LE PUISSANCE 4 ET LES MATHEMATIQUES

DUBUIS Michaël
LANDRA Nicolas
FERRIE Ludovic

SOMMAIRE

INTRODUCTION :

Lors du choix du sujet, nous avons choisi de prendre un jeu. Nous espérions y trouver de la facilité et un minimum de concret. Nous avons vite déchanté. Il nous a fallut mathématiser quelque chose que nous connaissions peut-être un peu trop. Il nous est rapidement apparu qu'il fallait reprendre les choses calmement et réviser un peu ce qu'était le puissance 4. Nous commencerons donc cet exposé par un petite leçon sur le puissance 4. Après cette révision nécessaire nous avons réussit à dégager du jeu des règles mathématiques : la transformation du plateau, des joueurs et des conditions de victoires.

Malgré la simplicité de ce jeu, il y a un nombre important de problèmes mathématiques. Nous verrons dans la seconde, la troisième et la quatrième partie de l'exposé différentes façons d'aborder ces problèmes : sous forme matricielle, sous forme de code en base 3 et enfin sous forme d'un tableau de poids.

La suite logique de cette mathématisation du puissance 4 est la création d'un logiciel sachant jouer sans jamais perdre. Nous verrons en dernière partie de cette analyse comment nous pouvons concevoir une intelligence artificielle capable de remporter face à un joueur.

I Rappels divers :

> Puissance 4 (appelé aussi parfois 4 en ligne) est un jeu de stratégie combinatoire abstrait, commercialisé pour la première fois en 1974 par a Milton Bradley Company, plus connue sous le nom de MB et détenue depuis 1984 par la société Hasbro.
>
> Le but du jeu est d'aligner 4 pions sur une grille comptant 6 rangées et 7 colonnes. Chaque joueur dispose de 21 pions d'une couleur (par convention, en général jaune ou rouge). Tour à tour les deux joueurs placent un pion dans la colonne de leur choix, le pion coulisse alors jusqu'à la position la plus basse possible dans ladite colonne suite à quoi c'est à l'adversaire de jouer. Le vainqueur est le joueur qui réalise le premier un alignement (horizontal, vertical ou diagonal) d'au moins quatre pions de sa couleur. Si alors que toutes les cases de la grille de jeu sont remplies aucun des deux joueurs n'a réalisé un tel alignement, la partie est déclarée nulle.

Wikipedia

II Les matrices :

II.a Le plateau & les joueurs :

Le plateau étant une grille de six lignes et sept colonnes, il nous a paru logique de transformer cette grille en matrice de six lignes et sept colonnes. Nous verrons par la suite que le fait que cette matrice ne soit pas une matrice carré a soulevé d'autres problèmes mathématiques.

La formalisation, quant à elle, est plus complexe de par les choix possibles.
Il semble naturel de noter le joueur un par un « 1 » dans la matrice, par un « 2 » le joueur deux et une case vide par un « 0 ». Après quelques tests et pour simplifier la recherche du tour de jeu, nous avons transformer le « 2 » du joueur deux en « -1 ». En effet nous avons testé, par somme de tous les facteurs de la matrice, le tour de jeu. Avec « 1 » et « 2 », il est difficile de trouver le tour de jeu sans opérations complexes. Il est impossible de le faire avec la parité. Il nous fallait donc deux facteurs que ne pouvaient être additionné ou qui s'annulaient.

Les complexes apportent à cette problématique une solution élégante. En donnant au joueur un la partie réelle et au joueur deux la partie irréelle, il était *facile* de savoir à qui cela venait de jouer. Cependant le choix des facteurs s'annulant nous apporte la même simplicité de solution. Jusqu'à ce stade les deux solutions étaient retenues.

Afin de savoir à qui cela venait de jouer, nous avons opté pour une solution très simple. Nous sommons tous les facteurs de la matrice. Avec la solution des complexes, il suffit de comparer la valeur réelle et la valeur imaginaire divisé par *i* de ce nombre complexe. Si la valeur « réelle » est plus important que la valeur « imaginaire » alors le joueur 1 vient de jouer et si les parties sont égales, le joueur 2 vient de jouer.

Exemple : $\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & i & 0 & 0 \\ 0 & 0 & 0 & i & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & i & 0 & 0 \end{pmatrix}$ La somme des facteurs de cette matrice = 3+3i

Donc $P_{re} = 3$ et $P_{im}/i = 3$
donc $P_{re} = P_{im}$
Le joueur 2 vient donc de jouer, cela vient au joueur 1.

En utilisant la solution des facteurs s'annulant, on aura une somme égale à 1 ou 0. Si la somme est égale à 1, le joueur 1 vient de joueur, si c'est égale à 0, le joueur 2 vient de jouer.

Exemple :

$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & -1 & 0 & 0 \end{pmatrix}$ La somme des facteurs de cette matrice

S = -1+(-1)+1+1+1+(-1) = 0
Le joueur 2 vient donc de jouer, cela vient au joueur 1.

La deuxième colle qui nous avons affronté résidait dans la fait de transformer la gravité, qui s'applique naturellement dans la grille du puissance 4, en opérations mathématiques. Après une recherche approfondie, nous n'avons trouver aucun opérateur ou fonction capable de résoudre ce problème. Nous avons donc décidé de créer un opérateur réalisant deux fonctions inventé pour l'occasion : *Gravite*.

Gravite sera donc G.

AddL(M_1,M_2) sera un opérateur remplaçant les zéro de la première ligne de la matrice M_2 par les chiffres, à la place que les zéros sus-cités, de la matrice ligne M_1 qui aura toujours autant de colonne que la matrice M_2.

Et enfin, S(M_2) sera un opérateur appliquant un principe de gravité à la matrice M_2. Ce principe sera explicité plus tard.

$$G(M_1,M_2) = S(M_2) \circ AddL(M_1,M_2)$$

Nous allons maintenant détailler les sous-fonctions AddL et S.

Soit une Matrice M_2, de *n* lignes et *m* colonnes.

L'opérateur S, que nous avons appelé *gravity*, va échanger, en partant du bas de chaque colonne et en remontant le premier zéro rencontré par le premier chiffre (ou nombre) non nul parcouru. Cette opération sera répétée sur chaque ligne jusqu'à attendre le haut de la colonne.

Exemple : Soit *a, b, c, d* et *e* des nombres différents de zéro.

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ e & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & c & 0 & d & 0 & d \\ 0 & 0 & 0 & 0 & 0 & a & 0 \\ 0 & a & a & b & 0 & 0 & 0 \end{pmatrix} \underset{\rightarrow}{\mathrm{S}} \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & c & 0 & 0 & 0 & 0 \\ e & a & a & b & d & a & d \end{pmatrix}$$

Soit une Matrice M_2, de *n* lignes et *m* colonnes et M_2 une matrice ligne de *m* colonnes.

L'opérateur AddL va remplacer sur la première ligne les zéros de la matrice M_2 par les nombres non-nuls de la matrice M_1.

Exemple : Soit *a, b, c, d* et *e* des nombres différents de zéro.

$$\begin{pmatrix} e & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ e & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & c & 0 & d & 0 & d \\ 0 & 0 & 0 & 0 & 0 & a & 0 \\ 0 & a & a & b & 0 & 0 & 0 \end{pmatrix} \# \begin{pmatrix} a & b & c & d & 0 & e & 0 \end{pmatrix} = \begin{pmatrix} e & b & c & d & 0 & e & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ e & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & c & 0 & d & 0 & d \\ 0 & 0 & 0 & 0 & 0 & a & 0 \\ 0 & a & a & b & 0 & 0 & 0 \end{pmatrix}$$

II.b Conditions de victoires et les vérifications :

Pour gagner, il faut que la somme 4 par 4 d'une ligne, colonne ou diagonale de la matrice soit égale à 4 fois le « nombre attribué au joueur ».

Exemple :

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 & -1 & 0 & 0 \\ 0 & 1 & -1 & -1 & 1 & 0 & 0 \end{pmatrix}$$

La difficulté réside dans le fait qu'il faut créer une fonction capable de vérifier en prenant la matrice entière si l'un des deux joueurs à gagné ou non. Afin de facilité nos recherches dans cette voie, nous avons commencé par décomposer en plusieurs fonctions vérifiant des sous-matrices. Ces fonctions vérifient soit un ligne, soit une colonne, soit une diagonale.

Vérifications des lignes :

Nous verrons dans un premier temps la vérification des lignes dans une sous-matrice.

Afin de vérifier si une ligne est gagnante, il nous faut additionner les facteurs de la ligne en question. Nous multiplierons donc notre matrice carré M par une matrice ligne remplie de zéro sauf sur la colonne qui a la même position que la ligne à vérifier. Puis nous multiplierons la matrice ainsi obtenue par une matrice colonne remplie de un.

Exemple :

Nous testons si la deuxième ligne est gagnante.

$$\begin{pmatrix}0 & 1 & 0 & 0\end{pmatrix} * \begin{pmatrix}a & b & c & d\\ e & f & g & h\\ i & j & k & l\\ m & n & p & q\end{pmatrix} = \begin{pmatrix}e & f & g & h\end{pmatrix}$$

puis

$$\begin{pmatrix}e & f & g & h\end{pmatrix} * \begin{pmatrix}1\\1\\1\\1\end{pmatrix} = \begin{pmatrix}e+f+g+h\end{pmatrix}$$

Nous obtenons après ces opérations une matrice 1x1 qui nous permet de savoir si cette ligne est gagnante ou non.

Vérifications des colonnes :

Afin de vérifier si une colonne est gagnante, il nous faut additionner les facteurs de la colonne en question. Nous multiplierons donc notre matrice carré M par une matrice colonne remplie de zéro sauf sur la ligne qui a la même position que la colonne à vérifier. Puis nous multiplierons la matrice ainsi obtenue par une matrice ligne remplie de un.

Exemple :

Nous testons si la deuxième colonne est gagnante.

$$\begin{pmatrix}a & b & c & d\\ e & f & g & h\\ i & j & k & l\\ m & n & p & q\end{pmatrix} * \begin{pmatrix}0\\1\\0\\0\end{pmatrix} = \begin{pmatrix}b\\f\\j\\n\end{pmatrix}$$

puis

$$\begin{pmatrix}1 & 1 & 1 & 1\end{pmatrix} * \begin{pmatrix}b\\f\\j\\n\end{pmatrix} = \begin{pmatrix}b+f+j+n\end{pmatrix}$$

Nous obtenons après ces opérations une matrice 1x1 qui nous permet de savoir si cette colonne est gagnante ou non.

Vérifications de la diagonale principale :

Les diagonales principales peuvent-être vérifiées facilement grâce à un opérateur matriciel existant, la trace de la matrice.

Définition : La trace d'une matrice, notée $Tr(A)$, est la somme des éléments diagonaux.

Exemple :

Nous testons si la diagonale principale est gagnante.

$$Tr(\begin{pmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & p & q \end{pmatrix}) = a + f + k + l$$

Vérifications de l'autre diagonale :

Après une recherche approfondie nous n'avons trouvé aucune fonction permettant de vérifier la seconde diagonale ni de faire la symétrie de la matrice afin d'utiliser de nouveau la trace. Nous avons donc pris la décision de créer cette fonction.

La fonction appelée SysM fera la sysmétrie géométrique de la matrice par rapport à une droite verticale passant par le milieu de la matrice.

Exemple :

Nous testons si l'autre diagonale est gagnante.

Pour ce faire nous allons commencer par utiliser SysM sur notre matrice donc :

$$\left(\begin{array}{cc|cc} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & p & q \end{array}\right) \underset{\rightarrow}{\text{SysM}} \left(\begin{array}{cc|cc} d & c & b & a \\ h & g & f & e \\ l & k & j & i \\ q & p & n & m \end{array}\right)$$

Puis nous utiliserons la trace, ce qui nous donne :

$$Tr(\begin{pmatrix} d & c & b & a \\ h & g & f & e \\ l & k & j & i \\ q & p & n & m \end{pmatrix}) = d + g + j + m$$

Extraction d'une matrice carrée :

Ces vérifications obtenue une nouvelle énigme c'est présenté à nous. Comment pouvons nous extraire des matrices 4 x 4 d'une matrice *n* x *m*

Nous avons le début d'une solution qui consiste à multiplier, notre matrice jeu de taille *n* x *m* par une matrice *m* x *n,* remplie de zéros sauf sur la diagonales des lignes que nous voulons récupérer. Puis nous multiplions à nouveau par une matrice de taille *m* x *m* remplie de zéros sauf sur la diagonale des colonnes que nous voulons extraire.

