GROUPE INP La Prépa

Projet d'informatique tronc commun - S4
Mastermind

Louan Giroud
Titouan Vial

Mars 2024

Un plateau du jeu Mastermind
https://commons.wikimedia.org/wiki/File:Mastermind.jpg#/media/Fichier:Mastermind.jpg

1 Dépot du projet

Nous avons pour ce projet travaillé en utilisant git afin de pouvoir nous organiser au mieux dans le travail sur le poste de chacun et avoir un historique de nos modifications. Le dépot github est accessible si vous souhaitez le visiter : https://github.com/Mityno/projet_info_s4.

2 Tests

Nous avons effectué les tests des fonctions de `common.py` dans le fichier `autotest.py`. Les tests des autres fonctions du projet ont été faits en analysant les logs des parties pour vérifier qu'il n'y avait pas d'incohérences dans les résultats.

Nous avons par exemple trouvé que notre fonction `evil_codemaker` pouvait renvoyer la combinaison proposée comme solution maximisant l'ensemble des solutions possibles car toutes les combinaisons possibles renvoyaient un ensemble de solution d'une seule solution. Nous avons donc rajouté une condition pour traiter ce cas particulier qui permettait de finir certaines parties du codebreaker 3 contre le codemaker 2 en 5 coups. Les fonctions les plus complexes ont été déboguées ainsi et le détail des corrections est expliqué dans le code en commentaire (on explique la démarche suivie par le code).

3 Questions

Notations

On notera dans la suite :

- le nombre de couleur du jeu : $C =$ `len(common.COLORS)`
- la longueur des combinaisons du jeu : $L =$ `common.LENGTH`
- le nombre total de possibilités de combinaisons du jeu : $n = C^L$

Question 3

On cherche ici à calculer l'espérance de la variable aléatoire discrète notée X qui compte le nombre de coups joués par le codebreaker 0 avant son succès.

Comme il y a remise, on a $X(\Omega) = \mathbb{N}^*$.

Soit $k \in \mathbb{N}^*$, on définit les évènements suivant :

- A_k : "obtenir une mauvaise combinaison au k^{ieme} tirage"
- $S_k = A_1 \cap A_2 \cap \cdots \cap A_{k-1} \cap \overline{A_k}$: "obtenir la bonne combinaison au k^{ieme} tirage en n'ayant obtenu que des mauvaises combinaisons aux $k-1$ tirages précédents"

On a donc : $\mathbb{P}(A_k) = \dfrac{n-1}{n}$

On arrête les tirages dès que la bonne combinaison est trouvée, on a des tirages avec remise donc les évènements A_k sont indépendants. Ainsi :

$$\begin{aligned}
\mathbb{P}(X = k) &= \mathbb{P}(S_k) \\
&= \mathbb{P}\left(\overline{A_k}\right) \prod_{i=1}^{k-1} \mathbb{P}(A_i) \\
&= \frac{1}{n} \prod_{i=1}^{k-1} \frac{n-1}{n} \\
&\boxed{= \frac{1}{n} \left(\frac{n-1}{n}\right)^{k-1}}
\end{aligned}$$

On détermine la probabilité que l'on trouve la bonne combinaison, c'est à dire la probabilité que l'on finisse, en calculant la probabilité de l'union des S_n qui sont, par définition, incompatibles :

$$\begin{aligned}
\mathbb{P}(\text{fin}) &= \mathbb{P}\left(\bigcup_{n\in\mathbb{N}^*} S_n\right) \\
&= \sum_{k=1}^{+\infty} \frac{1}{n}\left(\frac{n-1}{n}\right)^{k-1} = \frac{1}{n}\sum_{k=1}^{+\infty}\left(\frac{n-1}{n}\right)^{k-1} \\
&= \frac{1}{n}\frac{1}{1-\frac{n-1}{n}} \qquad \text{(somme géométrique, } \frac{n-1}{n} \in [-1,1]) \\
&= \frac{1}{n-(n-1)} = \frac{1}{1} \\
&= 1
\end{aligned}$$

On est donc assuré que le jeu finisse.

