TP noté Informatique
Couplages
J1 MI 1003, groupe B1, Université Bordeaux

Recommandations

Vous allez écrire tous vos programmes dans un seul et même fichier que vous allez appeler quelque chose du style `NOMPrenom.py`. N'oubliez pas de mettre `graphV3.py` et `bibV3.py` dans le fichier dans lequel vous compilez vos programmes.

Du reste, n'oubliez pas l'instruction à mettre au début de votre document :

```
from bibV3 import *
```

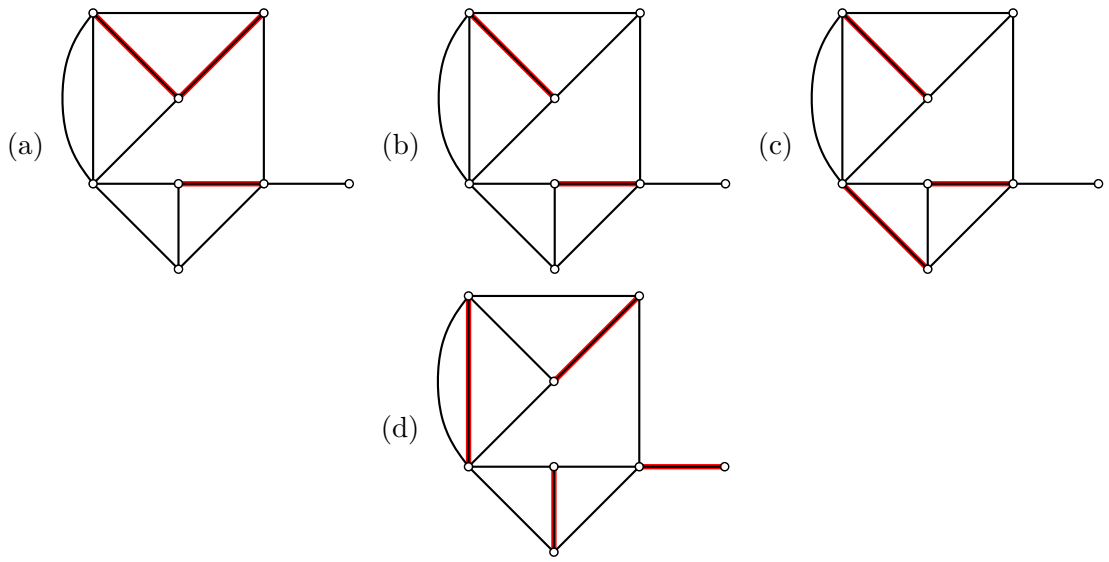
Vous pouvez également rajouter des commentaires en les précédant par des dièses # afin qu'ils ne soient pas lus lors de l'exécution de votre programme.

Exemple de problème

Vous êtes moniteur dans une colonie de vacances, et vous cherchez à former des binômes parmi les enfants pour une activité. Toutefois, certains enfants ne peuvent pas se supporter, c'est pourquoi vous ne pouvez pas mettre n'importe qui avec n'importe qui dans le même binôme. Le problème est donc le suivant : Vous avez un graphe de compatibilité entre enfants, et vous cherchez à former le plus de binômes compatibles possible. Comment faire ?

Modélisation théorique

Un **couplage** dans un graphe G est un sous-ensemble d'arêtes de G qui n'ont pas de sommets en commun. Autrement dit, un couplage est un sous-ensemble d'arêtes tel qu'aucun sommet n'est incident à plus de deux arêtes de ce sous-ensemble.



Par exemple, pour le graphe (a), les arêtes marquées ne forment pas un couplage car le sommet au centre du carré est incident à deux arêtes marquées. Par contre, les arêtes marquées pour le graphe (b), (c) et (d) forment des couplages.

La **taille** d'un couplage est le nombre d'arêtes qu'elle contient.

Un couplage est dit **maximal** (à ne pas confondre avec *maximum*) s'il n'existe pas de couplage plus gros le contenant. Par exemple, (b) n'est pas maximal car (c) est un couplage qui le contient. Par contre, (c) est maximal : on ne peut pas rajouter une autre arête dans ce sous-ensemble sans que celui devienne différent d'un couplage.

Attention, un couplage maximal n'est pas forcément de taille maximale. Par exemple, le couplage (c) est maximal, mais sa taille est plus petite que le couplage (d) !

Programmation

Pour tester les premières fonctions sur le graphe `Europe`, on pourra taper dans le fichier principal les lignes d'instructions suivantes :

```
marquerArete(listeAretesIncidentes(sommetNom(Europe,"France"))[0])
marquerArete(listeAretesIncidentes(sommetNom(Europe,"Autriche"))[1])
```

Exercice 1 Écrire une fonction `demarquer(G)` qui démarque tous les sommets et toutes les arêtes du graphe `G`.

Exercice 2 Écrire une fonction `incidentAreteMarquee(s)` qui renvoie `True` si `s` est incident à (au moins) une arête marquée ; `False` sinon.

Exercice 3 En déduire une fonction `marqueExtremites(G)` qui marque les sommets de `G` qui sont incidents à (au moins) une arête marquée.

Exercice 4 Écrire une fonction `nbAretesIncidentesMarquees(s)` qui compte le nombre d'arêtes incidentes à `s` qui sont marquées.

Exercice 5 En déduire une fonction `couplage(G)` qui renvoie `True` si les arêtes marquées forment un couplage ; `False` sinon.

Exercice 6 En utilisant la fonction `nbAretesIncidentesMarquees(s)`, écrire une fonction `tailleMarquees(G)` qui compte le nombre d'arêtes marquées dans le graphe `G`.

