**TRƯỜNG ĐẠI HỌC FPT**

****

**ASSIGNMENT 2**

**DATA STRUCTURE AND ALGORITHM**

*Người thực hiện*: **Huỳnh Lê Phúc Vinh – SE170460**

**Phan Hồng Trinh – SE170414**

Lớp **: SE1714**

Khoá  **: 17**

# LỜI CẢM ƠN

Chúng tôi xin chân thành cảm ơn đến quý Thầy TS. Bùi Thanh Hùng– Giảng viên

hướng dẫn đã truyền đạt những kiến thức nền tảng, những bài học lập trình, qua đó đã

giúp đỡ chúng tôi rất nhiều trong quá trình thực hiện bài báo cáo này.

# PHẦN ĐÁNH GIÁ CỦA GIẢNG VIÊN

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

TP. Hồ Chí Minh, ngày tháng năm (kí và ghi họ tên)

MỤC LỤC

[LỜI CẢM ƠN 2](#_Toc129819901)

[PHẦN ĐÁNH GIÁ CỦA GIẢNG VIÊN 3](#_Toc129819902)

[1. Giới thiệu về bài toán 5](#_Toc129819903)

[2. Giải quyết bài toán 6](#_Toc129819904)

[2.1 Chi tiết cấu trúc dữ liệu trong bài toán 6](#_Toc129819905)

[2.2 Sơ đồ giải thuật 8](#_Toc129819906)

[2.3 Hiện thực 10](#_Toc129819907)

[2.4 Kết Qủa Và Thảo Luận 17](#_Toc129819908)

[3. Kết Luận 18](#_Toc129819909)

[3.1 Một số kinh nghiệm chúng tôi rút ra được từ bài báo cáo này: 18](#_Toc129819910)

[3.2 Hướng phát triển tương lai 18](#_Toc129819911)

**BÀI TOÁN TÔ MÀU – COLORING PROBLEM**

## 1. Giới thiệu về bài toán

* + Bài toán tô màu là bài toán rất quan trọng trong lý thuyết đồ thị, trong đó yêu cầu tô màu các đỉnh của đồ thị sao cho không có hai đỉnh kề nhau cùng màu.
  + Bài toán có nhiều ứng dụng trong thực tế, ví dụ như tô màu các vùng đất trong bản đồ, tô màu các bức tranh số, tô màu các trang web, trong lập lịch làm việc, lập kế hoạch sản xuất, thiết kế mạch điện tử, thiết kế mạng máy tính và thiết kế đồ họa. Các phương pháp giải quyết bài toán Tô màu được sử dụng để tối ưu hóa các quá trình trong các lĩnh vực này, giúp tăng năng suất và hiệu quả của các hoạt động.

## 2. Giải quyết bài toán

### 2.1 Chi tiết cấu trúc dữ liệu trong bài toán

Bài toán tô màu là một bài toán tìm kiếm, nó liên quan đến việc gán màu cho các đỉnh trong đồ thị sao cho không có hai đỉnh nào kề nhau được gán cùng một màu.

Một cấu trúc dữ liệu thường được sử dụng trong bài toán này là danh sách kề (adjacency list). Đây là một cấu trúc dữ liệu để lưu trữ thông tin về các đỉnh của đồ thị và các cạnh nối giữa chúng. Các đỉnh được lưu trữ dưới dạng các nút (node) trong danh sách kề, mỗi nút chứa một danh sách các đỉnh kề với nó.

Một danh sách kề có thể được biểu diễn dưới dạng một mảng hai chiều, trong đó hàng thứ i chứa danh sách các đỉnh kề với đỉnh i. Ví dụ, nếu đỉnh 1 kề với đỉnh 2 và đỉnh 3, đỉnh 2 kề với đỉnh 1 và đỉnh 3, và đỉnh 3 kề với đỉnh 1 và đỉnh 2, thì danh sách kề có thể được biểu diễn như sau:

[ [2, 3], // danh sách các đỉnh kề với đỉnh 1

[1, 3], // danh sách các đỉnh kề với đỉnh 2

[1, 2] // danh sách các đỉnh kề với đỉnh 3 ]

Khi tô màu các đỉnh trong đồ thị, một mảng được sử dụng để lưu trữ màu được gán cho mỗi đỉnh. Mỗi phần tử trong mảng tương ứng với một đỉnh trong đồ thị. Ví dụ, nếu mảng màu ban đầu được khởi tạo với giá trị 0, tức là mỗi đỉnh chưa được gán màu, và khi gán màu cho đỉnh i, màu được lưu trữ tại vị trí i trong mảng. Ví dụ, nếu đỉnh 1 được gán màu 2, mảng màu có thể được cập nhật như sau:

[2, 0, 0] // đỉnh 1 được gán màu 2, các đỉnh còn lại chưa được gán màu

Ngoài ra, một số cấu trúc dữ liệu khác cũng có thể được sử dụng trong bài toán tô màu, chẳng hạn như ma trận kề (adjacency matrix) hoặc danh sách cạnh (edge list). Tuy ngoài danh sách kề, bài toán tô màu cũng có thể sử dụng ma trận kề để lưu trữ thông tin về đồ thị. Ma trận kề là một ma trận vuông, trong đó hàng và cột tương ứng với các đỉnh của đồ thị. Giá trị tại hàng i, cột j của ma trận kề bằng 1 nếu có cạnh nối giữa đỉnh i và j, ngược lại bằng 0.

Ví dụ, nếu đồ thị có 4 đỉnh và 3 cạnh nối giữa chúng. Thì ma trận kề của đồ thị có thể được biểu diễn như sau:

[ [0, 1, 1, 0], // hàng thứ 1 tương ứng với đỉnh 1

[1, 0, 1, 0], // hàng thứ 2 tương ứng với đỉnh 2

[1, 1, 0, 1], // hàng thứ 3 tương ứng với đỉnh 3

[0, 0, 1, 0] // hàng thứ 4 tương ứng với đỉnh 4 ]

Ma trận kề cung cấp thông tin về đồ thị dưới dạng một bảng dữ liệu có cấu trúc, giúp dễ dàng truy xuất các đỉnh và cạnh nối giữa chúng.

Danh sách cạnh (edge list) là một cấu trúc dữ liệu khác cũng có thể được sử dụng trong bài toán tô màu. Danh sách cạnh là một danh sách các cạnh trong đồ thị, trong đó mỗi cạnh được biểu diễn dưới dạng một cặp đỉnh. Ví dụ, danh sách cạnh của đồ thị ở trên có thể được biểu diễn như sau:

[ [1, 2], // cạnh nối giữa đỉnh 1 và đỉnh 2

[1, 3], // cạnh nối giữa đỉnh 1 và đỉnh 3

[2, 3] // cạnh nối giữa đỉnh 2 và đỉnh 3 ]

Danh sách cạnh là một cấu trúc dữ liệu đơn giản, giúp dễ dàng truy xuất thông tin về các cạnh trong đồ thị. Tuy nhiên, nó không cung cấp thông tin về đỉnh kề với mỗi đỉnh, do đó không phù hợp để sử dụng trong việc giải quyết bài toán tô màu. Tuy nhiên, danh sách cạnh vẫn có thể được sử dụng để xác định các đỉnh liên thông của đồ thị và phát hiện các thành phần liên thông của đồ thị.

Để phát hiện các thành phần liên thông của đồ thị, ta có thể sử dụng thuật toán duyệt đồ thị theo chiều rộng (breadth-first search - BFS) hoặc thuật toán duyệt đồ thị theo chiều sâu (depth-first search - DFS) trên danh sách cạnh. Trong quá trình duyệt, các đỉnh liên thông của đồ thị sẽ được đánh dấu và lưu trữ vào một danh sách riêng.

Tổng quát, mỗi cấu trúc dữ liệu có điểm mạnh và điểm yếu riêng, và sự lựa chọn cấu trúc dữ liệu phù hợp phụ thuộc vào các yêu cầu cụ thể của bài toán. Trong bài toán tô màu, danh sách kề là một cấu trúc dữ liệu phổ biến và hữu ích, trong khi ma trận kề và danh sách cạnh có thể được sử dụng để xác định các thành phần liên thông của đồ thị.

### 2.2 Sơ đồ giải thuật

**- Thuật giải tô màu tham lam(Greedy)**

Dùng màu thứ nhất tô cho tất cả các đỉnh của đồ thị mà có thể tô được, sau đó dùng

màu thứ hai tô tất cả các đỉnh của đồ thị còn lại có thể tô được và cứ như thế cho đến khi tô

hết tất cả các đỉnh của đồ thị.

