# Une expérience aléatoire

Yannick Le Bastard

17 janvier 2024

## Table des matières



**Résumé**
Cet article détaille le jet de trois dés équilibrés, associé à un jeu d'argent. Nous modélisons d'abord la situation de manière théorique à l'aide de graphes probabilistes, en appliquant les règles de parcours et les processus de réduction de tels graphes, ainsi que les règles de la valeur moyenne. Dans un regard comparé, nous utilisons la puissante notion de fonction génératrice, en se basant sur la remarquable méthode de marquage de Van Dantzig. Ce qui nous permet d'obtenir la loi exacte des variables aléatoires considérées.
Ceci dit, les outils théoriques mis en jeu dépassent largement le cadre du lycée. D'où l'idée d'effectuer des simulations afin de retrouver de manière approchée les résultats précédents. Et par là même, justifier l'intérêt d'une telle approche pour répondre à des problèmes théoriquement inaccessibles dans le secondaire.
Nous utiliserons le langage de programmation Python pour illustrer nos propos en mettant en valeur l'approche fonctionnelle conformément aux nouveaux programmes.

# 1 Aspect théorique d'un jeu probabiliste

Considérons le jet de trois dés équilibrés à six faces. Notons $S$ la somme obtenue.

1. Si $S \leq 7$, la partie est perdue,
2. Si $8 \leq S \leq 14$, on relance les trois dés,
3. Si $S \geq 15$, la partie est gagnée.

Nous nous proposons de répondre dans un premier temps aux trois questions suivantes :
— Calculer la probabilité de gain à ce jeu en modélisant sa loi,
— Calculer la durée moyenne d'une partie,
— Modéliser la loi de probabilité de la variable aléatoire $S$ somme de 3 dés.

## 1.1 Modélisation du jeu avec les graphes probabilistes

Les trois dés étant parfaits, nous pouvons prendre comme univers $\Omega = \{1;2;3;4;5;6\}^3$ que nous munissons de la probabilité uniforme. Soit $S$ la variable aléatoire réelle qui à chaque jet de trois dés équilibrés, associe la somme obtenue.
Il est évident que $S(\Omega) = \{3;4;\ldots;18\}$.

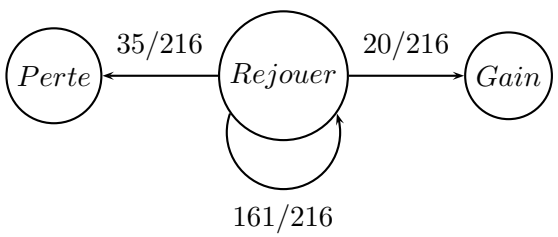
Détaillons un peu le calcul de $P(S \leq 7)$ :

| événement | déclinaisons possibles | nombre de cas favorables |
|---|---|---|
| (S=3) | 1+1+1 | 1 |
| (S=4) | 1+1+2 | 3!/2!=3 |
| (S=5) | 1+1+3 | 3!/2!=3 |
| | 1+2+2 | 3!/2!=3 |
| (S=6) | 1+1+4 | 3!/2!=3 |
| | 1+2+3 | 3!=6 |
| | 2+2+2 | 1 |
| (S=7) | 1+1+5 | 3!/2!=3 |
| | 1+2+4 | 3!=6 |
| | 1+3+3 | 3!/2!=3 |
| | 2+2+3 | 3!/2!=3 |

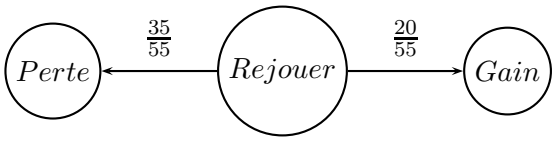
Ainsi, $P(S \leq 7) = \dfrac{2 \times 1 + 7 \times 3 + 2 \times 6}{6^3} = \dfrac{35}{216}$.

En effectuant un raisonnement similaire, nous trouvons que $P(S \geq 15) = \dfrac{20}{216}$.
Nous en déduisons que $P(8 \leq S \leq 14) = 1 - \dfrac{20+35}{216} = \dfrac{161}{216}$.

Nous pouvons dès lors créer un graphe probabiliste modélisant le jeu et qui comporte trois états : les deux états absorbants "Perte" : $(S \leq 7)$ et "Gain" : $(S \geq 15)$, ainsi que l'état "Rejouer" : $(8 \leq S \leq 14)$. Sans perte de généralité, nous pouvons supposer que ce dernier est l'état initial (la boucle au-dessus pouvant être parcourue 0, 1, 2, ...fois).



graphe qui peut se réduire (*c.f* annexe) à :




Si l'on note $X$ la variable aléatoire qui prend pour valeur 1 si Gain et 0 si Perte, on constate que $X$ suit la **loi de Bernoulli de paramètre** $p = \dfrac{20}{55} \approx 0,3636$.

Définissons maintenant la variable aléatoire $Y$ égale à la durée d'une partie. En regardant le premier graphe probabiliste représentant le jeu nous pouvons constater que : $Y(\Omega) = \mathbb{N}^*$ et que pour tout entier naturel $n \geq 1$ :

$$P(Y = n) = P(\text{faire } n-1 \text{ boucles suivi de perte ou de gain}) = \left(\frac{161}{216}\right)^{n-1} \times \frac{55}{216}$$

Le décalage "$n-1$" est dû au fait que l'on peut gagner ou perdre dès le premier lancer de dés.

Pour calculer l'espérance de $Y$, nous pouvons utiliser la seconde règle de la valeur moyenne (*c.f* annexe). Notons respectivement 0, 1 et 2 les états "Perte", "Rejouer" et "Gain" et $m_i$ le délai moyen d'absorption partant de l'état $i$ :

$$\begin{cases} m_0 = m_2 = 0 \\ m_1 = 1 + \dfrac{161}{216} m_1 + \dfrac{35}{216} m_0 + \dfrac{20}{216} m_2 \end{cases}$$

D'où $m_1 = \dfrac{216}{55} \approx 3.927$.

## 1.2 Modélisation du jeu avec les fonctions génératrices

Nous pouvons bien entendu nous servir de la loi de $Y$ pour retrouver sa fonction génératrice : $\phi_Y(t) = \dfrac{55}{216} \displaystyle\sum_{n\geq 1} \left(\frac{161}{216}\right)^{n-1} \times t^n$.

La méthode de marquage de Van Dantzig est plus élégante et plus pratique pour le calcul qui va suivre ; nous la rencontrerons en exercice.

Quoi qu'il en soit, le rayon de convergence de cette série entière est égal à $R = \dfrac{216}{161} > 1$.

On en déduit (c.f annexe) que $E(Y) = \phi'_Y(1)$.

Or, $\phi_Y(t) = \dfrac{55t}{216} \times \displaystyle\sum_{n\geq 1} \left(\frac{161t}{216}\right)^{n-1}$, soit pour $|t| < R$ :

$$\phi_Y(t) = \frac{55t}{216} \frac{1}{1 - \dfrac{161t}{216}} = \frac{55t}{216 - 161t}.$$

Un bref calcul nous assure alors que $\phi'_Y(1) = \dfrac{216}{55}$.

Nous retrouvons sans surprise le résultat précédent.

### 1.3 Loi de probabilité de la somme de trois dés

Il est évident que $S(\Omega) = \{3; 4; \ldots; 18\}$. Calculons $\text{Card}(S = n),\ 3 \leq n \leq 18$. Il ne s'agit rien d'autre que de résoudre l'équation entière

$$x_1 + x_2 + x_3 = n, \quad 1 \leq x_1, x_2, x_3 \leq 6, \quad 3 \leq n \leq 18$$

**Solutions de l'équation $x_1 + x_2 + x_3 = n$ :** La méthode visuelle expliquée ici est à retenir. Tout comme sa modification pour répondre au problème donné.

Considérons l'équation entière :

$$x_1 + x_2 + x_3 = n,\ x_i \in \mathbb{N}, n \geq 3 \quad (1)$$

Nous cherchons à déterminer le nombre de solutions $D_{3,n}$ de cette équation. Signalons que l'ordre dans lequel sont rangés les $x_i$ compte.

