****

## Sujets académiques – académies d’Amiens et de Lille

**Série S**

## Eléments de correction

# Sujet 1 : jeu des rochers

Situation 1 : découverte du jeu et observations

1. **a.** $p\_{1}= p\_{2}=p\_{3}=\frac{1}{3}$ **b.** $q\_{1}=q\_{2}=\frac{1}{2}$

**c.** $p\_{1}+p\_{2}=\frac{2}{3} $ **d.** $p\_{1}q\_{1}+p\_{2}q\_{1}+p\_{2}q\_{2}+p\_{3}q\_{2}=\frac{2}{3}$

1. **a. b.**

|  |  |  |  |
| --- | --- | --- | --- |
|  | $$A\_{1}$$ | $$A\_{2}$$ | $$A\_{3}$$ |
| **Rocher** | **1** | G | P | P |
| **2** | G | G | P |
| **3** | P | G | G |
| **4** | P | P | G |

**c.** **Non**: quelque soit l’intervalle où la grenade atterrie, la probabilité que le joueur gagne vaut $\frac{1}{2}$.

1. **Non**: quelque soit l’intervalle où la grenade atterrie, la probabilité que le joueur gagne vaut $\frac{2}{n}$.
2. **a.**
* D’après la question 3, si « les humanoïdes » sont placés aléatoirement de façon équiprobable alors les probabilités que le joueur gagne sont égales pour chacun des intervalles où la grenade atterrit.
* L’observation invalide raisonnablement l’égalité des probabilités de gagner pour chacun des intervalles où la grenade atterrit.
* Donc par contraposée, on peut raisonnablement penser que le programme ne place pas aléatoirement « l’humanoïde » derrière les rochers de façon équitable.

**b. i.** $p=\frac{1}{7}$

 **ii. Oui** car :

* Si la grenade est envoyée en $A\_{1}$, la probabilité que le joueur gagne est de $2p+p =\frac{3}{7}≈\frac{301}{698}$
* Si la grenade est envoyée en $A\_{2}$, la probabilité que le joueur gagne est de $p+p =\frac{2}{7}≈\frac{200}{704}$
* Si la grenade est envoyée en $A\_{3}$, la probabilité que le joueur gagne est de $p+p =\frac{2}{7}≈\frac{197}{695}$
* Si la grenade est envoyée en $A\_{4}$, la probabilité que le joueur gagne est de 2$p+p =\frac{3}{7}≈\frac{299}{703}$

Situation 2 : choix de $q\_{1}, …, q\_{n-1}$ en mode expert

1. ⚫ Probabilité que la grenade atterrisse en $A\_{1}$ et que le joueur gagne : $q\_{1}×2p+q\_{1}×p=\frac{3}{n+2}q\_{1}$
* Probabilité que la grenade atterrisse en $A\_{2}$ et que le joueur gagne : $q\_{2}×p+q\_{2}×p=\frac{2}{n+2}q\_{2}$
* L’égalité est une conséquence du problème visé.
1. Les situations sont identiques pour les rochers 1 et $n$ puis pour les rochers intermédiaires.

Donc $q\_{1}=q\_{n-1}$ et $q\_{2}=…=q\_{n-2}$. Puis on invoque la somme des probabilités égale à 1.

1. En invoquant les questions 1 et 2 (système 2x2) : $q\_{1}=\frac{2}{3n-5} $ et $q\_{2}=\frac{3}{3n-5} $
2. Question 3 à interpréter dans le contexte.

# Sujet 2 : les nombres étranches

Partie 1 : détermination des nombres étranches

102 ; 1024 ; 10240 ; 705 ; 70520 ; 705204

Partie 2 : recherche des nombres étranches à deux, puis trois chiffres

1. Toute démarche est acceptée.
2. **a.** 0, 1 ou 2.

**b.** $n\_{3}\in \left\{0 ;3 ;6 ;9\right\}$.

**c**. Sous la forme $3k+1$, $n\_{3}\in \left\{2 ;5 ;8\right\}$.

Sous la forme $3k+2$, $n\_{3}\in \left\{1 ;4 ;7\right\}$.

