A rendre le vendredi 16 octobre

Préliminaire concernant la programmation

Lorsque le candidat écrira une fonction ou une procédure, il pourra faire appel à une autre fonction ou procédure définie dans les questions précédentes. Enfin, si les paramètres d'une fonction ou d'une procédure à écrire sont supposés vérifier certaines hypothèses, il ne sera pas utile dans l'écriture de cette fonction ou de cette procédure de tester si les hypothèses sont bien vérifiées.
Dans les énoncés du problème, un même identificateur écrit dans deux polices de caractères différentes désignera la même entité, mais du point de vue mathématique pour la police en italique (par exemple n) et du point de vue informatique pour celle en romain (par exemple n).

Un *graphe* G est défini par deux ensembles X et E. L'ensemble X est un ensemble fini d'éléments appelés *sommets*. L'ensemble E est un ensemble de paires de sommets. Un élément $\{x, y\}$ de E est appelé *arête* de G ; x et y sont les *extrémités* de l'arête. L'*ordre* d'un graphe G est le nombre de sommets de G. Un graphe est représenté par un dessin où des cercles représentent les sommets et un trait joignant deux cercles représente une arête composée des deux sommets correspondant aux cercles. Si $\{x, y\}$ est une arête de G, on dit que x et y sont *voisins*. Le *degré* d'un sommet x est le nombre de voisins de x.
On dit que deux arêtes d'un graphe G sont *incidentes* si elles ont une extrémité en commun. On appelle *couplage dans* G un ensemble d'arêtes de G deux à deux non incidentes.
Un graphe G est dit *biparti* si on peut partitionner son ensemble de sommets X en deux sous-ensembles A et B ($A \neq \emptyset$, $B \neq \emptyset$, $A \cup B = X$, $A \cap B = \emptyset$) de sorte que toute arête ait une extrémité dans A et une extrémité dans B. Si les ensembles A et B ont même cardinal, on dit qu'il s'agit d'un *graphe biparti équilibré*. Dans tout le problème, on ne considère que des graphes bipartis équilibrés. On note n le cardinal commun aux ensembles A et B ; l'ordre du graphe est donc égal à $2n$. On suppose que l'on a toujours $n \geq 1$. Les sommets de A sont numérotés de 0 à $n - 1$ et nommés 0_A, 1_A, 2_A, …, $(n-1)_A$; les sommets de B sont numérotés de 0 à $n - 1$ et nommés 0_B, 1_B, 2_B, …, $(n-1)_B$. Une arête de G est toujours écrite en mettant d'abord l'extrémité qui est dans A puis celle qui est dans B.

On représente les graphes bipartis équilibrés par des schémas comme on peut le voir dans la figure 1 avec le graphe G_0, en représentant les sommets de A à gauche et les sommets de B à droite.

Pour G_0, n vaut 4 et l'ordre de G_0 vaut 8.
Les arêtes de G_0 sont :
$\{0_A,\ 0_B\}$, $\{0_A,\ 1_B\}$, $\{0_A,\ 2_B\}$, $\{1_A,\ 3_B\}$, $\{2_A, 0_B\}$, $\{2_A,\ 1_B\}$, $\{2_A,\ 2_B\}$, $\{2_A,\ 3_B\}$, $\{3_A, 3_B\}$.
Le sommet 0_A est de degré 3.
Le sommet 1_A est de degré 1.
Le sommet 2_A est de degré 4.
Le sommet 3_A est de degré 1.
Les sommets 0_B, 1_B et 2_B sont de degré 2.
Le sommet 3_B est de degré 3.

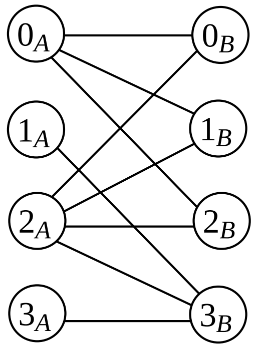

Figure 1 : le graphe G_0

Dans le graphe G_0, les arêtes $\{0_A,\ 0_B\}$ et $\{2_A, 3_B\}$ étant non incidentes, elles forment un couplage, nommé C_0, dont les arêtes sont dessinées en gras ci-contre ; on dit alors que dans ce couplage :

- le sommet 0_A est couplé au sommet 0_B, et réciproquement ;
- le sommet 2_A est couplé au sommet 3_B, et réciproquement ;
- les sommets 1_A, 3_A, 1_B et 2_B sont non couplés.

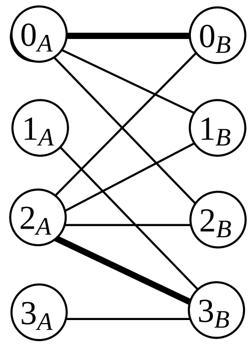

Figure 2 : le graphe G_0
et le couplage C_0

Le cardinal d'un couplage est le nombre d'arêtes de celui-ci ; par exemple le cardinal de C_0 vaut 2.