Nous distinguerons par exemple $n + 0 + 0 = n$ ; $0 + n + 0 = n$ ; $0 + 0 + n = n$. Ceci nous incite à remplacer les symboles "plus" par des "séparateurs" | et les nombres placés entre les séparateurs par des cercles $\circ$.
Par exemple, la solution $(1, 2, 4)$ de $x_1 + x_2 + x_3 = 7$ est représentée par $\circ|\circ\circ|\circ\circ\circ\circ$
Autre exemple : la solution $(0, 5, 2)$ de la même équation est représentée par $|\circ\circ\circ\circ\circ|\circ\circ$
Nous pouvons dès lors mettre en bijection l'ensemble des solutions de l'équation entière $x_1 + x_2 + x_3 = n$ avec l'ensemble des mots constitués de deux séparateurs | et de $n$ cercles $\circ$. Le cardinal de l'ensemble de ces mots est égal à $D_{3,n} = \binom{n+2}{2}$ (choix des places des 2 séparateurs parmi les $n + 2$ places possibles).
Attention ! Ce résultat implique que des variables $x_i$ peuvent être nulles, ce qui n'est pas le cas ici. Nous devons avoir $1 \leq x_i \leq 6 \quad (1 \leq i \leq 3)$. Il convient donc de modifier le résultat précédent.

Considérons maintenant l'équation :

$$y_1 + y_2 + y_3 = n,\ y_i \in \mathbb{N}^*, n \geq 3 \quad (2)$$

Il est clair qu'en posant $x_i = y_i - 1$, on se ramène à l'équation précédente (1). Le nombre $D^*_{3,n}$ des solutions de (2) est alors :

$$D^*_{3,n} = \binom{n-1}{2}$$

(on a remplacé $n$ par $n - 3$).

**Loi de la somme S de nos trois dés :** Ce qui précède, nous permet de conclure par équiprobabilité que : $\begin{cases} S(\Omega) = \{3; 4; 5; \ldots; 18\} \\ \forall n \in S(\Omega) \quad P(S = n) = \dfrac{\binom{n-1}{2}}{6^3} = \dfrac{(n-1)(n-2)}{432} \end{cases}$

## 2 Simulation de ce jeu

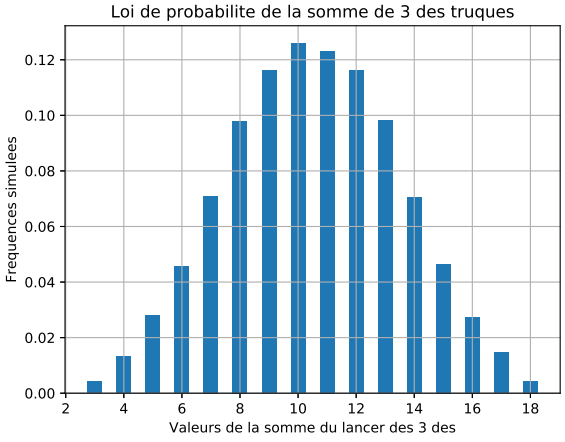
*Je remercie chaleureusement mon collègue Jean-Michel Dardié pour sa contribution numpy à cette partie.*

1. Nous commencerons par présenter la simulation de la variable aléatoire $S$ : somme des trois dés, étudiée de manière théorique auparavant.
2. Nous simulerons ensuite la loi de $X$, puis la loi de $Y$ et calculerons une valeur approchée de $E(Y) = \dfrac{216}{55}$.

## 2.1 Avec l'hypothèse d'équiprobabilité

**Loi simulée de $S$ :** L'intérêt de simuler la loi de $S$ tient au fait que celle-ci est difficile à appréhender de manière théorique pour des élèves du secondaire. Programmer un script la simulant n'est néanmoins pas simple, et il faut parfois se contenter de le présenter en justifiant son intérêt et en le décortiquant dans les grandes lignes. En voici un en "pur Python".

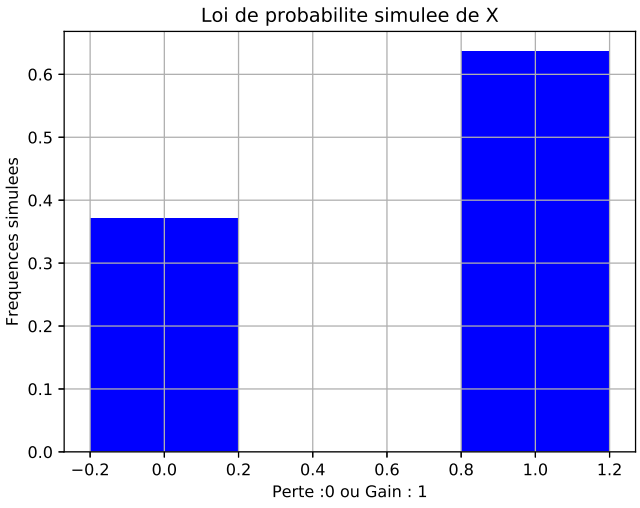
```
#Simulation de la loi de S : somme de trois des equilibres
def Lancer3Des():
    from random import randint
    return sum([randint(1,6) for i in range(3)])

def frequence(N):            #Frequence d'apparition de chaque somme
    L=[0 for i in range(3,19)]  #Sommes de 3 a 18 inclus
    for i in range(N):
        alea=Lancer3Des()    #on lance nos 3 des et on releve la somme
        for j in range(3,19):
            if alea==j:
                L[j-3]+=1      #Attention au decalage !
    return [L[i]/N for i in range(len(L))]

#Programme Principal
N=int(input("Combien de lancers ? "))
X=[i for i in range(3,19)]
Y=frequence(N)

import matplotlib.pyplot as plt
plt.figure()
plt.bar(X,Y,width=0.5)
plt.title("Loi de probabilite simulee de S")
plt.xlabel('Valeurs de la somme du lancer des 3 des')
plt.ylabel('Frequences simulees')
plt.grid()
plt.show()
```



Notons que puisque nous avons calculé la loi exacte de $S$, il est facile de tracer son histogramme. Notamment en utilisant la bibliothèque numpy (exercice laissé au lecteur).

**Loi simulée de $X$ :** Explicitons d'abord un script donnant la loi simulée de $X$ et sa représentation sous forme d'histogramme pour 10000 essais.

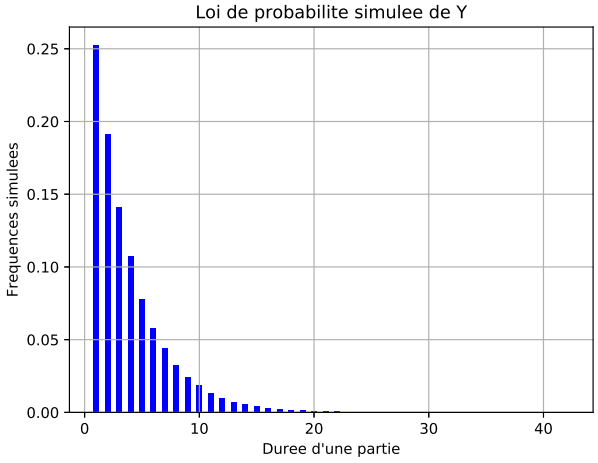
```
import matplotlib.pyplot as plt

def unePartie():
    from random import randint
    S=sum([randint(1,6) for i in range(3)])
    while 8<=S<=14:
        S=sum([randint(1,6) for i in range(3)])
    if S<=7:
        return 1
    else:
        return 0

def frequence(N):
    Gain=0      #nombre de gains et nombre d'essais avant fin de partie
    for i in range(N):
        Gain+=unePartie()
    return Gain/N

#Programme principal
N=int(input("Nombre de parties ? "))
#print("La duree moyenne d'une partie est de : ",frequence(N)[1])
X=[0,1] #0 pour perte et 1 pour gain
Y=[1-frequence(N),frequence(N)]

plt.figure()
plt.bar(X,Y,width=0.4, color='b')
plt.title("Loi de probabilite simulee de X")
plt.xlabel('Perte :0 ou Gain : 1')
plt.ylabel('Frequences simulees')
plt.grid()
plt.show()
```



Pour 10000 parties, nous trouvons une fréquence de gain d'environ 0,36. Ce qui corrobore notre calcul initial à l'aide du graphe probabiliste.

