ALGO2 – Algorithmique et Programmation 2

Fiche de TP supplémentaire - 2

Mastermind - Le robot joueur

Dans cette deuxième étape, vous allez réaliser la partie où c'est l'ordinateur qui cherche à deviner le code secret. Pour cela, vous allez enrichir les deux classes `Proposition` et `Mastermind`.

L'algorithme pour jouer est simple. Vous remarquerez d'abord qu'on peut énumérer toutes les combinaisons. Pour une taille de `4` et un nombre de couleurs de `6`, les combinaisons vont de `[0,0,0,0]` à `[5,5,5,5]`. On passe d'une proposition à la suivante en ajoutant `1` au chiffre le plus à gauche (ou à droite, peu importe, choisissez votre convention). Si le chiffre obtenu atteint le nombre de couleurs du mastermind, alors on le passe à zéro et on reporte la retenue sur le chiffre immédiatement à droite. La méthode `proposition_suivante(self)` de la classe `Proposition` retourne une nouvelle proposition qui est la combinaison suivante de `self`.

```
>>> p = Proposition([0,0,0,0])
>>> p2 = p.proposition_suivante()
>>> print(p2)
[1, 0, 0, 0]
>>> p = Proposition([1,2,3,4])
>>> p2 = p.proposition_suivante()
>>> print(p2)
[2, 2, 3, 4]
>>> p = Proposition([5,2,3,4])
>>> p2 = p.proposition_suivante()
>>> print(p2)
[0, 3, 3, 4]
>>> p = Proposition([5,5,5,5])
>>> p2 = p.proposition_suivante()
>>> print(p2)
[0, 0, 0, 0]
```

Pour jouer, une idée simple (qui permet de trouver assez rapidement la solution !) est donc d'énumérer toutes les combinaisons, et de s'arrêter lorsqu'on trouve une combinaison qui pourrait être le code secret : si c'était elle le code secret, elle aurait retourné le même nombre de pions bien placés et mal placés pour chaque proposition déjà faite que ce qui a été retourné. C'est cette proposition qui est jouée par l'ordinateur.

La classe `Proposition` doit maintenant posséder deux attributs `__bp` et `__mp` qui sont initialisés à zéro. La classe `Proposition` possède les nouvelles méthodes suivantes :

- `set_bp(self,bp)` qui permet d'affecter une valeur à l'attribut `__bp`
- `set_mp(self,mp)` qui permet d'affecter une valeur à l'attribut `__mp`
- `bp(self)`, `mp(self)` et `code(self)` qui retournent respectivement la valeur de `__bp`, `__mp` et une copie du code contenu dans la `Proposition`
- `proposition_suivante(self)` qui a déjà été discutée

— `valide(self,code)` dont la spécification est `Proposition,list(int) -> boolean` qui vérifie que les nombres de bien placés et de mal placés de `code` par rapport à `self` sont bien les mêmes que ceux mémorisés dans `__bp` et `__mp`.
— et toujours la méthode `__str__(self)` que vous n'oublierez pas de mettre à jour (on veut maintenant afficher le nombre de bien et de mal placés.

La classe `Mastermind` doit maintenant proposer un nouvel attribut qui est l'historique des propositions déjà faites. Elle possède également :

— une méthode `est_valide(self,code)` qui vérifie si `code` est valide pour chaque proposition de l'historique.
— une méthode `proposition(self)` qui retourne la prochaine combinaison valide en fonction de l'historique des propositions déjà faites. Si aucune proposition n'a déjà été faite, alors la première proposition est générée aléatoirement. Cette proposition est ajoutée à l'historique.
— une méthode `prend_score(self,bp,mp)` qui affecte le score de bien placés (`bp`) et de mal placés (`mp`) à la dernière proposition de l'historique.
— une méthode `partie_finie(self)` qui teste si la partie est finie (soit parce que l'ordinateur a gagné, soit parce qu'il a épuisé tous les essais permis).
— une méthode `__str__(self)` qui présente tout l'historique des propositions.

Enfin, vous adapterez le script principal proposé pour la semaine 1 afin d'être compatible avec la trace suivante. Vous ferez bien attention qu'il y a maintenant deux parties de mastermind en cours en même temps :

```
projet-mastermind$ python3 main_mastermind.py
Tour 1
Votre proposition ? [0,1,2,3]
La réponse est  1 bien placés et  2 mal placés.
Ma proposition est : [0, 4, 2, 0]
Mon score est : 0  bien placés,  2  mal placés.
Tour 2
Votre proposition ? [0,2,1,4]
La réponse est  1 bien placés et  1 mal placés.
Ma proposition est : [1, 0, 0, 1]
Mon score est : 1  bien placés,  0  mal placés.
Tour 3
Votre proposition ? [0,5,5,1]
La réponse est  0 bien placés et  1 mal placés.
Ma proposition est : [4, 2, 3, 1]
Mon score est : 2  bien placés,  2  mal placés.
Tour 4
Votre proposition ? [3,2,0,3]
La réponse est  1 bien placés et  1 mal placés.
Ma proposition est : [3, 2, 4, 1]
Mon score est : 1  bien placés,  3  mal placés.
Tour 5
Votre proposition ? [3,1,1,2]
La réponse est  4 bien placés et  0 mal placés.
Bravo, la partie est finie, vous avez gagné.
```