TP D'INFORMATIQUE N°1
Modélisation d'interactions coopératives itérées

1 Interaction coopératives itérées

On considère le jeu à deux joueurs suivant : lors d'une *manche*, les joueurs choisissent simultanément de *coopérer* ou de *trahir*. À l'issue de la manche :

- si les deux joueurs coopèrent, ils marquent chacun C points,
- si les deux trahissent, ils marquent chacun P points.
- si l'un coopère et l'autre trahit,
  - celui qui coopère (et s'est fait duper) marque D point,
  - celui qui a trahi marque T points,

C, P, D et T sont des entiers relatifs fixés au début du jeu, vérifiant $T > C > P > D$ et $2C > T+D$.

Lors d'une *partie*, les joueurs jouent N manches, chaque choix étant fait en fonction de ce qui s'est passé aux manches précédentes. Le gain d'un joueur est alors la somme des gains de ses manches, et son objectif est de maximiser ce gain, indépendamment du gain de l'autre joueur.

Un tel jeu peut par exemple modéliser le comportement d'individus collaborant pour produire des ressources. Il permet d'étudier la notion de confiance.

En binôme : faire une partie de 10 coups, avec les paramètres $T = 5, C = 3, P = 1, D = 0$.

2 Implémentation informatique

On représente une partie par deux tableaux de taille N, chaque tableau stockant les coups d'un joueur. À la manche d'indice i, on écrira dans chacune des deux cases d'indice i le choix du joueur correspondant, en représentant une trahison par l'entier 0 et une coopération par l'entier 1.

Une *stratégie* est une fonction prenant en argument l'indice i de la manche jouée et les deux tableaux de coups (supposés rempli de l'indice 0 à $i-1$), notés `moi` et `autre`, et qui renvoie 0 ou 1, selon le choix de trahir ou de coopérer.

Par exemple, la stratégie

```
def bisounours(i,moi,autre):
    return 1
```

coopère toujours. Notons les points suivants :

- une stratégie prend toujours en argument les 3 paramètres décrits, même si elle n'en tient pas compte ;
- la valeur renvoyée par une stratégie est simplement 0 ou 1, selon le coup qu'elle détermine sur cette manche précise. Le tableau des coups de la stratégie en sera déduit plus tard.

1. Implémenter en Python les stratégies suivantes :
   - **traître** : trahi toujours ;
   - **sympa irl** : alterne deux coopérations successives et une trahison.
2. Écrire une fonction **partie** qui prend en argument le nombre de manches N et les deux stratégies en interaction, et qui renvoie les deux tableaux de coups. On testera les stratégies précédentes ainsi que les suivantes avec cette fonction.

   Par exemple, `partie(5,traitre,sympa_irl)` doit renvoyer `[0,0,0,0,0],[1,1,0,1,1]`.

3. Écrire une fonction **score** qui prend en argument les paramètres T, C, P, D, ainsi que deux tableaux de coups supposés représenter une partie, et qui renvoie le score de chaque joueur. Par exemple, `score(5,3,1,0,[0,0,0,0,0],[1,1,0,1,1])` doit renvoyer `21,1`.
4. Implémenter les stratégies suivantes :
   - **rancunier** : coopère tant qu'il n'est pas trahi, et répond à une trahison en trahissant jusqu'à la fin ;
   - **miroir gentil** : commence par coopérer, puis joue le coup précédent de l'autre joueur ;
   - **miroir méchant** : commence par trahir, puis joue le coup précédent de l'autre joueur ;
   - **majorité** : trahi si et seulement si l'autre joueur a joué strictement plus de trahisons que de coopérations ;
   - **mastermind** : commence par une trahison et deux coopérations, puis :
     — si l'autre joueur a coopéré les trois premières fois, trahi jusqu'à la fin,
     — sinon, joue le coup précédent de l'autre joueur.
5. Écrire une fonction **indécis**, qui prend en argument un nombre flottant p, et qui renvoie la stratégie jouant à chaque coup une coopération avec probabilité p (on pourra importer le module `random` avec la commande `from random import *` et utiliser la fonction `random`). Attention, **indécis** est donc une fonction renvoyant une fonction.
6. Implémenter la stratégie **hacker**, qui remplace par effet de bord les coups précédents de l'autre joueur par des coopérations, et qui trahi systématiquement. La fonction `partie` est-elle vulnérable à cette forme de triche ? Si oui, procéder en urgence à une mise à jour de sécurité.
7. Écrire une fonction **tournoi** qui prend en argument un tuple de stratégies et les paramètres T, C, P, D, et :
   - fixe un nombre de manches tiré aléatoirement entre 500 et 1000 (on pourra utiliser la fonction `randint`),
   - fait se rencontrer chaque paire de stratégies
   - renvoie le tableau des scores cumulés de chaque stratégie.

3 Tournoi des 3/2

Implémenter une stratégie originale qui participera à un tournoi rassemblant bisounours, traître, sympa irl, rancunier, miroir gentil, miroir méchant, majorité, mastermind, indécis(0.4) et les stratégies des autres 3/2 de MP*/PC*, avec les paramètres $T = 5, C = 3, P = 1, D = 0$.

Cette stratégie est à déposer sur cahier de prepa avant le samedi 24 septembre à 12h. Elle sera notée selon la formule $10(x - 1,2)$ sur 20, où x est le score moyen par manche de la stratégie. J'attribuerai à ma discrétion des points malus aux stratégies ne respectant pas la spécification.

4 Tournoi des 5/2

Lors d'une manche, chaque joueur choisit un entier naturel ($< 10^9$). 0 correspond à une trahison, et un entier strictement positif correspond à une coopération avec le degré de confiance associé :

- Si les deux joueurs trahissent, ils ne marquent aucun point.
- Si les deux joueurs coopèrent, on considère le minimum n des deux degrés de confiance, et chacun marque $\frac{3(n-1)}{n}$ points.
- Sinon, le joueur qui a trahi gagne n points, et le joueur trahi perd n points, où n est le degré de confiance du joueur trahi.

Les stratégies Python prendront 4 arguments : les 3 premiers (`i, moi, autre`) sont similaires au jeu précédent, et le quatrième argument est un tableau `memoire`, qui sera vide avant la première manche, et que la stratégie pourra au besoin modifier par effet de bord pour enregistrer des informations d'une manche à l'autre. La note sera donnée par la formule $5(x + 1)$, où x est le score moyen par manche.