**Loi simulée de $Y$ :** Observons maintenant la loi de $Y$ (variable aléatoire qui peut a priori prendre une infinité de valeurs : 1, 2, 3, ...) et estimons son espérance.

```
import matplotlib.pyplot as plt
import numpy as np

def unePartie():
    from random import randint
    S=sum([randint(1,6) for i in range(3)])
    compteur=1      #nombre de coups par partie
    while 8<=S<=14:
        S=sum([randint(1,6) for i in range(3)])
        compteur+=1
    return compteur

def loi(N):
    L=[unePartie() for i in range(N)]
    return np.bincount(L,minlength=20)[1:]/N #explications plus tard

def frequence(N):
    coups=0     #duree d'une partie
    for i in range(N):
        coups+=unePartie()
    return coups/N

#Programme principal
N=int(input("Nombre de parties ? "))
print("La duree moyenne d'une partie est de : ",frequence(N))
Y=loi(N)
X=[i for i in range(1,len(Y)+1)]
plt.figure()
plt.bar(X,Y,width=0.5, color='b')
plt.title("Loi de probabilite simulee de Y")
plt.xlabel("Duree d'une partie")
plt.ylabel('Frequences simulees')
plt.grid()
plt.show()
```



Pour 10000 parties, nous trouvons une fréquence de durée d'environ 3,92. Ce qui corrobore notre calcul initial à l'aide de la seconde règle de la valeur moyenne ou des fonctions génératrices.

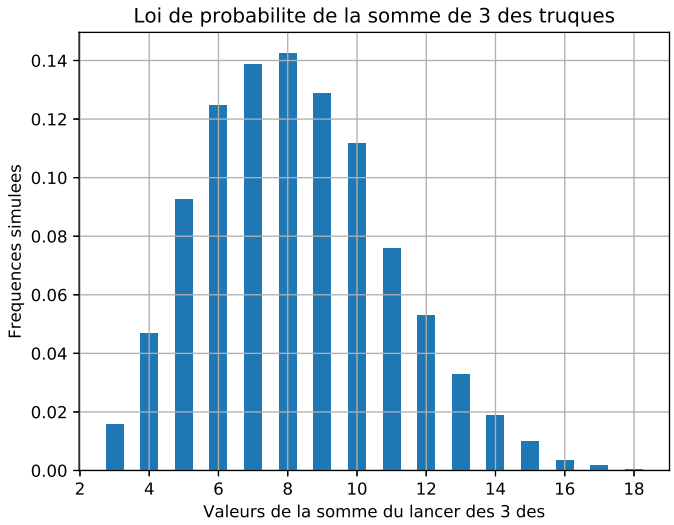
## 2.2 Sans l'hypothèse d'équiprobabilité

Première généralisation : nous supposons notre dé truqué de la manière suivante :
La probabilité de faire 1, 2 ou 3 est de 1/4 ; celle de faire 4, 5 ou 6 de 1/12.
Il convient de modifier les scripts précédents pour mettre en œuvre ceci. La notion de liste est alors mise à profit, ainsi que les méthodes qui s'y rattachent : liste définie par compréhension, somme, slicing.

**Loi de $S$ :** Commençons par la loi de $S$.

```
#Simulation de la loi de S : somme de trois des truques
def De():
    from random import random
    L=[1/4,1/4,1/4,1/12,1/12,1/12] #probas d'obtenir 1, 2, 3, 4, 5, 6
    C=[sum(L[:i+1]) for i in range(len(L))] #probas cumulees croissantes
    alea=random()            #jet du de
    for i in range(len(C)):
        if alea<=C[i]:
            return i+1       #numero sorti
            break    #on sort de la boucle des la premiere condition vraie

def Lancer3Des():
    return sum([De() for i in range(3)]) #somme obtenue pour un lancer

def frequence(N):                #Frequence d'apparition de chaque somme
    L=[0 for i in range(3,19)]   #Sommes de 3 a 18 inclus
    for i in range(N):
        alea=Lancer3Des()        #Jet de 3 des
        for j in range(3,19):
            if alea==j:
                L[j-3]+=1      #Attention au decalage !
    return [L[i]/N for i in range(len(L))]

#Programme Principal
N=int(input("Combien de lancers ? "))
X=[i for i in range(3,19)]
Y=frequence(N)

import matplotlib.pyplot as plt
plt.figure()
plt.bar(X,Y,width=0.5)
plt.title("Loi de probabilite de la somme de 3 des truques")
plt.xlabel('Valeurs de la somme du lancer des 3 des')
plt.ylabel('Frequences simulees')
plt.grid()
plt.show()
```

Avec 10000 expériences, nous obtenons l'histogramme des fréquences suivant (loi simulée de S) :



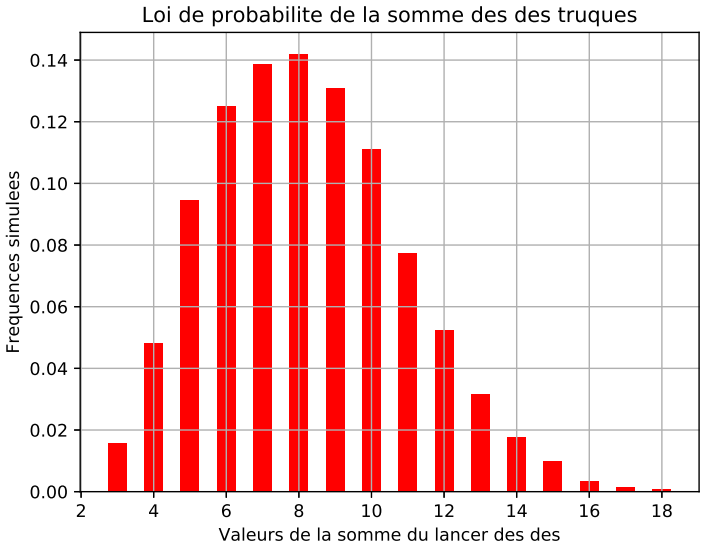

Mais il y a beaucoup plus rapide grâce aux méthodes spécifiques de la bibliothèque (ou module) numpy. Détaillons celles qui vont être mises en œuvre dans le script qui suit :

1. La bibliothèque numpy possède une "étagère" random (module pour les générateurs aléatoires), avec plein de livres traitant sur le hasard et ses simulations. Cette étagère possède à son tour un "livre" qui porte le même nom : random. Ce livre, en fait une fonction, permet de simuler la loi uniforme sur $[0;1[$. Par exemple, la commande np.random.random() renverra un réel pseudo-aléatoire compris entre 0 et 1. Mais il y a d'autres options :
   (a) On peut utiliser un alias pour simplifier l'écriture : import numpy.random as npr. Ainsi, npr.random() équivaut à numpy.random.random() !
   (b) npr.random(a,b) renvoie une matrice $a \times b$ avec tirages uniformes dans $[0\,;1]$.
2. L'instruction npr.choice($[a_1, a_2, \ldots, a_n]$, $p = [p_1, p_2, \ldots, p_n]$, size = (L,C) ) génère une matrice $L \times C$ de tirages indépendants dans $A = [a_1, a_2, \ldots, a_n]$, de loi $[p_1, p_2, \ldots, p_n]$,
3. L'instruction np.sum(tableau $L \times C$, axis=0) renvoie un tableau 1-D constitué des sommes des colonnes du tableau d'entrée,
4. L'instruction np.bincount(tableau 1-D, options) compte le nombre d'occurrences d'un tableau 1-dimensionnel d'*entiers naturels*.

```
import numpy as np
import numpy.random as npr
import matplotlib.pyplot as plt

def frequence(k,N):    #k nombre de des
    tirage=npr.choice(np.arange(1,7),p=[1/4,1/4,1/4,1/12,1/12,1/12],
    size=(k,N))
    #simule N lancers de k des avec la loi du de
    #on obtient une matrice k*N
    lancer=np.sum(tirage,axis=0) #l'option axis=0 somme chaque colonne
    return np.bincount(lancer,minlength=6*k+1)[k:]/N
    #le decalage est traite comme tel

#Programme Principal
k=3
N=int(input("Combien de lancers ? "))
```

```
### un tableau numpy offre plus de souplesse qu'une liste
#X=tuple(i for i in range(k,6*k+1)) est remplace par :
X=np.arange(k,6*k+1)
Y=frequence(k,N)

plt.figure()
plt.bar(X,Y,width=0.5, color='r')
plt.title("Loi de probabilite de la somme des des truques")
plt.xlabel('Valeurs de la somme du lancer des des')
plt.ylabel('Frequences simulees')
plt.grid()
plt.show()
```

