

Compte Rendu

Sae S1.02

I - Explication des programmes.....	2
1) Organisation des fichiers.....	2
2) Explication main.....	2
3) Bot débutant.....	3
4) Bot expert.....	3
5) Changement dans les scripts des jeux.....	4
II - Démonstration et tests.....	5
1) Test main.....	5
0 joueur réel:.....	5
1 joueur réel:.....	6
2 joueurs réels:.....	7
2) Test devinettes.....	7
3) Test allumettes.....	8
4) Test morpion.....	9
5) Test score.....	10
III - Comparaison des algorithmes.....	10
1) Pour le jeu des devinettes:.....	10
2) Pour le jeu des allumettes:.....	11
3) Pour le jeu du morpion:.....	11

I - Explication des programmes

1) Organisation des fichiers

Il y a 2 difficultés de bots:

Une difficulté débutant, correspondant à un bot qui fait tout au hasard.

une difficulté expert, correspondant à un bot qui prendra (en général) la stratégie la plus optimisée.

Nous avons donc créé 2 fichiers contenant les programmes pour chacun des bots.

Ils se nomment donc bot_debutant et bot_expert. Leurs fonctions seront donc appelées par les différents fichiers de jeu si c'est au tour des bots.

2) Explication main

Au début du programme, nous demandons le nombre de joueurs humains. Ainsi, si il y a 2 humains qui jouent, alors le programme fonctionnera exactement comme avant les modifications pour cette SAE et ne prendra pas en compte les bots. Cependant, s'il y a au moins un bot, nous demanderons à l'utilisateur le niveau de chaque bots. Ainsi, selon le niveau du bot, on crée un profil pré-existant afin de le mettre dans la structure de donnée qui répertorie les joueurs current. Voici les 4 profils pré-existant:

bot1D - bot de niveau débutant à la position de joueur 1

bot1E - bot de niveau expert à la position de joueur 1

bot2D - bot de niveau débutant à la position de joueur 2

bot2E - bot de niveau expert à la position de joueur 2

S'il y a moins de 2 joueurs humains, il y aura forcément un bot à la place du joueur 2 (s'il n'y a qu'un joueur humain, il sera forcément à la position de joueur 1). Mais s'il n'y a aucun joueur humain, il faut aussi un bot à la position 1. Nous avons 4 profils différents afin de différencier les scores lorsque l'on fait jouer 2 bots de même niveau entre eux.

En conséquence, les noms des bots sont bloqués pour les joueurs.

3) Bot débutant

Dans le 1er fichier que nous avons créé, `bot_débutant.py`, il y a donc les programmes qui permettent le bon fonctionnement du bot débutant qui jouera aléatoirement.

Tout d'abord, on importe le module `random` qui va alors nous servir pour plusieurs fonctions pour renvoyer des nombres aléatoires.

Pour le jeu des allumettes, nous avons une fonction **`allumette_bot`** qui permet de renvoyer un entier entre 1 et 3. Cette fonction permettra donc au bot de choisir combien d'allumettes enlever pendant la partie.

Pour le jeu des devinettes, nous avons 2 fonctions différentes.

`propose_bot` qui permet de retourner un nombre entre 1 et 100 en chaîne de caractère. Cette fonction permet donc au bot de débiter la partie de devinettes

`devine_bot` qui permet de retourner un nombre entre les bornes min et max qui correspondent à celle du jeu des devinettes en partie. Cette fonction permet donc au bot de proposer des nombres pour trouver le nombre du poseur.

Pour le jeu du morpion, nous avons une fonction **`morpion_bot`** qui permet de retourner un nombre entre 1 et 9, qui est disponible dans la partie du morpion, en chaîne de caractère.

4) Bot expert

Dans le 2ème fichier que l'on a créé, `bot_expert.py`, il y a donc les programmes qui permettent le bon fonctionnement du bot expert qui jouera intelligemment.

Pour le jeu des allumettes, on a une fonction **`allumette_bot`** qui retourne un entier en fonction du nombre d'allumettes qu'il reste modulo 4.

Si le reste est 0, alors il retourne 3.

Si le reste est 1, alors il retourne 2.

Si le reste est 2, alors il retourne 1.

Si le reste est 3, alors il retourne 2.

Pour le jeu des devinettes, nous avons 2 fonctions différentes:

`propose_bot` qui permet de retourner un entier aléatoirement, grâce au module `random`, entre 1 et 100. Cette fonction permet donc au bot de faire deviner un nombre si le bot doit faire deviner.

devine_bot qui va retourner un nombre entier de façon dichotomique. Il va renvoyer le nombre qu'il obtiendra en faisant la division euclidienne par 2 du nombre d'allumettes qu'il reste dans la partie.