Exemple :

Nous voulons extraire la sous matrice de l'angle en haut à gauche :

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} * \begin{pmatrix} A_{1;1} & A_{1;2} & A_{1;3} & A_{1;4} & A_{1;5} & A_{1;6} & A_{1;7} \\ A_{2;1} & A_{2;2} & A_{2;3} & A_{2;4} & A_{2;5} & A_{2;6} & A_{2;7} \\ A_{3;1} & A_{3;2} & A_{3;3} & A_{3;4} & A_{3;5} & A_{3;6} & A_{3;7} \\ A_{4;1} & A_{4;2} & A_{4;3} & A_{4;4} & A_{4;5} & A_{4;6} & A_{4;7} \\ A_{5;1} & A_{5;2} & A_{5;3} & A_{5;4} & A_{5;5} & A_{5;6} & A_{5;7} \\ A_{6;1} & A_{6;2} & A_{6;3} & A_{6;4} & A_{6;5} & A_{6;6} & A_{6;7} \end{pmatrix} =$$

$$\begin{pmatrix} A_{1;1} & A_{1;2} & A_{1;3} & A_{1;4} & A_{1;5} & A_{1;6} & A_{1;7} \\ A_{2;1} & A_{2;2} & A_{2;3} & A_{2;4} & A_{2;5} & A_{2;6} & A_{2;7} \\ A_{3;1} & A_{3;2} & A_{3;3} & A_{3;4} & A_{3;5} & A_{3;6} & A_{3;7} \\ A_{4;1} & A_{4;2} & A_{4;3} & A_{4;4} & A_{4;5} & A_{4;6} & A_{4;7} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Puis :

$$\begin{pmatrix} A_{1;1} & A_{1;2} & A_{1;3} & A_{1;4} & A_{1;5} & A_{1;6} & A_{1;7} \\ A_{2;1} & A_{2;2} & A_{2;3} & A_{2;4} & A_{2;5} & A_{2;6} & A_{2;7} \\ A_{3;1} & A_{3;2} & A_{3;3} & A_{3;4} & A_{3;5} & A_{3;6} & A_{3;7} \\ A_{4;1} & A_{4;2} & A_{4;3} & A_{4;4} & A_{4;5} & A_{4;6} & A_{4;7} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} * \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} =$$

$$\begin{pmatrix} A_{1;1} & A_{1;2} & A_{1;3} & A_{1;4} & 0 & 0 & 0 \\ A_{2;1} & A_{2;2} & A_{2;3} & A_{2;4} & 0 & 0 & 0 \\ A_{3;1} & A_{3;2} & A_{3;3} & A_{3;4} & 0 & 0 & 0 \\ A_{4;1} & A_{4;2} & A_{4;3} & A_{4;4} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Nous avons avec cette méthode réussie à mettre en avant toutes les sous matrices carrées possibles de la matrice jeu. Nous avons constater qu'appliquer notre système de vérification pour les lignes et les colonnes fonctionnait parfaitement. En effet ajouter des zéros n'influe en rien notre résultat. Cependant si nous extrayons une matrice qui n'est pas dans un angle nous ne pouvons pas appliquer la trace.

Nous avions pensé au départ à supprimer les zéros que nous rajoutions. Mais il nous est impossible de différentier les zéros rajoutés des zéros éventuels présents dans la matrice, dû au case vide.

Nous cherchons toujours une fonction existante permettant de décaler la matrice à vérifier dans un angle (en haut à gauche ou en bas à droite). Si nous ne trouvons pas, nous créerons un opérateur le faisant.

III La Base 3 :

III.a Le principe :

Après avoir longuement travaillé sur les matrices, il nous était essentiel de trouver d'autres méthodes mathématiques pour traiter le « puissance 4 ». La difficulté était que nous avions le regard tourné vers une solution qui nous paraissait maintenant la seule envisageable. Cependant, du fait que nous ayons deux informaticiens dans le groupe, il nous a très rapidement paru logique d'aborder le problème d'un point de vue binaire. Ne pouvant pas utiliser la base deux car les variables pouvaient avoir trois valeurs possibles. Nous avons donc opté pour la base trois.

Comme pour les matrices, nous avons transformer les cases vides en un « 0 » et les cases jouées par le joueur un par un « 1 », cependant, nous avons chois « 2 » pour les cases jouées par le joueur deux. Chaque ligne devenait ainsi un nombre en base 3. Nous donc mis en place un tableau de conversion :

Exposant Chiffre	6	5	4	3	2	1	0
1	729	243	81	27	9	3	1
2	1458	486	162	54	18	6	2

Exemple :

$(0\ 0\ 0\ 1\ 2\ 0\ 1)_{base3} = (27*1 + 9*2 + 1*1)_{base10} = 46_{base10}$
$(a_6\ a_5\ a_4\ a_3\ a_2\ a_1\ a_0)_{base3} = (a_6*p(a_6) + a_5*p(a_5) + a_4*p(a_4) + a_3*p(a_3) + a_2*p(a_2) + a_1*p(a_1) + a_0*p(a_0))_{base10}$

Combinaisons possibles :

Afin d'accélérer le calcul et ne pas le faire à la main, nous avons fait un programme calculant toutes les combinaisons et les écrivant dans un fichier texte ainsi que les combinaisons gagnantes et les écrivant dans un autre fichier texte. Le code source de ce programme est disponible dans les annexes. Nous avons donc établie une liste de toutes les combinaisons gagnantes :

0001111 : 41	0222222 : 729	1121111 : 1175	2112222 : 1863
0002222 : 81	1001111 : 770	1122220 : 1213	2121111 : 1904
0011110 : 121	1002222 : 810	1122221 : 1214	2122220 : 1942
0011111 : 122	1011110 : 850	1122222 : 1215	2122221 : 1943
0011112 : 123	1011111 : 851	1201111 : 1256	2122222 : 1944
0012222 : 162	1011112 : 852	1202222 : 1296	2201111 : 1985
0021111 : 203	1012222 : 891	1211110 : 1336	2202222 : 2025
0022220 : 241	1021111 : 932	1211111 : 1337	2211110 : 2065
0022221 : 242	1022220 : 970	1211112 : 1338	2211111 : 2066
0022222 : 243	1022221 : 971	1212222 : 1377	2211112 : 2067
0101111 : 284	1022222 : 972	1221111 : 1418	2212222 : 2106
0102222 : 324	1101111 : 1013	1222200 : 1450	2221111 : 2147
0111100 : 361	1102222 : 1053	1222201 : 1451	2222000 : 2161
0111101 : 362	1111000 : 1081	1222202 : 1452	2222001 : 2162
0111102 : 363	1111001 : 1082	1222210 : 1453	2222002 : 2163
0111110 : 364	1111002 : 1083	1222211 : 1454	2222010 : 2164
0111111 : 365	1111010 : 1084	1222212 : 1455	2222011 : 2165
0111112 : 366	1111011 : 1085	1222220 : 1456	2222012 : 2166
0111120 : 367	1111012 : 1086	1222221 : 1457	2222020 : 2167
0111121 : 368	1111020 : 1087	1222222 : 1458	2222021 : 2168
0111122 : 369	1111021 : 1088	2001111 : 1499	2222022 : 2169
0112222 : 405	1111022 : 1089	2002222 : 1539	2222100 : 2170
0121111 : 446	1111100 : 1090	2011110 : 1579	2222101 : 2171
0122220 : 484	1111101 : 1091	2011111 : 1580	2222102 : 2172
0122221 : 485	1111102 : 1092	2011112 : 1581	2222110 : 2173
0122222 : 486	1111110 : 1093	2012222 : 1620	2222111 : 2174
0201111 : 527	1111111 : 1094	2021111 : 1661	2222112 : 2175
0202222 : 567	1111112 : 1095	2022220 : 1699	2222120 : 2176
0211110 : 607	1111120 : 1096	2022221 : 1700	2222121 : 2177
0211111 : 608	1111121 : 1097	2022222 : 1701	2222122 : 2178
0211112 : 609	1111122 : 1098	2101111 : 1742	2222200 : 2179
0212222 : 648	1111200 : 1099	2102222 : 1782	2222201 : 2180
0221111 : 689	1111201 : 1100	2111100 : 1819	2222202 : 2181
0222200 : 721	1111202 : 1101	2111101 : 1820	2222210 : 2182
0222201 : 722	1111210 : 1102	2111102 : 1821	2222211 : 2183
0222202 : 723	1111211 : 1103	2111110 : 1822	2222212 : 2184
0222210 : 724	1111212 : 1104	2111111 : 1823	2222220 : 2185
0222211 : 725	1111220 : 1105	2111112 : 1824	2222221 : 2186
0222212 : 726	1111221 : 1106	2111120 : 1825	2222222 : 2187
0222220 : 727	1111222 : 1107	2111121 : 1826	
0222221 : 728	1112222 : 1134	2111122 : 1827	

Il y a **162** combinaisons gagnantes pour **2188** combinaisons totales. En bleu sont représentés les possibilités gagnantes en colonne et en diagonale.

III.b Les possibilités :

Une fois la liste établie, il reste à la traiter.

Méthode de récupération :

Une fois que notre jeu est établi, nous obtenons donc une liste de six nombres en base 10. Afin de vérifier les colonnes et les diagonales, il nous faut des fonctions qui récupéreront le 0, le 1 ou le 2 de toutes cases intéressantes dans le but d'obtenir un nouveau nombre qui représentera la colonne ou la diagonale à vérifier.

Raisonnement :

J'obtiens comme jeu ceci :

$$\begin{array}{rrrrrrrr} 0 & 0 & 0 & 0 & 0 & 0 & 0 = & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 = & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 = & 0 \\ 0 & 0 & 0 & 1 & 2 & 0 & 0 = & 45 \\ 0 & 0 & 2 & 2 & 1 & 0 & 0 = & 225 \\ 0 & 2 & 1 & 1 & 1 & 2 & 0 = & 609 \end{array}$$

Pour vérifier la première colonne, je vais prendre mes 6 chiffres et appliquer la formule :

$$f(n, p(c)) = \lfloor \frac{n}{p(\mathrm{c})} \rfloor \quad \text{avec } p(\mathrm{c}) = \text{poids de la colonne a récupérer .}$$

$\lfloor \frac{0}{729} \rfloor = 0$

$\lfloor \frac{0}{729} \rfloor = 0$

$\lfloor \frac{0}{729} \rfloor = 0$

$\lfloor \frac{45}{729} \rfloor = 0$

$\lfloor \frac{225}{729} \rfloor = 0$

$\lfloor \frac{609}{729} \rfloor = 0$

La première colonne est donc composé de 000000.

$\lfloor \frac{0}{243} \rfloor = 0$

$\lfloor \frac{0}{243} \rfloor = 0$

$\lfloor \frac{0}{243} \rfloor = 0$

$\lfloor \frac{45}{243} \rfloor = 0$

$\lfloor \frac{225}{243} \rfloor = 0$

$\lfloor \frac{609}{243} \rfloor = 2$

La deuxième colonne est donc de cette forme : 0000002.

$\lfloor \frac{0}{81} \rfloor = 0$

$\lfloor \frac{0}{81} \rfloor = 0$

$\lfloor \frac{0}{81} \rfloor = 0$

$\lfloor \frac{45}{81} \rfloor = 0$

$\lfloor \frac{225}{81} \rfloor = 2$

$\lfloor \frac{609}{81} \rfloor = 4$

La troisième colonne devrait-être de cette forme : 000024. Or il ne peut y avoir de 4.

