

**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH**

**KHOA CÔNG NGHỆ THÔNG TIN**



**MÔN HỌC: CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT**

**TIỂU LUẬN CUỐI KỲ**

**MATROID PARTITIONING**

**GVHD:** TS. Huỳnh Xuân Phụng

**SVTH:**

1. Phạm Bá Thành 21110923
2. Lê Quang Lâm 21110894
3. Đậu Nguyễn Nam Sơn 21110628

(DASA230179\_08)

**Mã lớp học:**

**DASA230179\_09**

**HỌC KÌ I NĂM HỌC 2022 - 2023**

Thành phố Hồ Chí Minh, tháng 12 năm 2022

**DANH SÁCH THÀNH VIÊN THAM GIA LÀM TIỂU LUẬN**

*Nhóm 16*

**Đề tài:** Matroid Partitioning

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Họ và tên** | **MSSV** | **Tỉ lệ hoàn thành (%)** |
| 1 | Phạm Bá Thành | 21110923 | 100% |
| 2 | Lê Quang Lâm | 21110894 | 100% |
| 3 | Đậu Nguyễn Nam Sơn | 21110628 | 100% |

**Nhận xét của giảng viên:**

….……………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………

**Điểm:** ………………

*Ngày 30 tháng 12 năm 2022*

*Giảng viên ký tên*

**MỤC LỤC**

# LÝ THUYẾT VỀ MATROID

# NGUỒN GỐC THUẬT TOÁN

# PHÂN TÍCH THUẬT TOÁN

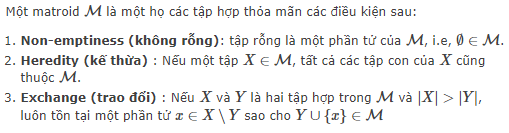
# ỨNG DỤNG TRONG GAME SODOKU

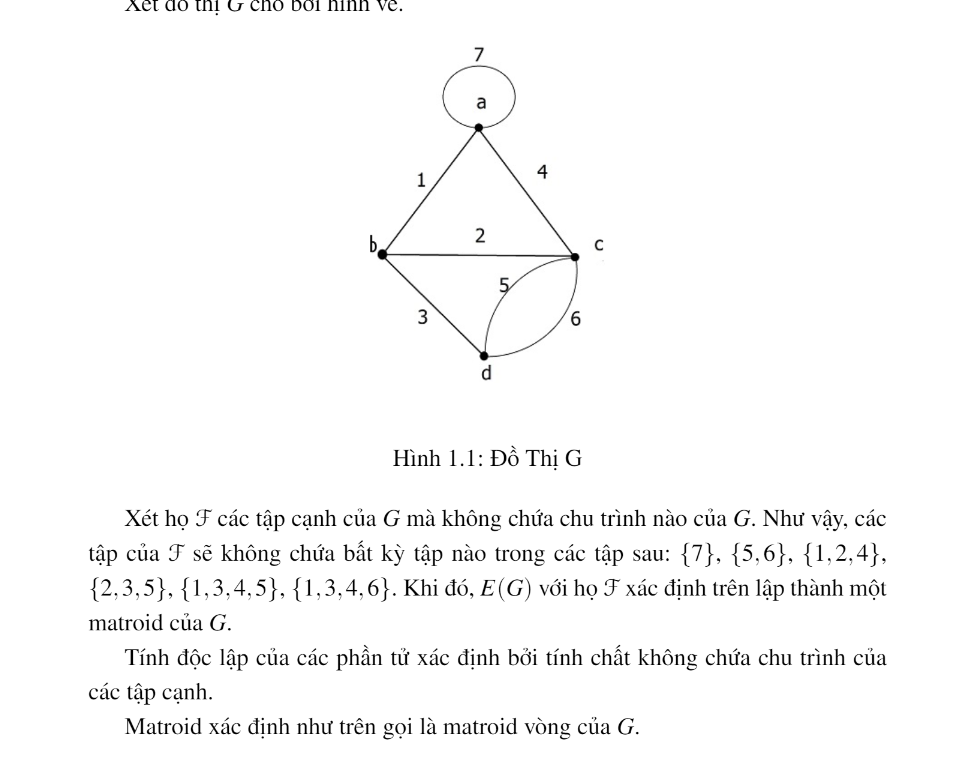
# THU HOẠCH KẾ QUẢ

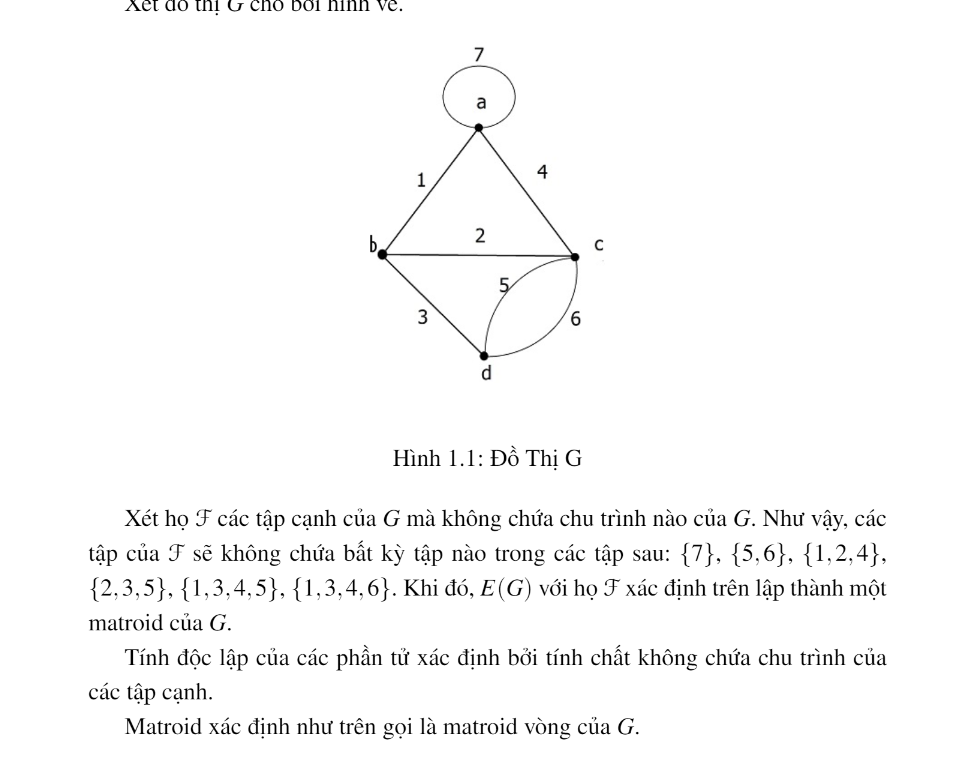
# SƠ ĐỒ KHỐI

# CODE

1. **Lý thuyết về Matroid**



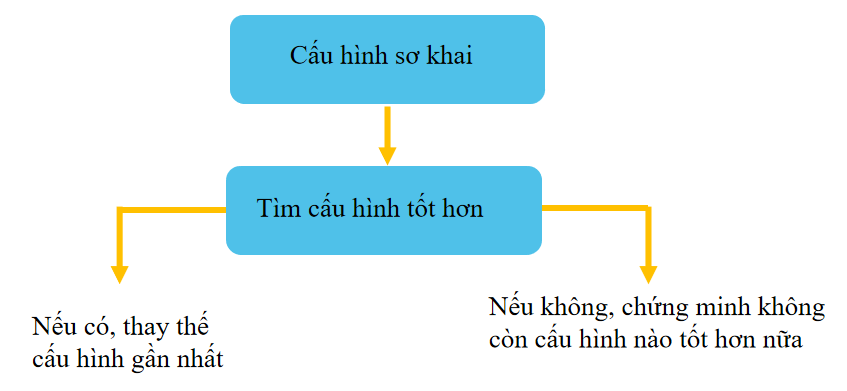




1. **Nguồn gốc thuật toán**

Cách tự nhiên để các phần tử được tô màu , các phần tử lúc bắt đầu màu của chúng được bỏ trống, và màu của các phần tử lần lượt được tô.

