I. Graphes bipartis équilibrés

On appelle graphe biparti équilibré tout graphe non orienté $G = (S, A)$ tel qu'il existe une partition (S_1, S_2) de S (c'est-à-dire que $S_1 \cup S_2 = S$, $S_1 \neq \emptyset$, $S_2 \neq \emptyset$ et $S_1 \cap S_2 = \emptyset$) telle qu'il n'existe aucune arête entre sommets de S_1 et entre sommets de S_2 et telle que $\text{card}(S_1) = \text{card}(S_2)$.

Dans toute la suite on considèrera un graphe biparti équilibré $G = (S, A)$ et (S_1, S_2) une partition comme ci-dessus. On notera n le cardinal de S_1 et S_2, $s_{i,1}$, $0 \leq i \leq n-1$, les éléments de S_1 et $s_{i,2}$, $0 \leq i \leq n-1$, ceux de S_2. On représentera un graphe biparti par un tableau $(g_{i,j})_{0 \leq i,j \leq n-1}$ de booléens tels que $g_{i,j}$ est vrai si et seulement s'il existe une arête entre $s_{i,1}$ et $s_{j,2}$.

Un couplage de ce graphe est un ensemble d'arêtes tel que tout sommet du graphe est au plus sur une arête. Un couplage du graphe biparti $G = (S, A)$ peut être représenté par un tableau t d'entiers de taille n tel que, si $s_{i,1}$ est couplé à $s_{2,j}$, la valeur $t.(i)$ vaut j et, si $s_{i,1}$ n'est couplé à aucun sommet, $t.(i)$ vaut -1.

Q1. a. Déclarer une matrice représentant le graphe biparti tel que $\text{card}(S_1) = \text{card}(S_2) = 4$ et les arête de G sont $\{\{s_{0,1}, s_{0,2}\}, \{s_{0,1}, s_{1,2}\}, \{s_{0,1}, s_{2,2}\}, \{s_{1,1}, s_{3,2}\}, \{s_{2,1}, s_{0,2}\}, \{s_{2,1}, s_{1,2}\}, \{s_{2,1}, s_{2,2}\}, \{s_{2,1}, s_{3,2}\}, \{s_{3,1}, s_{3,2}\}\}$.
b. Déclarer un tableau représentant le couplage $\{\{s_{0,1}, s_{0,2}\}, \{s_{2,1}, s_{3,2}\}\}$.

Q2. On souhaite pouvoir vérifier si un tableau de taille n est bien un couplage d'un graphe biparti de cardinal $2n$, c'est-à-dire si les arêtes proposées par le tableau sont bien des arêtes du graphe biparti. Ecrire une fonction de type `verifie : int array array -> int array -> bool` prenant en argument une matrice $(g_{i,j})_{0 \leq i,j \leq n-1}$ représentant un graphe biparti, un tableau $(c_i)_{0 \leq i \leq n-1}$ représentant un couplage vérifiant si les arêtes du couplage sont bien dans le graphe.

Q3. Ecrire une fonction `cardinal : int array -> int` renvoyant le nombre d'arêtes d'un couplage.

II. Un premier algorithme

On rappelle que le degré d'un sommet dans un graphe est le nombre d'arêtes dont il est une extrémité.

Pour trouver un couplage maximum C (c'est-à-dire de plus grand cardinal) d'un graphe biparti équilibré G, on propose pour commencer l'algorithme suivant :

```
C:=Vide
tant que G possède au moins une arête faire
   {s,s'}:=arête dont la somme des degrés des extrémités est minimal
   ajouter à C l'arête {s,s'}
   retirer de G les arêtes adjacentes à s et s'
fin faire
renvoyer C
```

Q4. Ecrire une fonction `arete_min : int array array -> int*int` prenant en argument un graphe biparti équilibré renvoyant, si le graphe possède au moins une arête, une arête (i, j) dont la somme des degrés est minimal (i étant le sommet de S_1, j celui de S_2), le couple $(-1, -1)$ si le graphe n'a pas d'arête.

Q5. Ecrire une fonction `supprime : int array array -> int*int -> unit` qui prend en argument un graphe biparti équilibré, une arête qui modifie le graphe en supprimant toutes les arêtes dont un sommet est l'un des sommets de l'arête en argument.