On constate que si les colonnes avant sont pleines, le calcule est faux. Il nous faut donc réviser la fonction précédente. Elle sera donc de la forme suivante :

$$f_c(n, p(\mathrm{c})) = \left\lfloor \frac{n - \sum_{i=c}^{i=7} \left(\left\lfloor \frac{n}{p(i)} \right\rfloor * p(i) \right)}{p(\mathrm{c})} \right\rfloor$$

On applique cette fonction pour trouver la troisième colonne, donc c=5, p(5)=81,p(6)=243 et p(7)=729.

$$f_5(0,p(5)) = \left\lfloor \frac{0-\sum_{i=5}^{i=7}\left(\left\lfloor \frac{0}{p(i)} \right\rfloor * p(i)\right)}{p(5)} \right\rfloor = \left\lfloor \frac{0-0}{81} \right\rfloor = 0$$

$$f_5(0,p(5)) = \left\lfloor \frac{0-\sum_{i=5}^{i=7}\left(\left\lfloor \frac{0}{p(i)} \right\rfloor * p(i)\right)}{p(5)} \right\rfloor = \left\lfloor \frac{0-0}{81} \right\rfloor = 0$$

$$f_5(0,p(5)) = \left\lfloor \frac{0-\sum_{i=5}^{i=7}\left(\left\lfloor \frac{0}{p(i)} \right\rfloor * p(i)\right)}{p(5)} \right\rfloor = \left\lfloor \frac{0-0}{81} \right\rfloor = 0$$

$$f_5(45,p(5)) = \left\lfloor \frac{45-\sum_{i=5}^{i=7}\left(\left\lfloor \frac{45}{p(i)} \right\rfloor * p(i)\right)}{p(5)} \right\rfloor = \left\lfloor \frac{45-0}{81} \right\rfloor = 0$$

$$f_5(225,p(5)) = \left\lfloor \frac{225-\sum_{i=5}^{i=7}\left(\left\lfloor \frac{225}{p(i)} \right\rfloor * p(i)\right)}{p(5)} \right\rfloor = \left\lfloor \frac{225-0}{81} \right\rfloor = 2$$

$$f_5(609,p(5)) = \left\lfloor \frac{609-\sum_{i=5}^{i=7}\left(\left\lfloor \frac{609}{p(i)} \right\rfloor * p(i)\right)}{p(5)} \right\rfloor = \left\lfloor \frac{609-486}{81} \right\rfloor = 1$$

La troisième colonne sera donc de cette forme : 000021.

Même si cette formule semble juste, il faudrait effectuer plus de tests afin de vérifier qu'elle fonctionne. On peut cependant avancer qu'elle ressemblera fortement à ça.

De même pour récupérer les diagonales, on appliquera à chaque ligne la fonction f_c , où c sera une suite croissante.

Il faudra après avoir récupérer les chiffres 0,1 ou 2 les concaténer ou les additionner après les avoir multipliés par $p(|\text{numéro de la ligne} - 6|)$. On obtiendra ainsi un nombre représentant la colonne.

Idées à développer :

Faute de temps, nous n'avons pas finit notre étude sur la base 3.

Il nous resterait à trouver une logique à la suite des nombres obtenus avec les combinaisons gagnantes. Cette suite logique nous permettrait (peut-être) de trouver une méthode pour tendre vers les coups gagnants.

IV Les poids :

Après les matrices et les bases 3, nous sommes revenu sur une théorie bien plus simpliste et bien plus facile à appliquer. Nous utiliserons comme support un simple tableau aux dimensions du jeu de puissance 4.

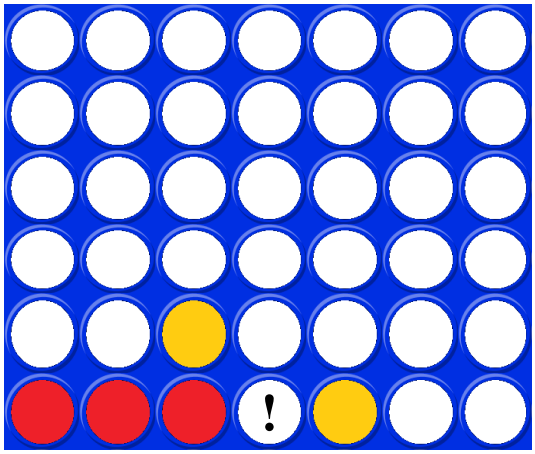
IV.a Idée principale :

La théorie des poids est une méthode permettant de jauger la pertinence des possibilités de jeu qui s'offre à nous. Elle existe d'ores et déjà pour des jeux tel que les échecs ou les dames chinoises. Cette méthode instaure dans la grille de jeu des valeurs calculables à partir de critères simples. Ces critères se cumulent pour donner un éventail de possibilités guidant le joueur vers une victoire dite « assurée », ou du moins plus guidée que par le simple hasard.

La théorie n'étant pas achevé il est naturel que les critères présents ne suffisent pas à assurer la victoire, mais selon les estimations que nous avons fait, une partie sur 2 est gagnante contre quelqu'un utilisant les même règles, et une partie sur 3 contre une personne jouant normalement.

Cette théorie est d'ailleurs appliquée aux autres jeux de plateau avec des résultats très probants.

L'idée de cette théorie vient du fait qu'un joueur jauge en permanence l'intérêt des cases jouables selon quelques critères bien définis. Par exemple si votre adversaire réussi à aligner 3 jetons, il vous paraitra naturel de bloquer cet alignement pour empêcher l'adversaire de gagner. La théorie des poids retranscrit cela en chiffre.



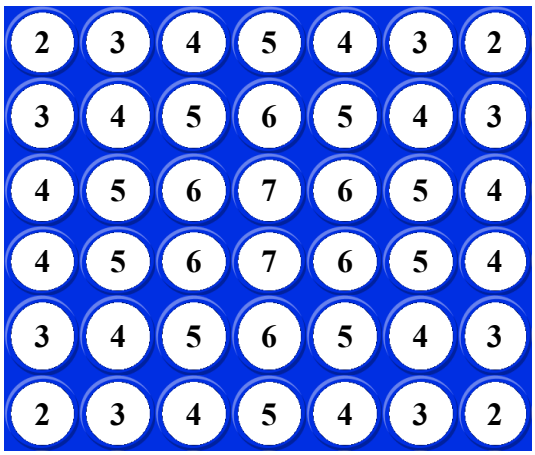
De la même manière il vous semblera naturel de privilégier un blocage ou un alignement assurant la victoire, et cette théorie peut, par des règles adaptées, le concrétiser. L'avantage étant de voir comment se déroule une partie « humaine » pour réussir à le transformer en « réflexion virtuelle ». Ainsi une intelligence artificielle réussirait à cumuler les capacités virtuelles et humaines pour réussir à gagner 100% des parties commencées par celle-ci. Nous avons déjà appris que le joueur qui commence la partie de Puissance 4 à 0% de chance de défaite s'il suit des coups en fonction de ceux de son adversaire. Le second joueur quant à lui ne peut que provoquer le match nul pour ne pas perdre.

Si cette théorie peut permettre le développement d'une intelligence artificielle, les parties se ressembleraient toutes et 1 partie sur 2 serait gagnée par le joueur commençant la partie, et la seconde serait une partie nulle. En effet, là où le cerveau ne peut prévoir que quelques coups à l'avance, l'ordinateur lui peut prévoir bien plus loin et ainsi prévoir 3, 4, 5 ou même 6 coups à l'avance. Donc plus les coups sont prévus à l'avance, plus les possibilités de jeu s'élargissent. Ce qui amène au fait de penser à un calcul qui nous permettrait de savoir si un ensemble de 3 coups serait plus « rentable » qu'un autre: La moyenne de la somme des valeurs obtenues par ces valeurs successives. Il serait possible d'envisager que cette « rentabilité » soit calculée non pas avec la moyenne, mais avec la différence de la somme des coups alliés, et la somme des coups adverses. Ceci prendrait en compte les concessions faites à l'adversaire.

IV.b La théorie :

Le puissance 4 est constitué de 7 colonnes et 6 lignes. Et on voit que certaines colonnes et lignes offrent beaucoup plus de possibilités de victoire que d'autres. De part cette observation on peut en déduire que les plus hautes valeurs seront celles qui offrent le plus de possibilités pour gagner. Cette observation faite, les possibilités d'alignements de 4 jetons sont comptées depuis un emplacement. Des valeurs sont attribuées à ces lignes et colonnes pour obtenir un tableau de valeur simple.

Plus la valeur d'une case est élevée et plus les chances de former une combinaison gagnante à partir de cette case sont grandes. Ces valeurs sont arbitraires mais révèlent dans les grandes lignes l'intérêt de placer ses jetons à tel ou tel emplacement.



poids des cellules

Voyant que ces valeurs ne sont pas suffisantes pour régler le problème, de nouvelles règles furent intégrées dans ce tableau. Il a donc fallu prendre en compte certaines observations qui sont faites par le joueur en position de « défense » comme en position « d'attaque ».On sait que lors d'une partie, il faut savoir défendre, et savoir attaquer. En effet on est obligé de bloquer une combinaison adverse pour ne pas perdre, parallèlement, on doit essayer de former une fourchette pour gagner. Savoir favoriser l'un ou l'autre est primordial dans ce type de jeu. Or ce tableau montre les valeurs non altérées par la présence de jetons alliés ou adverses.

Pour se faire de nouvelles règles sont intégrées et prennent en compte ces observations. De cette manière on pourra faire progresser le joueur vers une victoire plus sûre et plus rapide.

Les règles sont :

- A chaque jetons adverses, les 8 cases adjacentes voient leur valeur augmentée de 1 (les cases adjacente à ces cases là gagnent +0,5).
- Si 2 jetons adverses sont alignés à une distance maximum de 2 cases, la ligne, colonne ou diagonale reliant ces 2 jetons voit ses valeurs augmentées de 3. (Dans le but de contrer une fourchette ou un alignement simple).
- Un alignement de 3 jetons sur une distance de 4 cases donne la priorité absolu au prochain coup de jouer entre ces jetons, ou à l'extrémité libre.

Avec ces règles intégrées au jeu, les valeurs du tableau changent à chaque jeton posé. De cette manière on peut voir un comportement presque naturel. Si l'homme est capable de prévoir 2 ou même 3 coups à l'avance (avec de l'entrainement), l'ordinateur peut aisément en prévoir 3, 4 ou même 5.

Pour se faire avec cette théorie il suffirait au joueur de prévoir 2 coups, en faire la moyenne des valeurs, et comparée à d'autre coups prévus à l'avance. De cette manière on peut aussi faire varier le niveau de complexité de jeu d'un ordinateur en lui demandant de faire la moyenne de 3 ou 4 coups. Si l'on utilise la seconde possibilité de calcul, l'ordinateur devrait prendre la somme des valeurs des 2 joueurs pendant ces 3 coups et en faire la différence pour voir s'il en sort gagnant ou perdant.

V A.P.I. (Artificial Power Intelligent) :

La suite logique de notre sujet, fut de trouver les différentes stratégies de jeu, permettant de gagner. Nous nous sommes très rapidement orienté vers la création d'une Intelligence Artificielle, capable de jouer et de gagner au puissance 4.

La première étape fut de créer un programme faisant tourner le puissance 4 pour deux joueurs humains. Nous avions dès lors dans l'idée de créer un programme jouant à la place d'une des deux joueurs. Le code de ce programme est disponible dans les annexes.

Nous avons utilisé les différentes « fonctions » que nous avons précédemment développé pour la vérification des matrices, la définition du tour de jeu, ainsi que l'utilisation des matrices. Par contre, toute la partie opérateurs mathématiques n'est pas indispensable en informatique, même si les fonctions programmées ressemblent aux fonctions mathématiques.

Pour les vérifications, nous utilisons la position du dernier jeton joué afin d'éviter une récurrence dans la vérification. En effet, vérifier les cases avoisinantes la case du dernier jeton suffit largement à définir si l'on a gagné ou non.

Lors de nos recherches sur les stratégies, il nous est apparu plusieurs méthodes pour envisager une stratégie gagnante. Nous avons choisi d'en traiter deux : en utilisant une bibliothèque de fourchettes et en utilisant la méthode des poids.

V.a Avec les fourchettes :

Pour commencer, nous nous sommes fixer plusieurs règles :

- L'ordinateur jouera toujours en premier,
- Il commencera toujours par jouer au milieu,
- Le joueur humain ne se fera jamais avoir par un alignement simple.

Définition des fourchettes :

Dans une partie de *Puissance 4*, il existe ce que l'on appelle une fourchette. Il s'agit d'un cas de jeu ou le joueur piégé sera obligé de jouer dans une position le faisant perdre. Le joueur mettant en place la fourchette a fait naître deux positions lui permettant de gagner.

Exemple :

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \end{pmatrix}$$

Les deux positions notées en rouge permettent au joueur 1 de gagner. Le joueurs 2 n'a aucune chance en un seul tour d'empêcher le joueur 1 de gagner. Ceci est la fourchette la plus simple qu'il existe.

Autre exemple :

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 & 1 & 0 & 1 \end{pmatrix}$$

Si le joueur 2 ne joue pas à la position rouge, il perd et s'il joue à la position rouge, le joueur 1 peut jouer à la position verte et gagner. Il s'agit là encore d'une fourchette, bien que plus complexe.