Sự cải tiến của thuật toán



1. **Phân tích thuật toán**

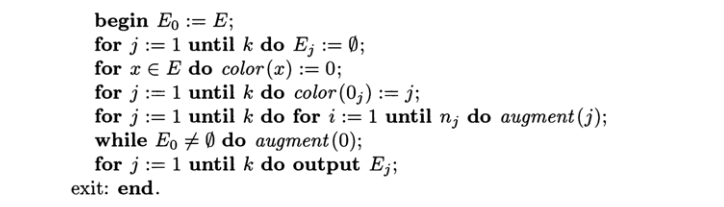
E : là một tập hữu hạn

k : là số tập con độc lập của E

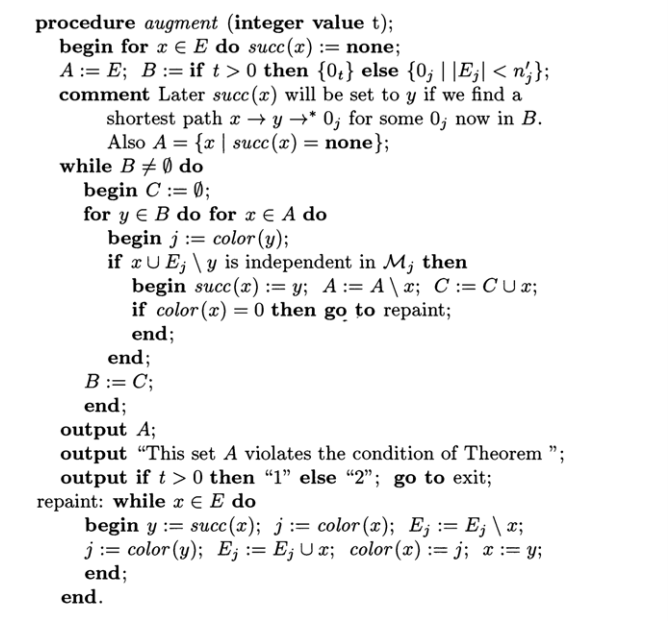
0j: là một phần tử tiêu biểu của màu j, màu của nó không xóa bỏ được

E1,E2,....,Ek : là các tập con độc lập của E

Color(x): màu của phần tử x



**Hàm augment ()**



Succ(x): hàm chỉ số nguyên kế tiếp x

Thực hiện hàm augment(1)

**B1:** cho tất cả phần tử thuộc E , có succ=null;

**B2:** Gán A=E;

t=1 >0 nên B={01};

**B3:** Vòng lặp(B!= null)

Gán C=null;

.Với y=01;

.x=e1(x chạy từ e1 đến e7);

.gán j=color(y=01);

.nếu xUE1\y độc lập trên matroid M

thì succ(x)=y; A=A\x; C=CUx ;

nếu màu x =0 thì repaint();

Repaint(x): Vòng lặp(dừng khi x ko thuộc E)

y= succ(x);

Repaint(x): Vòng lặp(dừng khi x ko thuộc E)

y= succ(x); j= màu x=0;

E0=E0\x;

j=màu y=1;

E1=E1Ux ;

màu x = 1;

x=y=01;

Tiếp tục B3 của hàm augment(1):

