Jetons dans un tableau

2018-2019

Nom, prénom et niveaux des élèves : Benoit COEUGNET, Raphaël BERNAS et Alex TRAN VAN NHIEU, classe de Première

Établissement : Lycée Blaise Pascal, Orsay

Enseignant(s) : M.MISSENARD, Mme COCHARD, Mme DAMONGEOT et M.JULLIOT

Sommaire

1 Présentation du sujet

On prend un tableau ABCD de h colonnes et v lignes. On déplace un jeton dans ce tableau de centre en centre des cases grâce à un vecteur de composantes (±n,±m) ou (±m,±n). Nous partons du point A et notre objectif est d'atteindre le point B sans sortir du tableau. Si on utilise le vecteur (0;1), on peut atteindre l'objectif en un nombre de déplacement d qui correspond à d = h - 1.

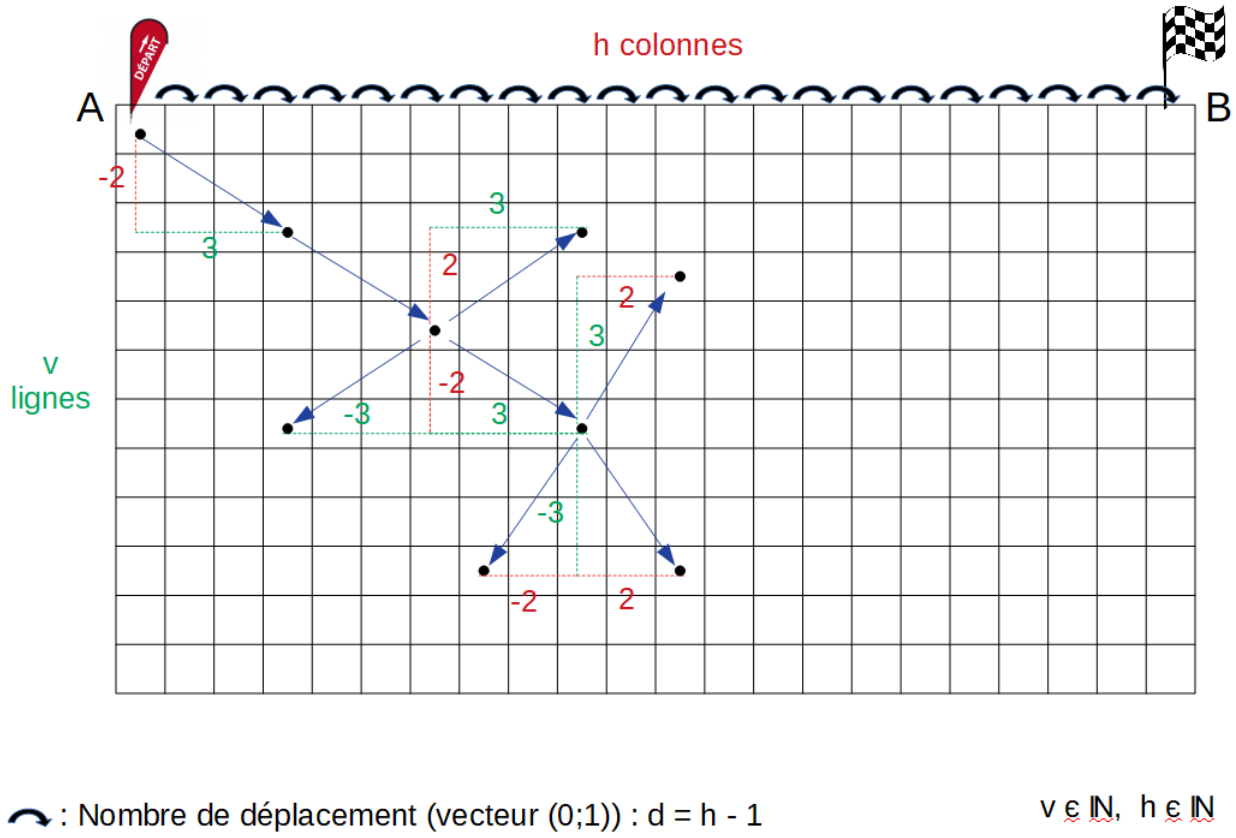

FIGURE 1 – Mise en situation avec le vecteur (2;3)

Le fait de construire des tableaux était un processus très long. Pour gagner du temps, et observer des généralisations, nous avons réalisés 2 programmes afin de faciliter nos recherches.