Première partie : généralités

❒ 1 – Exhiber un couplage de cardinal 3 dans G_0.

❒ 2 – Indiquer s'il existe dans G_0 un couplage de cardinal 4. Justifier la réponse.

Un graphe biparti équilibré d'ordre $2n$ est représenté par une matrice carrée de dimension $n \times n$ dont les lignes correspondent aux éléments de A et les colonnes aux éléments de B. Les cases de cette matrice sont indicées par (i, j) avec $0 \leq i \leq n - 1$, $0 \leq j \leq n - 1$ et contiennent des **valeurs booléennes** : la case d'indice (i, j) contient la valeur « vrai » (ou `true` dans le langage de programmation) si $\{i_A,\ j_B\}$ est une arête du graphe ; elle contient la valeur « faux » (ou `false` dans le langage de programmation) dans le cas contraire. Le graphe G_0 ci-dessus est donc représenté par la matrice suivante :

i \ j	0	1	2	3
0	vrai	vrai	vrai	faux
1	faux	faux	faux	vrai
2	vrai	vrai	vrai	vrai
3	faux	faux	faux	vrai

Figure 3 : la matrice représentant G_0

Un couplage est représenté par un tableau d'entiers indicé de 0 à $n - 1$. Soit i vérifiant $0 \leq i \leq n - 1$; si le sommet i_A est couplé avec le sommet j_B, la case d'indice i contient la valeur j ; si le sommet i_A n'est pas couplé, la case d'indice i contient une valeur égale à –1. Le couplage C_0 de G_0, formé des arêtes $\{0_A, 0_B\}$ et $\{2_A, 3_B\}$, est représenté par le tableau ci-dessous :

i	0	1	2	3
	0	–1	3	–1

Figure 4 : le tableau représentant C_0

Indications pour la programmation

Un vecteur est par la suite nommé aussi tableau.

On nomme *matrice* un tableau à deux dimensions (un vecteur de vecteurs).

La matrice représentant un graphe biparti équilibré d'ordre $2n$ est codée en OCaml par une matrice $n \times n$ de booléens. La matrice représentant G_0 est ainsi codée par :

```
let G0 = [|[|true; true; true; false|];
           [|false; false; false; true|];
           [|true; true; true; true|];
           [|false; false; false; true|];|];;
```

Un couplage est codé par un vecteur de n entiers ; le couplage C_0 est ainsi codé par :

```
let C0 = [|0; -1; 3; -1|];;
```

Une arête a est codée par un vecteur `a` de deux entiers, avec `a.(0)` dans A et `a.(1)` dans B.

❒ 3 – Soit G un graphe biparti équilibré d'ordre $2n$. On considère un tableau C d'entiers de longueur n et contenant dans ses cases indicées de 0 à $n - 1$ soit la valeur -1, soit une valeur comprise entre 0 et $n - 1$. Il s'agit de savoir si ce tableau C représente ou non un couplage dans G.
Écrire en OCaml une fonction `verifie` telle que

- si `G` est une matrice codant le graphe G,
- si `C` est un vecteur codant le tableau C, alors `verifie G C` renvoie `true` si le tableau C représente un couplage dans G et `false` sinon. Indiquer la complexité de la fonction `verifie`.

❒ 4 – On considère un tableau C, de longueur n, codant un couplage d'un graphe G. Il s'agit d'écrire une fonction qui calcule le cardinal de ce couplage.
Écrire en OCaml une fonction `cardinal` telle que, si `C` est un vecteur codant un couplage, alors `cardinal C` renvoie le cardinal de ce couplage.
Indiquer la complexité de la fonction `cardinal`.

Deuxième partie : un algorithme pour déterminer un couplage maximal

On dit qu'un couplage C dans un graphe G est *maximal* si toute arête de G n'appartenant pas à C est incidente à au moins une arête de C. Par exemple, le couplage C_0 de G_0 est maximal. Un couplage maximal de G n'est pas forcément de cardinal maximum parmi les couplages de G. On cherche à concevoir un algorithme qui détermine un couplage maximal dans un graphe biparti équilibré G.

L'algorithme, nommé *algo_approche*, est le suivant :

- on commence avec un couplage vide C ;
- tant que G possède au moins une arête :
  - on choisit une arête a de G dont la somme des degrés des extrémités soit minimum ;
  - on ajoute l'arête a au couplage C ;
  - on retire de G l'arête a et toutes les arêtes incidentes à a.

On admettra que le résultat est, par construction, un couplage maximal.

❒ 5 – Appliquer *algo_approche* au graphe G_0 (voir la figure 1 page 4).

On considère par la suite le graphe biparti équilibré G_1 d'ordre 12 représenté sur la figure 5.