Avec 10000 expériences, nous obtenons l'histogramme des fréquences suivant (loi simulée de S) :

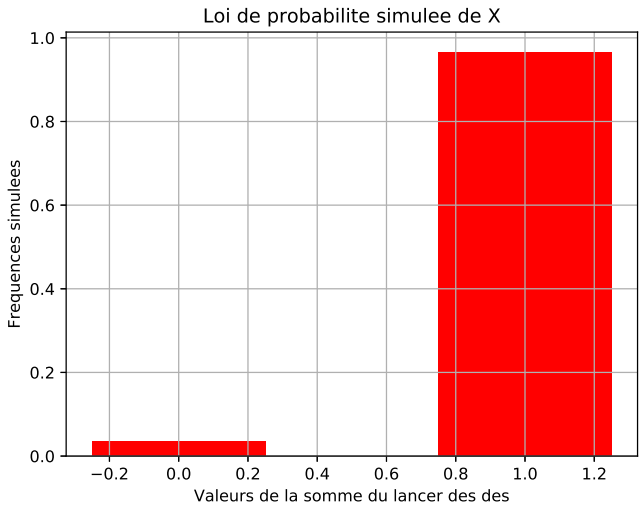



Nous pouvons maintenant simuler notre jeu de manière optimale en créant les lois simulées des variables aléatoires $X$ (1 si gain et 0 si perte), et $Y$ : durée d'une partie.

**Loi de $X$ :** Petite modification du script avec équiprobabilité.

```
#Loi de X avec 3 des truques
import numpy as np
import numpy.random as npr
import matplotlib.pyplot as plt

def De():
    from random import random
    L=[0,1/4,1/4,1/4,1/12,1/12,1/12] #probas d'obtenir 0, 1, 2, 3, 4, 5, 6
    #il est important de commencer a zero
    C=np.cumsum(L)                   #probas cumulees croissantes
    alea=random()                    #jet du de
    return np.searchsorted(C,alea) #classification du numero obtenu

def LancerkDes(k):
    return sum([De() for i in range(k)]) #somme obtenue pour un lancer de k des
```

```
def unePartie(k):
    S=LancerkDes(k)
    while 8<=S<=14:
        S=LancerkDes(k)
    if S<=7:
        return 1
    else:
        return 0

def frequence(k,N):
    Gain=0     #nombre de gains et nombre d'essais avant fin de partie
    for i in range(N):
        Gain+=unePartie(k)
    return Gain/N

#Programme principal
k=3
N=int(input("Nombre de parties ? "))
X=[0,1]
Y=[1-frequence(k,N),frequence(k,N)]

plt.figure()
plt.bar(X,Y,width=0.5, color='r')
plt.title("Loi de probabilite simulee de X")
plt.xlabel('Valeurs de la somme du lancer des des')
plt.ylabel('Frequences simulees')
plt.grid()
plt.show()
```




**Loi de $Y$ :** (mini-)Défi 1.

# 3 Somme de $k$ dés

## 3.1 Aspect théorique (cas équiprobable)

Le raisonnement effectué en lançant trois dés se généralise aisément au cas de $k \geq 4$ dés parfaits.
Nous pouvons mettre en bijection l'ensemble des solutions de l'équation entière $x_1 + x_2 + \cdots + x_k = n$ avec l'ensemble des mots constitués de $k-1$ séparateurs $|$ et de $n$ cercles $\circ$.
Le cardinal de l'ensemble de ces mots est égal à $D_{k,n} = \binom{n+k-1}{k-1}$ (choix des places des $k-1$ séparateurs parmi les $n+k-1$ places possibles).
Comme avant, ce résultat implique que des variables $x_i$ peuvent être nulles, ce qui n'est pas le cas ici. Nous devons avoir $1 \leq x_i \leq 6 \;\; (1 \leq i \leq k)$. Il convient donc de modifier le résultat précédent.

Considérons maintenant l'équation :

$$y_1 + y_2 + \cdots + y_k = n, \;\; y_i \in \mathbb{N}^*, n \geq k \;\; (*)$$

Il est clair qu'en posant $x_i = y_i - 1$, on se ramène à l'équation précédente (1). Le nombre $D^*_{k,n}$ des solutions de (*) est alors :

$$D^*_{k,n} = \binom{n-1}{k-1}$$

(on a remplacé $n$ par $n-k$).

**Loi de la somme S de nos $k$ dés :** Ce qui précède, nous permet de conclure par équiprobabilité que : $\begin{cases} S(\Omega) = \{k; k+1; k+2; \ldots; 6k\} \\ \forall n \in S(\Omega) \;\; P(S=n) = \dfrac{\binom{n-1}{k-1}}{6^k} \end{cases}$

## 3.2 Simulation

**En situation d'équiprobabilité :** Modifier le script avec 3 dés.

**En situation de non équiprobabilité :** Le script correspondant au lancer de $k$ dés a déjà été donné précédemment. Il suffit de changer la valeur de $k$ pour obtenir l'histogramme correspondant.

# 4 Exercices

## 4.1 Énoncés

**Exercice 1 :** Une urne contient 2 boules rouges et 3 boules noires. On prélève successivement et sans remise une boule de cette urne jusqu'à temps que son contenu soit unicolore. Soit T la variable aléatoire temps d'attente (nombre de tirages nécessaires).

1. Simulez à l'aide de l'approche fréquentiste la loi de T. Vous testerez ceci sur 10000 expériences et afficherez le résultat sous la forme d'un diagramme en barres, ainsi que le nombre moyen de tirages effectués.
2. Retrouvez le résultat de manière théorique.

**Exercice 2 :** On lance une pièce biaisée. La probabilité de faire Pile est de $p$. On posera $q = 1 - p$, probabilité de faire face. $X$ est la variable aléatoire réelle égale au nombre d'essais nécessaires pour obtenir pour la première fois une suite de trois Piles consécutifs.

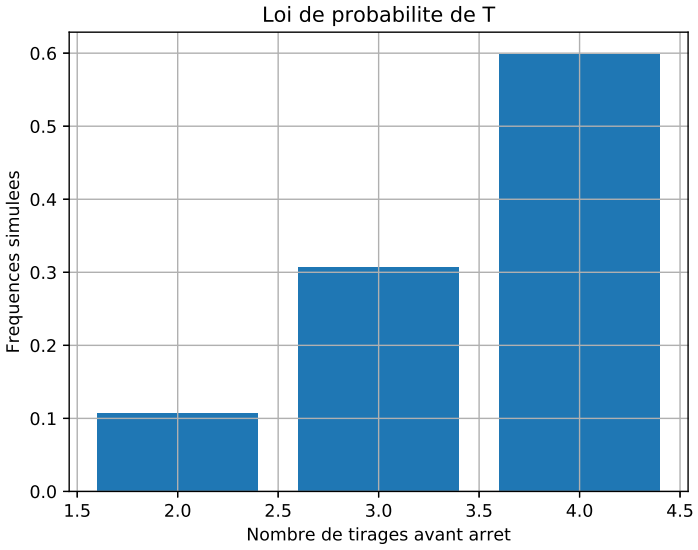
1. Dessiner un graphe probabiliste modélisant l'expérience précédente. En déduire le graphe d'un parcours non marqué.
2. En réduisant le graphe précédent, donner l'expression de la fonction génératrice de $X$.
3. Généraliser au cas de $n$ Piles consécutifs.
4. En déduire en fonction de $n$ et de $p$ le temps moyen d'attente. Appliquer avec $n = 3$ et $p = 0,25$.
5. Écrire un script qui demande à l'utilisateur :
   (a) de saisir la probabilité $p$ de faire Pile,
   (b) de saisir le nombre $n$ de Piles consécutifs souhaité,
   (c) Et qui simule la variable aléatoire temps moyen $T_{n,p}$ d'attente sur $N = 10000$ essais. Appliquer avec $n = 3$ et $p = 0,25$.