Pour le jeu du morpion, nous avons 5 fonctions:

morpion_bot qui est la fonction principale qui va d'abord vérifier si le bot joue avec les O ou avec les X.

Il va ensuite vérifier si le plateau de morpion est vide, grâce à la fonction **board_vide**. Si le plateau est vide, la fonction retourne un entier qui sera soit 1, 3, 7 ou 9, ce qui va permettre de choisir un coin du plateau.

Si le plateau n'est pas vide, il va vérifier s'il y a une possibilité de gagner grâce à la fonction **having_win_pos**, avec en paramètre le plateau de jeu et le signe du bot, et va donc retourner un entier qui correspond à la case disponible pour gagner.

Si le plateau n'est pas vide mais que le bot n'a pas de position pour gagner, il va alors vérifier si l'autre joueur a une case disponible pour gagner grâce à la fonction **having_win_pos**, avec en paramètre le plateau de jeu et le signe de l'autre joueur. et va donc retourner un entier qui correspond à cette case.

Si aucune de ses solutions n'est possible alors il joue aléatoirement.

La fonction **having_win_pos** regarde si il y a 2 signes qui sont alignés horizontalement, verticalement ou de manière oblique sur le plateau de jeu. La fonction retourne alors la case correspondante.

La fonction **play_random** permet de retourner un nombre entre 1 et 9, qui est disponible dans le partie du morpion, en chaine de caractère.

La fonction **get_voide** renvoie la case vide si dans une ligne, il y a 2 signes et 1 case vide.

La fonction **board_vide** retourne True ou False si le tableau de jeu est vide ou non. Il vérifie sur chaque ligne de chaque colonne si une case est remplie par le signe X ou O.

5) Changement dans les scripts des jeux

Chaque jeu, lorsqu'ils demandent une action au joueurs, doit maintenant prendre en compte si ce joueur est un bot ou non. Ainsi il regardent le nom du joueur et s'il correspond à l'un des noms des bots, alors il va appeler la bonne fonction en fonction du niveau du bot, qu'il trouve dans son nom (c'est la dernière lettre de leurs nom). Les fonctions réponse des bot ont les mêmes noms d'un fichier à l'autre, pour les deux bots, seul le fichier source change. Cela permet une cohérence dans le nom des fonctions.

Au début de chaque fichier jeu, on importe donc les fonctions des fichiers bots de cette manière:

```
import programme.bot_debutant as bot_debutant
```

```
import programme.bot_expert as bot_expert
```

Ainsi, si par exemple le fichier du jeu du morpion a besoin de la réponse du bot expert, il appellera la fonction de cette manière:

```
bot_expert.morpion_bot(), avec évidemment les arguments demandés par cette fonction.
```

II - Démonstration et tests

1) Test main

Avec l'implémentation des 2 nouveaux bots à nos programmes, l'affichage des sélections des joueurs est donc différents:

Après l'exécution du main, le programme demande à l'utilisateur combien il y a de joueurs réels.

```
Bienvenue !  
Veuillez entrer le nombre de joueur: █
```

```
Bienvenue !  
Veuillez entrer le nombre de joueur: 3  
n doit être compris entre 0 et 2 compris  
Veuillez entrer le nombre de joueur: j  
n doit être un entier  
Veuillez entrer le nombre de joueur: =  
n doit être un entier  
Veuillez entrer le nombre de joueur: █
```

Le nombre de joueurs réel doit être compris entre 0 et 3 et si l'utilisateur rentre un nombre supérieurs ou inférieurs, ou rentre autre chose qu'un nombre, le programme renvoie un message d'erreur.

0 joueur réel:

```
Bienvenue !  
Veuillez entrer le nombre de joueur: 0  
Selectionnez la difficulté du bot joueur 1  
D : débutant | E : expert ---> e  
Veuillez entrer le niveau de difficulté du bot joueur 2:  
D : débutant | E : expert ---> d█
```

Le programme va donc demander à l'utilisateur la difficulté des deux bots. L'utilisateur peut donc sélectionner d pour débutant ou e pour expert.

```

Bienvenue !
Veuillez entrer le nombre de joueur: 0
Selectionnez la difficulté du bot joueur 1
D : débutant | E : expert ---> t
Erreur, commande innexistante
D : débutant | E : expert ---> 5
Erreur, commande innexistante
D : débutant | E : expert ---> █

```

Si l'utilisateur choisit autre chose que ces 2 choix, le programme renvoie alors un message d'erreur.