On peut alors calculer l'espérance :

$$\begin{aligned}
\mathbb{E}(X) &= \sum_{k=1}^{+\infty} k\mathbb{P}\left(X=k\right) \\
&= \sum_{k=1}^{+\infty} \frac{k}{n}\left(\frac{n-1}{n}\right)^{k-1} \\
&= \frac{1}{n}\sum_{k=1}^{+\infty} k\left(\frac{n-1}{n}\right)^{k-1}
\end{aligned}$$

On reconnaît la somme partielle de la série dérivée de $\sum_{k\geq 0}\left(\frac{n-1}{n}\right)^k$ or :

$$\sum_{i=0}^{+\infty}\left(\frac{n-1}{n}\right)^k = \frac{1}{1-\frac{n-1}{n}}$$

En utilisant la dérivée de cette série convergente par rapport à $\frac{n-1}{n}$, on obtient :

$$\sum_{k=1}^{+\infty} k\left(\frac{n-1}{n}\right)^{k-1} = \frac{1}{\left(1-\frac{n-1}{n}\right)^2} = n^2$$

Ainsi $\mathbb{E}(X) = \frac{1}{n}\times n^2 = \boxed{n = C^L}$

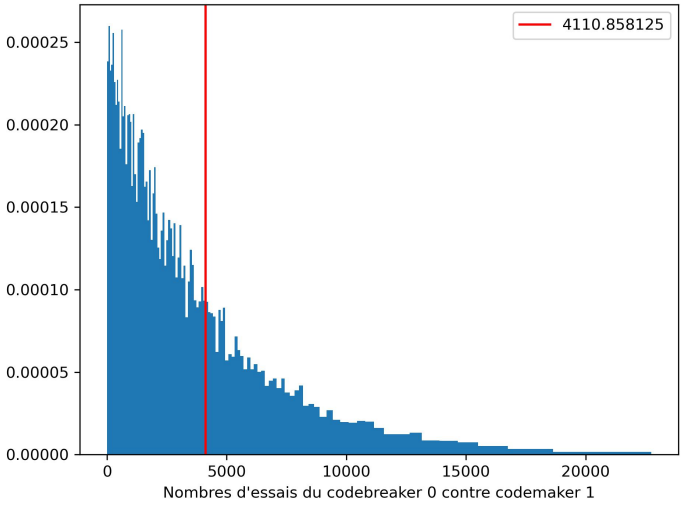

FIGURE 1 – Densité du nombres d'essais du codebreaker 0 contre le codemaker 1

On remarque que la moyenne du nombre d'essais (droite verticale rouge sur la figure 1) est à 4 110, ce qui est très proche de la valeur recherchée. On remarque aussi que le codebreaker 0 fait parfois énormément de tests avant de finir (plus de 20 000).

On a donc bien le même résultat que celui trouvé théoriquement, où pour $C = 8$ et $L = 4$ on a $\mathbb{E}(X) = n = 4\ 096$.

Question 4

On se place maintenant dans le cadre d'un tirage sans remise donc le codebreaker peut faire au maximum n tirages (puisque c'est le nombre total de possibilités). D'où $X(\Omega) = [\![1; n]\!]$.

Soit $k \in [\![1; n]\!]$, on définit les évènements suivant :

- A_k : "obtenir une mauvaise combinaisons au k^{ieme} tirage"
- E_k : "obtenir la bonne combinaison au k^{ieme} tirage en n'ayant obtenu que des mauvaises combinaisons aux $k-1$ tirages précédents", càd $E_k = \bigcap_{i=1}^{k-1} A_i \cap \overline{A_k}$

Qui sont de probabilités :

$$\begin{aligned}
\mathbb{P}(A_k) &= \frac{n-k}{n-(k-1)} \\
\mathbb{P}\left(\overline{A_k}\right) &= \frac{1}{n-(k-1)} \\
\mathbb{P}(A_k) + \mathbb{P}\left(\overline{A_k}\right) &= 1
\end{aligned}$$