Obtention des fourchettes :

Nous n'avons pas encore trouvé d'algorithme permettant de définir toutes les fourchettes. Nous en avons trouvé une dizaine de schémas de fourchettes possible soit environ 240 positions possible pour tout ces schémas. Si nous prenons en compte les variations possibles dans la partie de la matrice qui n'est pas concerné par la fourchette, le nombre de jeu possible est très important. Nous estimons qu'il nous manque environ un quart des schémas.

Pour découvrir les coups permettant d'arriver à ces différentes fourchettes, nous avons mis au point un algorithme partant du coup final. Pour arriver aux différents trajets pour finir par ce coup gagnant. Il faudrait parvenir à le programmer pour accélérer les calculs. Une fois cette algorithme trouvé et programmé, il suffirait de le coupler à un programmer définissant toutes les fourchettes et nous aurions alors notre programme obtenant les fourchettes ainsi que les trajets pour y arriver.

Ces trajets seront donc formater en arbres, avec à chaque extrémité le début et la fin. Nous supprimerons alors toutes les branches de l'arbre ne menant à la situation initiale, qui est le joueur 1 (ordinateur) joue en premier et au milieu.

Exemple :

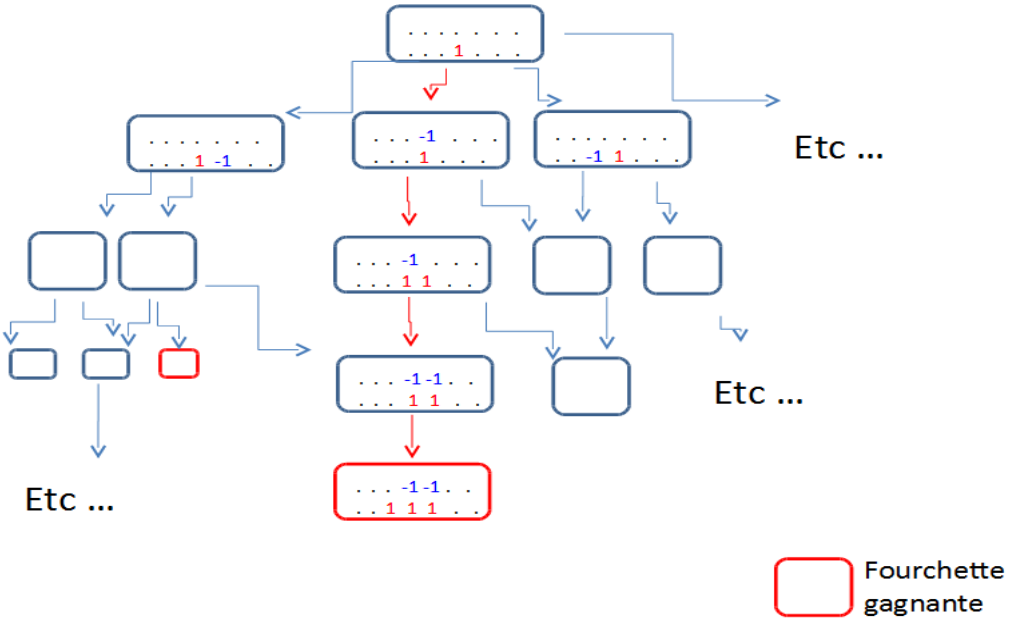
Voir Annexe 2.

Création de l'arbre final :

Une fois chacun de ces arbres créée (beaucoup d'arbres différents), non entrons ces arbres dans un ordinateur sous forme d'arbres composés de fils, où chaque nœud sera un coup possible dans l'arbre. Nous demanderons ensuite à l'ordinateur de supprimer les récurrences, ainsi plusieurs nœuds pourront avoir le même fils.

Donc, par exemple, le premier nœud aura forcément 7 fils, car le joueur 2 a toujours le choix de jouer à n'importe quelle colonne. Puis au fur et à mesure de la partie, les nœuds auront plus ou moins de fils car le joueur 1 réduira ses possibilités à celles utiles.

Schéma explicatif :



V.b Avec les poids :

Nous avons commencé à programmer, l'intelligence artificielle utilisant la méthode des poids avec les différentes règles. Nous n'avons pas réussi à le finir, en effet, la complexité des règles est d'un niveau bien trop élevé. Nous essayerons de le finir pour pouvoir le présenter à l'oral.

Nous avons tout de même créée un programme permettant de jouer à un joueur contre un joueur, nous avons aussi programmé la règle (des poids) disant que à chaque jeton joué par l'adversaire, les cases adjacentes sont incrémentées de 1.

Le code source est disponible, pour le programme complet gérant le puissance 4 en annexe 4. Et les fonctions gérant la première règle des poids en annexe 5.

V.c Conclusion concernant l'API :

Au cours de cette année, nous n'avons que peu utilisé la recherche par internet pour ne pas être influencé par les idées déjà présente sur le net. Cependant, à la fin de l'année, nous avons trouvé sur la toile des personnes ayant créée des IA pour gagner au puissance 4, comme par exemple Patrick Sterlin ou d'autres personnes ayant crée les programmes : Velena, Vianiato, Gladiator et d'autres encore (gratuits sur internet).

Pour la plupart, ils ont imaginé une autre conception de cette intelligence artificielle, ils ont utilisé l'algorithme dit de « min-max ».

L'algorithme min/max fonctionne suivant le principe suivant : on commence par construire un arbre des possibilités à partir de la situation courante. Chaque branche représente un coup (l'ordre des branches correspond à l'ordre des colonnes où l'on joue). L'idéal serait de construire l'arbre jusqu'à ce que la grille soit complètement remplie mais pour des raisons de taille mémoire et de temps de calcul, on limite la profondeur, ce qui donne une « vision » de quelques coups d'avance à l'IA. Cette profondeur est directement liée au niveau de l'IA choisie par l'utilisateur au début de

chaque partie (novice, moyen, expert).

Ensuite, on évalue toutes les feuilles de l'arbre en fonction du nombre et de la taille des alignements de pions de l'ordinateur et de son adversaire.

La fonction qui réalise cette construction, par récursivité, est appelée « Possibilite ».

Puis, on évalue tous les noeuds en remontant vers la racine : si la branche correspond à un tour où c'est à l'ordinateur de jouer, on choisit pour un père le maximum des évaluations de ses fils, et si la branche correspond à un tour où c'est à son adversaire, on choisit le minimum (on suppose ainsi que l'adversaire utilise la même stratégie que l'IA). La fonction d'évaluation exacte choisie permet de rendre compte, à la fois de la situation de l'ordinateur et de son adversaire. Il s'agit de :

- Si l'ordinateur gagne : **Coeff4**

- Si l'ordinateur perd : **- Coeff4**

- Sinon :

(nbre d'alignements de 3 pions du joueur - nbre d'alignements de 3 pions de l'adversaire)*coeff3
+ (nbre d'alignements de 2 pions du joueur - nbre d'alignements de 2 pions de l'adversaire)*coeff2

Coeff4 = 10 000
Coeff3 = 1 000
Coeff2 = 100
De cette façon, toute situation gagnante sera automatiquement choisie, et toute fonction « suicidaire » sera éliminée.
Seuls les alignements libres (qui laissent la possibilités d'être complétés) sont comptabilisés.

Citation de : *http://patrick.sterlin.free.fr/iamr.php#2*

Nous avons aussi constaté qu'il existait des concours d'informatique sur la programmation d'une telle IA. Il s'agit d'un concours se basant sur la simplicité du code, de la vitesse de gestion du jeu et de la fréquence de victoire. Cela représente un réel défi pour nous et nous essayerons de le finir pour l'oral voire même après si le temps nous fait défaut, pourquoi pas pour math en Jeans 2 ?

CONCLUSION :

L'année semble longue au début, mais finalement, il nous a manqué du temps. Nous avons énormément découvert, à la fois sur le puissance 4 et ses mathématiques et aussi sur la recherche mathématique dans son ensemble.

Il nous resterait à trouver comment extraire convenable les sous-matrices, une formule générale qui nous permettait d'extraire (quelque soit la taille) une sous-matrice de la matrice de mère. Puis inventer une fonction permettant la vérification de la diagonale d'une sous-matrice sans avoir à la sortir de la matrice mère.

Il faudrait que nous parvenions à trouver une logique à la suite des bases 3 et une méthode d'exploitation de cette suite. Il serait intéressant de programmer ou de concevoir une troisième IA basé sur les découverte des suite en base 3.

Il faudrait accroitre et développer les règles sur le poids afin que la victoire soit systématique. Cela nous permettrait de finir plus facilement la programmation de notre IA. Et enfin, si nous y parvenons à la confronter à d'autres IA disponibles sur le net et à notre méthode sur les bibliothèques de fourchettes.

Ce sujet est passionnant et bien que nous l'ayons pris de cette façon, nous aurions tout aussi put rechercher comment adapter le puissance 4 pour 3 joueurs, ou même plus. Nous aurions put tenter d'étudier le puissance 4 et ses mathématiques en trois dimensions, voire plus. Nous aurions put envisager le puissance 4 en changeant ses règles. Pourquoi pas devoir faire un L pour gagner ?

ANNEXES :

Annexes 1 : Les possibilités en base 3.

0000000 : 0	0001222 : 53	0010221 : 106	0012220 : 159	0021212 : 212
0000001 : 1	0002000 : 54	0010222 : 107	0012221 : 160	0021220 : 213
0000002 : 2	0002001 : 55	0011000 : 108	0012222 : 161	0021221 : 214
0000010 : 3	0002002 : 56	0011001 : 109	0020000 : 162	0021222 : 215
0000011 : 4	0002010 : 57	0011002 : 110	0020001 : 163	0022000 : 216
0000012 : 5	0002011 : 58	0011010 : 111	0020002 : 164	0022001 : 217
0000020 : 6	0002012 : 59	0011011 : 112	0020010 : 165	0022002 : 218
0000021 : 7	0002020 : 60	0011012 : 113	0020011 : 166	0022010 : 219
0000022 : 8	0002021 : 61	0011020 : 114	0020012 : 167	0022011 : 220
0000100 : 9	0002022 : 62	0011021 : 115	0020020 : 168	0022012 : 221
0000101 : 10	0002100 : 63	0011022 : 116	0020021 : 169	0022020 : 222
0000102 : 11	0002101 : 64	0011100 : 117	0020022 : 170	0022021 : 223
0000110 : 12	0002102 : 65	0011101 : 118	0020100 : 171	0022022 : 224
0000111 : 13	0002110 : 66	0011102 : 119	0020101 : 172	0022100 : 225
0000112 : 14	0002111 : 67	0011110 : 120	0020102 : 173	0022101 : 226
0000120 : 15	0002112 : 68	0011111 : 121	0020110 : 174	0022102 : 227
0000121 : 16	0002120 : 69	0011112 : 122	0020111 : 175	0022110 : 228
0000122 : 17	0002121 : 70	0011120 : 123	0020112 : 176	0022111 : 229
0000200 : 18	0002122 : 71	0011121 : 124	0020120 : 177	0022112 : 230
0000201 : 19	0002200 : 72	0011122 : 125	0020121 : 178	0022120 : 231
0000202 : 20	0002201 : 73	0011200 : 126	0020122 : 179	0022121 : 232
0000210 : 21	0002202 : 74	0011201 : 127	0020200 : 180	0022122 : 233
0000211 : 22	0002210 : 75	0011202 : 128	0020201 : 181	0022200 : 234
0000212 : 23	0002211 : 76	0011210 : 129	0020202 : 182	0022201 : 235
0000220 : 24	0002212 : 77	0011211 : 130	0020210 : 183	0022202 : 236
0000221 : 25	0002220 : 78	0011212 : 131	0020211 : 184	0022210 : 237
0000222 : 26	0002221 : 79	0011220 : 132	0020212 : 185	0022211 : 238
0001000 : 27	0002222 : 80	0011221 : 133	0020220 : 186	0022212 : 239
0001001 : 28	0010000 : 81	0011222 : 134	0020221 : 187	0022220 : 240
0001002 : 29	0010001 : 82	0012000 : 135	0020222 : 188	0022221 : 241
0001010 : 30	0010002 : 83	0012001 : 136	0021000 : 189	0022222 : 242
0001011 : 31	0010010 : 84	0012002 : 137	0021001 : 190	0100000 : 243
0001012 : 32	0010011 : 85	0012010 : 138	0021002 : 191	0100001 : 244
0001020 : 33	0010012 : 86	0012011 : 139	0021010 : 192	0100002 : 245
0001021 : 34	0010020 : 87	0012012 : 140	0021011 : 193	0100010 : 246
0001022 : 35	0010021 : 88	0012020 : 141	0021012 : 194	0100011 : 247
0001100 : 36	0010022 : 89	0012021 : 142	0021020 : 195	0100012 : 248
0001101 : 37	0010100 : 90	0012022 : 143	0021021 : 196	0100020 : 249
0001102 : 38	0010101 : 91	0012100 : 144	0021022 : 197	0100021 : 250
0001110 : 39	0010102 : 92	0012101 : 145	0021100 : 198	0100022 : 251
0001111 : 40	0010110 : 93	0012102 : 146	0021101 : 199	0100100 : 252
0001112 : 41	0010111 : 94	0012110 : 147	0021102 : 200	0100101 : 253
0001120 : 42	0010112 : 95	0012111 : 148	0021110 : 201	0100102 : 254
0001121 : 43	0010120 : 96	0012112 : 149	0021111 : 202	0100110 : 255
0001122 : 44	0010121 : 97	0012120 : 150	0021112 : 203	0100111 : 256
0001200 : 45	0010122 : 98	0012121 : 151	0021120 : 204	0100112 : 257
0001201 : 46	0010200 : 99	0012122 : 152	0021121 : 205	0100120 : 258
0001202 : 47	0010201 : 100	0012200 : 153	0021122 : 206	0100121 : 259
0001210 : 48	0010202 : 101	0012201 : 154	0021200 : 207	0100122 : 260
0001211 : 49	0010210 : 102	0012202 : 155	0021201 : 208	0100200 : 261
0001212 : 50	0010211 : 103	0012210 : 156	0021202 : 209	0100201 : 262
0001220 : 51	0010212 : 104	0012211 : 157	0021210 : 210	0100202 : 263
0001221 : 52	0010220 : 105	0012212 : 158	0021211 : 211	0100210 : 264