2 Programmation

Nous avons donc réalisé 2 programme différents. Le premier permet de savoir si un vecteur fonctionne ou pas pour un tableau donné. Le deuxième nous donne les vecteurs pour lesquels l'objectif peut être réalisé.

2.1 Trouver les solutions dans un tableau donné pour un vecteur de composantes données

La lettre A dans le tableau représente la case d'arrivée.

O 0					A 17
		O 3	O 8	O 13	
		O 16	O 1	O 6	O 11
O 9	O 14				O 4
O 2	O 7	O 12			X
		O 5	O 10	O 15	

FIGURE 2 – Tableau de 6*6 avec un vecteur de déplacement (2;3)

On peut voir dans la *figure 2* que le programme a trouvé une solution en 17 déplacements pour le tableau 6*6 avec le vecteur (2;3). La solution est visible : on peut connaître le chemin emprunté au programme pour trouver la solution en suivant les nombres dans les cases.

X				X	A
		X			
X				X	

FIGURE 3 – Tableau de 6*5 avec un vecteur de déplacement (2;2)

La *figure 3* nous montre un tableau où le programme n'a pas pu trouver de solution, et d'ailleurs il n'y en a pas (cf.Conditions d'arrêts). Les croix dans le tableau nous permettent de savoir tous les endroits par où est passé le programme. On peut bien voir que l'arrivée est impossible à atteindre.

2.1.1 Principe de récursivité

Dans cette partie, nous expliquerons le fonctionnement du premier programme.
A chaque case atteinte par le programme, celui-ci va tester tous les déplacements possibles à partir de cette case en les prenant dans un ordre déterminé arbitrairement (pour le point de départ, on a 2 déplacements possibles si $n \neq m$ et 1 si $n = m$. Pour certains points, on pourra avoir 8 déplacements au plus).

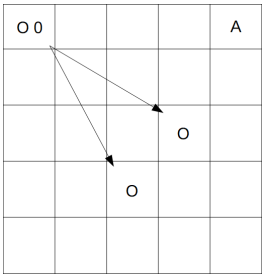

FIGURE 4 – 2 déplacements possibles à partir du points O0

Sur la *figure 4*, on voit qu'à partir du point en haut à gauche, on a deux possibilité de déplacement avec le vecteur (2;3). Cependant, nous avons arbitrairement donné un ordre d'exécution au programme, dans ce cas là, il va donc choisir le déplacement (3;2), soit celui d'en haut.

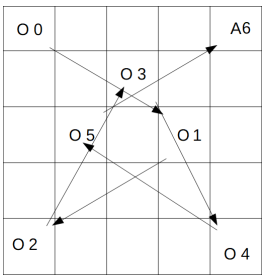

FIGURE 5 – 2 solutions possibles à partir du points O0

Sur la *figure 5*, on voit que le programme n'a pas toujours choisi le vecteur (2;3), mais il a aussi choisi le vecteur (3;2) ou (-3;2), par exemple, pour arriver à la fin.