D'après l'indépendance deux à deux des (A_k), on a alors :

$$\begin{aligned}
\mathbb{P}(E_k) &= \mathbb{P}\left(A_1 \cap A_2 \cap \cdots \cap A_{k-1} \cap \overline{A_k}\right) \\
&= \mathbb{P}\left(\overline{A_k}\right) \prod_{i=1}^{k-1} \mathbb{P}(A_i) \\
&= \frac{1}{n-(k-1)} \underbrace{\prod_{i=1}^{k-1} \frac{n-i}{n-(i-1)}}_{\frac{n-(k-1)}{n}} \qquad \text{(produit télescopique)} \\
&\boxed{= \frac{1}{n}}
\end{aligned}$$

On a donc que $\boxed{X \rightsquigarrow \mathcal{U}(n)}$, d'où $\mathbb{E}(X) = \dfrac{n+1}{2} = \dfrac{C^L+1}{2}$.
Pour $n = 4\ 096$, on a donc $\mathbb{E}(X) = 2\ 048{,}5$.

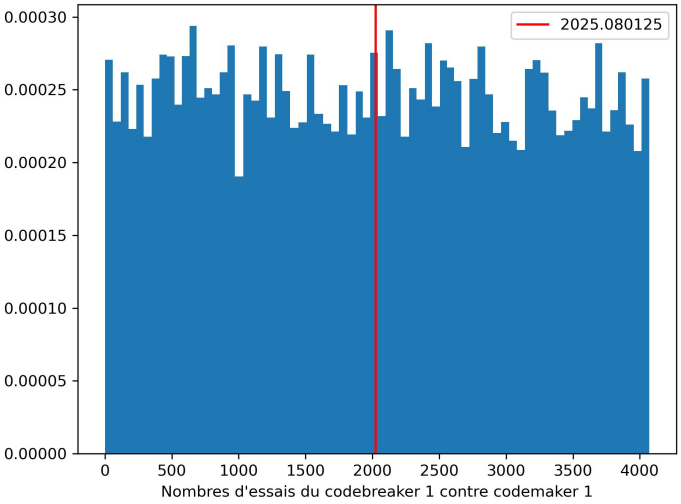

FIGURE 2 – Nombres d'essais du codebreaker 1 contre le codemaker 1

On a bien le même résultat que celui trouvé expérimentalement, figure 2 où pour $C = 8$ et $L = 4$ on obtenait une moyenne du nombre d'essai de 2 025 qui est proche des 2 048,5 coups attendus, et une répartition visiblement uniforme du nombre de coup avant de finir les parties.

Question 7

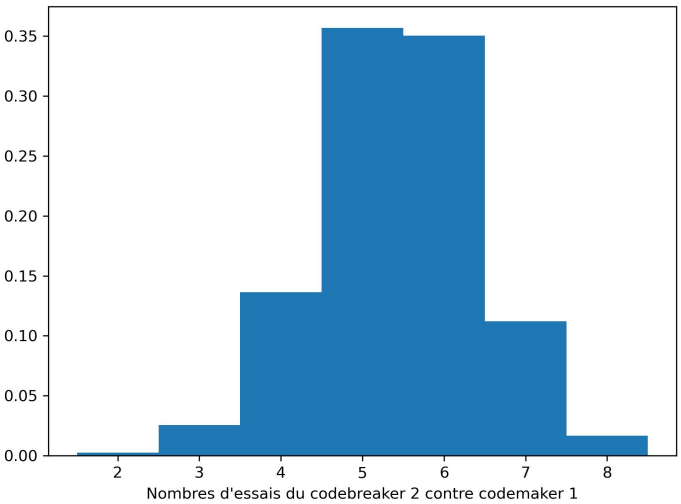

FIGURE 3 – Nombre d'essai du codebreaker 2 contre le codemaker 1

On observe sur la figure 3 que le codebreaker 2 est bien plus efficace que le codebreaker 1 : les parties du codebreaker 2 durent entre 1 et 9 coups, alors que celles du codebreaker 1 durent en moyenne plus de 2 000 coups. Ainsi la stratégie de réduire l'ensemble des combinaisons possibles à chaque évaluation retournée par le codemaker est réellement efficace pour finir une partie plus vite.

