Option informatique : Coloration d'un graphe

Rappel de Ocaml

- Opérateurs sur les entiers : `+`, `-`, `*`, et `mod` pour le modulo.
- Instructions conditionnelles : `if ... then ... else ...`. Type commun dans les blocs `then` et `else`, une seule instruction dans chaque bloc ou encadrement par des parenthèses ou `begin ... end`.
- boucles :
  - `for i = id to if do ... done ;` itère les opérations du corps de boucle (de type `unit`) pour tout $i \in [\![i_d, i_f]\!]$. Si cet ensemble est vide, le corps n'est pas exécuté.
  - `for i = id downto if do ... done ;`. Même chose, mais par pas de -1. On doit avoir $i_d \geq i_f$ pour qu'au moins un tour soit effectué.
  - `while ... do ... done ;` pour une boucle `while`.
- Tableaux : dans la suite, `t` est un tableau.
  - taille d'un tableau : `Array.length t`
  - élements d'un tableau de taille n : `t.(0)`,..., `t.(n-1)`.
  - modifier un élément : `t.(i) <- x`.
  - Création : `Array.make n x` crée un tableau de taille n rempli de x ;
  - Copie : `Array.copy t`.
- Références : impératives quand il s'agit de « modifier » une variable !
  - créer une référence vers `x` : `ref x`. Par exemple : `let i=ref 0 in ...`) ;
  - modifier la valeur pointée par la référence `r` : `r:= ...` ;
  - accéder à la valeur pointée : `!r`.
- Listes chaînées : c'est une structure immuable (pas de modification, simplement la création de nouveaux objets)
  - `[]` : liste vide ;
  - `x::q` liste obtenue par ajout de `x` en tête de `q`.
  - `List.hd q`, `List.tl q` : tête et queue d'une liste `q` ;
  - mais on procède souvent par filtrage sur le motif `x::q`.

Définitions et notations

On utilisera dans tout le problème les notations et définitions suivantes :

— une *paire* est un ensemble composé de deux éléments a et b *distincts* et est notée $\{a, b\}$ (ce qui est égal à $\{b, a\}$) ;

— un *graphe* G est composé d'un ensemble $S(G)$ d'éléments appelés *sommets* et d'une liste $A(G)$ de paires de sommets ; les éléments de A(G) sont appelés *arêtes* et ne sont pas nécessairement tous distincts ;

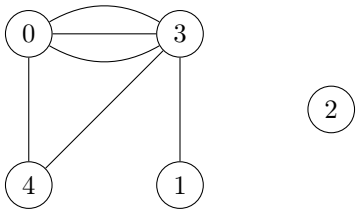
— le cardinal de $S(G)$ est noté $n(G)$ et on a toujours $S(G) = \{0, 1, \ldots, n(G) - 1\}$; un graphe est donc entièrement décrit par le couple $(n(G), A(G))$;

— deux sommets s et t de G sont dits *voisins* si $\{s, t\}$ est une arête de G ;

Exemple introductif : un graphe nommé G_{ex1}

On considère le graphe G_{ex1} dessiné ci-dessous et dont les caractéristiques sont :

$$n(G_{\text{ex1}}) = 5 \quad \text{et} \quad A(G_{\text{ex1}}) = \{\{0,4\},\{4,3\},\{0,3\},\{1,3\},\{0,3\},\{3,0\}\}$$



Il y a trois arêtes entre le sommet 0 et le sommet 3. Le sommet 0 a pour voisins les sommets 3 et 4, le sommet 1 a pour voisin le sommet 3, le sommet 2 n'a pas de voisin, le sommet 3 a pour voisins les sommets 0, 1 et 4, et le sommet 4 a pour voisins les sommets 0 et 3.

Indication pour la programmation

On définit les types suivants :

```
type arete = {a : int; b : int} ;;
type graphe = {n : int; ar : arete list} ;;
```

Ainsi, le graphe Gex1 de l'exemple introductif est défini par :

```
let gex1 = {n = 5; ar = [{a=0; b=4}; {a=4; b=3}; {a=0; b=3}; {a=1; b=3}; {a=0; b=3}; {a=3; b=0}]} ;;
```

On rappelle qu'on accède aux composantes d'un type produit via `objet.composante`, par exemple `gex1.ar` est la liste des arêtes du graphe `gex1`.

