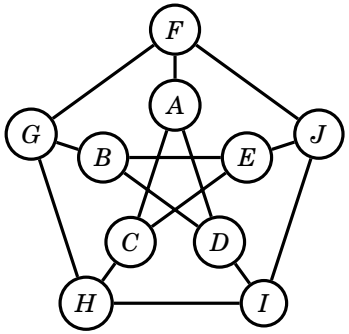
ITC – TD n°12-6
Coloration de graphe

I. Principe

On parle de N-colorier un graphe lorsqu'il s'agit d'attribuer à chaque sommet une couleur différente (ci-après désignée par un entier compris entre 1 à N, 0 représentant un sommet non colorié) de façon à ce que deux sommets adjacents aient une couleur différente. Si un tel coloriage existe, on dit que le graphe est N-coloriable.



1 – Proposer un 3-coloriage du graphe ci-contre.

2 – Si un graphe a pour degré maximal Δ, pouvez-vous intuiter une borne supérieure du nombre de couleurs nécessaires pour le colorier ?

II. Algorithme pour le coloriage

On propose de réaliser l'algorithme de backtracking de coloriage suivant, dans lequel le nombre de couleurs N est imposé :

- on choisit le sommet u non colorié ayant le moins de couleurs restantes possibles étant donné ses voisins ; si aucun tel sommet n'est trouvé, le graphe est colorié (cas de base) ;
- on essaie une des couleurs acceptables pour ce sommet u ;
- on appelle récursivement l'algorithme ;
  ▷ si le retour de l'appel récursif indique que le graphe est colorié, on termine en renvoyant l'information de graphe colorié ;
  ▷ si le retour de l'appel récursif indique que le graphe n'est pas coloriable avec cette couleur pour u, on essaie la couleur acceptable suivante pour u et on recommence l'appel récursif ;
- si toutes les couleurs acceptables pour u ont été essayées et qu'aucune n'a permis de colorier le graphe, on renvoie l'information de graphe non coloriable en l'état.

On reprend la représentation des graphes habituelle (dictionnaires de dictionnaires d'adjacence), et on ajoute un dictionnaire `couleurs` dont les clés sont les sommets u du graphe et les valeurs le nombre compris entre 1 et N correspondant à la couleur de u, et 0 si u n'est pas colorié.

3 – Écrire une fonction `liste_possibles(g, couleurs, u, Nmax)` qui prend en paramètres le graphe, les couleurs des sommets, un sommet u et le nombre de couleurs autorisées, et renvoie la liste des couleurs possibles pour u étant donné ses voisins.

4 –Écrire une fonction `sommet_contraint_max(g, couleurs, Nmax) -> str, list` qui détermine le sommet non colorié le plus contraint, c'est-à-dire celui avec le moins de couleurs autorisées dans l'état actuel de coloration, et renvoie ce sommet ainsi que la liste des couleurs possibles pour lui. Si tous les sommets sont coloriés, la fonction renvoie `''`, `[]`.

5 – Écrire la fonction `colorier_graphe(g, couleurs, Nmax) -> bool` qui met à jour le dictionnaire des couleurs suivant l'algorithme proposé, jusqu'à ce que le graphe soit colorié. Cette fonction renvoie `True` si le graphe est entièrement colorié, et `False` si une couleur n'a pas pu être attribuée au sommet le plus contraint.

6 – On pourra tester cet algorithme avec un des graphes habituels :

```
g = gr.recuperer_graphe('g4')
couleurs = {u: 0 for u in g}
Nmax = 3
colorier_graphe(g, couleurs, Nmax)
gr.afficher_graphe(g, labels_sommets=couleurs)
```

III. Aplication : résolution d'un sudoku

Un Sudoku est un jeu joué sur une grille de taille $N \times N$ (usuellement $N = 9$) dans lequel les cases doivent être remplies en utilisant une seule fois chaque nombre entre 1 et N sur chaque ligne, colonne, et carré interne de taille $\sqrt{N} \times \sqrt{N}$. Certaines cases ont une valeur fixée, et le but est de trouver un remplissage qui respecte ces cases et les règles.

Le problème de résolution du Sudoku peut se formuler comme une coloration de graphe : chaque sommet représente une case, et est relié par une arête à une case en interaction (même ligne, même colonne, même sous-carré). La grille initiale est donc un graphe partiellement coloré, et on doit le N-colorier.

On fournit une grille de test sous la forme d'un tableau 2D (une liste de listes), contenant des 0 dans les cases vides et d'autres valeurs dans les cases à valeurs imposées. Il est également fournit une fonction `print_sudoku` qui permet un affichage élégant de la grille dans la console.

7 – Écrire une fonction `générer_graphe_sudoku(grille)` qui prend en paramètre une grille et renvoie un graphe avec les bonnes arêtes. On pourra nommer les sommets en fonction des coordonnées, par exemple le sommet $i = 2$, $j = 5$ pourra avoir pour nom `'2:5'`.

8 – Écrire une procédure `résoudre_sudoku(grille)` qui prend en paramètre une grille, qui créé le graphe correspondant et le dictionnaire des couleurs, puis utilise `colorier_graphe` pour remplir les couleurs, et enfin remplit la grille.