L'ATTRACTION FOLLE

Auteur : Lionel Xavier

TI-Nspire™ - TI-Nspire™ CAS

Mots-clés : aléatoire, simulation, programme, matrice.

Fichiers associés : AttractionFolle_Nspire_eleve.pdf, AttractionFolle.tns.

1. Objectifs

- Approcher un problème de marche aléatoire.
- Utiliser différents registres pour comprendre le phénomène (simulation, matrices, programmation).

2. Énoncé

A la fête foraine, une nouvelle attraction vient de voir le jour. Une roue pouvant pivoter autour de son axe, est partagée en 6 secteurs égaux. Un seul secteur, le rouge, donne droit à un lot important. Les autres ne correspondent qu'à des lots de consolation. Pour gagner le gros lot, il suffit qu'à l'issue de la partie, le secteur rouge soit positionné en face de la flèche, comme ci-contre. Chaque introduction d'une pièce de 1 € fait tourner la roue, de manière aléatoire, d'un secteur vers la droite (D) ou vers la gauche (G).

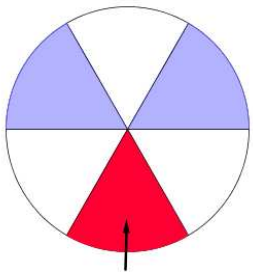


figure 1

3. Commentaires

L'idée de cette activité est de se familiariser avec les marches aléatoires sur un graphe eulérien et non sur un segment.

L'exploration, avec un nombre fort limité de mouvements, permet d'introduire ou de réutiliser des notions déjà rencontrées (simulation, matrices et graphes). Bien entendu, avec des élèves ayant déjà une bonne compréhension de ce que sont ces marches aléatoires, ou experts en matière de simulation sur la calculatrice, il peut être intéressant de simplifier l'énoncé, de façon à leur laisser une plus grande part d'initiative.

La question cachée de l'activité est celle du retour au point de départ lors d'une promenade aléatoire. Le fait toutefois de travailler sur une roue complique un peu la situation. Il faut que l'élève se rende compte que ce « retour » peut être obtenu de trois façons, puisque les tours complets, par la gauche ou par la droite, sont aussi des possibilités. Là encore, une plus grande part d'initiative peut être laissée à l'élève. L'activité suggère à ce propos qu'à partir de 6 € le problème « change » de nature. Il est à noter que l'utilisation des congruences permet ici d'éviter un test du type, si le joueur a obtenu 0 ou 6 ou –6 alors il a gagné.

Remarque :
Dans la partie algorithmique, on fait en sorte que le joueur se fixe un maximum de 20 € à dépenser dans le jeu, pour s'assurer que le jeu ait une fin. Il pourrait en effet voir la roue alterner indéfiniment entre deux positions ne donnant jamais droit au lot principal. Les élèves sont amenés à se poser la question du jeu infini dans la partie exploratoire.

Dans les questions **3 b** et **3 c**, on propose ici l'utilisation d'une fonction *gain*( ) plutôt que d'un programme. Ce peut être l'occasion d'aborder la différence entre ces deux notions. Bien entendu, on peut aussi rédiger un programme *gain()* qui agit sur une variable globale (pour la partie cumul).

L'activité peut être adaptée (suppression de la partie matricielle et des congruences) pour des élèves qui ne seraient pas en spécialité. Elle se placerait alors dans le cadre des activités algorithmiques.

4. Conduite de l'activité

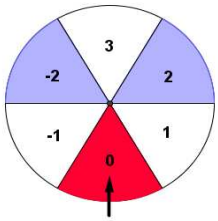
1) Première approche – Un cas particulier à 4 €

On suppose qu'avant que le joueur ne paye le premier euro, la roue est positionnée comme en figure 1.

a) Quel sera le secteur positionné en face de la flèche si la séquence de déplacements est **GDDD** ?

Il peut être plus simple de numéroter les cases de la roue comme ci-contre et de convenir alors qu'une rotation vers la droite correspond à la valeur +1 et qu'une rotation vers la gauche correspond lui à la valeur −1. Le déplacement donné en exemple s'écrit alors −1,+1,+1,+1 ce qui, si on additionne les 4 nombres, donne +2, numéro de la case finale.

