Projet Algorithmique et Complexité

1 Description du projet

Le but de ce projet est de concevoir et mettre en œuvre un algorithme heuristique de 3-coloriage d'un graphe.

2 Complexité du 3-coloriage

On a vu en cours que 3-Color est NP-complet. Cependant, on peut tenter de concevoir une heuristique qui permet de résoudre le problème pour un maximum de graphes.

a) Donnez un schéma algorithmique déterministe permettant de trouver à coup sûr un 3-coloriage d'un graphe $G = (V, E)$ lorsque celui-ci est 3-coloriable.

b) Évaluez sa complexité en fonction des dimensions du graphe ($n = |V|$ et $m = |E|$). En déduire que cet algorithme n'est pas applicable en pratique.

3 Algorithme heuristique

a) Proposez une stratégie de 3-coloriage d'un graphe en expliquant précisément en quoi elle favorise l'obtention d'un coloriage valide.

b) Déduisez-en un algorithme *polynômial* qui cherche un 3-coloriage d'un graphe donné. En cas d'échec, l'algorithme doit indiquer le sommet en conflit et les couleurs de ses voisins.

c) Proposez un algorithme qui vérifie si un sommet en conflit lors d'une tentative de 3-coloriage appartient à une 4-clique. Dans ce cas, l'échec du 3-coloriage est normal puisque celui-ci est impossible.

d) Proposez un algorithme qui vérifie qu'un 3-coloriage d'un graphe est valide.

e) Évaluez théoriquement la complexité de vos trois algorithmes.

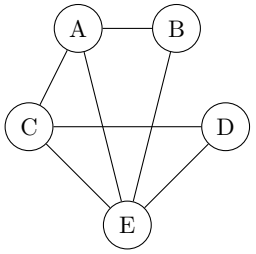
f) Mettre en œuvre un schéma algorithme utilisant vos trois algorithmes pour :

- chercher un 3-coloriage
- indiquer, en cas d'échec, si ce dernier est justifié par la présence d'une 4-clique
- vérifier, en cas de réussite (déroulement sans conflit), que le coloriage obtenu est valide
- afficher le 3-coloriage obtenu (couleurs des sommets) lorsqu'il est validé

et l'appliquer aux différents graphes fournis sur Arche, dont le format de fichier est le suivant :

- Un entier N indiquant le nombre de sommets
- La matrice d'adjacence de taille $N \times N$

Exemple :



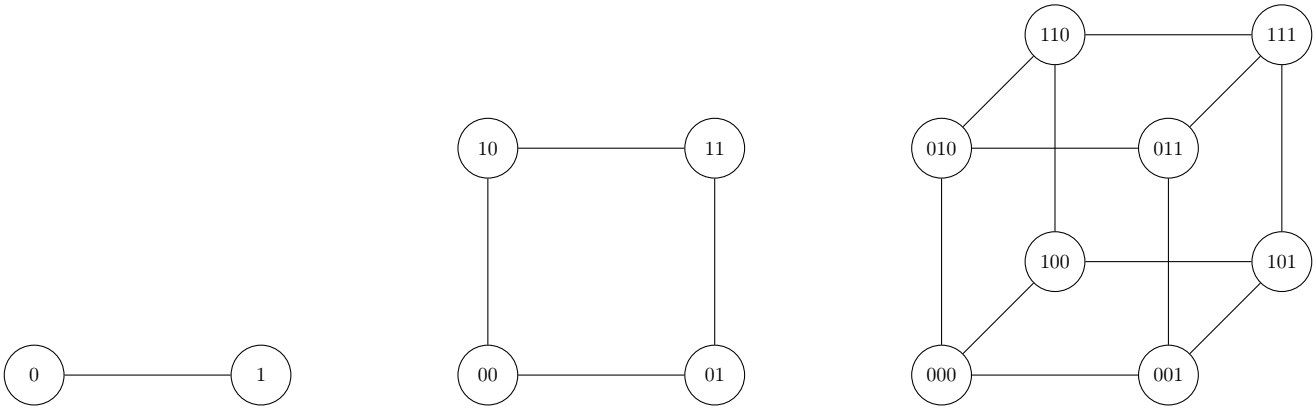

```
5
01101
10001
10011
00101
11110
```

Pour information, ce graphe est 3-coloriable.

4 Application aux N-cubes

Un N-cube est un graphe comportant 2^N sommets plongés dans un espace binaire à N dimensions. Chaque sommet est positionné selon la représentation binaire de son numéro (entre 0 et $2^N - 1$) et les dimensions sont ordonnées des poids faibles vers les poids forts. Ainsi, dans un 3-cube, le sommet 6 (110) est à la position 0 sur la 1ère dimension, 1 sur la 2nde et 1 sur la 3ème. Les arêtes du graphe relient des nœuds n'ayant qu'un seul bit différent dans leurs numérotations. Par exemple, dans le 2-cube, (00) a comme voisins (01) et (10) mais pas (11). On en déduit que les

arêtes représentent des déplacements entre nœuds sur une seule dimension à la fois. Ainsi, le degré de chaque sommet est N. La figure ci-dessous donne les N-cubes de dimension 1, 2 et 3.



a) Développez un programme qui génère des N-cubes jusqu'à la dimension 8 et les sauvegarde selon le format de fichier décrit dans la section précédente.

b) Appliquez votre schéma algorithmique de coloriage à ces N-cubes.

c) Quelle conjecture peut-on en déduire ?

d) Montrer formellement si cette conjecture est vraie ou fausse.

5 Résultats attendus

Les éléments attendus sont :

— Un fichier PDF comportant les réponses aux questions (sauf 3.f, 4.a et 4.b)
— Les fichiers sources de vos programmes (python, C ou Java)
— Un fichier texte contenant les résultats des exécutions demandées en 3.f et 4.b