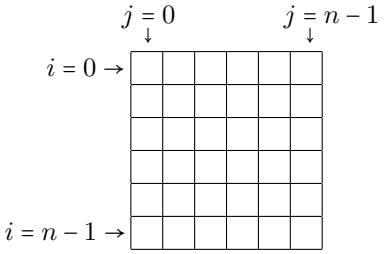
Corrigé du DM n°11

Informatique : Étude d'un jeu de plateau

1. **Reconnaissance du bord**

   Les cases du bord ou des coins sont repérées par les couples d'indices (i, j) tels que : $i = 0$ ou $i = n-1$ ou $j = 0$ ou $j = n-1$ (voir la figure ci-dessous).



   D'où le code :

```
def bord(i,j,n) :
    if i*j == 0 : return True   # i × j = 0 ⇔ i = 0 ou j = 0
    if i == n-1 or j == n-1 : return True
    return False
```

2. **Traitement du rebond**

   Une première version consiste à utiliser des événements de probabilités $\frac{1}{8}$:

```
import random as rd
def rebond(i,j) :
    choix = rd.randint(1,8)
    if choix == 1 : return (i-1,j-1)
    if choix == 2 : return (i-1,j)
    if choix == 3 : return (i-1,j+1)
    if choix == 4 : return (i,j-1)
    if choix == 5 : return (i,j+1)
    if choix == 6 : return (i+1,j-1)
    if choix == 7 : return (i+1,j)
    if choix == 8 : return (i+1,j+1)
```

   Alternativement, on peut choisir aléatoirement les décalages d'abscisse et d'ordonnée dans $\{-1, 0, 1\}$, en prenant soin d'exclure le cas où le jeton reste dans la case initialement désignée.

```
def rebond2(i,j) :
    while True :
        di , dj = rd.randint(-1,1) , rd.randint(-1,1)
        if di != 0 or dj != 0 : return (i+di,j+dj)
```

3. **Simulation d'un jeu**

   (a) On crée une matrice nulle de taille $n \times n$ grâce à la commande : `np.zeros((n,n))`

   (b) On se sert d'une boucle de taille n^2 pour jeter un à un les jetons. Le coefficient (i, j) de la matrice `plateau`, initialement nul, augmente de 1 à chaque fois qu'un jeton atterrit dans cette case.

```
import numpy as np
def jouer(n) :
    plateau = np.zeros((n,n))
    for _ in range(n**2) :
        i = rd.randint(0,n-1)
        j = rd.randint(0,n-1)
        if not bord(i,j,n) :
            k,l = rebond(i,j)
            plateau[k,l] += 1
    return plateau
```

4. **Case la plus peuplée**

(a) La fonction `jouer(n)` sert à créer un plateau rempli après le lancer des n^2 jetons.
On recherche ensuite le maximum des coefficients de la matrice `plateau`, grâce à une double boucle :

```
def populationMax(n) :
    plateau = jouer(n)
    max = 0
    for i in range(n) :
        for j in range(n) :
            if plateau[i,j] > max :
                max = plateau[i,j]
    return max
```

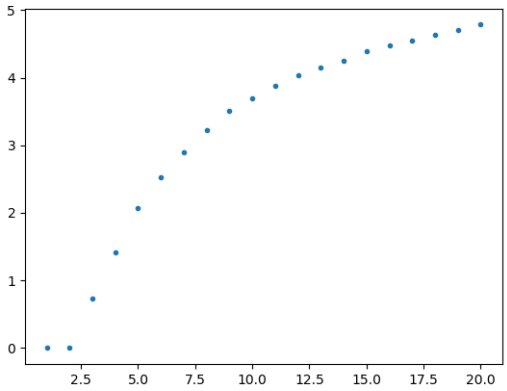
(b) On itère N fois la fonction `populationMax(n)`, et on calcule la moyenne des N résultats obtenus :

```
def popMaxMoyenne(n,N) :
    s = 0
    for _ in range(N) :
        s += populationMax(n)
    return s/N
```

(c) On utilise le module `matplotlib.pyplot`. On définit comme liste d'abscisses : $X = [\![1, n]\!]$, et comme liste d'ordonnées les valeurs renvoyées par `popMaxMoyenne(k,N)` pour $k \in [\![1, n]\!]$:

```
import matplotlib.pyplot as plt
def graphePopMax(n,N) :
    X = list(range(1,n+1))
    Y = []
    for k in range(1,n+1) :
        Y.append(popMaxMoyenne(k,N))
    plt.plot(X,Y,'.')
    plt.show()
```

(d) Figure renvoyée par `graphePopMax(20,1000)` :



On peut conjecturer que le nombre maximal de jetons dans une seule case est croissant quand n augmente. Il n'est pas clair qu'il converge vers une limite, ou qu'il diverge vers $+\infty$.

5. **Proportion de cases vides**

(a) On utilise un compteur et une double boucle pour compter les cases vides d'un plateau :

```
def casesVides(n) :
    plateau = jouer(n)
    compteur = 0
    for i in range(n) :
        for j in range(n) :
            if plateau[i,j] == 0 :
                compteur += 1
    return compteur
```

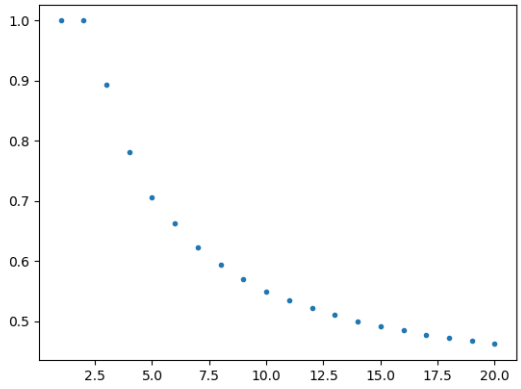
(b) La proportion de cases vide s'obtient en divisant le nombre de cases vides par n^2.
Comme à la question 4(b), on utilise une boucle de taille N pour effectuer une moyenne de ces proportions.

```
def ratioVide(n,N) :
    s = 0
    for _ in range(N) :
        s += casesVides(n)
    return s/(N*n**2)
```

(c) Même idée qu'à la question 4(c) :

```
def grapheCasesVides(n,N) :
    X = list(range(1,n+1))
    Y = []
    for k in range(1,n+1) :
        Y.append(ratioVide(k,N))
    plt.plot(X,Y,'.')
    plt.show()
```

(d) Figure renvoyée par `grapheCasesVides(20,1000)` :



La proportion de cases vides semblent être décroissante quand n augmente. Étant minorée par 0, elle converge nécessairement vers une limite ℓ. Il n'est pas clair que ℓ soit strictement positive ou nulle, bien qu'on puisse raisonnablement penser que $\ell \neq 0$.