Dans cette façon de procéder, le résultat −3 sera assimilé au résultat +3.



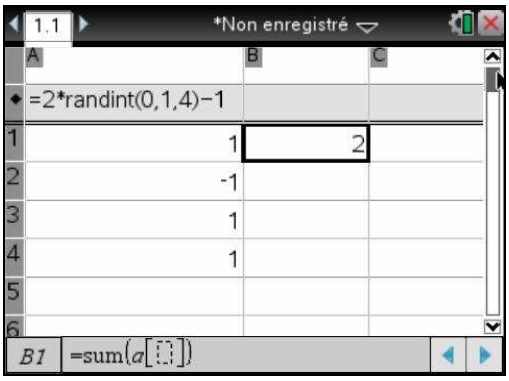
b) Simuler, à l'aide de la calculatrice, une série de 4 rotations successives.

Pour cela, générer, dans la colonne A du tableur, une suite de 4 nombres valant chacun +1 ou −1.

Ajouter les éléments de la colonne A dans la colonne B.

Relancer plusieurs fois la séquence d'instructions (ctrl R).

Que remarque-t-on sur les résultats ? Expliquer.



c) Est-il raisonnable de penser que le joueur puisse gagner le lot principal ?

d) Justifier que dépenser 5 € plutôt que 4, ne constitue pas une bonne stratégie.

e) En quoi le fait de jouer 6 € change-t-il les possibilités de gagner ?
Comment détecter que la somme correspond au secteur rouge (utiliser les congruences) ?

f) Écrire un programme **jeu** qui simule une partie à 6 € et affiche « gagné » ou « perdu », selon les cas.

On pourra noter ***s*** la variable, initialement à 0, qui donne le numéro du secteur atteint après chaque mouvement.

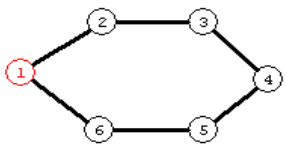
```
Define jeu()=Prgm
  Local s,i
  s:=0
  For i,1,6
    s:=s+2·randInt(0,1)−1
  EndFor
  If mod(s,6)=0 Then
    Disp "C'est gagné !"
  Else
    Disp "Perdu !"
  EndIf
EndPrgm
```

2) Deuxième approche – Via les matrices et pour toute somme engagée

On considère le graphe à six sommets ci-contre.

Les sommets sont les six secteurs. Deux sommets sont liés dès lors que les secteurs sont voisins sur la roue (il est possible de passer de l'un à l'autre en un seul mouvement).

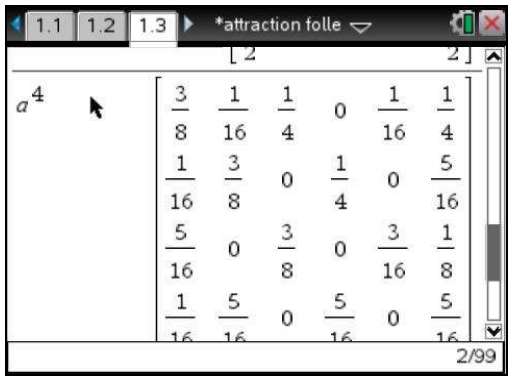
On suppose à nouveau que, initialement, le secteur rouge (ici le sommet 1) est en position « gagné ».



a) Donner la matrice de transition *a* de ce graphe.

b) Rappeler comment, à l'aide des puissances de a, on peut déterminer la probabilité pour qu'un joueur gagne en ayant dépensé exactement n €.

c) En déduire la probabilité de gagner en ayant dépensé exactement 4 € et comparer le résultat à votre réponse formulée en **1c**.



d) Désormais, le chariot est positionné de manière aléatoire à l'arrivée du joueur.
Comment, via la matrice a, peut-on obtenir la probabilité de gagner après exactement n rotations de la roue ?

Si la roue est initialement positionnée sur le secteur k, $1 \leq k \leq 6$*, il faut, après avoir calculé* a^n*, noter la valeur du coefficient* $(k,1)$.

e) Que faut-il modifier dans le cas où la roue à 1 chance sur 5 de rester immobile, 2 chances sur 5 de tourner d'un secteur à droite et 2 chances sur 5 de tourner d'un secteur à gauche ?