## 4.2 Solutions

**Exercice 1 :** Nous donnons ici un script "pur Python".

```
#Une urne contient 2 boules rouges et 3 boules noires
#on preleve successivement et sans remise 1 boule de cette urne
#jusqu'a temps que son contenu soit unicolore.
#Soit T la variable temps d'attente. Nous simulerons la loi approchee de T
#et en deduirons une valeur approchee de E(T).

def experience():
    from random import randint
    compteur=0
    nb_noires,nb_rouges=3,2
    while (nb_noires!=0 and nb_rouges!=0):
        alea=randint(1,nb_noires+nb_rouges)
        if alea<=nb_noires:
            nb_noires-=1
        else:
            nb_rouges-=1
        compteur+=1
    return compteur

def frequence(N):
    L=[0,0,0]
    for i in range(N):
        alea=experience()
        for j in range(2,5):
            if alea==j:
                L[j-2]+=1
    return [el/N for el in L]

#Programme Principal
N=int(input("Combien de lancers ? "))
X=tuple(i for i in range(2,5))
Y=frequence(N)

#nombre moyen de tirages effectues
```

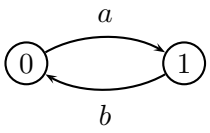
```
nb_tirages=0
for i in range(N):
    nb_tirages+=experience()
print("nombre moyen de tirages : ",nb_tirages/N)

#Histogramme de la loi simulee de T
import matplotlib.pyplot as plt
plt.figure()
plt.bar(X,Y,width=0.5)
plt.title("Loi de probabilite de T")
plt.xlabel('Nombre de tirages avant arret')
plt.ylabel('Frequences simulees')
plt.grid()
plt.show()
```



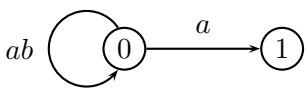

La solution théorique n'est pas lourde à mettre en œuvre (un arbre de probabilités convient) et nous la laissons à la sagacité du lecteur. Vous devez trouver :

| $x_i$ | 2 | 3 | 4 |
|---|---|---|---|
| $P(X = x_i)$ | 0,1 | 0,3 | 0,6 |

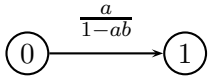
**Exercice 2 :** Donnons d'abord un résultat préliminaire très utile. Nous laissons sa démonstration à la sagacité du lecteur.
Le graphe :



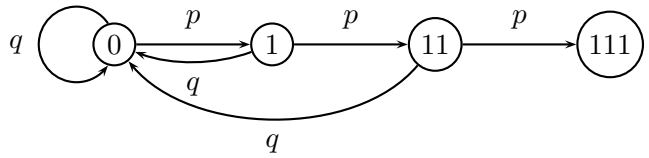

équivaut à :



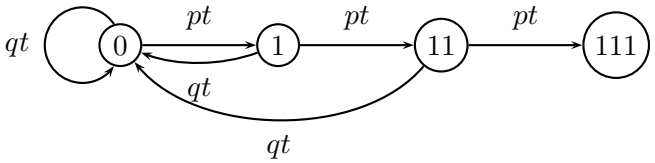

équivaut à :



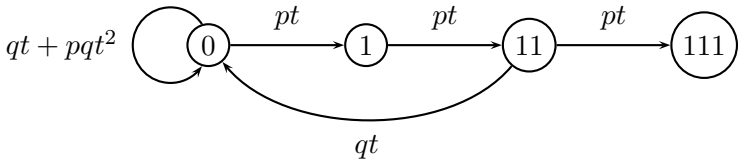

**Question 1** :
Nous poserons 0 pour Face et 1 pour Pile. Sans perte de généralité, on peut supposer que l'état initial est 0. Le graphe probabiliste associé à notre expérience aléatoire prend alors la forme :



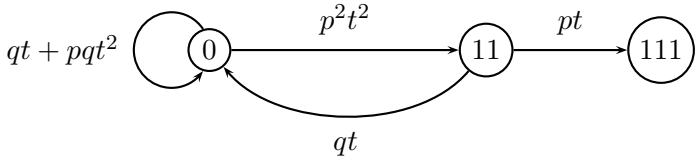

**Question 2** : Il suffit de multiplier chacune des probabilités indiquées par $t$, la probabilité de ne pas être marqué en faisant tourner la roue de Van Dantzig.



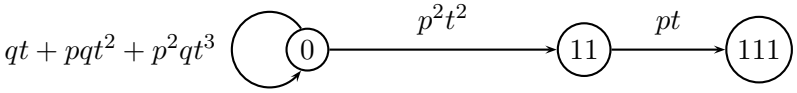

équivaut à :



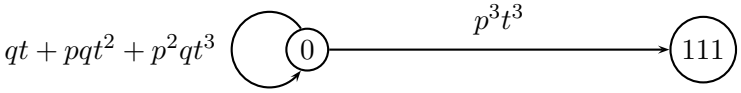

lui-même équivalent à :



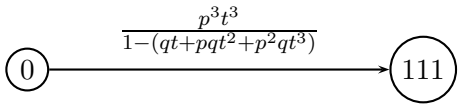

qui équivaut à :




équivaut à :



équivaut à :




De plus, $qt + pqt^2 + p^2qt^3 = \dfrac{qt(1-p^3t^3)}{1-pt}$, d'où :

$$\phi_X(t) = \frac{(pt)^3(1-pt)}{1-pt-qt(1-(pt)^3)} = \frac{(pt)^3(1-pt)}{1-t+qt(pt)^3}$$

soit :

$$\phi_X(t) = \frac{(1-pt)p^3t^3}{1-t+qp^3t^4}$$

**Question 3** :

On laisse le lecteur se persuader que $\phi_X(t) = \dfrac{(1-pt)p^nt^n}{1-t+qp^nt^{n+1}}$.

**Question 4** :

Le développement en série entière de $\phi_X(t)$ est laborieux. Ceci dit, le calcul de $\phi'_X(1)$, même plus simple, l'est aussi. X-Cas étant notre ami, demandons-lui un petit coup de pouce !

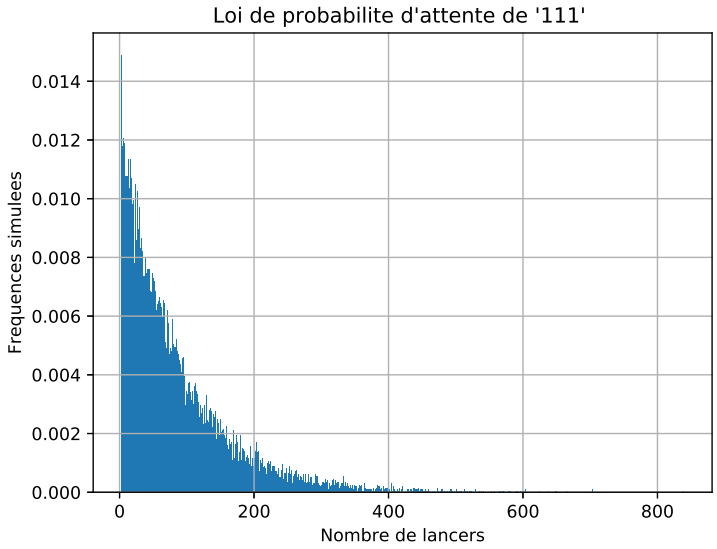
$E(X) = \phi'_X(1) = \dfrac{1-p-qp^n}{q^2p^n} = \dfrac{1-p^n}{qp^n}$.

Avec $n = 3$ et $p = 0,25$, nous trouvons un temps moyen d'attente de 84 lancers.