Partie I. Détermination des voisins des sommets

Question 1. Écrire en Caml une fonction `insere q s` de type `int list -> int -> int list` qui prend en argument une liste `q` d'entiers distincts triée par ordre croissant et un entier quelconque `s`, et qui renvoie :

— la liste d'entiers obtenue en insérant `s` dans `q` selon l'ordre croissant si la valeur `s` ne figure pas dans `q` ;

— la liste `q` si `s` figure dans `q`.

Question 2. Indiquer, en fonction de la longueur de `q`, la complexité dans le pire des cas de l'algorithme utilisé pour la fonction `insere`.

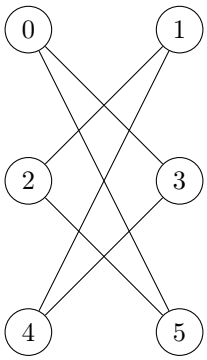
Question 3. Écrire une fonction `voisins g s` de type `graphe -> int -> int list` qui prend en arguments un graphe `g` et un sommet `s` et renvoie la liste triée par ordre croissant des voisins de `s` dans `g`. On utilisera pour cela la fonction `insere`.

Partie II. Un algorithme de bonne coloration du graphe

On appelle *coloration* d'un graphe G toute application c de l'ensemble S des sommets de G dans l'ensemble des entiers naturels. Pour tout sommet s, l'entier $c(s)$ est appelé la *couleur* de s pour la coloration c. Une coloration c est une *bonne coloration* de G si pour toute arête $\{s,t\}$ de G on a $c(s) \neq c(t)$; autrement dit si les extrémités de toute arête sont de couleurs différentes.

On considère l'algorithme de bonne coloration suivant. Les couleurs disponibles sont les entiers naturels ; la plus petite couleur est ainsi la couleur 0. On colorie les sommets les uns après les autres, dans l'ordre $0, 1, 2, \ldots, n(G)-1$; lorsqu'on considère un sommet s, on lui attribue la plus petite couleur disponible, c'est à dire la plus petite couleur qui n'est pas déjà la couleur d'un voisin de s.

Question 4. Que donne cet algorithme pour le graphe G_{ex2} ci-dessous ? Y a-t-il une autre bonne coloration de G_{ex2} utilisant moins de couleurs ?



On représentera une coloration d'un graphe G par un tableau à $n(G)$ cases à valeurs entières dont les indices correspondent aux sommets de G ; la case d'indice s contiendra la couleur du sommet s.

Question 5. Écrire en Caml une fonction `coloration g` de type `graphe -> int array` prenant en argument un graphe `g` et renvoyant le tableau des couleurs attribuées aux sommets de `g` par l'algorithme décrit ci-dessus.

Partie III. Définition du nombre chromatique de G

Soit G un graphe et p un entier strictement positif. On appelle *bonne p-coloration* de G une bonne coloration n'utilisant que des couleurs appartenant à l'ensemble $[\![0, p-1]\!]$. On introduit les notations suivantes :

— $\mathrm{BC}(G, p)$ est l'ensemble des bonnes p-colorations de G ;

— $\mathrm{fc}(G, p)$ est le cardinal de $\mathrm{BC}(G, p)$;

— $\mathrm{EC}(G) = \{p \in \mathbb{N}^* \mid \mathrm{fc}(G, p) \neq 0\}$.

Question 6. Montrer l'existence d'un unique entier $\mathrm{nbc}(G)$ tel que :

$$\mathrm{EC}(G) = \{p \in \mathbb{N}^* \mid p \geq \mathrm{nbc}(G)\}$$

On dit que $\mathrm{nbc}(G)$ est le *nombre chromatique* du graphe G ; c'est le nombre minimum de couleurs permettant de colorier les sommets de G de sorte que deux sommets voisins n'aient pas la même couleur.

Dénombrement. Les trois questions suivantes demandent de dénombrer des colorations. On pourra utiliser sans démonstration que :

— il y a $\binom{n}{p} = \frac{n!}{p!(n-p)!}$ manières de choisir p éléments distincts dans un ensemble de n éléments (pour $0 \leq p \leq n$, sinon on pose $\binom{n}{p} = 0$) ;

— le nombre de fonctions de $[\![0, n-1]\!]$ dans $[\![0, p-1]\!]$ est p^n ;

— il y a $k!$ séquences contenant une et une seule fois chaque entier de $[\![0, k-1]\!]$.