Il suffit d'adapter la matrice de transition a comme ci-contre :

$$\begin{pmatrix} 1/5 & 2/5 & 0 & 0 & 0 & 2/5 \\ 2/5 & 1/5 & 2/5 & 0 & 0 & 0 \\ 0 & 2/5 & 1/5 & 2/5 & 0 & 0 \\ 0 & 0 & 2/5 & 1/5 & 2/5 & 0 \\ 0 & 0 & 0 & 2/5 & 1/5 & 2/5 \\ 2/5 & 0 & 0 & 0 & 2/5 & 1/5 \end{pmatrix}.$$

3) Troisième approche – Un algorithme qui détecte la victoire.

Un joueur espère arriver à ses fins en engageant 20 euros au maximum. Il peut donc activer le chariot jusqu'à 20 fois mais peut décider de s'arrêter à l'issue de n'importe lequel des 20 mouvements.
Initialement, le chariot est positionné sur la case 0.

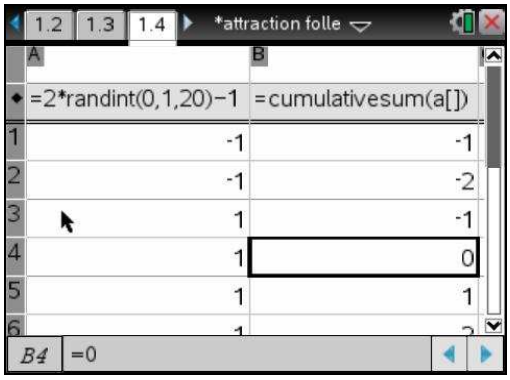
a) A l'aide du tableur

Générer une liste de 20 entiers égaux à +1 ou −1 dans la colonne A.

La TI-Nspire dispose d'une instruction **cumulativesum**, permettant d'obtenir dans la colonne B, de proche en proche, les sommes des entiers de la colonne A.

Comment le joueur détecte-t-il les passages du chariot en position 0 ?

Cela correspond aux 0 *dans la colonne* B.



Renouveler la simulation en appuyant sur ctrl R.

b) Avec un programme

Écrire un programme ***gain*** qui donne le nombre d'euros nécessaires pour gagner.

Algorithme
Initialiser p à 1 ou –1 (p : position)
Initialiser n à 1 (n : nombre de rotations)
Tant que le chariot n'est pas revenu en position initiale et que l'on a dépensé 20 € ou moins **faire**
Ajouter 1 ou –1 à p
Augmenter n de 1
Fin tant que
Si le secteur rouge a été atteint
Afficher Gagné en n euros
Sinon
Afficher Perdu
Fin Si

```
Define gain()=Func
  Local p,n
  2·randInt(0,1)−1→p
  n:=1
  While p≠0 and n<19
    p:=p+2·randInt(0,1)−1
    n:=n+1
  EndWhile
  If p=0 Then
    Disp "Gagné en ",n," euros."
  Else
    Disp "Perdu."
  EndIf
EndFunc
```

Remarque : Le fait d'initialiser p à ±1, et donc n à 1, revient à commencer la boucle après le premier mouvement de la roue. Il reste donc au maximum 19 mouvements à faire.

c) Écrire un programme ***cumul***, d'argument n, permettant de sommer les résultats lors de n exécutions du programme ***gain*** et qui affiche la moyenne obtenue.

Attention ! Le programme ***gain*** affichait du texte que l'on ne souhaite pas voir à chaque répétition. On utilisera donc plutôt un programme ***gain2***, dans lequel le texte aura été supprimé.

```
Define gain2()=Func                                   Terminé
  Local p,n
  2·randInt(0,1)−1→p
  n:=1
  While p≠0 and n<19
    p:=p+2·randInt(0,1)−1
    n:=n+1
  EndWhile
EndFunc
Define cumul(n)=Prgm                                  Terminé
  Local i,somme
  somme:=0
  For i,1,n
    somme:=somme+gain2()
  EndFor
  Disp approx(somme/n)
EndPrgm
cumul(100)
                                                      6.92
                                                      Terminé
                                                      2/3
```