứ tự.
* Mỗi hàng dọc phải có đủ 9 chữ số.
* Trong mỗi khối vuông 3x3 (tổng cộng 9 khối) cũng phải có mặt đủ 9 chữ số khác nhau.

# THUẬT TOÁN QUAY LUI

## Giới thiệu thuật toán

Quay lui (backtrack) là một kỹ 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. Người đầu tiên đề ra thuật toán này là nhà toán học người Mỹ D.H.Lehmer vàoo những năm 1950.

## Tư tưởng thuật toán

Dùng để giải bài toán liệt kê các cấu hình. Mỗi cấu hình được xây dựng bằng từng phần tử. Mỗi phần tử lại được chọn bằng cách thử tất cả các khả năng.

Các bước trong việc liệt kê cấu hình dạng X[1...n]:

* Xét tất cả các giá trị X[1] có thể nhận, thử X[1] nhận các giá trị đó. Với mỗi giá trị của X[1] ta sẽ:
* Xét tất cả giá trị X[2] có thể nhận, lại thử X[2] cho các giá trị đó. Với mỗi giá trị X[2] lại xét khả năng giá trị của X[3]...tiếp tục như vậy cho tới bước:
* ...
* ....
* Xét tất cả giá trị X[n] có thể nhận, thử cho X[n] nhận lần lượt giá trị đó.
* Thông báo cấu hình tìm được.

Bản chất của quay lui là một quá trình tìm kiếm theo chiều sâu(Depth-First Search).

## Mô hình thuật toán

Mã giả cho thuật toán quay lui:

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]];

}

}

}

}

## Áp dụng thuật toán vào trò chơi Sudoku

Ý tưởng: Mỗi bước tìm tập các giá trị khả dĩ để điền vào ô trống và sau đó đệ quy để điền ô tiếp theo.

Các bước của bài toán được thực hiện như sau:

B1: Đầu tiên, chúng ta tìm kiếm tất cả các ô trống (value = 0) vào lưu vị trí của các ô trống vào 1 danh sách.

# Tim vi tri để giải dau tien

def FirstPos(gridArray):

    a = []

    b = []

    for k in range(81):

        a.append({1,2,3,4,5,6,7,8,9})

    for i in range(9):

        for j in range(9):

            if gridArray[i][j] == 0 :

                b.append(i\*9+j)

                for ii in range(9):

                    a[i\*9+j].discard(gridArray[ii][j])

                    a[i\*9+j].discard(gridArray[i][ii])

                ii = i // 3

                jj = j // 3

                for i1 in range(ii \* 3, ii \* 3 + 3):

                    for j1 in range(jj \* 3, jj \* 3 + 3):

                        a[i\*9+j].discard(gridArray[i1][j1])

    min = 10

    x = -1

    y = -1

    se = {}

    for k in range(81):

        if (len(a[k]))<min and gridArray[k//9][k%9]==0:

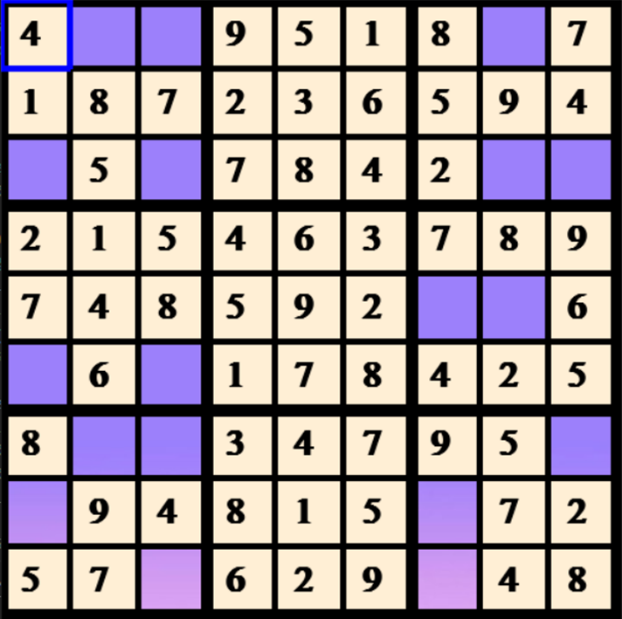
            x = k//9

            y = k%9

            min = len(a[k])

            se = a[k].copy()

    return (x,y,se)

**

Hình 4 Câu đố sudoku với các ô trống

B2: với mỗi ô trong danh sách, xét các giá trị nằm trong các hàng, cột và hộp <là hộp gồm 9 ô như trong giao diện> từ đó suy ra tập các giá trị thỏa mãn của ô được xét và lưu nó vào list.