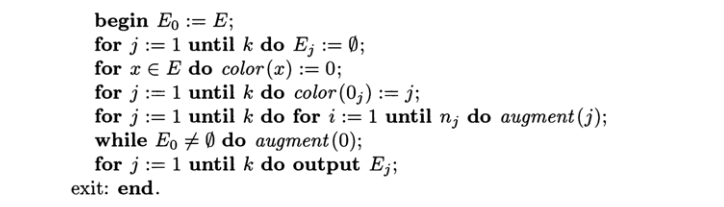
Chạy hết 2 vòng for

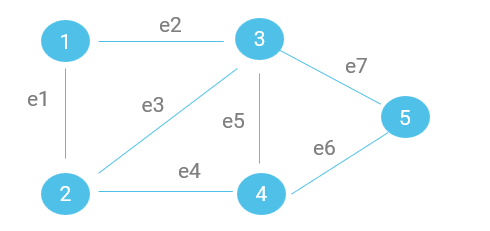
thì gán B=C;

Trong trường hợp xUEj\y phụ thuộc thì B=C=null;

Vòng lặp while sẽ dừng lại

**Hàm main()**





\*Cho tập E là các cạnh của đồ thị vô hướng

E={e1,e2,e3,e4,e5,e6,e7};

Yêu cầu : Tô màu các cây khung để bao quanh được hết đồ thị ( số lượgn cây khung phải nhỏ nhất)

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

**B1:** E0= E

**B2 :** Cho tất cả tập con của E = null;

**B3:** Cho tất cả các phần tử trong E có màu 0;

**B4:** Gán màu các phần tử 0j lần lượt bằng j (j->k);

**B5:** Với j=1,i=1;

Thực hiện hàm augment(1);

Mục đích của B5 là tìm tập hợp A chứa các phần tử ko độc lập, trái với các định lí . Để tìm các tập con độc lập ta qua B6

**B6:** Vòng lặp(dừng khi E0= null)

thực hiện hàm augment(0) cho tới khi dừng ;

**B7:** xuất các tập con độc lập E1.....Ek

Kết thúc thuật toán tìm được các tập cây khung E1 E2

Với

E1={e2,e3,e5,e7} màu đỏ

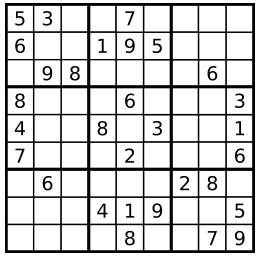
E2={e1,e3,e4,e6} màu vàng

E1 U E2 = E

Vậy tìm được ít nhất 2 cây khung để bao phủ toàn bộ đồ thị

1. **Ứng dụng trong game Sodoku**

Sudoku là một loại trò chơi logic, cách chơi là điền số từ 1 đến 9 vào những ô trống sao cho mỗi cột dọc, mỗi hàng ngang, mỗi phân vùng nhỏ (ô 3x3) có đủ các số từ 1 đến 9 mà không được lặp lại. Trò chơi có bảng câu đố hình vuông, mỗi chiều có 9 ô nhỏ, hợp thành 9 cột, 9 hàng và được chia thành 9 ô lớn 3x3. Một vài ô nhỏ được đánh số, đó là những manh mối duy nhất để bạn tìm lời giải.