**Question 5** : Nous allons utiliser un peu numpy.

```
#Temps d'attente du motif 111...11 (n fois "1")
#probabilite a chaque lancer de faire "1" : p
import numpy as np
import numpy.random as npr
import matplotlib.pyplot as plt

def Lancers(p,n):
    compteur,motif,chaine=0,"1"*n,""
    while chaine.count(motif)==0:
        alea=npr.random()
        if alea<=p:
            chaine+="1"
        else:
            chaine+="0"
        compteur+=1
    return compteur

def loi(p,n,N):
    L=[Lancers(p,n) for i in range(N)]
    return np.bincount(L,minlength=20)[n:]/N

#Programme Principal
n,p=3,0.25
N=int(input("Combien de lancers ? "))
```

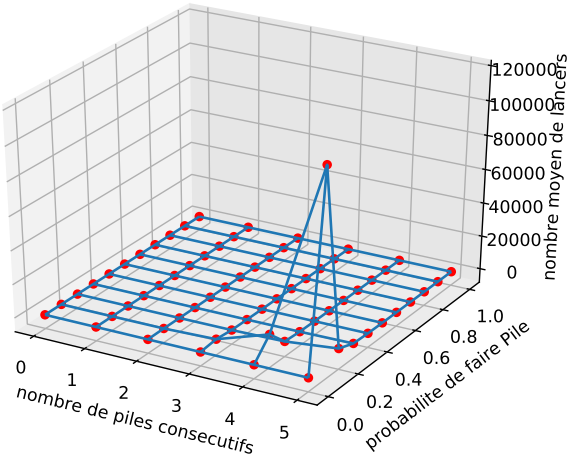
```
Y=loi(p,n,N)
X=np.arange(n,len(Y)+n)

plt.figure()
plt.bar(X,Y,width=0.8)
plt.title("Loi de probabilite d'attente de '111' ")
plt.xlabel('Nombre de lancers')
plt.ylabel('Frequences simulees')
plt.grid()
plt.show()
```




**Bonus** : Le tracé de la surface qui pour $N = 1000$ essais, avec $n$ variant de 0 à 5, et $p$ variant de 0 à 1 (pas de 0,1), renvoie le temps moyen d'attente. Ceci utilise matplotlib en 3D !

```
import numpy as np
import numpy.random as npr
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

def Lancers(p,n):
    compteur,motif=0,0
    while motif<n:
        alea=npr.random()
        if alea<=p:
            motif+=1
        else:
            motif=0
        compteur+=1
    return compteur          #le nombre d'essais avant arret

def frequence(p,n,N):
    attente=0
    for i in range(N):
        attente+=Lancers(p,n)
    return attente/N         #nombre moyen d'essais sur N experiences
```

```
#Programme principal
x=np.linspace(0,5,num=6)     #nombre de Piles consecutifs (0 a 5)
y=np.linspace(0,1,num=11)    #probabilite de faire Pile
N=1000                       #nombre d'experiences
tableau1=np.zeros((6,11))
np.set_printoptions(precision=3, suppress=True)
for n in x:
    for p in y:
        if int(n)==0 or p==0.: #pour eviter une boucle infinie
            tableau1[int(n),int(10*p)]=0
        else:
            tableau1[int(n),int(10*p)]=frequence(p,int(n),N)
fig=plt.figure()
ax=fig.add_subplot(111,projection='3d')
xv,yv=np.meshgrid(x,y)
ax.scatter(xv,yv,tableau1.T,c='r',marker='o')
ax.plot_wireframe(xv,yv,tableau1.T,rstride=1,cstride=1)
ax.set_xlabel('nombre de piles consecutifs')
ax.set_ylabel('probabilite de faire Pile')
ax.set_zlabel('nombre moyen de lancers')
plt.show()
```




Nous n'avons pas utilisé de chaine de caractère dans ce script modifié (plus rapide : merci Jean-Michel) !

# 5 Annexes

## 5.1 Graphes probabilistes

Nous sommes habitués lors de la modélisation probabiliste d'un énoncé faisant intervenir une expérience aléatoire, d'introduire les notions d'univers, d'événement et de variable aléatoire (réelle ou vectorielle). Il en ressort néanmoins une impression statique de l'expérience aléatoire considérée. Le point de vue qui sera adopté ici est celui de la dynamique, avec une évolution dans le temps.
Considérons par exemple la trajectoire d'une poussière sur la surface d'une nappe d'eau. On

peut découper cette surface en $n$ carrés élémentaires puis observer la présence de la poussière dans chacun de ces carrés au cours du temps, qui lui-même peut être discrétisé. Nous obtenons alors ce que nous définirons plus loin comme un *processus aléatoire discret*. Ainsi, nous qualifierons plus volontiers l'univers $\Omega$ d'*espace des états*, en référence aux systèmes dynamiques, plutôt qu'ensemble des événements.
L'étude des transitions de la particule d'un état à l'autre a des représentations commodes : à l'aide de graphes orientés et pondérés ou bien matriciellement. Ces deux approches sont complémentaires et l'utilisation de l'une plutôt que l'autre dépend avant tout du cas considéré. Notons que généralement, il est préférable de commencer par l'approche graphe probabiliste avant de passer à l'approche matricielle dans un souci de visualisation des états et de leurs transitions.

### 5.1.1 Arbres de probabilités

Nous sommes tous familiers de l'utilisation d'arbres de probabilités pour modéliser une situation dynamique : tirages successifs avec ou sans remise notamment. Rappelons les règles usuelles :

**Règle 1 :** La probabilité, partant d'un nœud donné de l'arbre de réaliser un parcours donné, est égale au produit de toutes les probabilités de transition (inscrites sur les segments) le long de ce parcours.

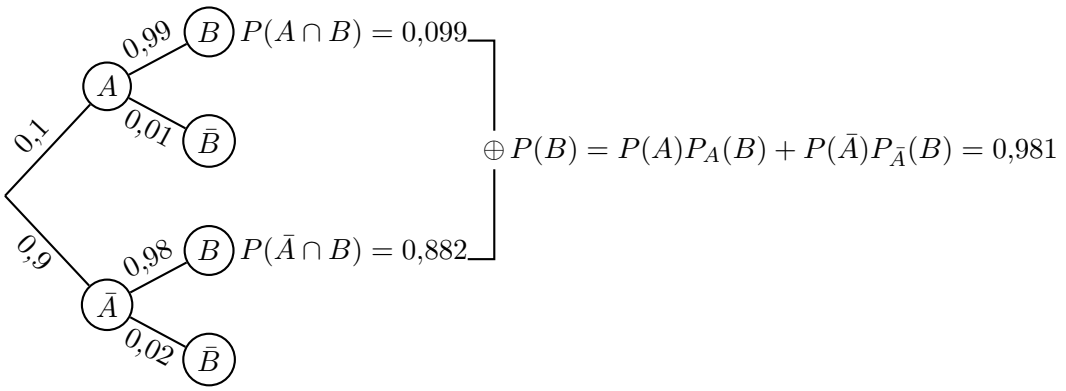
**Règle 2 :** La probabilité d'aller de $A$ à $B$ est la somme des probabilités de tous les chemins conduisant de $A$ à $B$.

**Règle 3 :** La somme des probabilités des segments issus d'un même nœud est égale à 1.

**Remarque :**

1. La règle 1 n'est rien d'autre que la traduction de la formule des probabilités composées : Si $P(A_1 \cap A_2 \cap \cdots \cap A_n) > 0$, alors :
$$P(A_1 \cap A_2 \cap \cdots \cap A_{n+1}) = P(A_1)P_{A_1}(A_2)P_{A_1 \cap A_2}(A_3) \ldots P_{A_1 \cap \ldots A_n}(A_{n+1})$$
2. La règle 2 n'est que la traduction de $P(C) = \sum_{\{i; x_i \in C\}} P(x_i)$, où $\Omega = \{x_i \ ; \ i \in I\}$.
3. La règle 3 est la traduction de la formule des probabilités totales.

**Exemple :** Un détaillant achète ses produits chez deux fournisseurs dont le premier, noté A lui fournit 10% de ses articles. Parmi les articles fournis par A, 99% n'ont pas de défaut de fabrication : événement B. Parmi les articles fournis par le second fournisseur, 98% n'ont pas de défaut de fabrication.
On prélève au hasard un article du stock du détaillant. Calculez la probabilité qu'il n'ait pas de défaut de fabrication.




Le calcul sus-mentionné est une application immédiate de la formule des probabilités totales. Avantage de l'arbre : nous percevons la dynamique du processus ! Généralisons-donc un peu... de manière heuristique en gardant les mêmes règles de parcours.