* **Lược đồ của thuật giải này như sau:**

m=1;

số đỉnh đã được tô = 0;

mọi đỉnh đều chưa được tô

**do**

**{**

for i=1 to n

if (đỉnh i là chưa xét và có thể tô được bằng màu m)

{ tô đỉnh i bằng màu m, đỉnh i trở thành đỉnh đã xét.

tăng số đỉnh đã được tô lên 1 đơn vị

}

m++

**}**

while (số đỉnh đã được tô<n)

**Ví dụ: Phương án đặt sách lên kệ sách**

Tại một cửa hàng sách, mới nhập về 12 quyển sách thuộc các loại sau:

Truyện cười: A, C, D, G.

Âm nhạc: B, H, K.

Lịch sử: E, J, L.

Khoa học: F, I.

Hãy sắp xếp những quyển sách này vào kệ sao cho số kệ sử dụng là ít nhất mà tuân theo

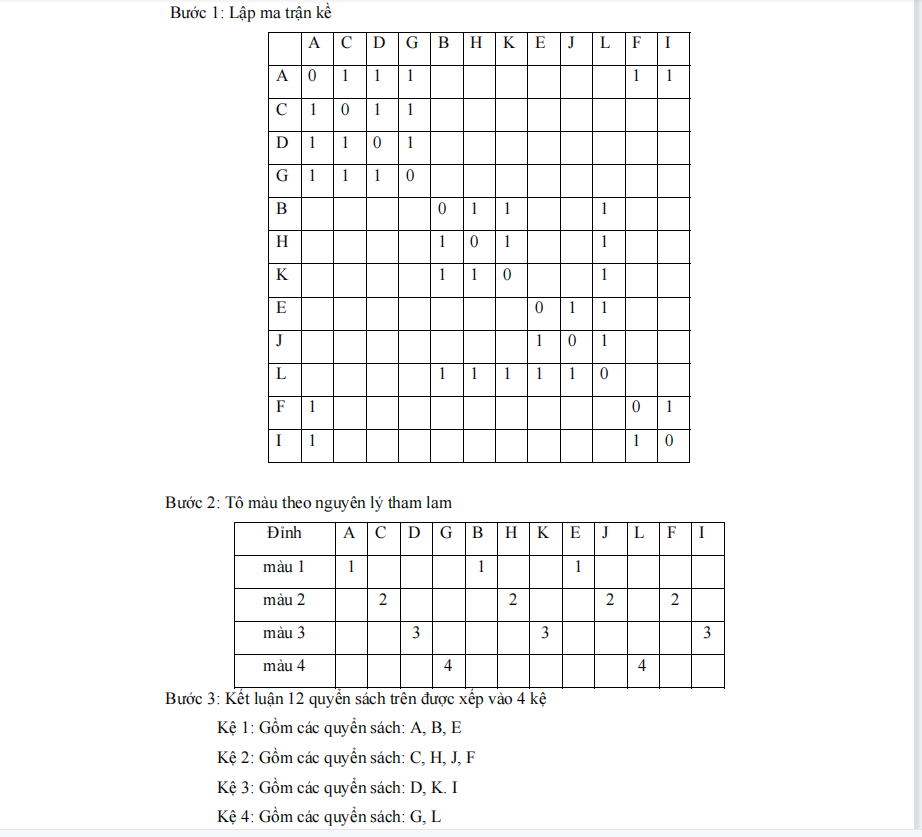
các yêu cầu sau:

-Các quyển sách cùng loại không được để chung một kệ.

-Quyển A không được để chung với sách khoa học.

-Quyển L không được để chung với sách âm nhạc.

Giải:



### 2.3 Hiện thực

**Code:**

import java.io.RandomAccessFile;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.HashMap;

import java.util.HashSet;

import java.util.List;

import java.util.Map;

import java.util.Scanner;

import java.util.Set;

import java.util.StringTokenizer;

/\*\*

\* Class GraphColoring \*

\*/

public class GraphColoring {

private int V, numOfColors;

private int[] color;

private int[][] graph;

private char ver[];

/\*\*

\* Function to assign color \*

\*/

public void graphColor(String s) {

//ver = "ACDGBHKEJLFI".toCharArray();

//this.V = v;

loadData(s);

//numOfColors = noc;

color = new int[V];

//graph = g;

try {

solve(0);

System.out.println("No solution");

} catch (Exception e) {

System.out.println("\nSolution exists ");

display();

}

}

/\*\*

\* function to assign colors recursively \*

\*/

public void solve(int v) throws Exception {

/\*\*

\* base case - solution found \*

\*/

if (v == V) {

throw new Exception("Solution found");