2.1.2 Conditions d'arrêts

Dans cette partie, nous expliciterons le retour en arrière du premier programme

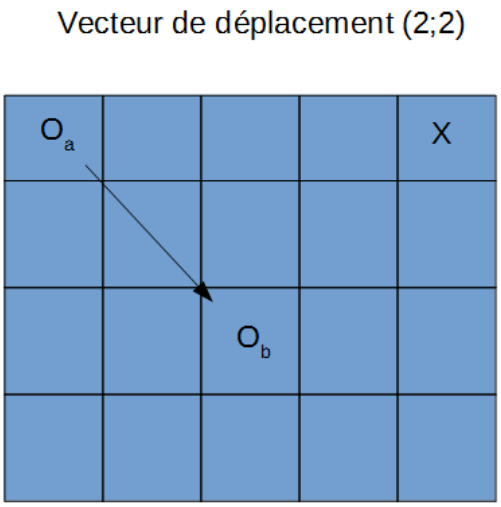

FIGURE 6 – Étape 1

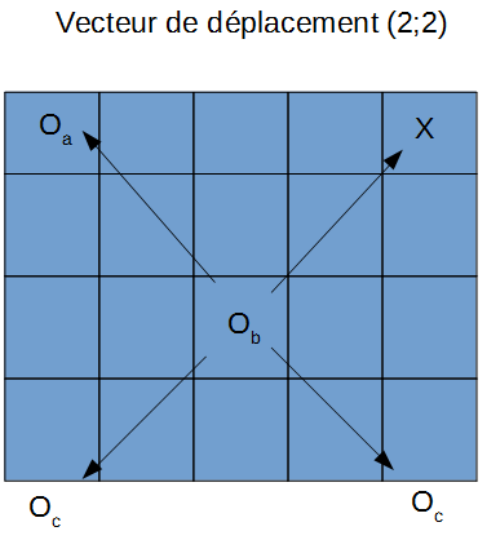

FIGURE 7 – Étape 2

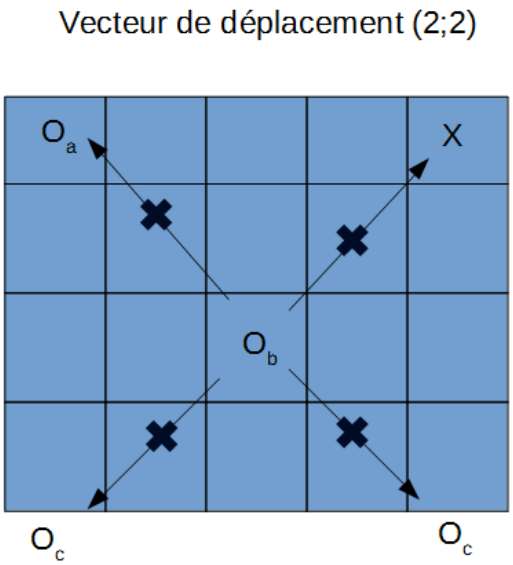

FIGURE 8 – Étape 3

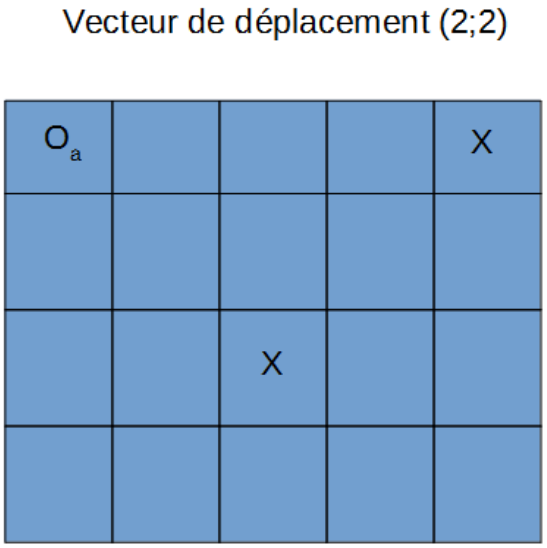

FIGURE 9 – Étape 4

2.2 Trouver tous les vecteurs pour lesquels on a une solution pour un tableau donné

Ce deuxième programme reprend le premier programme, mais à grande échelle. C'est à dire que ce deuxième programme fait tourner en boucle le premier programme avec différents entrées afin d'obtenir tous les vecteurs solutions pour un tableau donné :

10* 12	0	1	2	3	4	5	6	7	8	9	10	11	12
0	X	O	X	X	X	X	X	X	X	X	X	O	X
1	O	X	O	X	O	X	O	X	O	X	X	X	X
2	X	O	X	O	X	O	X	O	X	X	X	X	X
3	X	X	O	X	O	X	X	X	X	X	X	X	X
4	X	O	X	O	X	O	X	X	X	X	X	X	X
5	X	X	O	X	O	X	X	X	X	X	X	X	X
6	X	O	X	X	X	X	X	X	X	X	X	X	X
7	X	X	O	X	X	X	X	X	X	X	X	X	X
8	X	O	X	X	X	X	X	X	X	X	X	X	X
9	X	X	X	X	X	X	X	X	X	X	X	X	X
10	X	X	X	X	X	X	X	X	X	X	X	X	X

FIGURE 10 – Exemple avec le tableau 10*12

Dans le tableau, les croix désignent un vecteur qui ne permet pas de résoudre le problème et les ronds désignent un vecteur qui permet de résoudre le problème. Par exemple, pour ce tableau 10*12, le vecteur (3;2) est solution. Par contre, le vecteur (5;3) n'est pas solution.

