

Problème : Étude de réseaux sociaux

Objectif. Le but de cette partie est de regrouper des personnes par affinité dans un réseau social. Pour cela, on cherche à répartir les personnes en deux groupes de sorte à minimiser le nombre de liens d'amitié entre les deux groupes. Une telle partition s'appelle une coupe minimale du réseau.

Partie I : Réseaux sociaux

Structure de données. Nous supposons que les individus sont numérotés de 0 à $n - 1$ où n est le nombre total d'individus. Nous représenterons chaque lien d'amitié entre deux individus i et j par une liste contenant leurs deux numéros dans un ordre quelconque, c.-à-d. par la liste $[i, j]$ ou par la liste $[j, i]$ indifféremment.

Un réseau social R entre n individus sera représenté par une liste **reseau** à deux éléments où :

- **reseau[0]** = **n** contient le nombre d'individus appartenant au réseau
- **reseau[1]** = la liste non-ordonnée (et potentiellement vide) des liens d'amitié déclarés entre les individus

La figure 1 donne l'exemple d'un réseau social et d'une représentation possible sous la forme de liste. Chaque lien d'amitié entre deux personnes est représenté par un trait entre elles.

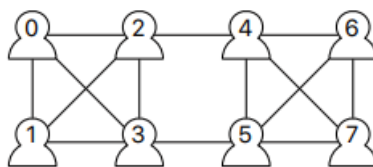
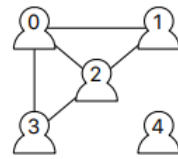


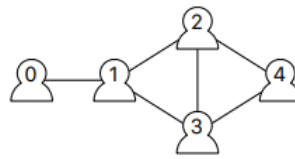
FIGURE 1 – Un réseau à 8 individus ayant 14 liens d'amitié déclarés et une de ses représentations possibles en mémoire sous forme d'une liste [Nombre d'individus, liste des liens d'amitié].

```
reseau = [ 8,
[ [0,1], [1,3], [3,2], [2,0], [0,3], [2,1], [4,5],
[5,7], [7,6], [6,4], [7,4], [6,5], [2,4], [5,3] ]
]
```

Question 1.1 Donner une représentation sous forme de listes pour chacun des deux réseaux sociaux ci-dessous :



Réseau A



Réseau B

Question 1.2 Écrire une fonction **creerReseauVide(n)** qui crée, initialise et renvoie la représentation sous forme de liste du réseau à n individus n'ayant aucun lien d'amitié déclaré entre eux.

Question 1.3 Écrire une fonction **estUnLienEntre(paire,i,j)** où *paire* est une liste à deux éléments et i et j sont deux entiers, et qui renvoie **True** si les deux éléments contenus dans *paire* sont i et j dans un ordre quelconque ; et renvoie **False** sinon.

Question 1.4 Écrire une fonction **sontAmis(reseau,i,j)** qui renvoie **True** s'il existe un lien d'amitié entre les individus i et j dans le réseau *reseau* ; et renvoie **False** sinon.

Question 1.5 Écrire une fonction **declareAmis(reseau,i,j)** qui modifie le réseau *reseau* pour y ajouter le lien d'amitié entre les individus i et j si ce lien n'y figure pas déjà.

Question 1.6 Écrire une fonction **listeDesAmisDe(reseau,i)** qui renvoie la liste des amis de i dans le réseau *reseau*.

Partie II : Partitions

Une partition en k groupes d'un ensemble A à n éléments consiste en k sous-ensembles **disjoints non-vides** $A_1 \cup \dots \cup A_k$ de A dont l'union est A , c-à-d. tels que $A_1 \cup \dots \cup A_k = A$ et pour tout $i \neq j$, $A_i \cap A_j = \emptyset$. Par exemple $A_1 = \{1, 3\}$, $A_2 = \{0, 4, 5\}$, $A_3 = \{2\}$ est une partition en trois groupes de $A = [[6]]$. Dans cette partie, nous implémentons une structure de données très efficace pour coder des partitions de $[[n]]$.