}

/\*\*

\* try all colours \*

\*/

for (int c = 1; c <= numOfColors; c++) {

if (isPossible(v, c)) {

/\*\*

\* assign and proceed with next vertex \*\

\*/

color[v] = c;

solve(v + 1);

/\*\*

\* wrong assignement \*

\*/

color[v] = 0;

}

}

}

/\*\*

\* function to check if it is valid to allot that color to vertex \*

\*/

public boolean isPossible(int v, int c) {

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

if (graph[v][i] == 1 && c == color[i]) {

return false;

}

}

return true;

}

/\*\*

\* display solution \*

\*/

public void display() {

//System.out.print("\nColors : \n");

for (int j = 1; j <= numOfColors; j++) {

System.out.print("Ke sach " + j + ": Gom cac quyen sach: ");

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

if (j == color[i]) {

System.out.print(ver[i] + " ");

}

}

System.out.println("");

}

}

//load file

void loadData(String fname) {

RandomAccessFile f;

// int i, j, x;

String s;

StringTokenizer t;

String s3 = "";

Map<String, Integer> bookIndices = new HashMap<>();

ArrayList<String> list = new ArrayList();

try {

f = new RandomAccessFile(fname, "r");

s = f.readLine();//just use the data in the k-th line

s = s.trim();

int c = Integer.parseInt(s);

numOfColors = c;

s = f.readLine();//just use the data in the k-th line

s = s.trim();

int n = Integer.parseInt(s);

this.V = n;

graph = new int[n][n];

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

for(int r2=0;r2<n;r2++) {

graph[r][r2] = 0;

}

}

for (int y = 1; y <= c; y++) {

s = f.readLine();

s = s.trim();

String s2 ="";

String txt[] = s.split(",");

for (int tt = 0; tt < txt.length; tt++) {

String s1 = txt[tt].trim();

s2 = s2.concat(s1);

s3 = s3.concat(s1);

}

list.add(s2);

for (String book : txt) {

bookIndices.put(book, bookIndices.size());

}

for ( int j = 0; j < txt.length; j++) {

for (int k = j + 1; k < txt.length; k++) {

int index1 = bookIndices.get(txt[j]);

int index2 = bookIndices.get(txt[k]);

graph[index1][index2] = 1;

graph[index2][index1] = 1;

}

}

}

this.ver = s3.toCharArray();

s=f.readLine();

s=s.trim();

int nRules =Integer.parseInt(s);

for(int i =1;i<=nRules;i++) {

s = f.readLine();

s = s.trim();

String[] line = s.split(" not in ");

String sach = line[0];

int ke = Integer.parseInt(line[1]);

int sachIndex =bookIndices.get(sach);

int keIndex = ke -1;

String sachkeIndex[] = list.get(keIndex).split("");

for(int y=0;y<sachkeIndex.length;y++){

int entry = bookIndices.get(sachkeIndex[y]);

graph[sachIndex][entry] = 1;

graph[entry][sachIndex] = 1;

}

}

/\* for (i = 0; i < n; i++) {

s = f.readLine();

s = s.trim();

t = new StringTokenizer(s);

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

x = Integer.parseInt(t.nextToken().trim());

graph[i][j] = x;

}

}\*/

f.close();

} catch (Exception e) {

}

}

/\*\*

\* Main function \*

\*/

public static void main(String[] args) {

Scanner scan = new Scanner(System.in);

System.out.println("Graph Coloring Algorithm Test\n");

/\*\*

\* Make an object of GraphColoring class \*

\*/

GraphColoring gc = new GraphColoring();

/\*\*

\* Accept number of vertices \*

\*/

/\* System.out.println("Enter number of verticesz\n");

int V = scan.nextInt();\*/

/\*\*

\* get graph \*

\*/

/\*System.out.println("\nEnter matrix\n");

int[][] graph = new int[V][V];

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

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

graph[i][j] = scan.nextInt();\*/

/\* System.out.println("\nNhap so loai sach(Enter number of color): "); int c = scan.nextInt();\*/

gc.graphColor("D:\\FPT\\PRF\\project\_CSD201\\Matrix.txt");}

### 2.4 Kết Qủa Và Thảo Luận

Việc đánh giá kết quả đạt được trong bài toán tô màu phụ thuộc vào các tiêu chí và yêu cầu của bài toán cụ thể. Tuy nhiên, một số tiêu chí chung để đánh giá kết quả có thể bao gồm:

Số lượng màu sử dụng: Một trong những tiêu chí quan trọng trong bài toán tô màu là sử dụng ít màu nhất có thể để tô đồ thị. Vì vậy, một kết quả tốt sẽ là kết quả tô được đồ thị bằng số lượng màu ít nhất có thể.

