TD 6 : Mastermind
Décembre 2012 - Janvier 2013

Ah noël ! La fête des joujoux où de nombreux fabriquant de babioles en plastiques s'en mettent plein les poches. Pour célébrer cette période faste, ce TD se propose de vous replonger dans un des jeux de votre enfance : le Mastermind. On va dans un premier temps coder des fonctions basiques qui nous permettront d'y jouer (mais sans y passer des heures). Ensuite, on tentera de faire en sorte que Maple joue (de façon pas trop bête) lui-même au Mastermind, et l'on écrira alors une proto-IA.

1 Les règles du jeu

J'imagine que vous avez déjà, sinon au moins entendu parler, joué au Mastermind. Si néanmoins ce n'est pas le cas, en voici donc les règles de façon synthétique.

Le Mastermind se joue à deux joueurs, que nous appellerons Alice et Bob (pour changer). Alice se tourne un peu les pouces, c'est donc son fonctionnement que nous programmerons en premier, pour que vous-même puissiez jouer le rôle de Bob (qui, lui, réfléchit un peu). Alice et Bob ont à leur disposition une grosse quantité de petites perles en plastique, qui peuvent être de 8 couleurs différentes.

Au début de la partie, Alice va se constituer une combinaison secrète en arrangeant de la façon qu'elle souhaite 4 de ces petites billes, par exemple Bleu-Rouge-Jaune-Vert, ou Rouge-Vert-Orange-Violet.

Le but de Bob est de deviner cette combinaison. Il a pour y parvenir le droit de soumettre à Alice des conjectures sur cette combinaison. Alice répond à ces conjectures en donnant à Bob à chaque fois le nombre de couleurs bien placées (avec autant de pions rouges), et le nombre de couleurs présentes dans la combinaison secrète, mais mal placées (par des pions blancs).

Si Bob parvient à deviner le secret d'Alice en moins de 12 questions, il gagne la partie. Dans le cas contraire c'est Alice qui remporte la victoire. Voici un exemple de partie, où chaque colonne indique une proposition de Bob, avec la dernière ligne représentant les pions rouges et blancs correspondant.

Bleu	Jaune	Jaune	Jaune	Violet	Violet							Violet
Bleu	Rouge	Vert	Violet	Vert	Rouge							Rouge
Jaune	Rouge	Violet	Rouge	Rouge	Bleu							Bleu
Jaune	Violet	Vert	Rouge	Bleu	Vert							Vert
0/1	1/1	1/1	0/1	1/3	4/0							secret

Bob gagne en 6 coups.

Bien sûr, personne n'est d'accord sur les règles : 6 couleurs, 8 ou 10 questions, droit de laisser des cases vides, droit de répéter la même couleur dans une combinaison, et ainsi de suite. Je vous laisse libres de choisir à quel jeu de règles vous obéirez. Dans la suite de ce TD, je me limiterai à seulement 6 couleurs, pour que notre IA finale soit plus rapide.

Exercice 1 (*Comptons !*)

Combien y-a-t-il de combinaisons possibles quand on ne joue qu'avec 6 couleurs, sans autoriser ni les cases vides ni les répétitions ?

Et qu'en est-il lorsqu'on applique ces deux règles et que le nombre de couleurs autorisées passe à 8 ?

2 Le jeu d'Alice

Tout ce que fait Alice, c'est choisir une combinaison intiale, et répondre par un couple d'entiers à une soumission de Bob (le couple pions rouges / pions blancs).

Exercice 2 (*Génération du secret*)

Coder la procédure `combinaison := proc ()` *qui renvoie une combinaison au hasard sous forme d'une liste à 4 éléments.*

On représentera chaque couleur par un entier entre 0 et 5. Notez que cette fonction est beaucoup plus facile à écrire si vous autorisez les duplications de couleurs.

Pour tirer un entier au hasard, il faut appeler la fonction `rand`. Cette fonction est un peu particulière : elle prend en entrée un intervalle `i..j` et renvoie une fonction qui ne prend pas de paramètre. Par exemple pour tirer un nombre entre 3 et 7 :

```
random_3_7 := rand(3..7) ;
x := random_3_7();
```

La question suivante est plus délicate, ne vous lancez pas tête baissée dans l'écriture.

Exercice 3 (*Pions rouges et pions blancs*)

Ecrire la procédure `rougesBlancs := proc (hypothese, secret)`*, qui à partir des arguments donnés comme des listes d'entiers, renvoie la liste à deux éléments contenant les nombres de pions rouges et blancs correspondant.*

Attention : si le secret est 1234 et que Bob propose 1100, alors on doit répondre "1 rouge, 0 blancs" et pas "1 rouge, 1 blanc". Commencez par déterminer le nombre de pions rouges, et évitez les redites lors du calcul du nombre de pions blancs.

On aimerait maintenant pouvoir jouer contre notre Alice artificielle de façon interactive, et c'est ce que nous allons implémenter maintenant.

Exercice 4 (*Jeu interactif*)

En utilisant les fonctions précédentes, écrire `jeu := proc()`*, la procédure de jeu. Celle-ci comportera typiquement une grosse boucle* `while` *qui s'incrémentera à chaque question que vous lui poserez, et qui attend que vous trouviez la solution ou que le nombre d'hypothèses dépasse 10. Vous aurez besoin des fonctions suivantes :*

- ▷ *Pour lire une hypothèse, donnée sous la forme de 4 entiers séparés par des espaces, vous utiliserez* `hypothese := scanf("%d %d %d %d");`
- ▷ *Pour afficher les nombres de la liste* `rb`*, vous écrirez :*
  `printf("%d pions rouges, %d pions blancs.\n",rb[1],rb[2]);`

A la fin de votre programme, vous de `printf` *pour afficher "Perdu" ou "Gagné en x coups", puis renvoyez un booléen indiquant si vous avez gagné.*

3 Rendre Bob intelligent

Ne perdez pas trop de temps à jouer contre l'ordi, et tâchez de réfléchir à comment élaborer une bonne stratégie pour le Mastermind.

Une stratégie raisonnable peut l'être à deux niveaux :

1. *Non-bêtise* : On ne joue que des coups raisonnables, c'est à dire en accord avec les informations déjà obtenues précédemment.
2. *Intelligence* : On joue non seulement des coups raisonnables, mais qui en plus peuvent nous apporter le plus d'information possible.

Exercice 5 (*Stratégie non-bête*)

On va se concentrer sur la réalisation d'une stratégie non-bête seulement, ce qui est déjà bien.

1. *Si l'on voit les combinaisons comme des entiers en base 6, il est possible de définir à partir d'une combinaison c un* successeur, *$c+1$. Le coder en Maple, et faire en sorte qu'il supporte le modulo (*ie $5555+1=0000$*).*
2. *Ecrire une fonction qui dans une partie, à partir de tout ce qui a été joué auparavant, parcourt les combinaisons dans l'ordre à partir d'une combinaison au hasard et joue la première combinaison cohérente rencontrée.*
3. *Faire jouer cette stratégie 100 fois et déterminer en combien de coups elle parvient à gagner en moyenne.*

Pour votre culture Résoudre rapidement le Mastermind à 6 couleurs est un problème qui a été beaucoup étudié. Il existe un algorithme simple et systématique pour le résoudre en 6 coups. Knuth parvient (mais de manière un peu plus compliquée) à descendre à 5, et en 1993 des chercheurs ont découvert un algorithme qui donnait le bon résultat en 4,3 coups en moyenne, sans jamais monter au-dessus de 6.