Le principe de cette structure de données est que les éléments de chaque groupe sont structurés par une relation filiale : chaque élément a un (unique) parent choisi dans le groupe et l'unique élément du groupe qui est son propre parent est appelé le représentant du groupe. On s'assure par construction que chaque élément i du groupe a bien pour ancêtre le représentant du groupe, c-à-d. que le représentant du groupe est bien le parent du parent du parent etc. (autant de fois que nécessaire) du parent de l'élément i . La relation filiale est symbolisée par une flèche allant de l'enfant au parent dans la figure 2 qui présente un exemple de cette structure de données. Dans l'exemple de cette figure, 14 a pour parent 11 qui a pour parent 1 qui a pour parent 9 qui est son propre parent.

Ainsi, 9 est le représentant du groupe auquel appartiennent 14, 11, 1 et 9. Notons que ce groupe contient également 8, 13 et 15.

A noter que la représentation n'est pas unique (si l'on choisit un autre représentant pour un groupe et une autre relation filiale, on aura une autre représentation du groupe).

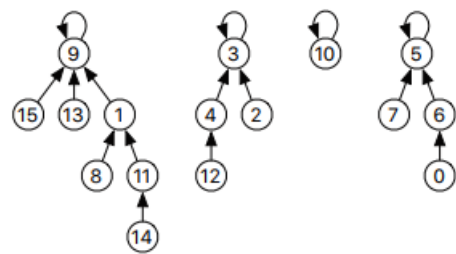
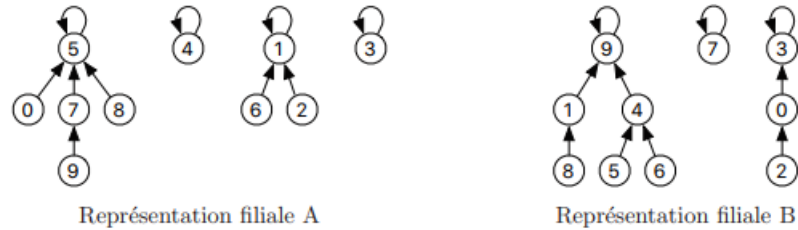


FIGURE 2 – Une représentation filiale de la partition suivante de $[[16]]$ en quatre groupes : $\{1, 8, 9, 11, 13, 14, 15\}$, $\{2, 3, 4, 12\}$, $\{10\}$ et $\{0, 5, 6, 7\}$ dont les représentants respectifs sont 9, 3, 10 et 5.

Pour coder cette structure, on utilise un tableau *parent* à n éléments où la case $parent[i]$ contient le numéro du parent de i . Par exemple, les valeurs du tableau *parent* encodant la représentation filiale donnée dans la figure 2 sont :

i :	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
parent[i] :	6	9	3	3	3	5	5	5	1	9	10	1	4	9	11	9

Question 1.7 Donner les valeurs du tableau *parent* encodant les représentations filiales des deux partitions de $[[10]]$ ci-dessous, et préciser les représentants de chaque groupe :



Initialement, chaque élément de $[[n]]$ est son propre représentant et la partition initiale consiste en n groupes contenant chacun un individu. Ainsi, initialement, $parent[i] = i$ pour tout $i \in [[n]]$.

Question 1.8 Écrire une fonction `creerPartitionEnSingletons(n)` qui crée et renvoie un tableau *parent* à n éléments dont les valeurs sont initialisées de sorte à encoder la partition de $[[n]]$ en n groupes d'un seul élément.

Nous sommes intéressés par deux opérations sur les partitions :

- Déterminer si deux éléments appartiennent au même groupe dans la partition.
- Fusionner deux groupes pour n'en faire plus qu'un. Par exemple, la fusion des groupes $A_1 = \{1, 3\}$ et $A_3 = \{2\}$ dans la partition de $[[6]]$ donnée en exemple au tout début de cette partie donnera la partition en deux groupes $A_2 = \{0, 4, 5\}$ et A_4 où $A_4 = A_1 \cup A_3 = \{1, 2, 3\}$.

