Rapport Algorithmique des graphes

Perrot Killiann & Espada Mora Matthieu & Nalet Antoine

Contents

Chapter 1

Page 1

Chapter 1

L'Étude des Coloriages de Graphes : Entre Culture et Utilité

Introduction

Les coloriages de graphes constituent un domaine d'étude fascinant, alliant la richesse culturelle des mathématiques à leur utilité pratique dans divers domaines. Dans cette exploration, nous plongerons dans la partie culturelle en examinant pourquoi les coloriages de graphes ont captivé l'attention des mathématiciens et quels défis stimulants ils soulèvent.

Partie culturelle

Les coloriages de graphes trouvent leurs racines dans la théorie des graphes, une branche des mathématiques. Leur attrait réside dans la manière dont ils permettent de visualiser et de comprendre les relations entre les entités représentées par les nœuds d'un graphe. Depuis les premières explorations de ce concept par Francis Guthrie au XIXe siècle avec le problème des quatre couleurs, les mathématiciens ont été intrigués par la question fondamentale : "Quel est le nombre minimum de couleurs nécessaires pour colorier un graphe de telle sorte que deux nœuds voisins aient des couleurs différentes ?" Cette question en apparence simple a donné naissance à une pléthore de résultats profonds, de conjectures célèbres, et d'applications dans divers domaines.

Partie utilitaire

Au-delà de leur intérêt académique, les coloriages de graphes trouvent des applications pratiques dans plusieurs domaines. En informatique, par exemple, la coloration de graphes est cruciale pour l'ordonnancement des tâches et la gestion des ressources. Dans les réseaux de télécommunications, les fréquences assignées aux canaux peuvent être modélisées par des coloriages de graphes pour éviter les interférences. Dans le domaine biologique, les coloriages de graphes sont utilisés pour représenter des interactions moléculaires complexes.

Conclusion

En conclusion, l'étude des coloriages de graphes offre une perspective fascinante qui transcende les frontières entre la culture mathématique pure et les applications concrètes. De la résolution du problème des quatre couleurs à l'application dans des domaines aussi divers que l'informatique et la biologie, les coloriages de graphes continuent d'enrichir notre compréhension des relations entre les entités et de nous offrir des outils précieux pour résoudre des problèmes du monde réel.

Les algorithmes utilisés

Algorithme Glouton

Structures de données utilisées:

- `adj` : Liste d'adjacence
- `result` : Liste des couleurs des sommets
- `available` : Liste des couleurs disponibles

Primitives utilisées :

- `addEdge` : Utilisée pour ajouter une arête entre deux sommets si l'utilisateur veut créer son graphe à la main

Algorithm 1: Greedy Coloring

Input:

- `adj`: Adjacency list representing the graph
- `V`: Number of vertices

Output:

- `result`: List of colors of the vertices

```
1  Function addEdge(adj, V, W):
2        adj[V].append(W);
3        adj[W].append(V);
4        return ajd;
5  Function greedyColoring(adj, V):
6         result[V] ← -1, -1, ..., -1 ;                    // Initialize the list of colors to -1
7         result[0] ← 0 ;                                  // Color the first vertex with the first color
8         available[V] ← False, False, ..., False ;        // Initialize the list of available colors to False
9         for u from 1 to V − 1 do
10  |              for i in ajd[u] do
11  |  |               if result[i] ≠-1 then
12  |  |  |                    available[result[i]] ← True;
13  |   end
14  |               cr ← 0;
15  |               while cr < v do
16  |  |               if available[cr] == false then
17  |  |  |                    break;
18  |   end
19  |               result[u] ← cr;
20 end
21                  for i in ajd[u] do
22  |                   if result[i] ≠ -1 then
23  |  |                      available[result[i]] ← False;
24 end
25       return result;
```

Note:-

La complexité dépend du nombre de sommets et des arêtes dans le graphe. Si V est le nombre de sommets et E le nombre d'arêtes dans le graphe , la complexité serait de l'ordre de $O(V \times E)$.

Algorithme Welsh-Powell

Structures de données utilisées:

- `G` : Dictionnaire représentant le graphe
- `Ma` : Matrice d'adjacence
- `sommets` : Liste des sommets du graphe
- `degres` : Liste des degrés des sommets du graphe
- `result` : Liste représentant les couleurs attribuées aux sommets
- `couleurs` : Liste des couleurs disponibles

Primitives utilisées :

- `MAdjacence` : Retourne une matrice d'adjacence Ma du graphe G ou Ma[i][j] = 1 si il existe une arête entre les sommets i et j, 0 sinon
- `Sommets` : Retourne une liste des sommets du graphe G
- `Voisinage` : Retourne un ensemble des voisins du sommet donné en paramètre

Algorithm 2: Welsh-Powell Coloring

Input:

- `G`: Adjacency list representing the graph

Output:

- `result`: List of colors of the vertices

```
1  Function MAdjacence(G):
2      n ← length(G);
3      Ma ← matrix of size n × n initialize with 0, 0, ..., 0 ;
4      for i from 0 to n − 1 do
5          for j from 0 to n − 1 do
6              if i == j then
7                  continue;
8              end
9              if j in G[i] then
10                 Ma[i][j] ← 1; // If there is an edge between vertex i and j, set the
                       corresponding entry to 1
11             end
12         end
13 end
14     return Ma;
15 Function Sommets(G):
16     return list of vertices in G;
17 Function Voisinage(G, sommet):
18     return neighbors of sommet in G;
```

Note:-

La complexité dépend du nombre de sommets dans le graphe. Si V est le nombre de sommets, la complexité serait de l'ordre de $O(V^2)$.