0100211 : 265
0100212 : 266
0100220 : 267
0100221 : 268
0100222 : 269
0101000 : 270
0101001 : 271
0101002 : 272
0101010 : 273
0101011 : 274
0101012 : 275
0101020 : 276
0101021 : 277
0101022 : 278
0101100 : 279
0101101 : 280
0101102 : 281
0101110 : 282
0101111 : 283
0101112 : 284
0101120 : 285
0101121 : 286
0101122 : 287
0101200 : 288
0101201 : 289
0101202 : 290
0101210 : 291
0101211 : 292
0101212 : 293
0101220 : 294
0101221 : 295
0101222 : 296
0102000 : 297
0102001 : 298
0102002 : 299
0102010 : 300
0102011 : 301
0102012 : 302
0102020 : 303
0102021 : 304
0102022 : 305
0102100 : 306
0102101 : 307
0102102 : 308
0102110 : 309
0102111 : 310
0102112 : 311
0102120 : 312
0102121 : 313
0102122 : 314
0102200 : 315
0102201 : 316
0102202 : 317
0102210 : 318
0102211 : 319
0102212 : 320
0102220 : 321
0102221 : 322
0102222 : 323
0110000 : 324
0110001 : 325
0110002 : 326
0110010 : 327
0110011 : 328
0110012 : 329
0110020 : 330
0110021 : 331
0110022 : 332
0110100 : 333
0110101 : 334
0110102 : 335
0110110 : 336
0110111 : 337
0110112 : 338
0110120 : 339
0110121 : 340
0110122 : 341
0110200 : 342
0110201 : 343
0110202 : 344
0110210 : 345
0110211 : 346
0110212 : 347
0110220 : 348
0110221 : 349
0110222 : 350
0111000 : 351
0111001 : 352
0111002 : 353
0111010 : 354
0111011 : 355
0111012 : 356
0111020 : 357
0111021 : 358
0111022 : 359
0111100 : 360
0111101 : 361
0111102 : 362
0111110 : 363
0111111 : 364
0111112 : 365
0111120 : 366
0111121 : 367
0111122 : 368
0111200 : 369
0111201 : 370
0111202 : 371
0111210 : 372
0111211 : 373
0111212 : 374
0111220 : 375
0111221 : 376
0111222 : 377
0112000 : 378
0112001 : 379
0112002 : 380
0112010 : 381
0112011 : 382
0112012 : 383
0112020 : 384
0112021 : 385
0112022 : 386
0112100 : 387
0112101 : 388
0112102 : 389
0112110 : 390
0112111 : 391
0112112 : 392
0112120 : 393
0112121 : 394
0112122 : 395
0112200 : 396
0112201 : 397
0112202 : 398
0112210 : 399
0112211 : 400
0112212 : 401
0112220 : 402
0112221 : 403
0112222 : 404
0120000 : 405
0120001 : 406
0120002 : 407
0120010 : 408
0120011 : 409
0120012 : 410
0120020 : 411
0120021 : 412
0120022 : 413
0120100 : 414
0120101 : 415
0120102 : 416
0120110 : 417
0120111 : 418
0120112 : 419
0120120 : 420
0120121 : 421
0120122 : 422
0120200 : 423
0120201 : 424
0120202 : 425
0120210 : 426
0120211 : 427
0120212 : 428
0120220 : 429
0120221 : 430
0120222 : 431
0121000 : 432
0121001 : 433
0121002 : 434
0121010 : 435
0121011 : 436
0121012 : 437
0121020 : 438
0121021 : 439
0121022 : 440
0121100 : 441
0121101 : 442
0121102 : 443
0121110 : 444
0121111 : 445
0121112 : 446
0121120 : 447
0121121 : 448
0121122 : 449
0121200 : 450
0121201 : 451
0121202 : 452
0121210 : 453
0121211 : 454
0121212 : 455
0121220 : 456
0121221 : 457
0121222 : 458
0122000 : 459
0122001 : 460
0122002 : 461
0122010 : 462
0122011 : 463
0122012 : 464
0122020 : 465
0122021 : 466
0122022 : 467
0122100 : 468
0122101 : 469
0122102 : 470
0122110 : 471
0122111 : 472
0122112 : 473
0122120 : 474
0122121 : 475
0122122 : 476
0122200 : 477
0122201 : 478
0122202 : 479
0122210 : 480
0122211 : 481
0122212 : 482
0122220 : 483
0122221 : 484
0122222 : 485
0200000 : 486
0200001 : 487
0200002 : 488
0200010 : 489
0200011 : 490
0200012 : 491
0200020 : 492
0200021 : 493
0200022 : 494
0200100 : 495
0200101 : 496
0200102 : 497
0200110 : 498
0200111 : 499
0200112 : 500
0200120 : 501
0200121 : 502
0200122 : 503
0200200 : 504
0200201 : 505
0200202 : 506
0200210 : 507
0200211 : 508
0200212 : 509
0200220 : 510
0200221 : 511
0200222 : 512
0201000 : 513
0201001 : 514
0201002 : 515
0201010 : 516
0201011 : 517
0201012 : 518
0201020 : 519
0201021 : 520
0201022 : 521
0201100 : 522
0201101 : 523
0201102 : 524
0201110 : 525
0201111 : 526
0201112 : 527
0201120 : 528
0201121 : 529
0201122 : 530
0201200 : 531
0201201 : 532
0201202 : 533
0201210 : 534
0201211 : 535
0201212 : 536
0201220 : 537
0201221 : 538
0201222 : 539
0202000 : 540
0202001 : 541
0202002 : 542
0202010 : 543
0202011 : 544
0202012 : 545
0202020 : 546
0202021 : 547
0202022 : 548
0202100 : 549
0202101 : 550
0202102 : 551
0202110 : 552
0202111 : 553
0202112 : 554
0202120 : 555
0202121 : 556
0202122 : 557
0202200 : 558
0202201 : 559
0202202 : 560
0202210 : 561
0202211 : 562
0202212 : 563
0202220 : 564

0202221 : 565
0202222 : 566
0210000 : 567
0210001 : 568
0210002 : 569
0210010 : 570
0210011 : 571
0210012 : 572
0210020 : 573
0210021 : 574
0210022 : 575
0210100 : 576
0210101 : 577
0210102 : 578
0210110 : 579
0210111 : 580
0210112 : 581
0210120 : 582
0210121 : 583
0210122 : 584
0210200 : 585
0210201 : 586
0210202 : 587
0210210 : 588
0210211 : 589
0210212 : 590
0210220 : 591
0210221 : 592
0210222 : 593
0211000 : 594
0211001 : 595
0211002 : 596
0211010 : 597
0211011 : 598
0211012 : 599
0211020 : 600
0211021 : 601
0211022 : 602
0211100 : 603
0211101 : 604
0211102 : 605
0211110 : 606
0211111 : 607
0211112 : 608
0211120 : 609
0211121 : 610
0211122 : 611
0211200 : 612
0211201 : 613
0211202 : 614
0211210 : 615
0211211 : 616
0211212 : 617
0211220 : 618
0211221 : 619
0211222 : 620
0212000 : 621
0212001 : 622
0212002 : 623
0212010 : 624
0212011 : 625
0212012 : 626
0212020 : 627
0212021 : 628
0212022 : 629
0212100 : 630
0212101 : 631
0212102 : 632
0212110 : 633
0212111 : 634
0212112 : 635
0212120 : 636
0212121 : 637
0212122 : 638
0212200 : 639
0212201 : 640
0212202 : 641
0212210 : 642
0212211 : 643
0212212 : 644
0212220 : 645
0212221 : 646
0212222 : 647
0220000 : 648
0220001 : 649
0220002 : 650
0220010 : 651
0220011 : 652
0220012 : 653
0220020 : 654
0220021 : 655
0220022 : 656
0220100 : 657
0220101 : 658
0220102 : 659
0220110 : 660
0220111 : 661
0220112 : 662
0220120 : 663
0220121 : 664
0220122 : 665
0220200 : 666
0220201 : 667
0220202 : 668
0220210 : 669
0220211 : 670
0220212 : 671
0220220 : 672
0220221 : 673
0220222 : 674
0221000 : 675
0221001 : 676
0221002 : 677
0221010 : 678
0221011 : 679
0221012 : 680
0221020 : 681
0221021 : 682
0221022 : 683
0221100 : 684
0221101 : 685
0221102 : 686
0221110 : 687
0221111 : 688
0221112 : 689
0221120 : 690
0221121 : 691
0221122 : 692
0221200 : 693
0221201 : 694
0221202 : 695
0221210 : 696
0221211 : 697
0221212 : 698
0221220 : 699
0221221 : 700
0221222 : 701
0222000 : 702
0222001 : 703
0222002 : 704
0222010 : 705
0222011 : 706
0222012 : 707
0222020 : 708
0222021 : 709
0222022 : 710
0222100 : 711
0222101 : 712
0222102 : 713
0222110 : 714
0222111 : 715
0222112 : 716
0222120 : 717
0222121 : 718
0222122 : 719
0222200 : 720
0222201 : 721
0222202 : 722
0222210 : 723
0222211 : 724
0222212 : 725
0222220 : 726
0222221 : 727
0222222 : 728
1000000 : 729
1000001 : 730
1000002 : 731
1000010 : 732
1000011 : 733
1000012 : 734
1000020 : 735
1000021 : 736
1000022 : 737
1000100 : 738
1000101 : 739
1000102 : 740
1000110 : 741
1000111 : 742
1000112 : 743
1000120 : 744
1000121 : 745
1000122 : 746
1000200 : 747
1000201 : 748
1000202 : 749
1000210 : 750
1000211 : 751
1000212 : 752
1000220 : 753
1000221 : 754
1000222 : 755
1001000 : 756
1001001 : 757
1001002 : 758
1001010 : 759
1001011 : 760
1001012 : 761
1001020 : 762
1001021 : 763
1001022 : 764
1001100 : 765
1001101 : 766
1001102 : 767
1001110 : 768
1001111 : 769
1001112 : 770
1001120 : 771
1001121 : 772
1001122 : 773
1001200 : 774
1001201 : 775
1001202 : 776
1001210 : 777
1001211 : 778
1001212 : 779
1001220 : 780
1001221 : 781
1001222 : 782
1002000 : 783
1002001 : 784
1002002 : 785
1002010 : 786
1002011 : 787
1002012 : 788
1002020 : 789
1002021 : 790
1002022 : 791
1002100 : 792
1002101 : 793
1002102 : 794
1002110 : 795
1002111 : 796
1002112 : 797
1002120 : 798
1002121 : 799
1002122 : 800
1002200 : 801
1002201 : 802
1002202 : 803
1002210 : 804
1002211 : 805
1002212 : 806
1002220 : 807
1002221 : 808
1002222 : 809
1010000 : 810
1010001 : 811
1010002 : 812
1010010 : 813
1010011 : 814
1010012 : 815
1010020 : 816
1010021 : 817
1010022 : 818
1010100 : 819
1010101 : 820
1010102 : 821
1010110 : 822
1010111 : 823
1010112 : 824
1010120 : 825
1010121 : 826
1010122 : 827
1010200 : 828
1010201 : 829
1010202 : 830
1010210 : 831
1010211 : 832
1010212 : 833
1010220 : 834
1010221 : 835
1010222 : 836
1011000 : 837
1011001 : 838
1011002 : 839
1011010 : 840
1011011 : 841
1011012 : 842
1011020 : 843
1011021 : 844
1011022 : 845
1011100 : 846
1011101 : 847
1011102 : 848
1011110 : 849
1011111 : 850
1011112 : 851
1011120 : 852
1011121 : 853
1011122 : 854
1011200 : 855
1011201 : 856
1011202 : 857
1011210 : 858
1011211 : 859
1011212 : 860
1011220 : 861
1011221 : 862
1011222 : 863
1012000 : 864