Thời gian chạy: Khi đối mặt với đồ thị lớn hoặc phức tạp, thời gian để tìm ra một giải pháp tốt có thể là yếu tố quan trọng. Vì vậy, một kết quả tốt sẽ là kết quả tô được đồ thị trong thời gian ngắn nhất có thể.

Sự chính xác: Một kết quả tốt phải đảm bảo đúng số lượng màu được sử dụng và mỗi đỉnh trong đồ thị đều được tô màu sao cho không có hai đỉnh kề nhau cùng màu.

Kết quả đạt được phụ thuộc vào thuật toán tô màu được sử dụng.

Ở project này ta dùng : Thuật giải tô màu tham lam (Greedy) là một phương pháp đơn giản và hiệu quả để giải quyết bài toán tô màu đồ thị. Tuy nhiên, độ chính xác của thuật toán này không đảm bảo đạt được kết quả tối ưu cho tất cả các đồ thị.

Trong trường hợp các đồ thị đơn giản, không quá lớn và có tính chất đặc biệt (như cây, đồ thị hai phía, đồ thị màu sắc), thuật giải tô màu tham lam có thể đạt được kết quả tối ưu trong thời gian ngắn.

Tuy nhiên, đối với các đồ thị phức tạp hơn, thuật giải tham lam có thể không đạt được kết quả tối ưu và cần phải sử dụng các thuật toán khác để đạt được kết quả tối ưu. Vì vậy, trong các bài toán thực tế, ta cần phải kết hợp nhiều thuật toán khác nhau và tối ưu hóa các tham số của thuật toán để đạt được kết quả tốt nhất trong thời gian ngắn nhất có thể.

Tóm lại, thuật giải tô màu tham lam là một phương pháp đơn giản và hiệu quả để giải quyết bài toán tô màu đồ thị. Tuy nhiên, độ chính xác của thuật toán này không đảm bảo đạt được kết quả tối ưu cho tất cả các đồ thị và cần phải được kết hợp với các thuật toán khác để đạt được kết quả tối ưu trong thời gian ngắn nhất có thể.

## 3. Kết Luận

### 3.1 Một số kinh nghiệm chúng tôi rút ra được từ bài báo cáo này:

1. Hiểu rõ bài toán trước khi giải quyết: Trước khi bắt đầu giải quyết bất kỳ bài toán nào, chúng ta nên hiểu rõ bài toán và ý nghĩa của nó. Điều này sẽ giúp cho việc giải quyết bài toán được hiệu quả hơn và giải pháp đưa ra sẽ phù hợp hơn với mục đích cần đạt được.
2. Lựa chọn cấu trúc dữ liệu phù hợp: Cấu trúc dữ liệu là một yếu tố quan trọng trong giải quyết bất kỳ bài toán nào. Việc lựa chọn cấu trúc dữ liệu phù hợp giúp cho việc giải quyết bài toán trở nên dễ dàng và nhanh chóng hơn.
3. Cần phải tìm hiểu kỹ các giải thuật: Các giải thuật là những công cụ quan trọng giúp cho việc giải quyết bài toán được hiệu quả hơn. Việc tìm hiểu kỹ các giải thuật giúp cho chúng ta lựa chọn giải thuật phù hợp nhất với bài toán cần giải quyết.

### 3.2 Hướng phát triển tương lai

Bài toán tô màu của chúng tôi chỉ áp dụng những kiến thức cơ bản của giải thuật Greedy để thực hiện những lệnh đơn giản như tô màu hai đỉnh không gần kề của đồ thị. Vì vậy trong tương lai, chúng tôi dự định sẽ nghiên cứu thêm về những vấn đề sau:

* Nâng cao hiệu suất của giải thuật bằng cách áp dụng các giải thuật khác như DFS ( Depth-First Search ), BFS ( Breadth-First Search ).
* Mở rộng bái toán tô màu : Bài toán tô màu có thể được mở rộng để giải quyết các bài toán liên quan như tô màu đồ thị có trọng số, tô màu trên mô hình 3D và hình dạng đa giác...

**Tài liệu tham khảo :**

*[1]Artificial Intelligence*

*George F.Luger & William A.Stubblefield*

*[2]Artificial Intelligence*

*Patrick Henry Winston – Addion \_ Wesley 1995*

*[4]Trí tuệ nhân tạo- Đại học Sài Gòn*