clement.rouvroy@ens.psl.eu / rouvroy@phare.normalesup.org (le deuxième sera toujours valable)

Coloration d'un graphe & autres

Consignes Le candidat respectera le langage imposé (C) et ne devra pas utiliser de librairie hors-programme. Il est invité à faire des schémas clairs et précis de ses algorithmes lors des appels au correcteur, c'est à la fois un gain de temps pour les deux et une manière de montrer qu'il a compris ce qu'il manipule. Les preuves écrites doivent être formelles, sauf si la consigne précise que ce n'est pas nécessaire, la rigueur sera évaluée. L'examinateur ne débugera pas votre code, en revanche en cas de doute ("ai-je le droit à telle ou telle librairie ?", "je suis sortie de la VM, comment y revenir ?", "ai-je le droit à une indication pour cette question ?") n'hésitez pas à poser votre question. Elle ne vous dévalorisera pas, si la réponse peut vous retirer des points, l'examinateur vous demandera avant de vous donner la réponse si vous l'acceptez (si vous n'avez pas de chance, le jour de l'oral il vous retirera les points rien que pour avoir posé la question, par exemple si vous demandez "comment trier une liste en $O(n\log(n))$?" ou un autre résultat classique du programme, ça sera sûrement retenu contre vous). Enfin, si l'examinateur n'est pas à votre portée quand vous avez besoin d'une aide / d'une question oral à donner, gardez le bras levé et lisez la suite, ne restez jamais passif.

Introduction du sujet

La coloration de graphe est un problème très classique en informatique et qui a des applications absolument partout. En fait, c'est un problème NP-Complet, ce qui siginifie que tout problème "difficile" (dans un sens que vous verrez en deuxième année) peut s'y ramener, on peut par exemple résoudre la satisfaction d'une formule logique ou le problème de réservation de salles avec cette technique.

Ce TP introduit la coloration de graphe puis passe à d'autres problèmes sur les graphes, vous pouvez traiter le sujet dans l'ordre que vous voulez après avoir traité la coloration.

I Pendant que ça charge

Question 1 de cours

1. Quelle est la complexité de l'algorithme de Dijkstra ?
2. Comment marche (l'idée de) l'algorithme de Floyd-Warshall ?
3. Donner un avantage (en terme de propriété algorithmique) du parcours en largeur par rapport au parcours en profondeur (autre que "plus facile à coder").
4. Qu'est-ce qu'une fonction récursive terminale ?

II Coloriage

Cette partie est une introduction aux coloriages et doit impérativement être traitée. Les questions sont surtout théoriques.

Dans tout ce sujet on coloriera avec l'ensemble $\mathbb{N}$ pour des raisons de simplicité (sinon il faut implémentr une ensemble de couleurs, c'est possible mais ça implique d'avoir un ensemble fini,

or il est facile de trouveer un graphe qui a besoin d'un nombre de couleurs plus grand que celui disponible dans n'importe quel ensemble fini fixé).

Définition 2.1 coloriage

Etant donné un graphe (non-orienté) $G = (\mathcal{V}, E)$, une coloration est une fonction $c : \mathcal{V} \to \mathbb{N}$ vérifiant la propriété

$$\forall (u, v) \in E, c(u) \neq c(v)$$

On appelle nombre chromatique $\chi(G) = \min_{c \text{ fonction de coloriage sur } G} \max_{v \in \mathcal{V}} c(v)$

L'image en introduction du sujet vous montre un exemple de coloriage. En particulier vous pouvez remarquer que quand on dessine un coloriage on utilise bien des couleurs et non des numéros (on le fait que quand il n'y a pas trop de couleurs évidemment).

Question 2 à l'écrit

Pour chacun des graphes suivants, donner la valeur de $\chi(G)$.

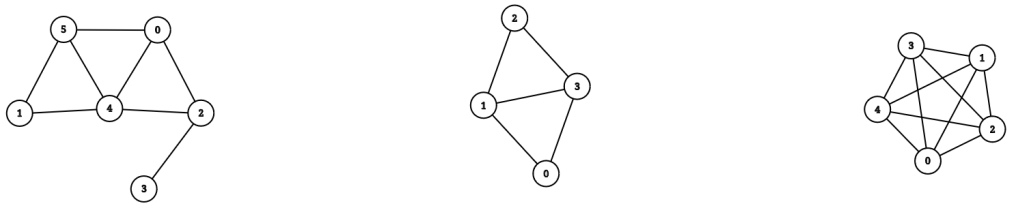


Figure 1: Les 3 graphes A,B et C (modulo affichage)

Question 3 à l'écrit

Donner un coloriage très simple qui marche sur tous les graphes.