1012001 : 865
1012002 : 866
1012010 : 867
1012011 : 868
1012012 : 869
1012020 : 870
1012021 : 871
1012022 : 872
1012100 : 873
1012101 : 874
1012102 : 875
1012110 : 876
1012111 : 877
1012112 : 878
1012120 : 879
1012121 : 880
1012122 : 881
1012200 : 882
1012201 : 883
1012202 : 884
1012210 : 885
1012211 : 886
1012212 : 887
1012220 : 888
1012221 : 889
1012222 : 890
1020000 : 891
1020001 : 892
1020002 : 893
1020010 : 894
1020011 : 895
1020012 : 896
1020020 : 897
1020021 : 898
1020022 : 899
1020100 : 900
1020101 : 901
1020102 : 902
1020110 : 903
1020111 : 904
1020112 : 905
1020120 : 906
1020121 : 907
1020122 : 908
1020200 : 909
1020201 : 910
1020202 : 911
1020210 : 912
1020211 : 913
1020212 : 914
1020220 : 915
1020221 : 916
1020222 : 917
1021000 : 918
1021001 : 919
1021002 : 920
1021010 : 921
1021011 : 922
1021012 : 923
1021020 : 924
1021021 : 925
1021022 : 926
1021100 : 927
1021101 : 928
1021102 : 929
1021110 : 930
1021111 : 931
1021112 : 932
1021120 : 933
1021121 : 934
1021122 : 935
1021200 : 936
1021201 : 937
1021202 : 938
1021210 : 939
1021211 : 940
1021212 : 941
1021220 : 942
1021221 : 943
1021222 : 944
1022000 : 945
1022001 : 946
1022002 : 947
1022010 : 948
1022011 : 949
1022012 : 950
1022020 : 951
1022021 : 952
1022022 : 953
1022100 : 954
1022101 : 955
1022102 : 956
1022110 : 957
1022111 : 958
1022112 : 959
1022120 : 960
1022121 : 961
1022122 : 962
1022200 : 963
1022201 : 964
1022202 : 965
1022210 : 966
1022211 : 967
1022212 : 968
1022220 : 969
1022221 : 970
1022222 : 971
1100000 : 972
1100001 : 973
1100002 : 974
1100010 : 975
1100011 : 976
1100012 : 977
1100020 : 978
1100021 : 979
1100022 : 980
1100100 : 981
1100101 : 982
1100102 : 983
1100110 : 984
1100111 : 985
1100112 : 986
1100120 : 987
1100121 : 988
1100122 : 989
1100200 : 990
1100201 : 991
1100202 : 992
1100210 : 993
1100211 : 994
1100212 : 995
1100220 : 996
1100221 : 997
1100222 : 998
1101000 : 999
1101001 : 1000
1101002 : 1001
1101010 : 1002
1101011 : 1003
1101012 : 1004
1101020 : 1005
1101021 : 1006
1101022 : 1007
1101100 : 1008
1101101 : 1009
1101102 : 1010
1101110 : 1011
1101111 : 1012
1101112 : 1013
1101120 : 1014
1101121 : 1015
1101122 : 1016
1101200 : 1017
1101201 : 1018
1101202 : 1019
1101210 : 1020
1101211 : 1021
1101212 : 1022
1101220 : 1023
1101221 : 1024
1101222 : 1025
1102000 : 1026
1102001 : 1027
1102002 : 1028
1102010 : 1029
1102011 : 1030
1102012 : 1031
1102020 : 1032
1102021 : 1033
1102022 : 1034
1102100 : 1035
1102101 : 1036
1102102 : 1037
1102110 : 1038
1102111 : 1039
1102112 : 1040
1102120 : 1041
1102121 : 1042
1102122 : 1043
1102200 : 1044
1102201 : 1045
1102202 : 1046
1102210 : 1047
1102211 : 1048
1102212 : 1049
1102220 : 1050
1102221 : 1051
1102222 : 1052
1110000 : 1053
1110001 : 1054
1110002 : 1055
1110010 : 1056
1110011 : 1057
1110012 : 1058
1110020 : 1059
1110021 : 1060
1110022 : 1061
1110100 : 1062
1110101 : 1063
1110102 : 1064
1110110 : 1065
1110111 : 1066
1110112 : 1067
1110120 : 1068
1110121 : 1069
1110122 : 1070
1110200 : 1071
1110201 : 1072
1110202 : 1073
1110210 : 1074
1110211 : 1075
1110212 : 1076
1110220 : 1077
1110221 : 1078
1110222 : 1079
1111000 : 1080
1111001 : 1081
1111002 : 1082
1111010 : 1083
1111011 : 1084
1111012 : 1085
1111020 : 1086
1111021 : 1087
1111022 : 1088
1111100 : 1089
1111101 : 1090
1111102 : 1091
1111110 : 1092
1111111 : 1093
1111112 : 1094
1111120 : 1095
1111121 : 1096
1111122 : 1097
1111200 : 1098
1111201 : 1099
1111202 : 1100
1111210 : 1101
1111211 : 1102
1111212 : 1103
1111220 : 1104
1111221 : 1105
1111222 : 1106
1112000 : 1107
1112001 : 1108
1112002 : 1109
1112010 : 1110
1112011 : 1111
1112012 : 1112
1112020 : 1113
1112021 : 1114
1112022 : 1115
1112100 : 1116
1112101 : 1117
1112102 : 1118
1112110 : 1119
1112111 : 1120
1112112 : 1121
1112120 : 1122
1112121 : 1123
1112122 : 1124
1112200 : 1125
1112201 : 1126
1112202 : 1127
1112210 : 1128
1112211 : 1129
1112212 : 1130
1112220 : 1131
1112221 : 1132
1112222 : 1133
1120000 : 1134
1120001 : 1135
1120002 : 1136
1120010 : 1137
1120011 : 1138
1120012 : 1139
1120020 : 1140
1120021 : 1141
1120022 : 1142
1120100 : 1143
1120101 : 1144
1120102 : 1145
1120110 : 1146
1120111 : 1147
1120112 : 1148
1120120 : 1149
1120121 : 1150
1120122 : 1151
1120200 : 1152
1120201 : 1153
1120202 : 1154
1120210 : 1155
1120211 : 1156
1120212 : 1157
1120220 : 1158
1120221 : 1159
1120222 : 1160
1121000 : 1161
1121001 : 1162
1121002 : 1163
1121010 : 1164

1121011 : 1165
1121012 : 1166
1121020 : 1167
1121021 : 1168
1121022 : 1169
1121100 : 1170
1121101 : 1171
1121102 : 1172
1121110 : 1173
1121111 : 1174
1121112 : 1175
1121120 : 1176
1121121 : 1177
1121122 : 1178
1121200 : 1179
1121201 : 1180
1121202 : 1181
1121210 : 1182
1121211 : 1183
1121212 : 1184
1121220 : 1185
1121221 : 1186
1121222 : 1187
1122000 : 1188
1122001 : 1189
1122002 : 1190
1122010 : 1191
1122011 : 1192
1122012 : 1193
1122020 : 1194
1122021 : 1195
1122022 : 1196
1122100 : 1197
1122101 : 1198
1122102 : 1199
1122110 : 1200
1122111 : 1201
1122112 : 1202
1122120 : 1203
1122121 : 1204
1122122 : 1205
1122200 : 1206
1122201 : 1207
1122202 : 1208
1122210 : 1209
1122211 : 1210
1122212 : 1211
1122220 : 1212
1122221 : 1213
1122222 : 1214
1200000 : 1215
1200001 : 1216
1200002 : 1217
1200010 : 1218
1200011 : 1219
1200012 : 1220
1200020 : 1221
1200021 : 1222
1200022 : 1223
1200100 : 1224
1200101 : 1225
1200102 : 1226
1200110 : 1227
1200111 : 1228
1200112 : 1229
1200120 : 1230
1200121 : 1231
1200122 : 1232
1200200 : 1233
1200201 : 1234
1200202 : 1235
1200210 : 1236
1200211 : 1237
1200212 : 1238
1200220 : 1239
1200221 : 1240
1200222 : 1241
1201000 : 1242
1201001 : 1243
1201002 : 1244
1201010 : 1245
1201011 : 1246
1201012 : 1247
1201020 : 1248
1201021 : 1249
1201022 : 1250
1201100 : 1251
1201101 : 1252
1201102 : 1253
1201110 : 1254
1201111 : 1255
1201112 : 1256
1201120 : 1257
1201121 : 1258
1201122 : 1259
1201200 : 1260
1201201 : 1261
1201202 : 1262
1201210 : 1263
1201211 : 1264
1201212 : 1265
1201220 : 1266
1201221 : 1267
1201222 : 1268
1202000 : 1269
1202001 : 1270
1202002 : 1271
1202010 : 1272
1202011 : 1273
1202012 : 1274
1202020 : 1275
1202021 : 1276
1202022 : 1277
1202100 : 1278
1202101 : 1279
1202102 : 1280
1202110 : 1281
1202111 : 1282
1202112 : 1283
1202120 : 1284
1202121 : 1285
1202122 : 1286
1202200 : 1287
1202201 : 1288
1202202 : 1289
1202210 : 1290
1202211 : 1291
1202212 : 1292
1202220 : 1293
1202221 : 1294
1202222 : 1295
1210000 : 1296
1210001 : 1297
1210002 : 1298
1210010 : 1299
1210011 : 1300
1210012 : 1301
1210020 : 1302
1210021 : 1303
1210022 : 1304
1210100 : 1305
1210101 : 1306
1210102 : 1307
1210110 : 1308
1210111 : 1309
1210112 : 1310
1210120 : 1311
1210121 : 1312
1210122 : 1313
1210200 : 1314
1210201 : 1315
1210202 : 1316
1210210 : 1317
1210211 : 1318
1210212 : 1319
1210220 : 1320
1210221 : 1321
1210222 : 1322
1211000 : 1323
1211001 : 1324
1211002 : 1325
1211010 : 1326
1211011 : 1327
1211012 : 1328
1211020 : 1329
1211021 : 1330
1211022 : 1331
1211100 : 1332
1211101 : 1333
1211102 : 1334
1211110 : 1335
1211111 : 1336
1211112 : 1337
1211120 : 1338
1211121 : 1339
1211122 : 1340
1211200 : 1341
1211201 : 1342
1211202 : 1343
1211210 : 1344
1211211 : 1345
1211212 : 1346
1211220 : 1347
1211221 : 1348
1211222 : 1349
1212000 : 1350
1212001 : 1351
1212002 : 1352
1212010 : 1353
1212011 : 1354
1212012 : 1355
1212020 : 1356
1212021 : 1357
1212022 : 1358
1212100 : 1359
1212101 : 1360
1212102 : 1361
1212110 : 1362
1212111 : 1363
1212112 : 1364
1212120 : 1365
1212121 : 1366
1212122 : 1367
1212200 : 1368
1212201 : 1369
1212202 : 1370
1212210 : 1371
1212211 : 1372
1212212 : 1373
1212220 : 1374
1212221 : 1375
1212222 : 1376
1220000 : 1377
1220001 : 1378
1220002 : 1379
1220010 : 1380
1220011 : 1381
1220012 : 1382
1220020 : 1383
1220021 : 1384
1220022 : 1385
1220100 : 1386
1220101 : 1387
1220102 : 1388
1220110 : 1389
1220111 : 1390
1220112 : 1391
1220120 : 1392
1220121 : 1393
1220122 : 1394
1220200 : 1395
1220201 : 1396
1220202 : 1397
1220210 : 1398
1220211 : 1399
1220212 : 1400
1220220 : 1401
1220221 : 1402
1220222 : 1403
1221000 : 1404
1221001 : 1405
1221002 : 1406
1221010 : 1407
1221011 : 1408
1221012 : 1409
1221020 : 1410
1221021 : 1411
1221022 : 1412
1221100 : 1413
1221101 : 1414
1221102 : 1415
1221110 : 1416
1221111 : 1417
1221112 : 1418
1221120 : 1419
1221121 : 1420
1221122 : 1421
1221200 : 1422
1221201 : 1423
1221202 : 1424
1221210 : 1425
1221211 : 1426
1221212 : 1427
1221220 : 1428
1221221 : 1429
1221222 : 1430
1222000 : 1431
1222001 : 1432
1222002 : 1433
1222010 : 1434
1222011 : 1435
1222012 : 1436
1222020 : 1437
1222021 : 1438
1222022 : 1439
1222100 : 1440
1222101 : 1441
1222102 : 1442
1222110 : 1443
1222111 : 1444
1222112 : 1445
1222120 : 1446
1222121 : 1447
1222122 : 1448
1222200 : 1449
1222201 : 1450
1222202 : 1451
1222210 : 1452
1222211 : 1453
1222212 : 1454
1222220 : 1455
1222221 : 1456
1222222 : 1457
2000000 : 1458
2000001 : 1459
2000002 : 1460
2000010 : 1461
2000011 : 1462
2000012 : 1463
2000020 : 1464