Question 7. Soit G un graphe n'ayant aucune arête. Déterminer $\mathrm{nbc}(G)$ et, pour tout $p \in \mathbb{N}^*$, déterminer $\mathrm{fc}(G, p)$ en fonction de $n(G)$ et de p.

Question 8. Soit G un graphe tel que toute paire de sommets soit une arête. Déterminer $\mathrm{nbc}(G)$ en fonction de $n(G)$ et, pour tout $p \in \mathbb{N}^*$, déterminer $\mathrm{fc}(G, p)$ en fonction de $n(G)$ et de p.

Question 9. On considère le graphe G_{ex1} de l'exemple introductif. Déterminer $\mathrm{nbc}(G_{\mathrm{ex1}})$ et, pour tout $p \in \mathbb{N}^*$, déterminer $\mathrm{fc}(G_{\mathrm{ex1}}, p)$ en fonction de p.

Partie IV. Les applications H et K

On dit qu'un sommet d'un graphe est *isolé* s'il n'a aucun voisin. Par exemple, le sommet 2 est isolé dans le graphe G_{ex1}. Étant donnés un graphe G et un sommet de G non isolé s, on note $\mathrm{prem\_voisin}(G, s)$ le plus petit voisin de s dans G, c'est-à-dire le voisin de s qui a le plus petit numéro. Par exemple, $\mathrm{prem\_voisin}(G_{\mathrm{ex1}}, 0)$ est le sommet 3.

Question 10. Rédiger une fonction `prem_voisin g s` de type `graphe -> int -> int` qui prend en argument un graphe `g` et un sommet `s` et retourne $\mathrm{prem\_voisin}(\mathtt{g}, \mathtt{s})$. Cette fonction ne prévoira pas le cas où `s` est un sommet isolé de `g`.

On considère un graphe G possédant au moins une arête. On note prem_ni(G) le plus petit sommet non isolé du graphe G, c'est à dire le sommet non isolé qui a le plus petit numéro. Par exemple, prem_ni(G_{ex1}) est le sommet 0.

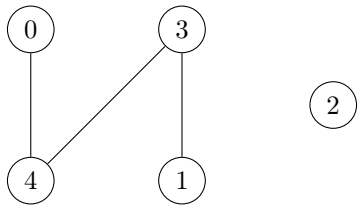
Question 11. Écrire une fonction `prem_ni g` de type `graphe -> int` qui prend en argument un graphe `g` et retourne prem_ni(`g`). Cette fonction ne prévoira pas le cas où `g` ne possède aucune arête.

Rappelons d'abord que dans la liste $A(G)$ des arêtes d'un graphe G, il est possible qu'une même arête soit répétée. Étant donné un graphe G possédant au moins une arête, on pose :

— $s_1 = \text{prem\_ni}(G)$;

— $s_2 = \text{prem\_voisin}(G, s_1)$.

Par exemple, pour le graphe G_{ex1}, $s_1 = 0$ et $s_2 = 3$. On note $H(G)$ le graphe obtenu à partir de G en supprimant toutes les arêtes entre s_1 et s_2. Par exemple, le graphe $H(G_{\text{ex1}})$ représenté ci-dessous est caractérisé par $n(H(G_{\text{ex1}})) = 5$ et $A(H(G_{\text{ex1}})) = \{\{0, 4\}, \{4, 3\}, \{1, 3\}\}$.



Question 12. Rédiger en Caml une fonction `h g` de type `graphe -> graphe` qui prend en argument un graphe `g` et retourne le graphe H(`g`). Cette fonction ne prévoira pas le cas où `g` ne possède aucune arête. *Remarque :* votre fonction doit renvoyer un nouveau graphe et ne pas « détruire » `g`.