En observant ces tableaux, nous avons essayé de trouver une logique pour les vecteurs solution. Les lois que nous avons trouvé sont expliqués dans la partie suivante.

3 Théorie

3.1 Lois démontrées

Si n divise d et le quotient est pair Alors (m;n) solution avec m quelconque
On peut réutiliser la démonstration précédente

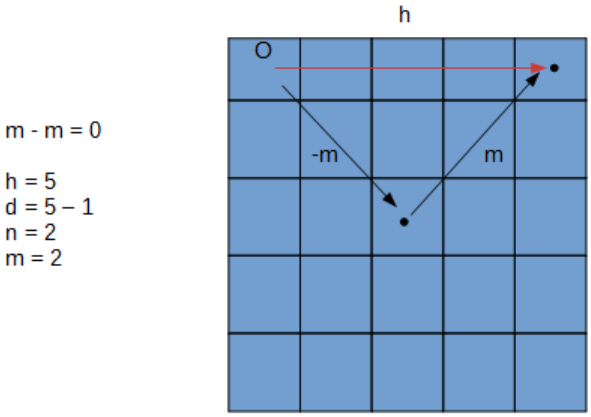

FIGURE 11 – Loi 1 : loi des diviseurs - extension

- Si 2(n + m) divise d Alors le couple est solution

  Alors (m,n) est solution

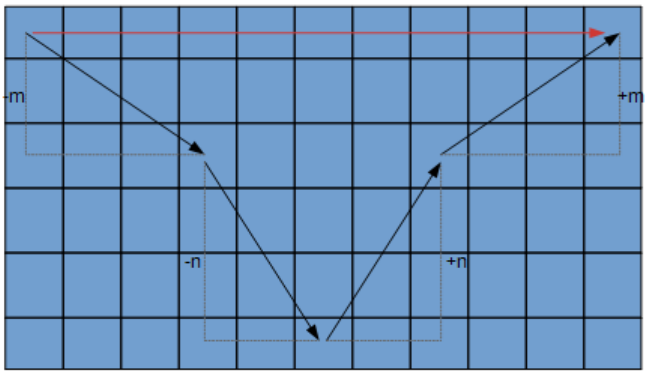

FIGURE 12 – Loi 2

- Si d est pair

  Alors (1;1) est solution

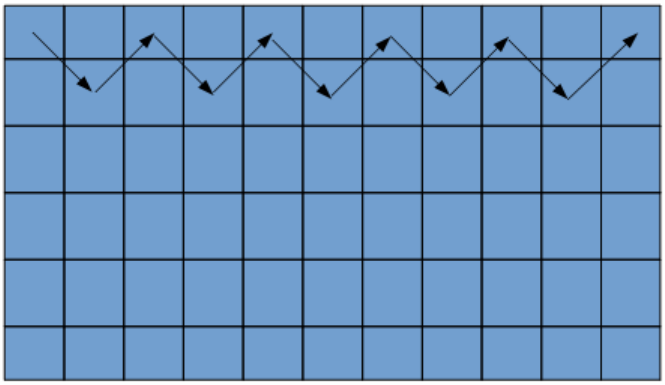

FIGURE 13 – Loi 3

3.2 Lois conjecturées

Soit h le nombre de colonnes.
Si h est pair, les couples solution seront des couples de parité différente et dont la somme est strictement inférieur à h.

4 Conclusion

Pour finir, afin de répondre à notre sujet, nous avons d'abord créé un programme (cf. partie programmation) qui permettrait avec un tableau donné de connaître les couples qui seraient solutions. Cependant, nous souhaitions pouvoir déterminer à un tableau donné, les couples qui seraient solutions sans utiliser le programme. Nous avons donc cherché des lois qui régissent cela (cf. partie théorie).