L'utilisateur peut choisir soit débutant contre expert soit débutant contre débutant soit expert contre expert.

1 joueur réel:

```

Bienvenue !
Veuillez entrer le nombre de joueur: 1
Joueur 1
Entrez un nom: bot1D
Ce nom est indisponible.
Entrez un nom: bot1E
Ce nom est indisponible.
Entrez un nom: bot2E
Ce nom est indisponible.
Entrez un nom: bot2D
Ce nom est indisponible.
Entrez un nom:
Votre nom ne peut pas être vide !
Entrez un nom: █

```

Si l'utilisateur choisit un nom parmi bot1D, bot1E, bot2D, bot2E ou alors un nom vide, le programme renvoie un message d'erreur.

```

Bienvenue !
Veuillez entrer le nombre de joueur: 1
Joueur 1
Entrez un nom: joachim
Veuillez entrer le niveau de difficulté du bot joeuur 2:
D : débutant | E : expert ---> e█

```

Par rapport au 2ème joueur qui est donc un bot, l'utilisateur doit donc choisir d ou e.

2 joueurs réels:

```
Bienvenue !  
Veuillez entrer le nombre de joueur: 2  
Joueur 1  
Entrez un nom: andreas  
Joueur 2  
Entrez un nom: joachim
```

Si l'utilisateur choisit un nom parmi bot1D, bot1E, bot2D, bot2E ou alors un nom vide, le programme renvoie un message d'erreur.

2) Test devinettes

Lorsque 2 bots joue l'un contre l'autre, le jeu se fait automatiquement, il suffit juste d'appuyer sur entrée pour pouvoir passer les différentes étapes d'affichage sur le terminal.

Dans ce cas les 2 bots sont des bots débutants.

```
bot1D commence à faire deviner !  
bot2D joue...  
  
nombre = 76  
  
C'est moins.  
bot2D joue...  
  
nombre = 68  
  
C'est moins.  
bot2D joue...  
  
nombre = 51  
  
Bravo, vous avez trouvé en 3 coups !
```

```
à bot2D de faire deviner !  
bot1D joue...
```

```
nombre = 30
```

```
C'est plus.
```

```
Bravo, vous avez trouvé en 9 coups !
```

```
bot1D à gagné 3pts | bot2D à gagné 9pts
```

Après que le bot1D fasse deviner, le bot2D fait à son tour deviner.
A la fin, le programme affiche le nombre de points des 2 bots.

3) Test allumettes

Pour le jeu des allumettes il suffit d'appuyer sur Entrée pour passer les étapes lorsque c'est un bot qui joue.

Dans ce cas, 2 bots débutant s'affrontent:

```
I I I I I I I I I I I I I I I I I I I I I I  
bot1D joue...
```

On appuie sur Entrée pour que le bot joue.

```
Il reste 18 allumettes !  
  
I I I I I I I I I I I I I I I I I I I I I I  
bot2D joue...
```

Et ainsi de suite jusqu'à la fin de la partie.

```
bot1D a gagné !
```


4) Test morpion

```
bot1D est O, bot2D est X
7 | 8 | 9
-----
4 | 5 | 6
-----
1 | 2 | 3

à : bot1D
bot1D joue...
```

Il suffit, comme pour les autres jeux, d'appuyer sur Entrée pour que le bot joue.

```
7 | 0 | 9
-----
4 | 5 | 6
-----
1 | 2 | 3

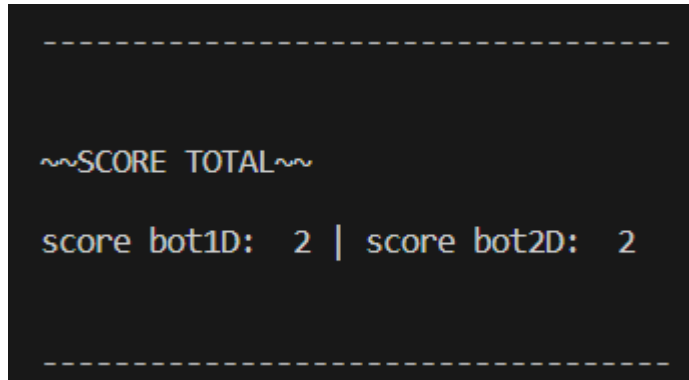
à : bot2D
bot1D joue...
```

```
X | 0 | 0
-----
0 | X | 0
-----
X | X | 0

bot1D à gagné !
```

5) Test score

Pour le score, tout fonctionne normalement sachant que les bots ont des profils comme tous les utilisateurs.



```
-----  
  
~~SCORE TOTAL~~  
  
score bot1D:  2 | score bot2D:  2  
  
-----
```

III - Comparaison des algorithmes

Afin d'obtenir des données sur les victoires des bots, nous avons créé de nouveaux scripts qui nous permettront d'obtenir des fichiers contenant les données. Nous avons donc dû aussi créer des scripts alternatif des scripts des jeux afin de les faire tourner automatiquement, et aussi afin de récupérer le temps de réponse des bots.