Question 8

Le codemaker 2 que nous avons implémenté parcourt l'ensemble des combinaisons encore possibles pour choisir la solution qui permet d'avoir, après l'évaluation de la combinaison proposée, un ensemble des combinaisons encore possibles de taille maximale.

Pour comparer toutes les solutions qu'il pourrait choisir, on cherche le maximum de la longueur de combinaisons possibles après l'évaluation, càd qu'on cherche la solution qui supprime le moins de combinaisons possibles. Ainsi, on se ramène à une recherche d'un minimum et l'on peut appliquer un principe de backtracking pour améliorer les performances de notre codemaker 2.

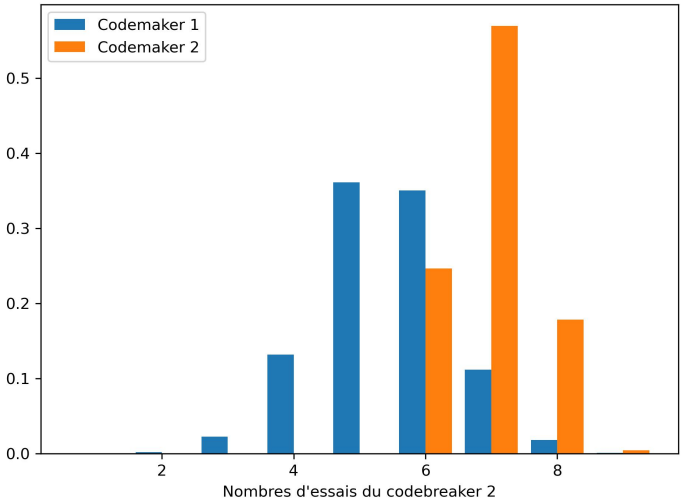

FIGURE 4 – Nombre d'essais du codebreaker 2 contre le codemaker 1 (bleu) et 2 (orange)

On observe sur la figure 4 que les parties contre le codemaker 2 sont longues, souvent 7 coup ou plus. Il arrive parfois que des parties soient jouées 6 coups, mais pas moins.

Alors que celles contre le codemaker 1, peuvent être plus courtes, on voit que certaines on été faites en 2 coup, mais sur un plus grand nombre de parties, on pourrait avoir des parties en 1 coup, et on a au maximum 8 coup.

Ce sont des résultats concluants, puisque l'on cherchait à faire un codemaker rallongeant les parties le plus possible, c'est bien ce que l'on constate après implémentation.

Question 11

En se plaçant dans une situation ou :

- `COLORS = [A, B, C, D, E, F]`
- `LENGTH = 2`

Si on a déjà testé les combinaisons suivantes et reçu les évaluations associées :

- $AB \rightarrow (1, 0)$
- $AC \rightarrow (1, 0)$

On sait que la combinaison finale sera de la forme suivante :
AX avec $X \in [D, E, F]$

En testant uniquement des combinaisons encore possibles, on obtiendrait des informations minimes, on ne supprimerait qu'une combinaison à la fois. Alors qu'en testant une combinaison comme DE on aurait beaucoup plus d'informations. En essayant cette combinaison, il y a trois évaluations possibles qui nous donnent toutes quelle est la bonne combinaison :

- Si on obtient $(1, 0)$, on sait que la bonne combinaison est AE
- Si on obtient $(0, 1)$, on sait que la bonne combinaison est AD
- Si on obtient $(0, 0)$, on sait que la bonne combinaison est AF

Ainsi dans ce cas, tester une solution que l'on sait fausse nous donne plus d'information (on est sûr d'avoir la bonne réponse en 2 coups) que de tester dans une combinaison encore possible (qui aboutirait dans le pire cas à la bonne réponse en 3 coups). En gardant le même principe mais en ajoutant plus de couleurs, on va rapidement voir que tester des solutions fausses ajoute peu de coups (1 toutes les quelques couleurs, selon la valeur de `LENGTH`) alors qu'en se restreignant aux combinaisons possibles le nombre de coups pour le pire cas augmente linéairement avec le nombre de couleurs supplémentaires. Ainsi, il est clairement plus intéressant dans certains cas de jouer une combinaison qui n'est plus possible.