### 5.1.2 Un exemple modèle

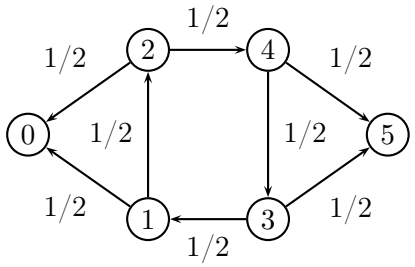
Yom suait à grosses gouttes. Faut dire qu'il n'avait pas fait les choses à moitié en marchant activement depuis le ministère de l'agriculture, situé à deux pas des Invalides. L'été parisien commençait tout juste à poindre le bout de son nez qu'il en avait asséché sa gorge. Vite, se désaltérer... Une terrasse accueillante près de l'hôtel de ville lui tendait les bras. La chaise en osier craqua subrepticement lorsqu'il s'assit nonchalamment en étirant ses membres alourdis. Ses lèvres frémissantes saluèrent le demi pression qu'il savoura goulument... jusqu'au moment fatidique de l'addition ! Cinq euros !
"Bigre ! Diantre ! Purée de forficules ! Paris sera toujours Paris !" se dit-il en essuyant sa moustache du revers de l'index.
Il farfouilla dans sa poche pour y trouver sa bourse en cuir élimée par les ans. Mais celle-ci ne contenait plus qu'un seul et malheureux euro. En regardant le nom du troquet : "Pie et Matou", un frisson glacé parcourut son échine. Un de ses collègues venant d'Ardèche et dans la dèche y avait vécu un moment douloureux face aux trois taulières : Brigida, dite Bri la géomètre ; Christelle, dite Cricri l'arène, et Nathalie, dite Nath l'Aligot.
Mais les trois amazones semblèrent compréhensives lorsqu'il leur déclara ne pouvoir leur restituer leur dû qu'à hauteur d'un euro ; d'autant que les distributeurs alentours étaient en panne comme de par hasard !

— C'est pas grave mon biquet, déclara Brigida un sourire en coin. Je te propose un petit jeu qui va te plaire.
— Lequel ? demanda Yom les lèvres sèches.
— J'ai une pièce parfaitement honnête comme moi, dit Brigida d'un air entendu. Tu vas la lancer autant de fois que nécessaire pour gagner mon dû de cinq euros. Si tu fais Pile, tu gagnes la manche, sinon tu la perds. Voici les règles : tant que tu disposes d'un ou deux euros, tu joues ton pécule. Si tu as trois ou quatre euros, tu joues le complément à cinq euros... à moins que tu ne perdes entre temps. Mais si tu perds...
— Quoi ? interrogea Yom le regard hagard.
— Je réaliserai ton rêve d'artiste : tu entreras en Seine !

Yom sortit un papier de sa poche afin de calculer ses chances de rester au sec. Rien de tel qu'un bon petit graphe pour se détendre. Voici ce qu'il scribouilla :

L'état de départ est noté 1 (comme un euro, ce dont je dispose) et ceux de fin (appelés états absorbants) sont notés 0 et 5 (comme zéro et cinq euros). Entre-temps, je peux passer par les

états 2, 4 et 3. En construisant le graphe petit à petit en fonction du résultat obtenu à chaque tour, j'obtiens donc :




Ok ! Ok ! Les chemins menant au gain sont ceux qui partant de 1 ont pour terminaison 5, soit :

— $C_1 :\ 1 \longrightarrow 2 \longrightarrow 4 \longrightarrow 5$
— $C_2 :\ 1 \longrightarrow 2 \longrightarrow 4 \longrightarrow 3 \longrightarrow 5$

tout ceci précédé de $n$ boucles, où $n$ est un entier naturel éventuellement nul et la boucle le chemin $1 \longrightarrow 2 \longrightarrow 4 \longrightarrow 3 \longrightarrow 1$, de probabilité $\left(\frac{1}{2}\right)^4 = \frac{1}{16}$.
Bon, ça m'étonnerait que je boucle une infinité de fois, mais je dois quand même le prendre en considération. Résumons donc tout ça...

$$P(\text{Gain}) = \sum_{n=0}^{+\infty} \underbrace{\left(\frac{1}{16}\right)^n \times \frac{1}{8}}_{n \text{ boucles suivi de } C_1} + \sum_{n=0}^{+\infty} \underbrace{\left(\frac{1}{16}\right)^n \times \frac{1}{16}}_{n \text{ boucles suivi de } C_2} = \frac{1}{5}.$$

Comme dirait Cambronne : M. . .E ! Ca sent la mise en bière tout ça ! marmonna Yom. Je suis vraiment mouillé dans une drôle d'histoire !

### 5.1.3 Chaines de Markov

1. On note $S$ l'ensemble des *états* qu'il est possible de visiter au cours de notre suite d'expériences aléatoires. On supposera que cet ensemble est fini ou infini dénombrable.
2. On appelle *probabilité de transition* de $i$ vers $j$, et on note $p_{ij}$ la probabilité de passer de l'état $i$ à l'état $j$ au cours d'un pas de temps. A priori, $p_{ij}$ dépend de $n$ et l'on devrait noter $p_{ij}(n)$, mais nous travaillerons uniquement sur des probabilités de transition indépendantes de l'instant considéré.
3. Un état $i$ est dit *absorbant* s'il vérifie $p_{ii} = 1$. On note $B$ l'ensemble des états absorbants de $S$ et on l'appelle *le bord* de $S$.
4. On appelle *état intérieur* un élément de $S \setminus B$.

**Définition :**

— Définition naïve : La donnée de $S$, ensemble des états, des $p_{ij}$, probabilités de transitions entre états, ainsi que de l'état initial $(a_0, a_1, \dots)$ définit une *chaine de Markov*.
— Définition plus rigoureuse : Soit $(X_n)_{n\geq 0}$ une suite de variables aléatoires à valeurs dans l'ensemble $S$ des états que l'on peut supposer égal à $\mathbb{N}$. On dit que cette suite est une *chaine de Markov* si pour tout entier $n \geq 1$ et toute suite $(i_0, \dots, i_{n-1}, i, j)$ d'éléments de $S$ tel que $P(B_n) \overset{\text{def}}{=} P(X_0 = i_0 \cap \dots \cap X_{n-1} = i_{n-1} \cap X_n = i) > 0$, on ait $P_{B_n}(X_{n+1} = j) = P_{X_n=i}(X_{n+1} = j)$.
On peut comprendre ceci comme : dans l'évolution au cours du temps, l'état du processus à l'instant $n+1$ ne dépend que de celui-ci à l'instant $n$ précédent, mais non de ses états antérieurs. Le processus est *sans mémoire*.

**Définition :**

1. Une chaine de Markov est dite *absorbante* si son bord $B$ est non vide : il y a au moins un état absorbant.
2. Une chaine de Markov est dite *homogène* (en temps) si la probabilité $P_{X_n=i}(X_{n+1}=j)$ ne dépend pas de $n \geq 0$. On la note $p_{ij}$ et on l'appelle *probabilité de transition* (en une étape) de l'état $i$ à l'état $j$.

**Remarque :** Dans ce chapitre, nous étudierons les deux cas : chaines de Markov absorbantes, et chaines sans bord. Dans tous les cas, elles seront supposées homogènes.

**Propriété :**

1. Pour tout couple d'entiers $(i,j)$ on a $p_{ij} \geq 0$,
2. Pour tout $i \in S$, $\sum_{j \in S} p_{ij} = 1$.

**Remarque :** La propriété précédente ne dit rien d'autre que pour chaque $i \in S$, l'application $B \mapsto \sum_{j \in S} p_{ij}$ définit une mesure de probabilité sur $S$.

**Règles de parcours** Les règles présentées ici ont une importance pratique capitale ! La règle 3, dont nous donnerons une autre version ultérieurement, se différencie des deux autres qui sont identiques à celles présentées pour les arbres de probabilité, car elle permet de calculer non pas une probabilité, mais une durée moyenne de parcours *i.e* une espérance.

**Règle 1 :** La probabilité, partant d'un état donné du graphe de réaliser un parcours donné, est égale au produit de toutes les probabilités de transition le long de ce parcours.