Question 4 à l'écrit

1. Si un sommet a un voisinage de taille k, que peut-on en déduire de sa couleur ? (preuve attendue)
2. En déduire une borne supérieure du nombre chromatique χ d'un graphe.

Question 5 code

Après avoir définie un type `graphe` par matrice d'adjacence qui contient en plus un tableau d'étiquette entière pour chaque sommet (la couleur d'un sommet), écrire une fonction `int est_bien_colorie(graphe g)` qui dit si la coloration associée est bien un coloriage (renvoie 0 si ce n'est pas un coloriage et sinon renvoie le nombre de couleurs utilisées).

Question 6 code

Ecrire une fonction `int colorie(graphe g)` qui colorie G en respectant la borne de la question 4.

Question 7 code

Implémentez des tests.

Cette question rapporte autant de points que la 6.

En pratique on peut faire bien mieux que cet algorithme trivial mais pour une première fois, c'est déjà bien :).

III Niveau de connexité

Définition 3.1 niveau de connexité

Soit G un graphe non-orienté, on note $\xi(G)$ son niveau de connexité, à savoir le nombre minimal d'arêtes qu'il faut lui retirer pour qu'il ne soit plus connexe.

Question 8 à l'écrit

Indiquer $\xi(G)$ pour chacun des graphes de la question 2. Que dire de $\xi(G)$ pour un graphe non-connexe ?

On a la borne triviale: $\xi(G) \leq n$ avec n le nombre de sommets.

Question 9 à l'écrit

En sachant qu'on doit retirer au plus $|\mathcal{V}|$ arêtes de G pour trouver $\xi(G)$, combien (au $+$) de combinaisons d'arêtes à retirer allez-vous tester ?

Question 10 code

Déduire de la borne un algorithme `int xi(graphe g)` qui renvoie $\xi(G)$. Quelle est sa complexité ?

Question 11 code

Implémentez des tests.

Cette question rapporte autant de points que la 10.

IV Isomorphisme de graphes

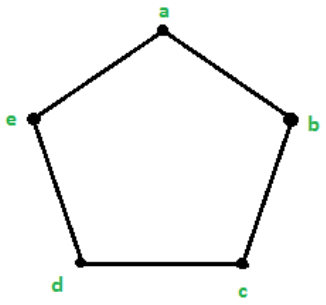
Définition 4.1 morphisme

On dit que deux graphes G_1 et G_2 sont homomorphes si il existe un renommage des sommets de G_1 tels que $\forall (u,v) \in E_1, (\varphi(u), \varphi(v)) \in E_2$.

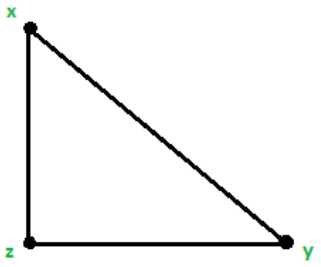
En particulier, φ est une fonction de V_1 vers V_2.

G_1 et G_2 sont dits isomorphes si φ est un isomorphisme.

Voici un exemple de morphisme:



Graph G



Graph G'

Question 12 à l'écrit

Donner un morphisme entre ces deux graphes. Est-ce un isomorphisme ?

Question 13 code

Ecrire une fonction `bool isomorphe(graphe g1, graphe2)` qui renvoie vrai si et seulement si G_1 et G_2 sont isomorphes. Quelle est la complexité en temps en fonction de la taille de G_1 et de G_2 ?

Conseil. Subdivisiez le problème en plusieurs sous-fonctions (une pour générer les bijections de $[\![1; |\mathcal{V}|;]\!]$, une pour appliquer une bijection à un graphe, une pour regarder si deux graphes sont égaux).

Admis. Le problème d'isomorphisme de graphe est difficile et donc vous ne pouvez pas trouver une solution qui est dans Pol(taille de l'entrée) à la fin, c'est tout à fait normal.

Question 14 code

Implémentez des tests.

Cette question rapporte autant de points que la 13.

V Déjà fini ?

Question 15 à l'écrit

Un peu de cryptographie

On considère le problème d'isomorphisme de graphe, qui peut se formaliser ainsi:

Etant donné G_1 et G_2, existe-t-il un isomorphisme entre G_1 et G_2 ?

Alice aimerait bien prouver à Bob qu'elle connaît un isomorphisme entre G_1 et G_2, mais sans donner l'isomorphisme à Bob, et sans rien apprendre à Bob sur cet isomorphisme.

Proposez un protocol (une série d'appels d'Alice vers Bob et de Bob vers Alice) qui permet de faire ça.