Question 12

Le codebreaker 3 utilise un algorithme minimax pour choisir quelle combinaison il propose. En effet il va déterminer la combinaison qui permet de réduire au maximum l'ensemble des solutions disponibles en prenant en compte le fait que le codemaker 2 cherche à maximiser la taille cet ensemble. Il teste donc toutes les combinaisons à chaque coup pour déterminer la combinaison la plus optimale.

On observe que sur une centaine de parties, les parties contre le codemaker 2 se finissent toujours en 6 coups pour $C = 8$ et $L = 4$, on a donc toujours le même style de partie (d'ailleurs, avec les mémoïsations effectuées, pour une exécution du programme c'est toujours la même partie qui sera jouée).

Par contre, contre le codemaker 1, le codebreaker 3 est plus rapide, on observe sur la figure 5, que le codebreaker 3 finit les parties en 6 coups au maximum, et il lui arrive régulièrement de finir en 5 coups ou moins (environ une partie sur deux). Nous avons un exemple en 3 coups sur nos tests, mais rien ne l'empêcherai de finir en 1 coup. On constate par contre qu'il a moins de chance que le codebreaker 2 de finir en 4 coups ou moins (environ 15% des parties pour le codebreaker 2 et moins 5% des parties pour le codebreaker 3).

Le codebreaker 3 est plus régulier que le codebreaker 2, il finit en moyenne toutes ses parties en 5 ou 6 coups, alors que le codebreaker 2 les finit en 4 à 8 coups (répartition plus disparate).

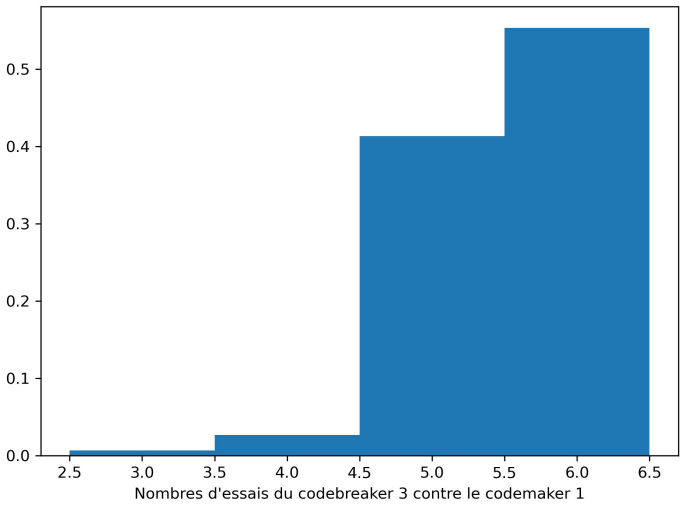

FIGURE 5 – Codebraker 3 contre codemaker 1

En faisant les calculs de moyenne, on trouve que le codebreaker 3 a une moyenne d'environ 5.5 coups contre le codemaker 1, et le codebreaker 2 a une moyenne d'environ 5.4 coups. Les deux codebreakers ont donc des résultats très similaire en moyenne, mais pas forcément sur les parties prises en particulier, où le codebreaker 2 peut être bien plus performant. Ce n'est pas très étonnant car le codebreaker 3 cherche à s'assurer de finir les parties en peu de coup, et pour cela, il joue des coups qui peuvent être en dehors de l'ensemble des solutions possibles. Donc il est moins probable pour lui de finir la partie sur un coup de chance et essayant "au hasard" la bonne solution.