Coloration d'un graphe (Mines 2003)

Durée : libre

Définitions et notations

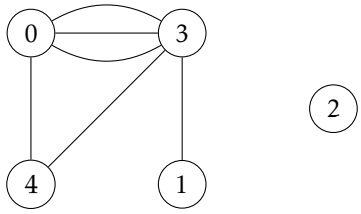
On utilisera dans tout le problème les notations et définitions suivantes :

- 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) = $\{0,1,\ldots,n(\mathrm{G})-1\}$; un graphe est donc entièrement décrit par le couple $\big(n(\mathrm{G}),\mathrm{A}(\mathrm{G})\big)$;
- deux sommets s et t de G sont dits *voisins* si $\{s,t\}$ est une arête de G ;

Exemple introductif : un graphe nommé *Gex1*

On considère le graphe *Gex1* dessiné ci-dessous et dont les caractéristiques sont :

$$n(Gex1) = 5 \quad \text{et} \quad \mathrm{A}(Gex1) = \Big\{\{0,4\},\{4,3\},\{0,3\},\{1,3\},\{0,3\},\{3,0\}\Big\}.$$



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.

Indications pour la programmation

On définit les types suivants :

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

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

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

Partie I. Détermination des voisins des sommets

Question 1. Écrire en Caml une fonction **`insere`** qui prend en argument une liste L d'entiers distincts triée par ordre croissant et un entier quelconque s, et qui renvoie :

- la liste d'entiers obtenue en insérant s dans L selon l'ordre croissant si la valeur s ne figure pas dans L ;
- la liste L si s figure dans L.

```
insere : int list -> int -> int list
```

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

Question 3. Écrire une fonction **`voisins`** 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`**.

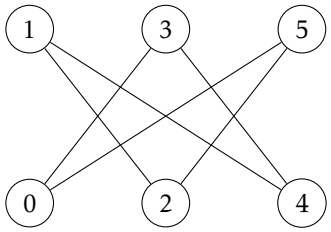
```
voisins : graphe -> int -> int list
```

Partie II. Un algorithme de bonne coloration d'un 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 non nuls. 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 non nuls ; la plus petite couleur est ainsi la couleur 1. On colorie les sommets les uns après les autres, dans l'ordre $0, 1, 2, \ldots, n(\mathrm{G}) - 1$; 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 *Gex2* ci-dessous ? Y a-t-il une autre bonne coloration de *Gex2* utilisant moins de couleurs ?



On représentera une coloration d'un graphe G par un tableau à $n(\mathrm{G})$ 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`** prenant en argument un graphe G et renvoyant le tableau des couleurs attribuées aux sommets de G par l'algorithme décrit ci-dessus.

```
coloration : graphe -> int vect
```

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 $[\![1, p]\!]$. On introduit les notations suivantes :

- $\mathrm{BC}(\mathrm{G}, p)$ est l'ensemble des bonnes p-colorations de G ;
- $\mathrm{fc}(\mathrm{G}, p)$ est le cardinal de $\mathrm{BC}(\mathrm{G}, p)$;
- $\mathrm{EC}(\mathrm{G}) = \left\{ p \in \mathbb{N}^* \;\middle|\; \mathrm{fc}(\mathrm{G}, p) \neq 0 \right\}$.

Question 6. Montrer l'existence d'un unique entier $\mathrm{nbc}(\mathrm{G})$ tel que : $\mathrm{EC}(\mathrm{G}) = \left\{ p \in \mathbb{N}^* \;\middle|\; p \geqslant \mathrm{nbc}(\mathrm{G}) \right\}$.
On dit que $\mathrm{nbc}(\mathrm{G})$ 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.

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

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

Question 9. On considère le graphe *Gex1* de l'exemple introductif. Déterminer $\mathrm{nbc}(\mathit{Gex1})$ et, pour tout $p \in \mathbb{N}^*$, déterminer $\mathrm{fc}(\mathit{Gex1}, p)$ 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 *Gex1*.

