# U.E. S2IN426 - CCTP

**Ce DM est à faire seul ou à deux et à rendre exclusivement par dépôt sur Moodle avant le lundi 1er mai 24h00.**

**L'organisation du code, sa clarté, sa facilité de lecture et les commentaires (en une phrase ou deux ce que fait la fonction/méthode, à quoi correspondent précisément les paramètres en entrée, quel est le résultat retourné, ...) seront pris en compte dans l'évaluation du DM (pour une part non négligeable).**

**Vous avez le choix du langage de programmation : C ou Python.**

**Vous devez rendre deux versions : la première correspondant à la section 2 du sujet. La deuxième à la section 3.**

**N'oubliez pas d'indiquer vos numéros, noms et prénoms en commentaire au début du code.**

---

L'objectif de ce TP est de coder un jeu de Kalaha en mode console et d'utiliser une résolution MinMax pour les coups joués par la machine.

## 1 Le Kalaha

Le jeu de Kalaha est un jeu de stratégie d'origine Africaine de la famille de l'Awele dont c'est une variante. C'est un jeu à deux joueurs qui utilise un plateau constitué de deux rangées de six cases (les champs) et d'un grenier pour chacun des joueurs. Au début du jeu 4 graines sont déposées dans chacun des douze champs.

**Règles :** Le but du jeu est de s'emparer du maximum de graines.

Chacun son tour un joueur prend toutes les graines qui sont dans l'un de ses six champs et en dépose une dans chacune des cases suivantes du plateau en le parcourant dans le sens anti-horaire. Il dépose exactement une graine dans chacune des cases : que ce soit un de ses champs, son grenier ou bien les champs de l'adversaire. Mais s'il arrive jusqu'au grenier de l'adversaire il saute celui-ci sans y déposer de graine et continu à déposer les graines dans les champs suivants.

Si la dernière graine qu'il dépose est située dans un champ qui lui appartient et qui est vide, il dépose celle-ci dans son grenier et s'empare de toutes les graines qui sont dans le champ adverse en face du sien (et il dépose aussi dans son grenier).

Le jeu se termine lorsque les six champs d'un des joueurs sont vides, il ne peut alors plus jouer. L'autre joueur ramasse alors toutes les graines de ses champs et les dépose dans son grenier.

Le gagnant est celui qui a le maximum de graines dans son grenier.

## 2 Mise en place du jeu en mode humain vs humain

Dans cette première partie il s'agit de mettre en place toutes les composantes permettant de jouer au Kalaha en mode console entre deux joueurs humains.

Vous devez entre autre :

— Définir les types nécessaires pour représenter le plateau du jeu ainsi que les deux joueurs.
— Coder la phase d'initialisation du jeu : configuration initiale du plateau, joueur qui commence, etc...
— Afficher l'état du plateau à l'écran (avec les numéros des champs à joueur pour faciliter le jeu)
Un exemple d'affichage possible :

```
                Joueur A
        6    5    4    3    2    1
-----------------------------------------
| GA |  8 |  0 |  1 |  3 |  1 |  7 |    |
|  4 |----------------------------|  3 |
|    |  6 |  6 |  5 |  1 |  1 |  2 | GB |
-----------------------------------------
        1    2    3    4    5    6
                Joueur B
```

— Tester si un coup est possible : le champ choisi ne doit pas être vide
— Jouer un coup selon les règles : c'est à dire, en fonction du joueur dont c'est le tour et du champ choisi, répartir les graines dans les différentes cases (champs et éventuellement grenier, mais pas dans le grenier de l'adversaire) du plateau, et si le cas se présente, s'emparer des graines de l'adversaire.
— Coder la boucle de jeu : chaque joueur humain joue à son tour un coup. La machine n'a qu'un rôle d'arbitre : initialiser le jeu, donner la main à tour de rôle à chaque joueur, vérifier que les coups choisis sont valides, afficher le plateau après chaque coup, annoncer la fin de la partie quand plus aucun coup n'est possible et désigner le gagnant.

## 3 Implantation de la stratégie de l'ordinateur

Cette deuxième partie a pour but d'implanter un algorithme MinMax afin de pouvoir jouer en mode humain vs ordinateur et ordinateur vs ordinateur.