Question 1.9 Écrire une fonction **representant(parent,i)** qui utilise le tableau *parent* pour trouver et renvoyer l'indice du représentant du groupe auquel appartient i dans la partition encodée par le tableau *parent*.

Pour réaliser la fusion de deux groupes désignés par l'un de leurs éléments i et j respectivement, on applique l'algorithme suivant :

1. Calculer les représentants p et q des deux groupes contenant i et j respectivement.
2. Faire $\text{parent}[p] = q$.

La figure 3 présente la structure filiale obtenue après la fusion des groupes contenant respectivement 6 et 14 de la figure 2.

Question 1.10 Écrire une fonction **fusion(parent,i,j)** qui modifie le tableau *parent* pour fusionner les deux groupes contenant i et j respectivement.

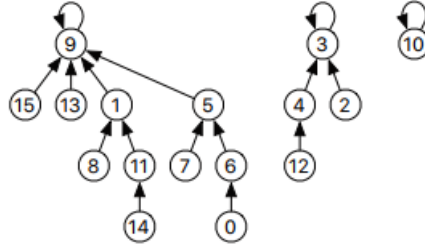


FIGURE 3 – Représentation filiale obtenue après la fusion des groupes contenant respectivement 6 et 14 de la figure 2.

Pour l'instant, la structure de données n'est pas très efficace comme le montre la question suivante.

Question 1.11 Proposer une suite de $(n - 1)$ fusions dont l'exécution à partir de la partition en n singletons de $[[n]]$, nécessite de l'ordre de n^2 opérations élémentaires.

Pour remédier à cette mauvaise performance, une astuce consiste à compresser la relation filiale après chaque appel à la fonction **representant(parent,i)**. L'opération de compression consiste à faire la chose suivante :

Si p est le résultat de l'appel à la fonction **representant**(parent,i), modifier le tableau parent de façon à ce que chaque ancêtre (c-à-d. parent de parent . . . de parent) de i , i y compris, ait pour parent direct p . Noter bien que même si un appel à **representant**(parent,i) renvoie le représentant de i elle modifie également le tableau parent.

Si l'on reprend l'exemple de la figure 2, le résultat de l'appel **representant**(parent,14) est 9, que l'on a calculé en remontant les ancêtres successifs de 14 :

11, 1 puis 9. L'opération de compression consiste alors à donner la valeur 9 aux cases d'indices 14, 11, et 1 du tableau parent. La structure filiale obtenue après l'opération de compression menée depuis 14 est illustrée dans la figure 4.

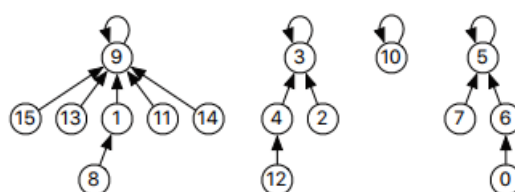


FIGURE 4 – Résultat de la compression depuis 14 dans la représentation filiale de la figure 2.

Question 1.12 Modifier votre fonction **representant**(parent,i) pour qu'elle modifie le tableau parent pour faire pointer directement tous les ancêtres de i vers le représentant de i une fois qu'il a été trouvé.

Afin d'afficher de manière lisible la partition codée par un tableau parent, on souhaite calculer à partir du tableau parent la liste des listes des éléments des différents groupes. Une sortie possible pour le tableau parent correspondant à la figure 2 serait :

```
[ [ 15, 8, 1, 9, 11, 13, 14 ],
  [ 4, 3, 2, 12 ],
  [ 7, 5, 6, 0 ],
  [ 10 ] ]
```

Question 1.13 Écrire une fonction **listeDesGroupes**(parent) qui renvoie la liste des différents groupes codés par le tableau parent sous la forme d'une liste des listes des éléments des différents groupes.

