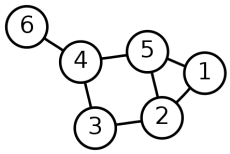
Coloration de graphe en `C++`

Prérequis : Pour mener à bien ce travail, il est nécessaire d'avoir installé les outils classiques pour le `C++` (compilateur, `CMake`, etc.).

Contexte du travail : On s'intéresse ici à l'implémentation en `C++` d'une méthode de coloration de graphe. Pour ce travail, on considère qu'un graphe est un objet composé d'un ensemble de sommets et d'un ensemble d'arêtes. Chaque arête est un lien entre exactement 2 sommets (distincts ou non). On considère que les arêtes ne sont pas *orientées*, ainsi le graphe est dit *non orienté*. Pour fixer les idées, voici un exemple de graphe à 6 sommets et 7 arêtes :



Voir `https://fr.wikipedia.org/wiki/Théorie_des_graphes`.

D'un point de vue informatique, un graphe répond à une interface `IGraph` définissant les fonctionnalités suivantes :

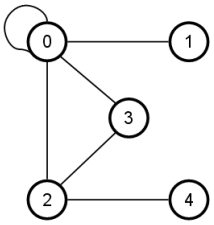
- `nbNodes()` : nombre de sommets du graphe ;
- `nbEdges()` : nombre d'arêtes du graphe ;
- `degree(int i)` : nombre d'arêtes connectant le sommet i ;
- `adjacency(int i)` : liste des sommets connectés (par une arête) au sommet .

Question 1. En vous aidant du cours, proposez une implémentation de l'interface à l'aide d'une classe abstraite.

Dans la suite, vos implémentations de classes de graphe ainsi que les algorithmes devront s'appuyer sur cette interface.

Question 2. Implémentez une classe `Graph` en utilisant une représentation par *liste d'adjacence*. Dans cette représentation, une liste des sommets connectés est associée à chaque sommet.

Par exemple, pour le graphe suivant :

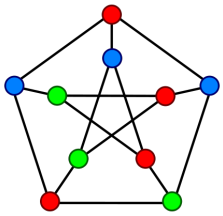


les listes d'adjacence par sommet sont les suivantes :

- sommet 0 : 0, 1, 2, 3
- sommet 1 : 0
- sommet 2 : 0, 3, 4
- sommet 3 : 0, 2
- sommet 4 : 2

Votre implémentation devra répondre au contrat d'utilisation `IGraph`. Vous pourrez par exemple définir une méthode `addEdge(int i,int j)` pour ajouter dynamiquement des arêtes au graphe. Votre implémentation devra également permettre un affichage du graphe compatible avec l'opérateur de flux standard.

En théorie des graphes, la coloration consiste à assigner à chaque sommet un couleur tel que 2 sommets adjacents ont des couleurs différentes. Ainsi, pour un graphe donné, la coloration est un vecteur de couleurs de la taille du nombre de sommets du graphe. Les couleurs sont représentées par des entiers naturels. Pour fixer les idées, voici un exemple de coloration d'un graphe avec 3 couleurs (rouge=0, vert=1, bleu=2) :



Voir `https://fr.wikipedia.org/wiki/Coloration_de_graphe`

D'un point de vue informatique, un algorithme de coloration répond à une interface `IGraphColoring` définissant la fonctionnalité suivante :

- `coloring(const IGraph G)` : coloration du graphe G.

Question 3. Proposer une implémentation de l'interface `IGraphColoring`.

Question 4. Implémentez l'algorithme de coloration dite *gloutonne* :

Pour tout sommet s du graphe :

1. On regarde l'ensemble des couleurs attribuées aux sommets voisins du sommet s ;
2. On en déduit la plus petite couleur disponible ;
3. On attribue la couleur au sommet s.

Question 5. Implémentez l'algorithme de coloration de Welsh et Powell :

1. Trier les sommets par ordre de degrés décroissants ;
2. Choisir une couleur et l'attribuer aux sommets en parcourant la liste : les sommets colorés ne doivent pas avoir de voisins colorés avec la couleur. A la fin de cette étape, une partie des sommets est colorée avec la couleur ;
3. Répéter l'étape 3. en choisissant une nouvelle couleur et en l'appliquant à la liste des sommets non colorés ;
4. Continuer jusqu'à ce que tous les sommets soient colorés.

Vous présenterez vos résultats sur différentes formes de graphe dont par exemple :

- `https://fr.wikipedia.org/wiki/Graphe_de_Golomb`
- `https://fr.wikipedia.org/wiki/Graphe_couronne`