Exercice 7 Écrire une fonction `areteLibre(G)` qui renvoie une arête `a` avec les deux extrêmités non marquées si cela existe ; `None` si cela n'existe pas.

On cherche maintenant à marquer un sous-ensemble d'arêtes d'un graphe G formant un couplage maximal. L'algorithme serait le suivant :

1. Démarquer tous les sommets et toutes les arêtes de G.
2. Tant qu'il existe une arête avec deux extrémités non marquées, marquer cette arête et ses extrêmités.

Exercice 8 Écrire une fonction `couplage_maximal(G)` qui implémente l'algorithme précédent. (On pourra légèrement modifier `areteLibre(G)` si on le souhaite, mais ce n'est pas obligatoire.)

On veut maintenant rajouter de l'aléatoire, afin d'obtenir plus de (voire toutes les ?) couplages maximaux.

Exercice 9 Grâce à la fonction `melange(l)`, écrire une fonction `areteLibreAleatoire(G)` qui renvoie **de manière aléatoire** une arête `a` avec les deux extrêmités non marquées si cela existe ; `None` si cela n'existe pas.

Exercice 10 Utiliser la fonction précédente pour écrire la version aléatoire de `couplage_maximal(G)`, qu'on appellera `couplage_maximal_aleatoire(G)`.

Plutôt que de considérer aléatoirement une arête libre, on va chercher à prendre heuristiquement une arête libre qui est incidente à un sommet de petit degré.

Exercice 11 Écrire une fonction `sommetMinimal(G)` qui retourne un sommet non marqué `s` relié à un autre sommet non marqué tel que le nombre de voisins non marqués soit minimal parmi de tels sommets.

Exercice 12 Implémenter l'algorithme suivant sous le nom de `couplage_maximal_heuristique(G)` :

1. Démarquer tous les sommets et toutes les arêtes de G.
2. Tant qu'il existe une arête avec deux extrémités non marquées :
   (a) Prendre un sommet sommet non marqué `s` relié à un sommet non marqué tel que le nombre de voisins non marqués soit minimal.
   (b) Marquer ce sommet.
   (c) Marquer une arête qui relie `s` à un voisin non marqué `t`.
   (d) Marquer `t`.

Exercice 13 On suppose qu'on est dans un graphe biparti. Écrire une fonction `cheminAlternant(G)` qui renvoie un chemin alternant s'il existe (i.e un chemin qui relie deux sommets non incident à une arête marqué, tel que le chemin alterne avec arêtes marquées / arêtes non marquées), `None` sinon.

Exercice 14 Utiliser la fonction précédente pour résoudre le problème de couplage **de taille** maximale dans un graphe biparti.

Fonctions du module *graphV3*

L'argument G est un graphe	
`listeSommets(G)`	retourne la *liste* des *sommets* de G.
`nbSommets(G)`	retourne le *nombre* de *sommets* de G.
`sommetNom(G,etiquette)`	retourne le *sommet* de G désigné par son *nom* (étiquette). Exemple : `sommetNom (Europe, 'Italie').`
`sommetNumero(G,i)`	retourne le *sommet* numéro i dans G ; la numérotation commence Ã 0.
`dessinerGraphe(G)` ou simplement `dessiner(G)`	demande (très poliment) au logiciel *Graphviz* de dessiner le graphe

L'argument s est un sommet	
`listeVoisins(s)`	retourne la *liste* des *voisins* de s.
`degre(s)`	retourne le *degré* de s.
`nomSommet(s)`	retourne le *nom* (étiquette) de s.
`colorierSommet(s,c)`	colorie s avec la couleur c. Exemples de couleurs : `'red'`, `'green'`, `'blue'`, `'white'`.
`couleurSommet(s)`	retourne la *couleur* de s.
`marquerSommet(s)` `demarquerSommet(s)`	marque ou démarque s.
`estMarqueSommet(s)`	retourne `True` si s est marqué, `False` sinon.
`listeAretesIncidentes(s)`	retourne la *liste* des arêtes *incidentes* Ã s.

L'argument a est une arête	
`nomArete(a)`	retourne le *nom* (étiquette) de a.
`marquerArete(a)` `demarquerArete(a)`	marque ou démarque a.
`estMarqueeArete(a)`	retourne `True` si a est marquée, `False` sinon.

Arguments : un sommet s et une arête a	
`sommetVoisin(s,a)`	retourne le voisin de s en suivant l'arête a.
L'argument est une liste	
`melange(u)`	retourne une copie mélangée aléatoirement de la liste u. Exemple : `melange(listeSommets(cube))`.

La liste `graphes` contient les graphes `tgv2005`, `fig22`, `Europe`, `Koenigsberg` et `Petersen`. La liste `graphesPlanairesReguliers` contient les graphes `tetraedre`, `cube`, `octaedre` (huit triangles), `dodecaedre` (douze pentagones) et `icosaedre` (vingt triangles). Les fonctions suivantes permettent de construire des graphes de taille variable :

`construireComplet(n)`	retourne le graphe complet K_n . Exemple : `K5 = construireComplet(5)`.
`construireBipartiComplet(m,n)`	retourne K_{mn} . Exemple : `K34 = construireBipartiComplet(3,4)`.
`construireArbre(d,h)`	retourne l'arbre de hauteur h dont chaque sommet possède d fils. Exemple : `arbre = construireArbre(2,3)`.
`construireGrille(m,n)`	retourne la grille rectangulaire avec m lignes et n colonnes. Exemple : `grille = construireGrille(4,6)`.
`construireTriangle(n)`	retourne la grille triangulaire d'ordre n. Exemple : `t5 = construireTriangle(5)`.