Partie III. Algorithme randomisé pour la coupe minimum

Revenons à présent à notre objectif principal : Trouver une partition des individus d'un réseau social en deux groupes qui minimise le nombre de liens d'amitiés entre les deux groupes. Pour résoudre ce problème nous allons utiliser l'algorithme randomisé suivant :
Entrée : un réseau social à n individus

1. Créer une partition P en n singletons de $[[n]]$
2. Initialement aucun lien d'amitié n'est marqué
3. Tant que la partition P contient au moins trois groupes et qu'il reste des liens d'amitié non-marquées dans le réseau faire :
 - (a) Choisir un lien uniformément au hasard parmi les liens non-marqués du réseau, notons-le $[i, j]$.
 - (b) Si i et j n'appartiennent pas au même groupe dans la partition P , fusionner les deux groupes correspondants.
 - (c) Marquer le lien $[i, j]$.
4. Si P contient $k \geq 3$ groupes, faire $k - 1$ fusions pour obtenir deux groupes.
5. Renvoyer la partition P .

La figure 5 présente une exécution possible de cet algorithme randomisé sur le réseau de la figure 1.

Question 1.14 Écrire une fonction `coupeMinimumRandomisee(reseau)` qui renvoie le tableau parent correspondant à la partition calculée par l'algorithme ci-dessus.

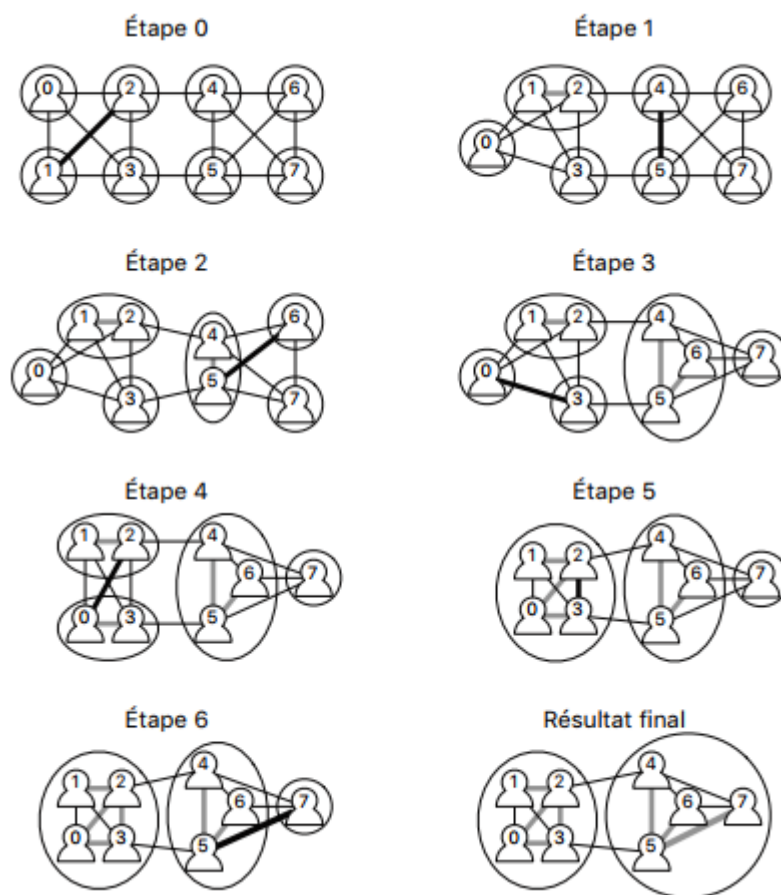


FIGURE 5 – Une exécution de l’algorithme randomisé sur le réseau de la figure 1 où les liens sélectionnés aléatoirement sont dans l’ordre : $[2,1]$, $[4,5]$, $[6,5]$, $[0,3]$, $[2,0]$, $[3,2]$ et $[5,7]$. Les liens représentés en noir épais sont les liens sélectionnés au hasard à l’étape courante ; les liens épais et grisés sont les liens marqués par l’algorithme ; les ronds représentent la partition à l’étape courante.

Question 1.15 Écrire une fonction **tailleCoupe(reseau, parent)** qui calcule le nombre de liens entre les différents groupes de la partition représentée par **parent** dans le réseau *reseau*.

_____ FIN _____

Bonne chance