Q6. Ecrire une fonction `couplage_maximum1 : int array array -> int array` prenant en argument un graphe biparti équilibré renvoyant un couplage donné par l'algorithme proposé.

Q7. Etudier la complexité de cet algorithme et prouver qu'il renvoie un couplage maximum.

III. Chemins augmentants et couplages maximums

On rappelle l'algorithme vu en cours de recherche de couplage maximum dans un graphe biparti :

```
C:=Vide
chemin:=chemin augmentant pour C et G
tant que chemin<>Vide faire
   C:= différence symétrique de C et chemin
   chemin:= chemin augmentant pour C et G
fin faire
renvoyer C
```

Pour trouver un chemin augmentant pour le couplage C, on va procéder de la façon suivante :
- on considère comme atteints tous les sommets non couplés de S_1 et on leur attribue comme antécédent -1
- si $s_2 \in S_2$ est voisin d'un sommet s_1 atteint et $\{s_1, s_2\}$ n'est pas dans C, s_2 est atteint et son antécédent est s_1
- si $s_1 \in S_1$ est voisin d'un sommet s_2 atteint et $\{s_1, s_2\}$ est dans C, s_1 est atteint et son antécédent est s_2.

Si un sommet non couplé $s_2 \in S_2$ est atteint, on aura un chemin augmentant en remontant les antécédents successifs depuis s_2.
S'il n'y a plus de sommets non atteints pouvant être atteint et si aucun sommet non couplé de S_2 n'est atteint, il n'y a pas de chemin augmentant.

Q8. Essayer l'algorithme sur le graphe donné dans la question 1.

On considèrera quatre tableaux :
- un tableau noté `c` représentant le couplage
- un tableau noté `r` (comme réciproque) de taille n dont l'élément d'indice j est -1 si $s_{j,2}$ n'est pas couplé et i si $s_{j,2}$ est couplé avec $s_{i,1}$
- un tableau `p1` de taille n contenant les prédécesseurs des sommets de S_1
- un tableau `p2` de taille n contenant les prédécesseurs des sommets de S_2.

Q9. Ecrire une fonction `couplage_reciproque : int array -> int array` prenant en argument un tableau représentant le couplage `c` renvoyant le tableau `r` tel que `r.(j)` vaut `i` si `c.(i)` vaut `j` et `-1` si `j` n'est pas couplé.

Q10. Ecrire une fonction `augmenter_couplage : int array -> int array -> int array -> int array -> int -> unit` prenant en arguments un couplage `c`, son couplage réciproque `r`, des tableaux de prédécesseurs `p1` et `p2` et `fin` l'indice de fin d'un chemin augmentant (donc d'un élément de S_2). Cette fonction modifiera les couplages `c` et `r` par le chemin augmentant issu de `fin`.

On rappelle que si un sommet `s` est atteint, ses voisins non encore atteints seront atteints et auront pour prédécesseur `s` selon la procédure décrite plus haut.

Q11. Ecrire des fonctions récursives croisées `cherche1` et `cherche2` prenant en argument une matrice `g` codant un graphe biparti et quatres tableaux `c`, `r`, `p1`, `p2`, un sommet `s` qui modifie les tableaux `p1`, `p2` et s'arrête lorsqu'un sommet non couplé de S_2 a été atteint ou lorsque tous les sommets qui peuvent être atteints l'ont été sans atteindre aucun sommet non couplé de S_2. Lorsqu'on découvre un sommet non couplé de S_2, on renverra l'indice de ce sommet, sinon on renverra -1.

Q12. Ecrire une fonction `chemin_augmentant` prenant en argument une matrice représentant un graphe biparti équilibré, quatre tableaux représentant un couplage, son couplage réciproque, deux tableaux de prédécesseurs de valeurs initiales -1. Cette fonction renverra -1 s'il n'existe pas de chemin augmentant, l'indice de fin d'un chemin augmentant s'il en existe un ; elle modifiera les tableaux de prédécesseurs afin de pouvoir récupérer un chemin augmentant s'il existe.

Q13. Ecrire une fonction `couplage_maximum2` prenant en argument une matrice représentant un graphe biparti équilibré renvoyant un couplage maximum.

Q14. Etudier la complexité de cet algorithme.