Division des enseignements en informatique

TP Mastermind

Nous nous proposons d'écrire une version du jeu Mastermind. Le but de ce jeu est de découvrir une combinaison de pions de différentes couleurs. Le nombre de tentatives (de coups) est limité. A chaque coup, le joueur reçoit des indications sur la proximité de la combinaison qu'il a proposée. Ces indications sont le nombre de pions bien placés et le nombre de pions présents dans la combinaison mais mal placés.

Dans la version du jeu que nous allons proposer, les couleurs sont au nombre de 8. Dans notre version initiale du jeu, il ne sera pas possible d'avoir une couleur en double dans une combinaison. A chaque couleur idnetifiée par une lettre, nous associerons son nom et un code en utilisant des dictionnaires :

```
codes_couleurs = {"R":0, "B":1, "J":2, "V":3, "N":4, "M":5, "F":6, "O":7}
couleurs_codes = {0:"R", 1:"B", 2:"J", 3:"V", 4:"N", 5:"M", 6:"F", 7:"O"}
noms_couleurs = {"R":"Rouge", "B":"Bleu", "J":"Jaune", "V":"Vert", "N":"
    Noir", "M":"Marron", "F":"Fushia", "O":"Orange"}
```

Un découpage du programme de jeu en différentes méthodes est proposé. Seules les méthodes principales sont indiquées. Vous êtes libres d'en ajouter si vous le jugez utile.

1. Ecrivez une fonction `verification(solution, proposition)` qui retourne le nombre de pions bien placés et le nombre de pions présents dans la combinaison, mais mals placés. La solution et la proposition sont fournies sous forme de liste d'entiers de même taille.
2. Ajoutez une fonction `affiche_indications(nb_places, nb_couleurs, proposition)` qui met en forme les indications fournies au joueur (nombre de pions bien placés, nombre de pions de la bonne couleur mais mal placés. Nous souhaitons un affichage de la forme : bonne couleur - rappel proposition - bien placé, par exemple :

```
 *  *  *  *  RBJV
       *  *  BRJV  *  *
```

3. Ecrivez une fonction permettant la saisie d'une proposition par le joueur. Cette saisie aura la forme d'une chaîne de caractères constituée des lettres représentant les couleurs. La fonction que vous écrirez devra gérer convenablement : la saisie d'une combinaison de taille différente de celle de la combinaison recherchée et l'utilisation d'un caractère ne correspondant à aucune couleur.
4. Ecrivez la méthode permettant de jouer. Cette méthode aura entre autres paramètres la combinaison à découvrir (liste d'entiers) et le nombre de coups autorisés pour découvrir la bonne combinaison.
5. Afin de pouvoir jouer contre un ordinateur, prévoyez une fonction générant de manière aléatoire une combinaison de pions d'une taille donnée. Cette combinaison sera retournée sous forme d'une liste d'entiers.
6. Complétez le jeu en permettant au joueur de choisir le niveau difficulté : nombre de couleurs utilisées, nombre de pions de la combinaison et nombre de coups autorisés pour découvrir la bonne combinaison.
7. Ajoutez la possibilité d'avoir des couleurs en double dans une combinaison.

Aller plus loin

Nous nous intéressons ici à la performance de la fonction de vérification.

1. Commencez par évaluer la complexité de la fonction que vous avez écrite (notation en 0(f(n)).
2. En utilisant le script `TP_mastermind_benchmark.py` réalisez un graphique représentant le temps de calcul de la méthode verification en fonction du nombre de pions de la combinaison. Ce script utilisant `matplotlib`, installez cette bibliothèque sur votre poste en tapant dans une invite de commande : `pip3 install matplotlib --proxy=10.0.4.2:3128` (la commande peut légèrement varier en fonction des machines : demander l'aide du professeur si besoin).

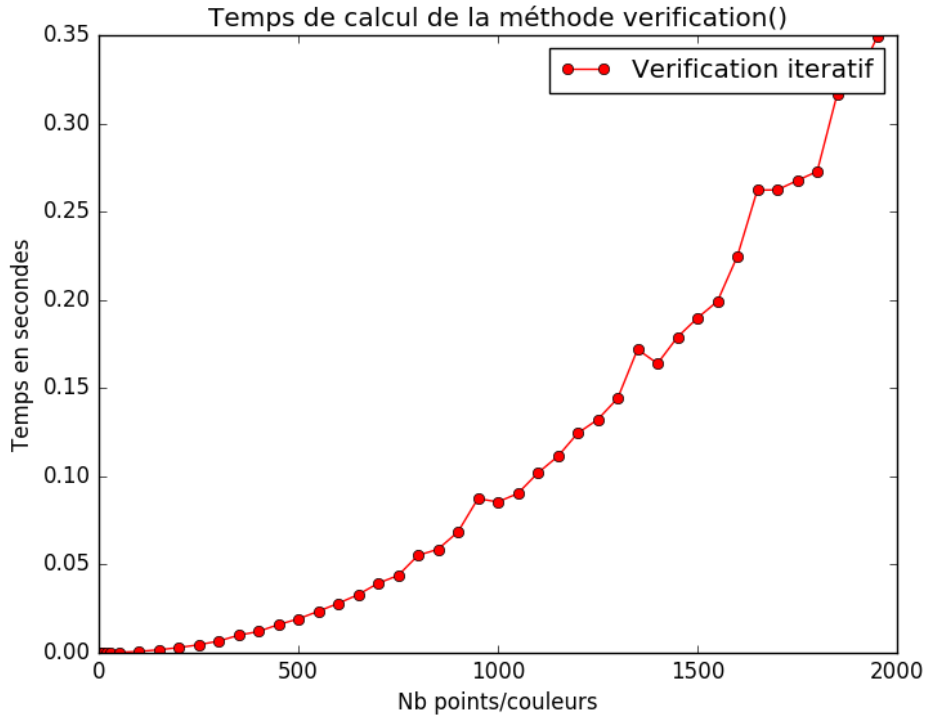

FIGURE 1 – Temps de calcul de verification() en fonction du nombre de points de la combinaison

Pour améliorer la complexité de l'algorithme, nous utiliserons des outils Python issus de la programmation fonctionnelle (https://fr.wikipedia.org/wiki/Programmation_fonctionnelle) :

— `zip()` qui permet de combiner plusieurs listes entre elles (cf. https://docs.python.org/3/library/functions.html#zip) ;
— `Counter` qui permet de compter le nombre d'occurence des éléments d'une liste et retourne le résultat sous forme de dictionnaire (cf. https://docs.python.org/3/library/collections.html#collections.Counter).

3. Proposez des améliorations de votre fonction `verification()` et comparer à l'aide du script précédent les temps de calcul des différentes méthodes.