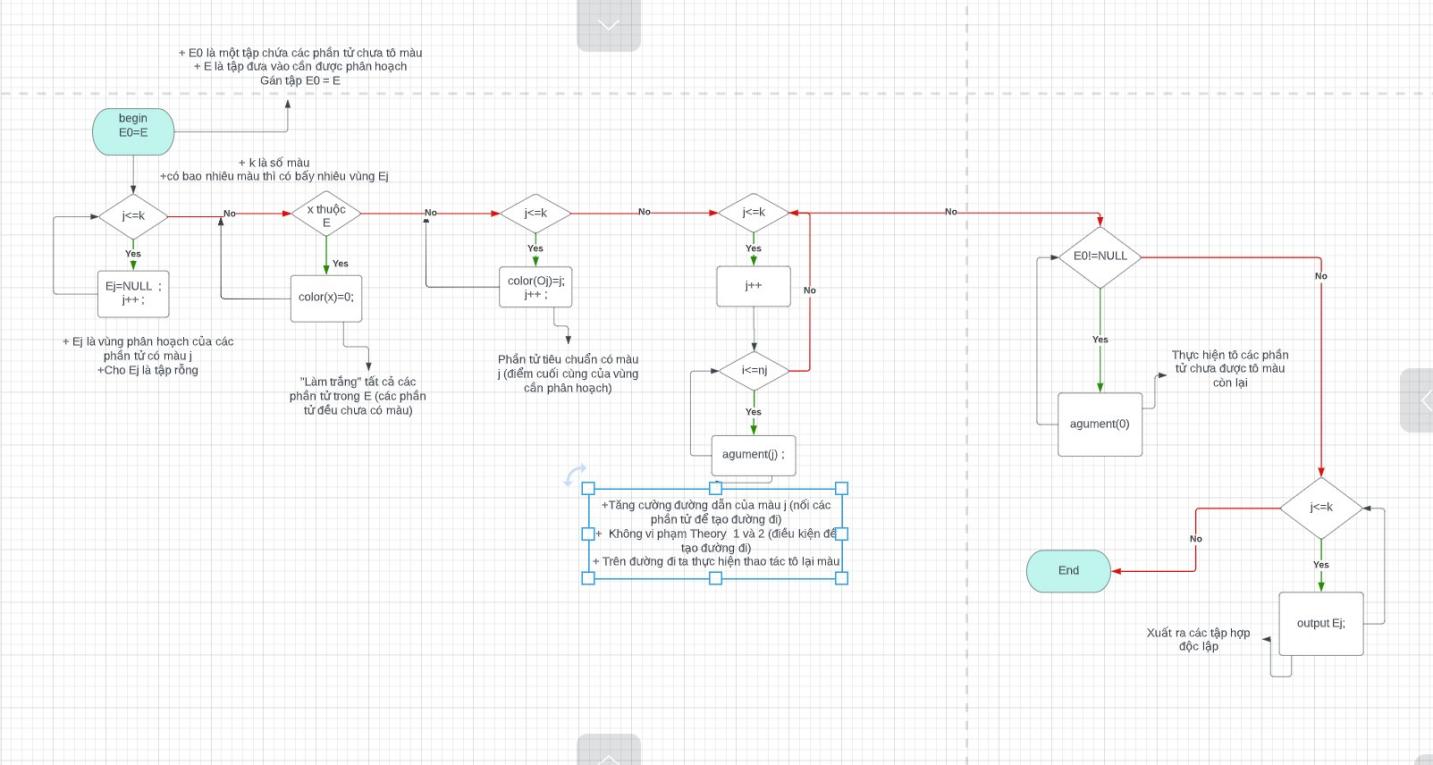
Trên quan điểm bảng câu đố này là một đồ thị, và mỗi ô vuông nhỏ (cell) là một đỉnh của đồ thị. Như vậy, đồ thị sẽ có 81 đỉnh. Những cặp đỉnh nào có ràng buộc sẽ có cạnh giữa chúng. Để giải quyết bài toán thì ta áp dụng thuật toán quay lui và thực hiện tô màu các đỉnh, tương ứng với các ô.