B3: tìm ô có len(list) min và điền giá trị thỏa mãn (là các giá trị trong list).

# Giai dua tren thuat toan quay lui

def SolveGrid(gridArray, i, j):

    global IsSolving

    IsSolving = False

    i = FirstPos(gridArray)[0]

    j = FirstPos(gridArray)[1]

    print(str(i)+" "+str(j))

    if i==-1 and j==-1:

        return True

    pygame.event.pump()  # được gọi mỗi vòng lặp

    for V in range(1, 10):  # thử các giá trị từ 1-9

        if IsUserValueValid(gridArray, i, j, V):  # Nếu giá trị khả dụng, thêm nó vào ô

            gridArray[i][j] = V

            if SolveGrid(gridArray, i, j):  # Nếu giá trị là chính xác, hãy giữ lại

                return True

            else:  # Nếu sai xóa nó khỏi ô, để ô trống

                gridArray[i][j] = 0

        screen.fill((255, 255, 255))

        DrawGrid()

        DrawSelectedBox()

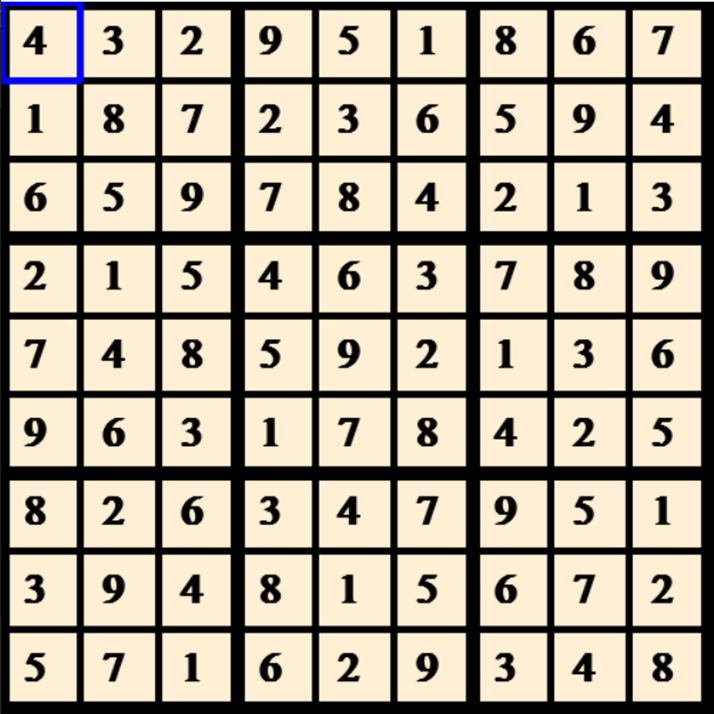
        DrawModes()

        pygame.display.update()

        pygame.time.delay(0)

    return False

B4: Thực hiện lặp 3 bước trên, đến khi hết ô trống thì dừng lại.



Hình 6 Câu đố sudoku được giải

# CÀI ĐẶT TRÒ CHƠI

## Các chức năng của trò chơi

### Tương tác với người dùng

Các phím chức năng người dùng có thể nhấn:

* C: Xoá toàn bộ câu đố
* X: Xoá một ô tại vị trí đã chọn
* E: Mở câu đỗ mức độ dễ
* A: Mở câu đố mức độ trung bình
* H: Mở câu đố random
* Space: Tự động giải câu đố

Người dùng có thể chọn các ô trống và nhập số để giải câu đố.