En lançant le script contenu dans le fichier "get_stat.py", on obtient trois fichiers, contenant les résultats des jeux ainsi que le temps de réponse moyen en milliseconde des bots, sur un nombre de parties données.

Il y a un fichier contenant les résultats opposant un bot débutant contre un bot débutant, un opposant un bot expert contre un bot expert, et enfin un bot débutant contre un bot expert. Voici les résultats que nous avons obtenues sur 10 000 parties :

1) Pour le jeu des devinettes:

Débutant contre débutant : 74765 contre 75196. Ils ont un nombre de points relativement proche. Celui ne dépend uniquement que de l'aléatoire, ils ont tous les deux 50% de chance d'avoir plus de points que l'autre.

Expert contre expert : 57819 contre 58025. Les points sont moins hauts, Il leur faut beaucoup moins d'essais pour trouver le chiffre que pour les débutants, en moyenne.

Débutant contre expert : 57996 pour le débutant contre 75159 pour l'expert. Les chiffres sont similaires à ceux du dessus.

Leurs temps de réponses sont en moyenne d' environ 0.0004 millisecondes pour le débutant contre 0.0013 ms. Évidemment, le bot expert prend plus de temps que le bot débutant, puisque celui utilise la dichotomie et doit faire plus de calculs qu'un bot qui renvoie simplement un chiffre aléatoire.

2) Pour le jeu des allumettes:

Débutant contre débutant : 5043 contre 4957. Tout comme le jeu des devinettes, et comme le morpion, ces chiffres sont uniquement liés à l'aléatoire, et donc ils ont 50% de chance de gagné.

Expert contre expert : 4990 contre 5010. Étant donné qu'ils utilisent la même stratégie, le gagnant est déterminé par lequel joue en premier. Ainsi, ils ont eux aussi 50% de chance de gagner.

Expert contre débutant : 27 pour le débutant contre 9973 pour l'expert. Ici, le seul moyen pour le débutant de gagner est de compter sur la chance. Soit, qu'il joue par pur hasard la même stratégie que l'expert tout en jouant en premier.

Niveau temps de réponse, ceci sont assez similaires: 0.0011 pour l'expert contre 0.0015. L'expert est même plus rapide cette fois-ci, même en regardant sur les autres fichiers. Le script de l'expert se base uniquement sur des structures conditionnelles et des calculs de modulo. Il y a donc fort à parier que ceci est moins coûteux en calcul que de générer un nombre aléatoire.

3) Pour le jeu du morpion:

Débutant contre débutant : 4353 contre 4364. Ici, tout comme les autres jeux, les bots jouant à l'aléatoire, ils ont 50% de chance de gagner.

Expert contre expert : 2319 contre 2438. Les chiffres sont plus bas que pour les débutants, ce qui veut dire qu'il y a eu plus d'égalité, ce qui est compréhensible quand dans leurs scripts, ils essayent en permanence de bloquer toutes les possibilités de gagner pour l'adversaire contrairement aux débutants. Leur seul moyen de gagner sur l'autre est de se retrouver dans une situation où ils ont deux possibilités pour gagner à la fois, ce qui peut arriver avec de la chance.

Expert contre Débutant : 326 pour le débutant contre 7999 pour l'expert. Comme précédemment, le seul moyen pour le débutant de gagner, est de commencer et de réussir à avoir 2 possibilité de gagner en une fois. Ce qui peut arriver avec de la chance, expliquant les quelques fois où le bot expert a perdu. Celui n'est donc pas infailible étant donné qu'il ne peut pas prévoir plus d'un coup à l'avance.

Pour leur temps de réponse, cette fois-ci celui de l'expert est bien plus haut avec 0.0312ms contre 0.0066ms pour le débutant. Bien que les 2 doivent vérifier quel case est libre, l'expert regarde en plus s'il y a des case gagnante pour l'un ou l'autre des adversaires ce qui augmente grandement son temps de réponse.

