# TME - semaines 1 à 3

Ce mini-projet est à réaliser pendant les séances de TME 1 à 3, et sera à rendre sur moodle quelques jours avant les soutenances (les soutenances auront lieu pendant les séances de TD4 et TP4). Sur moodle, vous trouverez un document récapitulant ce qui est attendu dans votre rendu (code + rapport). Pour les étudiants qui ne peuvent être présents le jour de la soutenance, vous devez impérativement contacter votre chargé de TD afin de lui fournir un justificatif et prévoir avec lui une autre date de soutenance (au plus près de la semaine de rendu). En effet, les soutenances sont obligatoires, et toute absence injustifiée sera sanctionnée par la note de zéro.

**Attention :** Ce projet doit obligatoirement être réalisé en binôme. Si vous ne trouvez pas de binôme, vous devez envoyer un mail à votre chargé de TD pour qu'il vous en trouve un rapidement.

## 1 Problème et affectation

On s'intéresse au problème d'affection des étudiants dans le master informatique de Sorbonne Université. Ce master comporte 9 parcours : ANDROIDE, BIM, DAC, IMA, RES, SAR, SESI, SFPN, STL. Vous pouvez aller sur le site du master, à l'adresse `www-master.ufr-info-p6.jussieu.fr/`, pour avoir une description de ces différents parcours. Le recrutement dans chacun de ces parcours se fait selon les dossiers, et selon les vœux exprimés par les étudiants. L'objectif de cet exercice est d'utiliser l'algorithme de Gale-Shapley pour déterminer efficacement des affectations stables dans ce problème. Commencer par récupérer les fichiers "PrefSpe.txt" et "PrefEtu.txt" sur le Moodle de l'UE.

**Fichier PrefSpe.txt :** il correspond aux préférences des 9 parcours sur un groupe fictif de 11 étudiants (les parcours sont numérotés de 0 à 8, et les étudiants de 0 à 11). La première ligne contient le nombre d'étudiants. La deuxième contient les capacités d'accueil des parcours (2 pour ANDROIDE, 1 pour BIM, etc). Les 9 lignes suivantes correspondent aux préférences des parcours sur les étudiants. Par exemple, la ligne "0 ANDROIDE 7 9 [...] 2" signifie que l'étudiant 7 est classé premier par le parcours numéroté 0 (ANDROIDE), que l'étudiant 9 est classé deuxième par ce même parcours, etc, et que l'étudiant 2 est le dernier.

**Fichier PrefEtu.txt :** il correspond aux préférences des étudiants. La première ligne contient le nombre d'étudiants (ici 11). Puis, le fichier contient une ligne par étudiant donnant ses préférences sur les parcours. Par exemple, la ligne "0 Etu0 5 [...] 4" signifie que l'étudiant numéro 0, nommé Etu0, a classé le parcours SAR (numéro 5) premier, etc, et que le parcours RES (numéro 4) est le dernier de son classement.

Les fonctions suivantes sont à coder en langage Python. Deux fichiers contenant quelques instructions utiles en Python vous sont fournis ("main.py" et "exemple.py"). Si vous n'avez jamais utilisé Python, lisez la fiche d'aide sur le langage python.

**Q1.** Écrivez une fonction lisant le fichier des préférences des étudiants sur les masters (PrefEtu.txt) et qui retourne une matrice $C_E$ qui en ligne $i$ contient le classement des parcours selon les préférences de l'étudiant $i$. De même, écrivez une fonction qui retourne une matrice $C_P$ correspondant aux préférences des parcours.

Ces matrices de préférences seront données en entrée de l'algorithme de Gale-Shapley pour trouver des affectations stables des étudiants dans les parcours.

**Q2.** On souhaite tout d'abord coder l'extension vue en cours de l'algorithme de Gale-Shapley au problème des hôpitaux "côté étudiants". Pour que votre implémentation soit efficace, il faut que les opérations suivantes soient les plus rapides possibles :

1. Trouver un étudiant libre à chaque itération.
2. Étant donné un étudiant libre, trouver le prochain parcours à qui faire une proposition.

3. Étant donné un étudiant $i$ et un parcours $j$, trouver la position de l'étudiant $i$ dans le classement du parcours $j$.
4. Étant donné un parcours, trouver l'étudiant le moins préféré par le parcours parmi ceux qui lui sont affectés.
5. Remplacer un étudiant par un autre dans l'affectation courante d'un parcours.