### Thông báo của trò chơi

* Kết thúc trò chơi:
  + Người chơi điền hết các ô trống: thông báo “Chiến thắng!”
* Người chơi chọn tự động giải:
  + Nếu giải thành công: Thông báo “Đã giải xong câu đố!”
  + Nếu không tìm được lời giải: Thông báo “Không tìm thấy lời giải!”
* Trong khi chơi, nếu người dùng điền số không thoả mãn: thông báo “Giá trị không chính xác”

### Tạo câu đố random

Để tạo một câu đố random chắc chắc có lời giải sẽ được thực hiện theo các bước:

* Tạo một ma trận trống để chứa câu hỏi, giá trị hàng đầu tiên của ma trận là hoán vị của các phần tử [1, 2, 3, 4, 5, 6, 7, 8, 9]

7 5 4 | 1 2 6 | 9 3 8

0 0 0 | 0 0 0 | 0 0 0

0 0 0 | 0 0 0 | 0 0 0

- - - - - - - - - - - - -

0 0 0 | 0 0 0 | 0 0 0

0 0 0 | 0 0 0 | 0 0 0

0 0 0 | 0 0 0 | 0 0 0

- - - - - - - - - - - - -

0 0 0 | 0 0 0 | 0 0 0

0 0 0 | 0 0 0 | 0 0 0

0 0 0 | 0 0 0 | 0 0 0

* Dùng giải thuật quay lui (bên trên) để giải ma trận câu hỏi:

7 5 4 | 1 2 6 | 9 3 8

1 2 3 | 4 8 9 | 5 6 7

6 8 9 | 3 5 7 | 1 2 4

- - - - - - - - - - - - -

2 1 5 | 6 3 4 | 7 8 9

3 4 6 | 7 9 8 | 2 1 5

8 9 7 | 2 1 5 | 3 4 6

- - - - - - - - - - - - -

4 3 8 | 5 7 1 | 6 9 2

5 6 1 | 9 4 2 | 8 7 3

9 7 2 | 8 6 3 | 4 5 1

* Random ra số lượng các ô hiện, random vị trí các ô hiện, lưu lại ma trận mới:

0 0 6 | 0 3 8 | 0 0 9

5 0 0 | 1 4 9 | 3 6 0

4 0 9 | 5 0 7 | 0 1 2

- - - - - - - - - - - - -

1 4 3 | 0 7 2 | 5 9 8

2 8 5 | 3 0 1 | 7 0 6

6 0 7 | 4 0 0 | 1 0 3

- - - - - - - - - - - - -

9 0 0 | 0 2 3 | 6 8 0

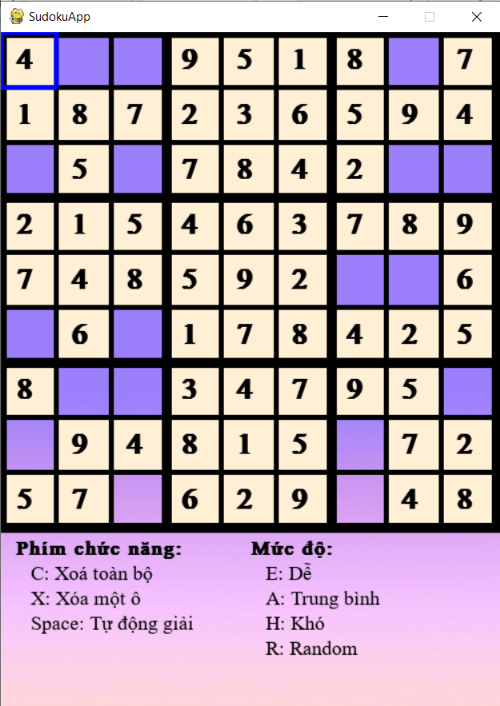
3 0 2 | 8 1 0 | 0 0 0

0 7 4 | 0 5 0 | 0 3 1

* Tạo thành câu hỏi hoàn chỉnh

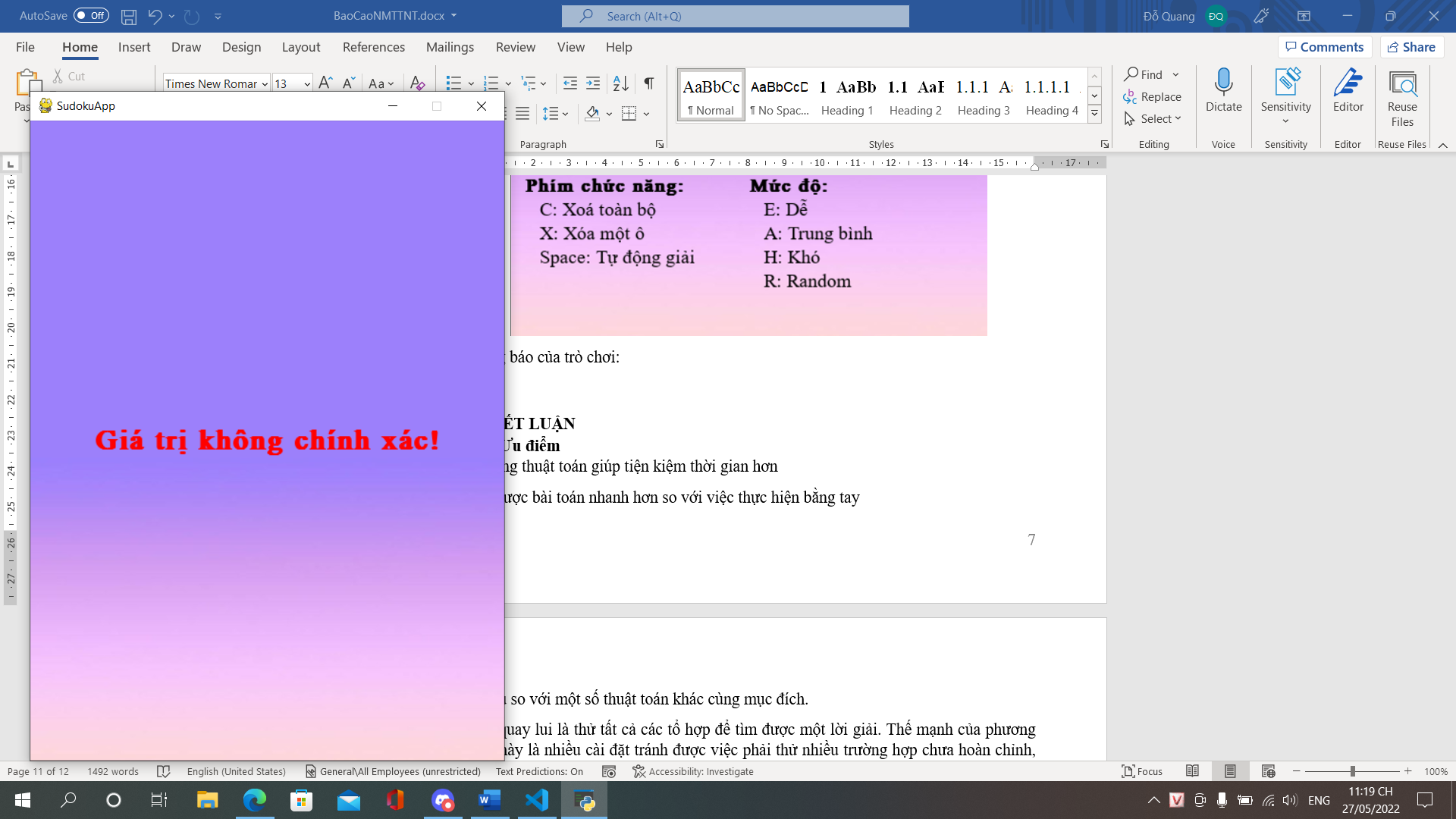
## Giao diện trò chơi

Giao diện chính của trò chơi:



Hình 7 Giao diện chính của trò chơi

Thông báo của trò chơi:



Hình 8 Thông báo “Giá trị không chính xác!”

Background pattern

Description automatically generated with medium confidence

Hình 9 Thông báo chiến thắng

Background pattern

Description automatically generated with medium confidence

Hình 10 Thông báo “Đã giải xong câu đố!”

Background pattern

Description automatically generated with medium confidence

Hình 11 Thông báo không thể tìm thấy lời giải!”

# KẾT LUẬN

## 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

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.