Vous devez compléter/modifier le programme de la partie 1 (en particulier la boucle de jeu) afin de pouvoir faire jouer l'ordinateur avec l'algorithme MinMax à profondeur bornée à la place d'un ou des deux joueurs.

Vous compléterez votre programme pour choisir choisir au début de la partie : le type des joueurs (ordinateur ou humain) ainsi que la profondeur d'exploration du Minmax dans le cas où au moins un des joueurs est l'ordinateur (inutile d'aller au delà de la profondeur 6 ou 7 : à chaque tour il peut y avoir jusqu'à 6 coups possibles, à la profondeur 7 l'arbre d'exploration peut donc contenir jusqu'à $6^7 = 279936$ feuilles ...)

### Calcul des scores - Evaluation des états du jeu

Une fois la partie terminée, le joueur gagnant est celui dont le nombre total de graines (grenier + champs) est le plus élevé. Cependant au cours de jeu, les graines déposées dans le grenier d'un joueur ne peuvent en être retirées, contrairement à celles déposées dans les champs. Un état du jeu est donc d'autant plus favorable à un joueur qu'il possède de graines dans son greniers, mais il faut aussi prendre en compte les graines des champs qui y seront ajoutées si l'état du jeu est final.

On propose donc, pour calculer le score du joueur $j$ dans un état $e$ :

$$sc(e,j) = 3 \times G(e,j) + \sum_{1..6} C_i(e,j)$$

grenier — une case

où $G(e,j)$ est le nombre de graines présentes dans le grenier du joueur $j$ et les $C_i(e,j)$ sont les nombres de graines présentes dans les $i$ champs du joueur $j$.

Par exemple, dans l'état du jeu donné plus haut, le score du joueur A est $3 \times 4 + 7 + 1 + 3 + 1 + 0 + 8 = 32$ et le score du joueur B est $3 \times 3 + 6 + 6 + 5 + 1 + 1 + 2 = 30$

On utilisera la fonction de score pour évaluer tous les états du jeu, qu'ils soient finaux ou intermédiaires, et c'est aussi cette fonction qu'on utilisera dans l'heuristique d'évaluation des états pour l'algorithme minmax à profondeur bornée. Ainsi la valeur d'un état, final ou intermédiaire, sera :

$$v(e) = sc(e,A)/sc(e,B)$$

### Mise en œuvre du minmax

L'algorithme minmax à profondeur bornée que vous devez implanter est résumé par les deux fonctions suivantes :

```
# Choix du coup à jouer par la machine
# e : état du jeu
# j : A ou B selon que la machine est le joueur A ou le joueur B
# Le joueur A choisit le coup qui maximise la valeur de l'état dans lequel il arrive
# le joueur B choisit celui qui minimise cette valeur

Fonction choixDuCoup(e, j)
Début
   S(e,j) := les états obtenus lorsque le jouer j joue un coup (valide) à partir de l'état e
   si j = A alors
      choix := le coup qui mène à un état e' parmi S(e,j) tel que minmax(e', A) est maximum
   sinon
      choix := le coup qui mène à un état e' parmi S(e,j) tel que minmax(e', B) est minimum
  fin si
  retourner choix
Fin
```

```
# Heuristique de calcul de la valeur d'un état du jeu (transparents 204-205 du cours)
# e est l'état du jeu
# pmax est la profondeur maximum d'exploration de l'arbre
# j est le joueur dont c'est le tour de jouer
# Résultat : la valeur de l'état e

Fonction minmax(e, pmax, j)
Début
   si e est final ou si pmax = 0 alors  # état final ou profondeur max atteinte
        retourner v(e)
   sinon
        si j = A alors
            # on remonte valmax, le maximum des w(e') parmi les fils de e,
            # c'est à dire les états du jeu dans lesquels on arrive après que **le joueur B**
            # a joué un coup
            valmax := 0
            pour e' dans S(e,B) faire
                valmax := max(valmax, minmax(e', pmax-1, B))
            fin pour
            retourner valmax
          mettre des condition pour qu'il ignore les 0
        sinon
            # on calcule valmin, le minimum des v(e') parmi les fils de
            # c'est à dire les états du jeu dans lesquels on arrive après que **le joueur A**
            # a joué un coup
            valmin := infini je peut le remplacer par 999
            pour e' dans S(e,A) faire
                valmin := min(valmin, minmax(e', pmax -1, A))
            retourner valmin
        fin si
    fin si
Fin
```