**d.** 15 nombres étranches de longueur 2 multiples de 3, 15 nombres étranches de longueur 2 égaux à 1 modulo 3 et 15 égaux à 2 modulo 3. Avec la question 2. c. : $15×4+15×3+15×3=150$.

**e.** $15×2+15×2+15×1=75$.

1. **a.**

|  |  |  |
| --- | --- | --- |
| $$i$$ | $$m$$ | $$c$$ |
| 0 | **62** | **1** |
| 1 | **621** | **2** |

**b.** $compteur=1$.

**c.** $compteur=0$.

**d**. Cet algorithme teste si un nombre entier à 3 chiffres est étranche.

Partie 3 : Dénombrement des nombres étranches à quatre et cinq chiffres

1. **a)** $\overbar{n\_{1}n\_{2}n\_{3}n\_{4}}=\overbar{n\_{1}n\_{2}}×100+\overbar{n\_{3}n\_{4}}=\overbar{n\_{3}n\_{4}}+4×\left(25×\overbar{n\_{1}n\_{2}}\right)$.

**b)** Utilisation attendue de la définition de la division euclidienne et de « divisible par 4 ».

**c)** 15 nombres étranches à 3 chiffres finissent par $n\_{3}=0$, 15 autres par $n\_{3}=1$, … 15 autres par $n\_{3}=9$.

* Si $n\_{3}$ est pair, $n\_{4}\in \left\{0 ;4 ;8\right\}$
* Si $n\_{3}$ est impair, $n\_{4}\in \left\{2 ;6\right\}$

Un dénombrement donne exactement $15×\left(5×3+5×2\right)=375$ nombres étranches à 4 chiffres.

1. Chaque nombre étranche à 4 chiffres peut être complété par 0 ou 5 à droite pour former un nombre étranche à 5 chiffres. On a donc exactement $375×2=750$ nombres étranches à 5 chiffres.

Aparté culturelle des concepteurs sur les nombres étranches :

* Il existe un et un seul nombre étranche à 10 chiffres sans répétition : 3816547290
* Il existe exactement 2492 nombres étranches terminaux (dont il est impossible d’adjoindre un chiffre à droite pour former un nouveau nombres étranches plus long).
* La liste des nombres étranches est finie. Le plus grand nombre étranche est 3608528850368400786036725. Il est le seul à s’écrire avec 25 chiffres.
* Pour obtenir cette liste et ce plus grand nombre étranche, on recourt à la programmation informatique. Aucune démonstration mathématique de la finitude de cette liste n’est connue à ce jour.
* Des précautions sont à prendre dans l’écriture de l’algorithme puisque la taille des nombres en jeu pose problème. La calculatrice des élèves (ou même les ordinateurs à architecture 32 bits) ne peuvent stocker les nombres entiers que jusque  $2^{32}=4294967296$.
* Il faut donc recourir à une astuce arithmétique pour décomposer l’entier $n$ testé modulo 23. Ci-après : deux versions algorithmiques informatisées du test du caractère « étranche ». La première fonction « EstEtranche(n) » traduit fidèlement la définition mathématique (exploitable en classe) ; la seconde fonction « TestEtranche(n) » est adaptée avec cette contrainte architecturale des machines.

from math import \*

def EstEtranche(n): *#programmation théorique sans tenir compte des problèmes de stockages des entiers au-delà de 2^32*

 f=len(str(n))

 Reponse=True

 k=0

 while k<f and Reponse==True:

 if int(n/10\*\*k)%(f-k)>0:

 Reponse = False

 k=k+1

 return Reponse

def TestEtranche(n): *#version décomposant n en nombres inférieurs à 2^32 et travaillant modulo f-k*

 f=len(str(n))

 Reponse=True

 k=0

 while k<f and Reponse==True:

 m=int(n/10\*\*10) *# On écrit ici n = m x 10^10 + r pour avoir m, 10^10 et r inférieurs à 2^32*

 r=n%10\*\*10

 m=int(m/10\*\*k)

 r=int(r/10\*\*k)

 if m%(f-k)\*((10\*\*10)%(f-k))+r%(f-k)>0:

 Reponse = False

 k=k+1

 return Reponse