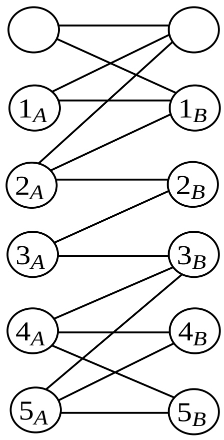

Figure 5 : le graphe G_1

❒ 6 – On applique *algo_approche* au graphe G_1. Déterminer la première arête a_1 choisie par *algo_approche* ; tracer le graphe obtenu après suppression de a_1 et des arêtes incidentes à a_1. Montrer que le couplage obtenu par *algo_approche* est de cardinal au plus 5 et indiquer s'il est de cardinal maximum parmi les couplages de G_1.

❒ 7 – Soit G un graphe biparti équilibré d'ordre $2n$. Il s'agit d'écrire en langage de programmation une fonction *arete_min* qui détermine une arête de G dont la somme des degrés des extrémités soit minimum. Si le graphe possède au moins une arête, cette fonction modifie un tableau a de deux entiers reçu en paramètre pour mettre dans les deux cases de a les numéros des deux extrémités d'une arête qui atteint ce minimum ; dans ce cas, la fonction renvoie la valeur « vrai » (`true`) ; sinon, elle renvoie la valeur « faux » (`false`).

Écrire en OCaml une fonction `arete_min` telle que :

- si `G` est une matrice codant le graphe G,
- si `a` est un vecteur de deux entiers,

alors `arete_min G a` effectue les opérations décrites ci-dessus, en modifiant le tableau `a` dans le cas où G possède au moins une arête.

Indiquer la complexité de la fonction `arete_min`.

❒ 8 – Il s'agit d'écrire en langage de programmation une fonction ou une procédure *supprimer* qui supprime d'un graphe biparti équilibré une arête *a* donnée et toutes les arêtes incidentes à *a*.
Écrire en OCaml une fonction `supprimer` telle que :
• si `G` est une matrice codant un graphe biparti équilibré *G*,
• si `a` est un vecteur de deux entiers codant une arête *a* de *G*,alors `supprimer G a` modifie `G` pour que, après modifications, `G` code le graphe obtenu à partir de *G* en supprimant *a* et toutes les arêtes incidentes à *a*.
Indiquer la complexité de la fonction `supprimer`.

❒ 9 – Il s'agit de définir en langage de programmation l'algorithme *algo_approche* décrit au début de la deuxième partie.
Écrire en OCaml une fonction `algo_approche` telle que, si `G` est une matrice qui code un graphe biparti équilibré *G*, `algo_approche G` effectue *algo_approche* à partir d'une copie de `G` et renvoie un vecteur codant le couplage obtenu.
Indication : on pourra utiliser sans la définir une fonction `dupliquer_matrice` telle que, si `G` est une matrice codant un graphe biparti équilibré *G*, `dupliquer_matrice G` renvoie une matrice identique à `G`. Cette fonction sera utilisée pour que la fonction `algo_approche` ne modifie pas le contenu de la matrice reçue en paramètre.
Indiquer la complexité de la fonction `algo_approche`.

Troisième partie : recherche exhaustive d'un couplage de cardinal maximum

❒ 10 – Soit G un graphe biparti équilibré d'ordre $2n$. Il s'agit d'écrire en langage de programmation une fonction nommée *une_arete* qui recherche une arête quelconque de G. Si G possède au moins une arête, la fonction mémorise **la première arête rencontrée** a dans un tableau (ou vecteur en Caml) de deux entiers reçu en paramètre ; elle arrête alors sa recherche et renvoie la valeur « vrai » (`true`) ; sinon, la fonction renvoie la valeur « faux » (`false`).
Écrire en OCaml une fonction `une_arete` telle que :

- si `G` est une matrice codant le graphe G,
- si `a` est un vecteur de deux entiers destiné à coder l'arête a,

alors `une_arete G a` effectue les opérations décrites ci-dessus, en modifiant le vecteur `a` dans le cas où G possède au moins une arête

❒ 11 – On cherche à établir un algorithme **récursif**, nommé *meilleur_couplage*, qui permette de déterminer un couplage de cardinal maximum dans un graphe biparti équilibré. Le principe est le suivant.
Si le graphe courant ne contient aucune arête, le cardinal maximum d'un couplage est 0 et aucun sommet n'est couplé.
Dans le cas contraire, l'algorithme considère une arête quelconque a du graphe courant et recherche successivement :

- un couplage de cardinal maximum parmi les couplages du graphe courant ne contenant pas a
- un couplage de cardinal maximum parmi les couplages du graphe courant contenant a.

L'algorithme déduit alors un couplage de cardinal maximum.
Écrire en OCaml une fonction récursive `meilleur_couplage` telle que, si `G` est une matrice codant un graphe biparti équilibré G, `meilleur_couplage G` renvoie un vecteur codant un couplage de cardinal maximum dans G. La fonction utilisera le principe décrit plus haut.
Indication : on pourra utiliser sans la définir une fonction `dupliquer_matrice` telle que, si `G` est une matrice codant un graphe biparti équilibré G, alors `dupliquer_matrice G` renvoie une matrice identique à `G`.