Question 10. Étant donnés un graphe G et un sommet de G non isolé s, on note 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, prem_voisin(*Gex1*, 0) est le sommet 3.

Rédiger une fonction **`prem_voisin`** qui prend en argument un graphe G et un sommet s et retourne prem_voisin(G, s). Cette fonction ne prévoira pas le cas où s est un sommet isolé de G.

```
prem_voisin : graphe -> int -> int
```

Question 11. 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(*Gex1*) est le sommet 0.

Écrire une fonction **`prem_ni`** 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.

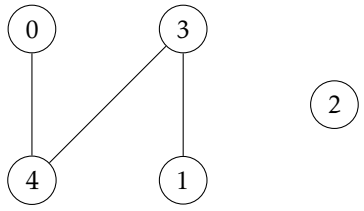
```
prem_ni : graphe -> int
```

Question 12. Rappelons 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 = prem_ni(G) ;

s_2 = prem_voisin$\big($G, prem_ni(G)$\big)$.

Par exemple, pour le graphe *Gex1*, $s_1 = 0$ 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(*Gex1*) représenté ci-dessous est caractérisé par $n\big(\text{H}(Gex1)\big) = 5$ et $\text{A}\big(\text{H}(Gex1)\big) = \big\{\{0,4\},\{4,3\},\{1,3\}\big\}$.



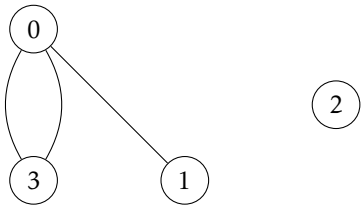
Rédiger en Caml une fonction **`h`** qui prend en argument un graphe G et retourne le graphe H(G). Cette fonction ne prévoira pas le cas où G ne possède aucune arête.

```
h : graphe -> graphe
```

Question 13. 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 st 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(*Gex1*) représenté ci-dessous est décrit par $n\Big(\mathrm{K}(Gex1)\Big) = 4$ et $\mathrm{A}\Big(\mathrm{K}(Gex1)\Big) = \Big\{\{0,3\},\{3,0\},\{1,0\}\Big\}$.



Rédiger en Caml une fonction **k** 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.

```
k : graphe -> graphe
```

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 : $\mathrm{BC}(\mathrm{G},p) \subset \mathrm{BC}(\mathrm{H}(\mathrm{G}),p)$.

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

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

Question 16. En utilisant la formule obtenue à la question précédente, décrire en termes simples un algorithme récursif de calcul de $\mathrm{fc}(\mathrm{G},p)$ (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 $\mathrm{fc}(\mathrm{G},p)$ est la restriction à $\mathbb{N}^*$ d'un polynôme en p de degré $n(\mathrm{G})$. On appelle *polynôme chromatique* de G et on note $\mathrm{P_C}(\mathrm{G})$ ce polynôme.

Partie VI. Calcul du polynôme $\mathbf{P_C(G)}$ 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 **vect_length** qui retourne le nombre de cases d'un tableau donné en argument.

Question 18. Rédiger en Caml une fonction **difference** qui prend en argument deux polynômes p et q vérifiant $\deg q < \deg p$ et qui retourne le polynôme $p - q$.

```
difference : int vect -> int vect -> int vect
```

Question 19. Écrire en Caml une fonction **Pc** qui prend en argument un graphe G et renvoie le polynôme $\mathrm{P_C}(\mathrm{G})$.

```
Pc : graphe -> int vect
```

Question 20. Dans cette question, il s'agit de calculer la valeur $\mathrm{P}(x)$ 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** prenant en arguments un polynôme P et un entier x et qui retourne l'entier $\mathrm{P}(x)$.

```
eval : int vect -> int -> int
```

Question 21. Ecrire enfin une fonction **nbc** qui calcule le nombre chromatique nbc(G) d'un graphe G.

```
nbc : graphe -> int
```