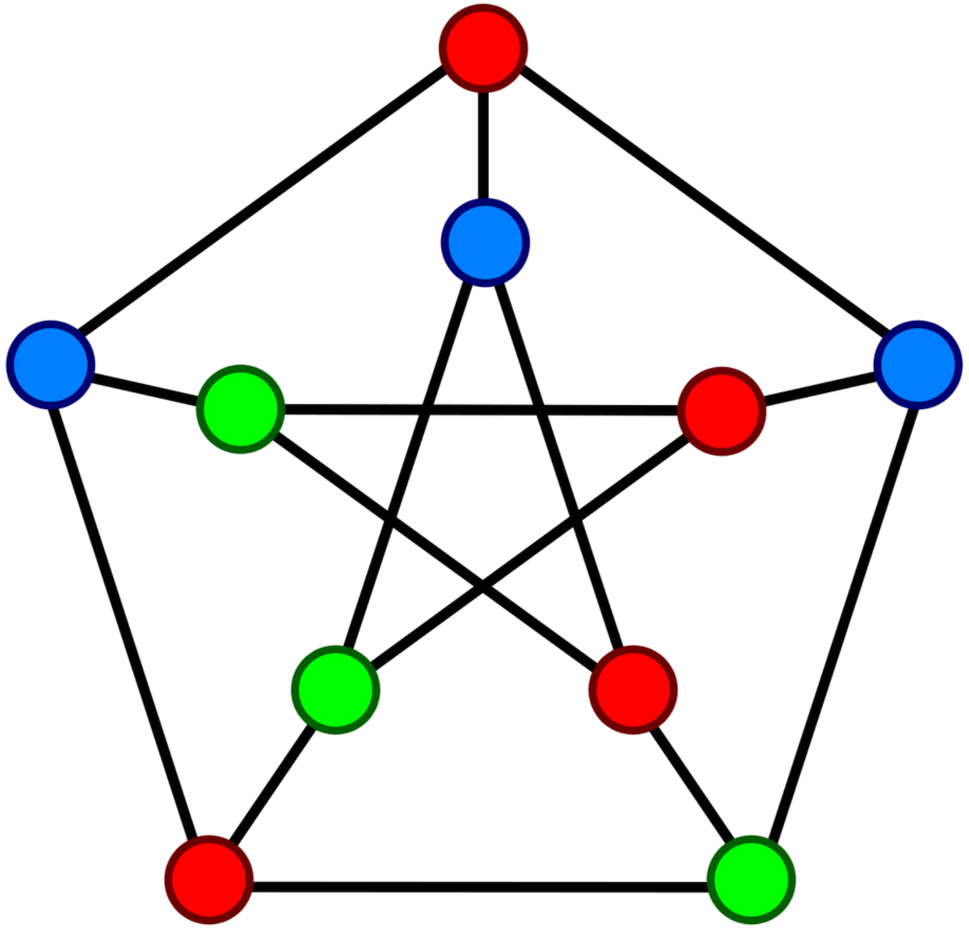
Issam Dziri

Application d'un algorithme approché pour résoudre le problème de coloration de graphe

Séance 2



00

L'algorithme choisi

Pour résoudre notre problème de coloration de graphe, j'ai choisi l'algorithme de BackTracking.
Cet algorithme permet de tester systématiquement l'ensemble des affectations potentielles à un problème donné de manière beaucoup plus optimisé qu'un simple test itératif.

01

Le Pseudo-code de l'algorithme

Pseudo-code

```
bool graphColoring(bool graph[V][V], int m, int i, int
color[V])
{
if (i == V) {
if (isSafe(graph, color)) {
printSolution(color);
return true;
}
return false;
}
for (int j = 1; j <= m; j++)
{
color[i] = j;
if (graphColoring(graph, m, i + 1, color))
return true;
color[i] = 0;
}
return false;
}
```

02

L'utilité des différents paramètres

Les paramètres

Input :
Un graphe matriciel 2D[V][V] où V est le nombre de sommets dans le graphe et graph[V][V] est une représentation matricielle d'adjacence du graphe. Une valeur graph[i][j] est 1 s'il y a une arête directe de i à j, sinon graph[i][j] est 0. Un entier m est le nombre maximum de couleurs pouvant être utilisées

Output:
Un tableau color[V] qui devrait avoir des nombres de 1 à m. color[i] doit représenter la couleur attribuée au ième sommet. Le code doit également retourner false si le graphique ne peut pas être coloré avec m couleurs

03

L'application de l'algorithme pour résoudre notre problème

Exemple :

```
Input: graph = {0, 1, 1, 1},
               {1, 0, 1, 0},
               {1, 1, 0, 1},
               {1, 0, 1, 0}
Output:
Solution Exists:
Following are the assigned colors
1 2 3 2
Explanation:
By coloring the vertices with following colors,
adjacent vertices does not have same colors
Input: graph = {1, 1, 1, 1},
               {1, 1, 1, 1},
               {1, 1, 1, 1},
               {1, 1, 1, 1}
Output: Solution does not exist.
Explanation: No solution exits.
```