2000021 : 1465
2000022 : 1466
2000100 : 1467
2000101 : 1468
2000102 : 1469
2000110 : 1470
2000111 : 1471
2000112 : 1472
2000120 : 1473
2000121 : 1474
2000122 : 1475
2000200 : 1476
2000201 : 1477
2000202 : 1478
2000210 : 1479
2000211 : 1480
2000212 : 1481
2000220 : 1482
2000221 : 1483
2000222 : 1484
2001000 : 1485
2001001 : 1486
2001002 : 1487
2001010 : 1488
2001011 : 1489
2001012 : 1490
2001020 : 1491
2001021 : 1492
2001022 : 1493
2001100 : 1494
2001101 : 1495
2001102 : 1496
2001110 : 1497
2001111 : 1498
2001112 : 1499
2001120 : 1500
2001121 : 1501
2001122 : 1502
2001200 : 1503
2001201 : 1504
2001202 : 1505
2001210 : 1506
2001211 : 1507
2001212 : 1508
2001220 : 1509
2001221 : 1510
2001222 : 1511
2002000 : 1512
2002001 : 1513
2002002 : 1514
2002010 : 1515
2002011 : 1516
2002012 : 1517
2002020 : 1518
2002021 : 1519
2002022 : 1520
2002100 : 1521
2002101 : 1522
2002102 : 1523
2002110 : 1524
2002111 : 1525
2002112 : 1526
2002120 : 1527
2002121 : 1528
2002122 : 1529
2002200 : 1530
2002201 : 1531
2002202 : 1532
2002210 : 1533
2002211 : 1534
2002212 : 1535
2002220 : 1536
2002221 : 1537
2002222 : 1538
2010000 : 1539
2010001 : 1540
2010002 : 1541
2010010 : 1542
2010011 : 1543
2010012 : 1544
2010020 : 1545
2010021 : 1546
2010022 : 1547
2010100 : 1548
2010101 : 1549
2010102 : 1550
2010110 : 1551
2010111 : 1552
2010112 : 1553
2010120 : 1554
2010121 : 1555
2010122 : 1556
2010200 : 1557
2010201 : 1558
2010202 : 1559
2010210 : 1560
2010211 : 1561
2010212 : 1562
2010220 : 1563
2010221 : 1564
2010222 : 1565
2011000 : 1566
2011001 : 1567
2011002 : 1568
2011010 : 1569
2011011 : 1570
2011012 : 1571
2011020 : 1572
2011021 : 1573
2011022 : 1574
2011100 : 1575
2011101 : 1576
2011102 : 1577
2011110 : 1578
2011111 : 1579
2011112 : 1580
2011120 : 1581
2011121 : 1582
2011122 : 1583
2011200 : 1584
2011201 : 1585
2011202 : 1586
2011210 : 1587
2011211 : 1588
2011212 : 1589
2011220 : 1590
2011221 : 1591
2011222 : 1592
2012000 : 1593
2012001 : 1594
2012002 : 1595
2012010 : 1596
2012011 : 1597
2012012 : 1598
2012020 : 1599
2012021 : 1600
2012022 : 1601
2012100 : 1602
2012101 : 1603
2012102 : 1604
2012110 : 1605
2012111 : 1606
2012112 : 1607
2012120 : 1608
2012121 : 1609
2012122 : 1610
2012200 : 1611
2012201 : 1612
2012202 : 1613
2012210 : 1614
2012211 : 1615
2012212 : 1616
2012220 : 1617
2012221 : 1618
2012222 : 1619
2020000 : 1620
2020001 : 1621
2020002 : 1622
2020010 : 1623
2020011 : 1624
2020012 : 1625
2020020 : 1626
2020021 : 1627
2020022 : 1628
2020100 : 1629
2020101 : 1630
2020102 : 1631
2020110 : 1632
2020111 : 1633
2020112 : 1634
2020120 : 1635
2020121 : 1636
2020122 : 1637
2020200 : 1638
2020201 : 1639
2020202 : 1640
2020210 : 1641
2020211 : 1642
2020212 : 1643
2020220 : 1644
2020221 : 1645
2020222 : 1646
2021000 : 1647
2021001 : 1648
2021002 : 1649
2021010 : 1650
2021011 : 1651
2021012 : 1652
2021020 : 1653
2021021 : 1654
2021022 : 1655
2021100 : 1656
2021101 : 1657
2021102 : 1658
2021110 : 1659
2021111 : 1660
2021112 : 1661
2021120 : 1662
2021121 : 1663
2021122 : 1664
2021200 : 1665
2021201 : 1666
2021202 : 1667
2021210 : 1668
2021211 : 1669
2021212 : 1670
2021220 : 1671
2021221 : 1672
2021222 : 1673
2022000 : 1674
2022001 : 1675
2022002 : 1676
2022010 : 1677
2022011 : 1678
2022012 : 1679
2022020 : 1680
2022021 : 1681
2022022 : 1682
2022100 : 1683
2022101 : 1684
2022102 : 1685
2022110 : 1686
2022111 : 1687
2022112 : 1688
2022120 : 1689
2022121 : 1690
2022122 : 1691
2022200 : 1692
2022201 : 1693
2022202 : 1694
2022210 : 1695
2022211 : 1696
2022212 : 1697
2022220 : 1698
2022221 : 1699
2022222 : 1700
2100000 : 1701
2100001 : 1702
2100002 : 1703
2100010 : 1704
2100011 : 1705
2100012 : 1706
2100020 : 1707
2100021 : 1708
2100022 : 1709
2100100 : 1710
2100101 : 1711
2100102 : 1712
2100110 : 1713
2100111 : 1714
2100112 : 1715
2100120 : 1716
2100121 : 1717
2100122 : 1718
2100200 : 1719
2100201 : 1720
2100202 : 1721
2100210 : 1722
2100211 : 1723
2100212 : 1724
2100220 : 1725
2100221 : 1726
2100222 : 1727
2101000 : 1728
2101001 : 1729
2101002 : 1730
2101010 : 1731
2101011 : 1732
2101012 : 1733
2101020 : 1734
2101021 : 1735
2101022 : 1736
2101100 : 1737
2101101 : 1738
2101102 : 1739
2101110 : 1740
2101111 : 1741
2101112 : 1742
2101120 : 1743
2101121 : 1744
2101122 : 1745
2101200 : 1746
2101201 : 1747
2101202 : 1748
2101210 : 1749
2101211 : 1750
2101212 : 1751
2101220 : 1752
2101221 : 1753
2101222 : 1754
2102000 : 1755
2102001 : 1756
2102002 : 1757
2102010 : 1758
2102011 : 1759
2102012 : 1760
2102020 : 1761
2102021 : 1762
2102022 : 1763
2102100 : 1764

2102101 : 1765
2102102 : 1766
2102110 : 1767
2102111 : 1768
2102112 : 1769
2102120 : 1770
2102121 : 1771
2102122 : 1772
2102200 : 1773
2102201 : 1774
2102202 : 1775
2102210 : 1776
2102211 : 1777
2102212 : 1778
2102220 : 1779
2102221 : 1780
2102222 : 1781
2110000 : 1782
2110001 : 1783
2110002 : 1784
2110010 : 1785
2110011 : 1786
2110012 : 1787
2110020 : 1788
2110021 : 1789
2110022 : 1790
2110100 : 1791
2110101 : 1792
2110102 : 1793
2110110 : 1794
2110111 : 1795
2110112 : 1796
2110120 : 1797
2110121 : 1798
2110122 : 1799
2110200 : 1800
2110201 : 1801
2110202 : 1802
2110210 : 1803
2110211 : 1804
2110212 : 1805
2110220 : 1806
2110221 : 1807
2110222 : 1808
2111000 : 1809
2111001 : 1810
2111002 : 1811
2111010 : 1812
2111011 : 1813
2111012 : 1814
2111020 : 1815
2111021 : 1816
2111022 : 1817
2111100 : 1818
2111101 : 1819
2111102 : 1820
2111110 : 1821
2111111 : 1822
2111112 : 1823
2111120 : 1824
2111121 : 1825
2111122 : 1826
2111200 : 1827
2111201 : 1828
2111202 : 1829
2111210 : 1830
2111211 : 1831
2111212 : 1832
2111220 : 1833
2111221 : 1834
2111222 : 1835
2112000 : 1836
2112001 : 1837
2112002 : 1838
2112010 : 1839
2112011 : 1840
2112012 : 1841
2112020 : 1842
2112021 : 1843
2112022 : 1844
2112100 : 1845
2112101 : 1846
2112102 : 1847
2112110 : 1848
2112111 : 1849
2112112 : 1850
2112120 : 1851
2112121 : 1852
2112122 : 1853
2112200 : 1854
2112201 : 1855
2112202 : 1856
2112210 : 1857
2112211 : 1858
2112212 : 1859
2112220 : 1860
2112221 : 1861
2112222 : 1862
2120000 : 1863
2120001 : 1864
2120002 : 1865
2120010 : 1866
2120011 : 1867
2120012 : 1868
2120020 : 1869
2120021 : 1870
2120022 : 1871
2120100 : 1872
2120101 : 1873
2120102 : 1874
2120110 : 1875
2120111 : 1876
2120112 : 1877
2120120 : 1878
2120121 : 1879
2120122 : 1880
2120200 : 1881
2120201 : 1882
2120202 : 1883
2120210 : 1884
2120211 : 1885
2120212 : 1886
2120220 : 1887
2120221 : 1888
2120222 : 1889
2121000 : 1890
2121001 : 1891
2121002 : 1892
2121010 : 1893
2121011 : 1894
2121012 : 1895
2121020 : 1896
2121021 : 1897
2121022 : 1898
2121100 : 1899
2121101 : 1900
2121102 : 1901
2121110 : 1902
2121111 : 1903
2121112 : 1904
2121120 : 1905
2121121 : 1906
2121122 : 1907
2121200 : 1908
2121201 : 1909
2121202 : 1910
2121210 : 1911
2121211 : 1912
2121212 : 1913
2121220 : 1914
2121221 : 1915
2121222 : 1916
2122000 : 1917
2122001 : 1918
2122002 : 1919
2122010 : 1920
2122011 : 1921
2122012 : 1922
2122020 : 1923
2122021 : 1924
2122022 : 1925
2122100 : 1926
2122101 : 1927
2122102 : 1928
2122110 : 1929
2122111 : 1930
2122112 : 1931
2122120 : 1932
2122121 : 1933
2122122 : 1934
2122200 : 1935
2122201 : 1936
2122202 : 1937
2122210 : 1938
2122211 : 1939
2122212 : 1940
2122220 : 1941
2122221 : 1942
2122222 : 1943
2200000 : 1944
2200001 : 1945
2200002 : 1946
2200010 : 1947
2200011 : 1948
2200012 : 1949
2200020 : 1950
2200021 : 1951
2200022 : 1952
2200100 : 1953
2200101 : 1954
2200102 : 1955
2200110 : 1956
2200111 : 1957
2200112 : 1958
2200120 : 1959
2200121 : 1960
2200122 : 1961
2200200 : 1962
2200201 : 1963
2200202 : 1964
2200210 : 1965
2200211 : 1966
2200212 : 1967
2200220 : 1968
2200221 : 1969
2200222 : 1970
2201000 : 1971
2201001 : 1972
2201002 : 1973
2201010 : 1974
2201011 : 1975
2201012 : 1976
2201020 : 1977
2201021 : 1978
2201022 : 1979
2201100 : 1980
2201101 : 1981
2201102 : 1982
2201110 : 1983
2201111 : 1984
2201112 : 1985
2201120 : 1986
2201121 : 1987
2201122 : 1988
2201200 : 1989
2201201 : 1990
2201202 : 1991
2201210 : 1992
2201211 : 1993
2201212 : 1994
2201220 : 1995
2201221 : 1996
2201222 : 1997
2202000 : 1998
2202001 : 1999
2202002 : 2000
2202010 : 2001
2202011 : 2002
2202012 : 2003
2202020 : 2004
2202021 : 2005
2202022 : 2006
2202100 : 2007
2202101 : 2008
2202102 : 2009
2202110 : 2010
2202111 : 2011
2202112 : 2012
2202120 : 2013
2202121 : 2014
2202122 : 2015
2202200 : 2016
2202201 : 2017
2202202 : 2018
2202210 : 2019
2202211 : 2020
2202212 : 2021
2202220 : 2022
2202221 : 2023
2202222 : 2024
2210000 : 2025
2210001 : 2026
2210002 : 2027
2210010 : 2028
2210011 : 2029
2210012 : 2030
2210020 : 2031
2210021 : 2032
2210022 : 2033
2210100 : 2034
2210101 : 2035
2210102 : 2036
2210110 : 2037
2210111 : 2038
2210112 : 2039
2210120 : 2040
2210121 : 2041
2210122 : 2042
2210200 : 2043
2210201 : 2044
2210202 : 2045
2210210 : 2046
2210211 : 2047
2210212 : 2048
2210220 : 2049
2210221 : 2050
2210222 : 2051
2211000 : 2052
2211001 : 2053
2211002 : 2054
2211010 : 2055
2211011 : 2056
2211012 : 2057
2211020 : 2058
2211021 : 2059
2211022 : 2060
2211100 : 2061
2211101 : 2062
2211102 : 2063
2211110 : 2064