On considère un graphe G possédant au moins une arête. On définit s_1 et s_2 comme dans la question précédente ; on peut remarquer la relation $s_1 < s_2$. On construit alors un graphe noté $K(G)$ de la façon suivante :

— on construit $H(G)$;

— dans $H(G)$ on superpose s_1 et s_2 ; plus précisément :

  — on considère successivement chaque arête de la liste des arêtes de $H(G)$; pour chacune d'entre elles, on renumérote ses extrémités :

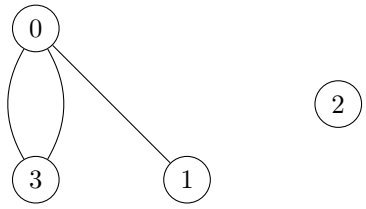
    — une extrémité de valeur strictement inférieure à s_2 est inchangée ;

    — une extrémité de valeur s_2 prend la valeur s_1 ;

    — une extrémité de valeur strictement supérieure à s_2 est décrémentée de 1 ;

  — on diminue de 1 le nombre de sommets du graphe.

Par exemple, le graphe $K(G_{\text{ex1}})$ représenté ci-dessous est décrit par $n(K(G_{\text{ex1}})) = 4$ et $A(K(G_{\text{ex1}})) = \{\{0, 3\}, \{3, 0\}, \{1, 0\}\}$.



Question 13. Rédiger en Caml une fonction `k g` de type `graphe -> graphe` qui prend en argument un graphe `g` et retourne le graphe K(`g`). Cette fonction ne prévoira pas le cas où `g` ne possède aucune arête. De même, votre fonction doit renvoyer un nouveau graphe et ne pas « détruire » `g`.

Partie V. Fonction fc(G, p) et polynôme chromatique

Soit G un graphe possédant au moins une arête et p un entier non nul.

Question 14. Montrer l'inclusion : $\text{BC}(G, p) \subset \text{BC}(H(G), p)$.

Question 15. Comparer les cardinaux de $\text{BC}(K(G), p)$ et de $\text{BC}(H(G), p) \backslash \text{BC}(G, p)$, et en déduire l'égalité :

$$\text{fc}(G, p) = \text{fc}(H(G), p) - \text{fc}(K(G), p)$$

Question 16. En utilisant la formule obtenue à la question précédente, décrire en termes simples un algorithme récursif de calcul de $\mathrm{fc}(G, p)$; ici, on ne demande pas de rédiger cet algorithme en Caml. On prouvera que cet algorithme se termine.

Question 17. Le graphe G étant fixé, montrer que la fonction qui à p associe la quantité $\mathrm{fc}(G, p)$ est la restriction à $\mathbb{N}^*$ d'un polynôme en p de degré $n(G)$. On appelle polynôme chromatique de G et on note $P_C(G)$ ce polynôme.

Partie VI. Calcul du polynôme $P_C(G)$ et de nbc(G)

On ne considérera dans cette partie que des polynômes à une variable et à coefficients entiers. Un polynôme de degré d sera représenté par un tableau à $(d+1)$ cases, la case d'indice i contenant le coefficient du monôme de degré i. On pourra, pour connaître le degré d'un polynôme, utiliser la fonction `Array.length` qui retourne le nombre de cases d'un tableau donné en argument.

Question 18. Rédiger en Caml une fonction `difference p q` de type `int array -> int array -> int array` qui prend en argument deux polynômes `p` et `q` vérifiant $\deg \mathtt{q} < \deg \mathtt{p}$ et qui retourne le polynôme $\mathtt{p} - \mathtt{q}$.

Question 19. Écrire en Caml une fonction `pc g` de type `graphe -> int array` qui prend en argument un graphe `g` et renvoie le polynôme $P_C(\mathtt{g})$.

Question 20. Dans cette question, il s'agit de calculer la valeur $P(x)$ d'un polynôme P en une valeur entière x. On n'utilisera pas de fonction prédéfinie du langage qui calculerait les puissances d'un entier. Les calculs seront faits avec les quatre opérations arithmétiques ordinaires. De plus, la complexité de l'algorithme utilisé sera nécessairement du même ordre de grandeur que le degré du polynôme considéré.
Écrire en Caml une fonction `eval p x` de type `int array -> int -> int` prenant en arguments un polynôme `p` et un entier `x` et qui retourne l'entier $\mathtt{p}(\mathtt{x})$.

Question 21. Écrire enfin une fonction `nbc g` de type `graphe -> int` qui calcule le nombre chromatique $\mathrm{nbc}(\mathtt{g})$ d'un graphe `g`.