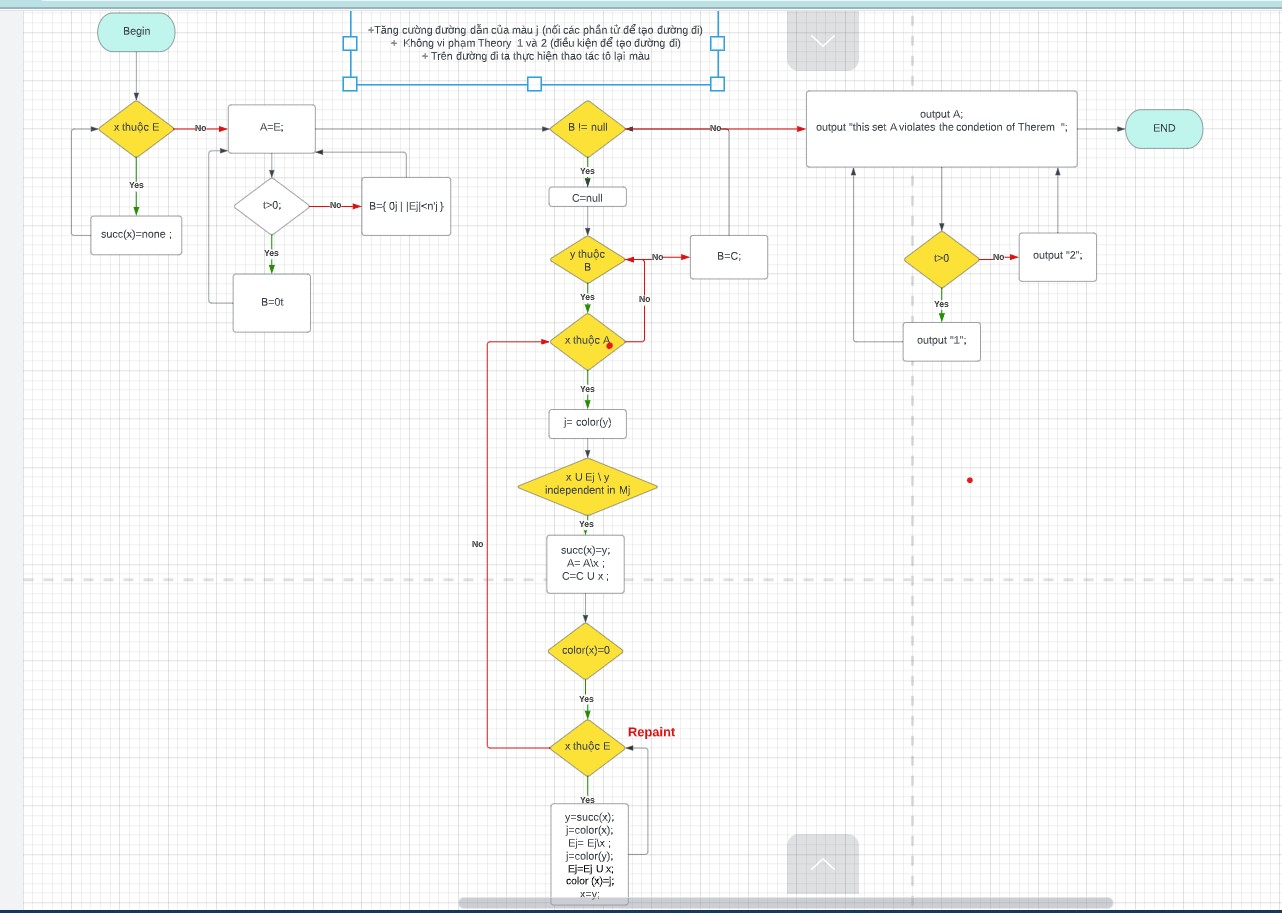
1. **Thu hoạch kết quả**

Sau khi nghiên cứu bài báo về thuật toán Phân hoạch Matroid, nhóm em đã thu hoạch được một số nội dung như sau:

* Một là, ôn tập lại kiến thức về tiếng Anh, tăng cường khả năng dịch thuật và khả năng phân tích bài báo.
* Hai là, vận dụng kiến thức đã học từ môn Cấu trúc dữ liệu và Giải thuật để viết thuật toán, giải thích công thức trong bài báo. Vận dụng kiến thức về backtracking, ứng dụng graph để viết code mô phỏng chương trình.
* Ba là, áp dụng được những kiến thức đã học vào thực tế, giải quyết các bài toán cơ bản.