Discuter quelles sont les structures de données à utiliser pour que ces opérations soient les plus rapides possibles et les moins gourmandes en espace mémoire. Justifiez vos choix de structures en donnant la complexité de ces opérations.

**Q3.** Codez l'algorithme de Gale-Shapley "côté étudiants" avec les structures de données que vous aurez choisies. Quelle est la complexité de votre algorithme ?

**Q4.** Proposez une adaptation de Gale-Shapley au problème des hôpitaux pour avoir une version "côté parcours". Discutez ensuite quelles sont les structures de données à utiliser pour optimiser ses opérations, donnez leur complexité et codez votre algorithme.

**Q5.** Appliquez les algorithmes "côté étudiants" et "côté parcours" sur les deux fichiers tests. Quelle(s) affectation(s) obtenez-vous ?

**Q6.** Écrivez une méthode prenant en entrée une affectation (et les matrices de préférences), et renvoyant la liste des paires instables. Vérifiez ensuite la stabilité de l'affectation (ou des affectations) obtenue(s) dans les questions précédentes.

## 2 Évolution du temps de calcul

**Q7.** Écrivez deux méthodes prenant en paramètre un nombre $n$ d'étudiants :
- l'une générant un matrice $C_E$ des préférences de ces $n$ étudiants sur les 9 parcours du master (préférences aléatoires),
- l'autre générant une matrice $C_P$ des préférences des 9 parcours du master sur les $n$ étudiants (préférences aléatoires).

**Q8.** Mesurez le temps de calcul de vos algorithmes de Gale-Shapley pour différentes valeurs de $n$ en suivant les instructions suivantes :
- Faire varier $n$ de 200 à 2000 par pas de 200.
- Faire plusieurs tests (au moins 10) pour chaque valeur de $n$ pour avoir une valeur significative (la moyenne).
- Pour chaque test, définir les capacités d'accueil des parcours de sorte que la somme soit égale à $n$. On pourra générer des capacités de manière déterministe (et par exemple à peu près équilibrées entre les parcours).
- Tracer une courbe représentant le temps de calcul (moyen) en fonction de $n$.

**Q9.** Quelle complexité observez-vous pour vos algorithmes ? Est-ce cohérent avec leur complexité théorique ?

**Q10.** Faites de même pour le nombre d'itérations de vos algorithmes. Est-ce cohérent avec une analyse théorique ?

## 3 Equité et PL(NE)

On souhaite maintenant assurer une certaine équité entre les étudiants. Dans un premier temps, on souhaite trouver une solution (pas forcément stable) qui maximise l'utilité minimale des étudiants (on définit leurs utilités par les scores de Borda). Ainsi, on veut trouver le plus petit $k$ tel qu'il existe une affectation où tout étudiant a un de ses $k$ premiers choix (c-à-d une utilité d'au moins $m - k$ pour chaque étudiant).

**Q11.** Sur l'exemple, écrivez (dans votre rapport) un PLNE permettant de savoir s'il existe une affectation où tout étudiant a un de ses $k$ premiers choix (pour un $k$ fixé).

**Q12.** Pour $k = 3$, écrivez une méthode permettant de générer un fichier .lp correspondant au PLNE, puis résolvez le PLNE à l'aide de Gurobi (voir la fiche d'aide sur la programmation linéaire). Existe-t-il une solution ?

**Q13.** Trouvez le plus petit $k$ tel qu'il existe une solution. Donnez ensuite une solution maximisant l'utilité minimale des étudiants.

On souhaite à présent garantir une certaine efficacité totale en plus de l'équité entre les étudiants.

**Q14.** Sur l'exemple, écrivez (dans votre rapport) un PLNE permettant de trouver une solution qui maximise la somme des utilités (étudiants et parcours), puis résolvez-le avec Gurobi. Quelle utilité moyenne obtient-on ? Quelle utilité minimale des étudiants obtient-on ?

**Q15.** Soit $k^*$ le plus petit $k$ trouvé à la question 13. Sur l'exemple, écrivez (dans votre rapport) un PLNE maximisant la somme des utilités (étudiants et parcours) parmi les solutions où chaque étudiant a un de ses $k^*$ premiers choix. Résolvez le PLNE à l'aide de Gurobi.

**Q16.** Comparez les différentes solutions obtenues (GS côté étudiants, GS côté parcours, solutions des questions 13, 14 et 15) : stabilité, utilité moyenne, utilité minimale des étudiants. Utilisez la question 6 pour les comparer aussi selon le nombre de paires instables.