2211111 : 2065
2211112 : 2066
2211120 : 2067
2211121 : 2068
2211122 : 2069
2211200 : 2070
2211201 : 2071
2211202 : 2072
2211210 : 2073
2211211 : 2074
2211212 : 2075
2211220 : 2076
2211221 : 2077
2211222 : 2078
2212000 : 2079
2212001 : 2080
2212002 : 2081
2212010 : 2082
2212011 : 2083
2212012 : 2084
2212020 : 2085
2212021 : 2086
2212022 : 2087
2212100 : 2088
2212101 : 2089
2212102 : 2090
2212110 : 2091
2212111 : 2092
2212112 : 2093
2212120 : 2094
2212121 : 2095
2212122 : 2096
2212200 : 2097
2212201 : 2098
2212202 : 2099
2212210 : 2100
2212211 : 2101
2212212 : 2102
2212220 : 2103
2212221 : 2104
2212222 : 2105
2220000 : 2106
2220001 : 2107
2220002 : 2108
2220010 : 2109
2220011 : 2110
2220012 : 2111
2220020 : 2112
2220021 : 2113
2220022 : 2114
2220100 : 2115
2220101 : 2116
2220102 : 2117
2220110 : 2118
2220111 : 2119
2220112 : 2120
2220120 : 2121
2220121 : 2122
2220122 : 2123
2220200 : 2124
2220201 : 2125
2220202 : 2126
2220210 : 2127
2220211 : 2128
2220212 : 2129
2220220 : 2130
2220221 : 2131
2220222 : 2132
2221000 : 2133
2221001 : 2134
2221002 : 2135
2221010 : 2136
2221011 : 2137
2221012 : 2138
2221020 : 2139
2221021 : 2140
2221022 : 2141
2221100 : 2142
2221101 : 2143
2221102 : 2144
2221110 : 2145
2221111 : 2146
2221112 : 2147
2221120 : 2148
2221121 : 2149
2221122 : 2150
2221200 : 2151
2221201 : 2152
2221202 : 2153
2221210 : 2154
2221211 : 2155
2221212 : 2156
2221220 : 2157
2221221 : 2158
2221222 : 2159
2222000 : 2160
2222001 : 2161
2222002 : 2162
2222010 : 2163
2222011 : 2164
2222012 : 2165
2222020 : 2166
2222021 : 2167
2222022 : 2168
2222100 : 2169
2222101 : 2170
2222102 : 2171
2222110 : 2172
2222111 : 2173
2222112 : 2174
2222120 : 2175
2222121 : 2176
2222122 : 2177
2222200 : 2178
2222201 : 2179
2222202 : 2180
2222210 : 2181
2222211 : 2182
2222212 : 2183
2222220 : 2184
2222221 : 2185
2222222 : 2186

Annexe 2 : Exemple détaillé d'une fourchette.

Annexe 3 : Code source du programme calculant les possibilités en base 3.

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
   int a,b,c,d,e,f,g,cpt=0,cpt_g=0;
   FILE* ft=fopen("possiblilites_trinaire.txt","w");
   FILE* fg=fopen("gagant_trinaire.txt","w");
   for(a=0;a<3;a++)
   {
      for(b=0;b<3;b++)
      {
         for(c=0;c<3;c++)
         {
            for(d=0;d<3;d++)
            {
               for(e=0;e<3;e++)
               {
                  for(f=0;f<3;f++)
                  {
                     for(g=0;g<3;g++)
                     {
                        fprintf(ft,"%d%d%d%d%d%d%d : %d\n",a,b,c,d,e,f,g,cpt);
                        cpt++;
                        if(a==b && b==c && c==d && a!=0 || b==c && c==d && d==e && b!
=0 || c==d && d==e && e==f && c!=0 || d==e && e==f && f==g && d!=0)
                        {
                           cpt_g++;
                           fprintf(fg,"%d%d%d%d%d%d%d : %d\n",a,b,c,d,e,f,g,cpt);
                        }
                     }
                  }
               }
            }
         }
      }
      fprintf(fg,"\nIl y a %d combinaisons gagnantes pour %d combinaisons
totales.\n",cpt_g,cpt+1);
   }
```

Annexe 4 : Code source du jeu pour deux joueurs humains

```
include <stdio.h>
#include <stdlib.h>

int M[6][7];
float P[6][7];
int aqui=1;
int GAGNER=0;

void init()
{
   int i, j;
   // Tableau M initialisé à 0 !
   for(i=1;i>=6;i++)
   {
      for(j=1;j>=7;j++)
      {
         M[i][j]=0;
      }
   }
}

void afficher ()
{
   int i,j;
   system("cls");
   for(i=1;i<=6;i++)
   {
      for(j=1;j<=7;j++)
      {
         if(M[i][j]==1) printf(" J ");
         if(M[i][j]==-1) printf(" R ");
         if(M[i][j]==0) printf(" 0 ");
      }
      printf("\n");
   }
}

int jouer()
{
   int pos=0,i;
   printf("Cela vient au joueur ");
   if(aqui==1)
   {
      printf("1 de joueur.\n");
   }
   else
   {
      printf("2 de joueur.\n");
```

```
    }
    printf("A quelle case voulez-vous jouer ? ");
    scanf("%d",&pos);
    while(!(pos>0 && pos<8))
    {
       b:printf("A quelle case voulez-vous jouer ? ");
       scanf("%d",&pos);
    }
    for(i=6;i>=1;i--)
    {
       if(M[i][pos]==0)
       {
          M[i][pos]=aqui;
          break;
       }
       if(i==1 && M[i][pos]!=0)
       {
          printf("La colonne est pleine !!!\n");
          pos=0;
          goto b;
       }
    }
    if(aqui==1) aqui=-1;
    else aqui=1;
    return pos;
}

void verif_colonne(int pos)
{
    int i, temp;
    for(i=1;i<=3;i++)
    {
       temp = M[i][pos]+M[i+1][pos]+M[i+2][pos]+M[i+3][pos];
       if(temp==4 || temp==-4) break;
    }
    if(temp==4)
    {
       GAGNER=1;
    }if(temp==-4)
    {
       GAGNER=2;
    }
}

void verif_ligne()
{
    int i,j,temp;
    for(i=1;i<=6;i++)
    {
       temp=0;
```

```
      for(j=1;j<=4;j++)
      {
         temp += M[i][j];
         if(temp==4)
         {
            GAGNER=1;
            goto end;
         }
         if(temp==-4)
         {
            GAGNER=2;
            goto end;
         }
      }
   }
   end:printf("");

}

void verif_diagonale()
{
   int i,j,temp;
   for(i=1;i<=3;i++)
   {
      for(j=1;j<=4;j++)
      {
         temp=M[i][j]+M[i+1][j+1]+M[i+2][j+2]+M[i+3][j+3];
         if(temp==4)
         {
            GAGNER=1;
            goto end;
         }
         if(temp==-4)
         {
            GAGNER=2;
            goto end;
         }
      }
   }
   for(i=6;i>=4;i--)
   {
      for(j=1;j<=4;j++)
      {
         temp=M[i][j]+M[i-1][j+1]+M[i-2][j+2]+M[i-3][j+3];
         if(temp==4)
         {
            GAGNER=1;
            goto end;
         }
         if(temp==-4)
```

```
            {
               GAGNER=2;
               goto end;
            }
         }
      }
      end:printf("");
}

void verif(int pos)
{
   if(GAGNER==0)verif_colonne(pos);
   if(GAGNER==0)verif_ligne();
   if(GAGNER==0)verif_diagonale();
}

int main()
{
   int pos;
   init();
   while(GAGNER==0)
   {
      afficher();
      pos=jouer();
      verif(pos);
   }
   afficher();
   printf("\nLe joueur %d gagne !\n",GAGNER);
}
```

Annexe 5 : Fonctions gérant la première règle des poids

```
void init_poids()
{
   //Le tableau des poids
   P[1][1] = 2; P[1][2] = 3; P[1][3] = 4;P[1][4] = 5; P[1][5] = 4; P[1][6] = 3;P[1][7] = 2;
   P[2][1] = 3; P[2][2] = 4; P[2][3] = 5;P[2][4] = 6; P[2][5] = 5; P[2][6] = 4;P[2][7] = 3;
   P[3][1] = 4; P[3][2] = 5; P[3][3] = 6;P[3][4] = 7; P[3][5] = 6; P[3][6] = 5;P[3][7] = 4;
   P[4][1] = 4; P[4][2] = 5; P[4][3] = 6;P[4][4] = 7; P[4][5] = 6; P[4][6] = 5;P[4][7] = 4;
   P[5][1] = 3; P[5][2] = 4; P[5][3] = 5;P[5][4] = 6; P[5][5] = 5; P[5][6] = 4;P[5][7] = 3;
   P[6][1] = 2; P[6][2] = 3; P[6][3] = 4;P[6][4] = 5; P[6][5] = 4; P[6][6] = 3;P[6][7] = 2;
}

int API_joue()
{
   int i,j;
   float temp=0,pos;
   for(j=1;j<=7;j++)
   {
      for(i=6;i>=1;i--)
      {
         if(M[i][j]==0)
         {
            if(temp<P[i][j])
            {
               pos=j;
               temp = P[i][j];
            }
            break;
         }
      }
   }
   return pos;
}

void API_j(int x, int y)
{
   int i,j;
   /** REGLE AUGMENTER LES 8 CASES ADJACENTES DE 1 */
   for(i=x-1;i<=x+1;x++)
   {
      if(i<0 || i>7) goto plus;
      for(j=y-1;j<=y+1;y++)
      {
         if(j<0 || j>8) goto plplus;
         /*ACTIONS*/
         P[i][j]+=1;
         plplus:printf("");
```

```
        }
        plus:printf("");
    }
    /***/
}

void API_o(int i, int j)
{

}
```