1. **Sơ đồ khối**





Link:

https://lucid.app/lucidchart/ac6e749b-4fe7-42c8-972f- 6547637b11d0/edit?viewport\_loc=179%2C28%2C2613%2C1182%2C0\_0&invitationId=inv\_00446a71- fc65-4f82-b46c-48a6429e2ba1

1. **Code**

**Thuật toán**

#include <stdio.h>

#include <stdlib.h>

#define SIZE 50

#define input "dothi.txt"

void docDoThi(int A[][SIZE], int &n)

{

FILE \*fi = fopen(input,"rt");

if (fi == NULL){

printf("Khong mo duoc file");

exit(0);

}

fscanf(fi,"%d",&n);

for (int i=0; i<n; i++){

for (int j=0; j<n; j++){

fscanf(fi,"%d",&A[i][j]);

}

}

fclose(fi);

}

void toMau(int A[][SIZE], int n, int Mau[])

{

int BangMau[SIZE][SIZE] = {0};

int mauvuato;

for(int i=0; i<n; i++){

for(int k=0; k<n; k++){

if(BangMau[i][k] == 0){

Mau[i] = k;

mauvuato = k;

break;

}

}

for(int j=i+1; j<n; j++){

if(A[i][j] > 0){

BangMau[j][mauvuato] = 1;

}

}

}

}

void xuat(int A[], int n, int mau)

{

for(int i=0; i<n; i++){

if(A[i] == mau){

printf("%d ", i+1);

}

}

printf("\n");

}

int main()

{

int A[SIZE][SIZE], Mau[SIZE], n;

docDoThi(A,n);

toMau(A, n, Mau);

int maugiongnhau=0;

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

{

xuat(Mau,n,maugiongnhau);

maugiongnhau++;

}

return 0;

}

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

**Sodoku**

#include <iostream>

using namespace std;

bool isValidPlacement(char board[][9], int row, int col, char c) {

//kiem tra hang co trung phan tu khong

for(int i = 0; i < 9; i++)

if(board[row][i] == c) return false;

//kiem tra cot co trung phan tu khong

for(int i = 0; i < 9; i++)

if(board[i][col] == c) return false;

//kiem tra trong khoi 3x3 co phan tu trung khong qua 9 toa do goc

for(int i = 0; i < 3; i++)

for(int j = 0; j < 3; j++)

if(board[row/3\*3 + i][col/3\*3 + j] == c)

return false;

return true;

}

bool solve(char board[][9]) {

//Duyet het bang, tim cac phan tu trong ('.')

for(int i = 0; i < 9; i++){

for(int j = 0; j < 9; j++){

//Neu tim duoc phan tu '.', ta tim cach dien chu so vao o do

if (board[i][j] == '.'){

// Lặp lại tất cả các số nguyên từ 1 đến 9 để điền vào

// '1'+ k chuyen ve char

for(int k = 0; k < 9; k++){

//Kiem tra dieu kien chen

if (isValidPlacement(board, i, j, k +'1')){

// Dien gia tri vao va tiep tuc backtracking

board[i][j] = '1'+ k;

if(solve(board))

//Tiep tuc tim kiem cau hinh cua phan tu tiep theo cho den khi dien

//tat ca cac o

return true;

else

// Neu chua duoc thi ta tim kiem mot gia tri khac

board[i][j] = '.';

}

}

return false;

}

}

}

return true;

}

void solveSudoku(char board[][9]) {

solve(board);

}

void printArray(char A[][9]) {

for(int i = 0; i < 9; i++){

for (int j = 0; j < 9; j++){

cout << A[i][j] << ' ';

}

cout << endl;

}

}

int main()

{

char A[9][9] = {

{'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'},

};

solveSudoku(A);

printArray(A);

system("pause");

return 0;

}