**Règle 2 :** La probabilité, partant d'un état intérieur $i$ donné, d'atteindre un quelconque sous ensemble $T$ du bord $B$ est égale à la somme des probabilités de tous les chemins menant de $i$ à $T$. Cette règle reste valable entre deux états intérieurs $i$ et $j$.

**Règle 3 :** La durée moyenne $m_i$ des parcours aléatoires allant de l'état $i$ au bord $B$ est la moyenne pondérée des longueurs des parcours de $i$ à $B$, chaque longueur de parcours $\ell_k$ étant pondérée par la probabilité $p_k$ de ce parcours.

**Règles de la valeur moyenne** Ces outils permettent de simplifier les règles de parcours présentées précédemment. Notamment la règle 3 de durée moyenne de parcours. On note $S = \{1, 2, \ldots, n\}$ l'ensemble des états.

**Définition :** On appelle *fonction de probabilité* la fonction définie sur $S$ à valeurs dans $[0;1]$, qui à chaque état $i$ associe sa probabilité d'être absorbée dans un sous-ensemble $T \subset B$. On la note $p_i^{(T)}$ où, si aucune confusion n'est à craindre $p_i$.

La formule des probabilités totales nous dit alors que pour tout état intérieur $i$ :

$$p_i = \sum_{k=1}^{n} p_{ik} p_k$$

Pour le bord : $p_i = 1$ si $i \in T$ et $p_i = 0$ si $i \in B \setminus T$.

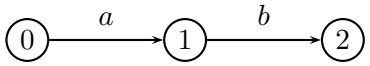
**Théorème fondamental :**

1. Première règle de la valeur moyenne : La valeur de la fonction de probabilité en un état intérieur $i$ est la moyenne pondérée de ses valeurs en les états voisins de $i$.
2. **Seconde règle de la valeur moyenne** : La valeur du délai d'absorption en un état intérieur $i$ est de 1 plus la moyenne pondérée des délais d'absorption en les états voisins.

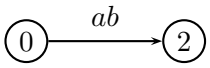
### 5.1.4 Réduction de graphes

Les premières et secondes règles de parcours nous permettent de simplifier avantageusement certains graphes probabilistes. Citons notamment :

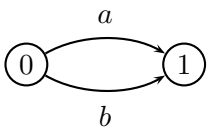
**(a) Suppression d'un nœud :** La règle 1 de parcours se traduit par :



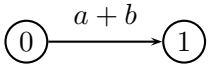

équivaut à :



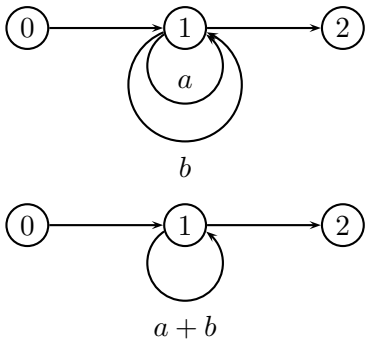

**(b) Réunions de branches en parallèles :** La règle 2 de parcours se traduit par :




équivaut à :



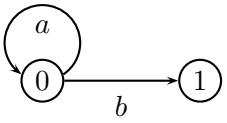

**(c) Réunions de boucles issues d'un même nœud :** Ce n'est qu'un cas particulier de ce que nous avons vu en (b).



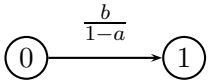

équivaut à :

**(d) Synthèse :** Simplification de graphes faisant intervenir des boucles :

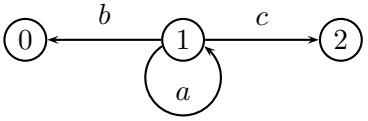
Cas fréquent 1 :



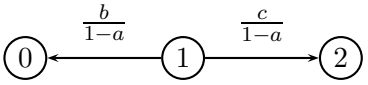

équivaut à :



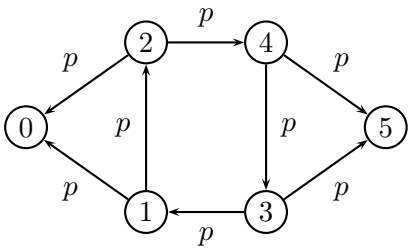

Cas fréquent 2 :



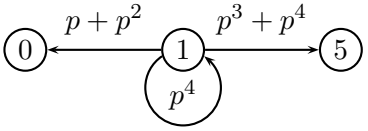

équivaut à :



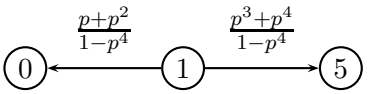

Appliquons ces résultats pour retrouver la probabilité de gain au jeu des taulières en simplifiant petit à petit le graphe. Nous poserons $p = 1/2$ :




L'état 1 est fondamental.
Regroupons les branches $1 \to 0$ et $1 \to 2 \to 0$ de probabilités respectives $p$ et $p^2$ qui mènent à l'état absorbant 0 ; les branches $1 \longrightarrow 2 \longrightarrow 4 \longrightarrow 5$ et $1 \longrightarrow 2 \longrightarrow 4 \longrightarrow 3 \longrightarrow 5$ de probabilités respectives $p^3$ et $p^4$ qui mènent à l'état absorbant 5, sans oublier la seule boucle du graphe $1 \longrightarrow 2 \longrightarrow 4 \longrightarrow 3 \longrightarrow 1$, de probabilité $p^4$. Nous obtenons le graphe suivant :




lui-même équivalent à :




On en déduit $P(\text{Gain}) = \frac{p^3+p^4}{1-p^4} = 0,2$. Pas de surprise !

**Théorème :** Si le bord $B$ d'une chaine de Markov est non vide, la probabilité, partant d'un état intérieur quelconque, d'atteindre $B$ est égale à 1.

## 5.2 Fonctions génératrices

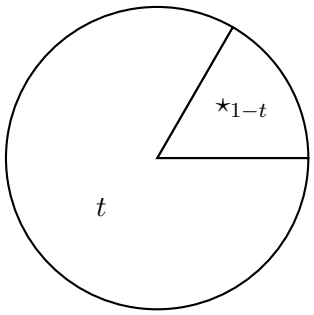
Considérons une chaine de Markov absorbante. Nous savons qu'une particule, partant d'un état donné, finit par être absorbée avec la probabilité 1.
Soit $X$ la variable aléatoire réelle donnant le nombre de transitions avant l'absorption.
$X(\Omega) = [0; N] \cap \mathbb{N}$ ou $X(\Omega) = \mathbb{N}$. Sans perte de généralité, on peut supposer $X(\Omega) = \mathbb{N}$ quitte à poser $P(X = i) = 0$ à partir d'un certain rang.

**Notation 1-1-1** : on posera $p_i := P(X = i)$ et $q_i := P(X > i)$. La fonction $i \mapsto q_i$ s'appelle *fonction de survie* de la particule pour des raisons évidentes.

**Remarque :** $E(X) = \sum_{i \geq 0} q_i$.
Décrivons maintenant la géniale méthode (dixit Maitre Yoda) inventée par le mathématicien hollandais *David Van Dantzig* (1900-1959) pour donner un sens aux séries entières $\phi(t) = \sum_{i \geq 0} p_i t^i$, où $(p_i)_{i \geq 0}$ est une distribution de probabilité. En l'occurrence, nous considérerons essentiellement ici la distribution de probabilité de la variable aléatoire $X$ précédente.

**Méthode de Van Dantzig :** Elle comporte deux étapes que nous allons décrire précisément. Considérons un état intérieur d'une chaine de Markov absorbante. C'est le point de départ de la particule.

1. Cette particule effectue une transition par unité de temps.
2. A chaque transition, on lance la roulette suivante :



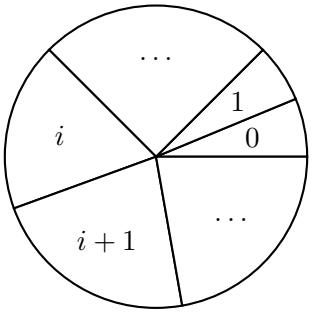

où la probabilité de tomber dans la zone marquée est $1 - t$ et celle de ne pas y tomber est $t$.

En vertu de la formule des probabilités totales, avec le système complet d'événements $\{(X = i)\}_{i \geq 1}$, la probabilité de l'événement $A$ : "avoir un parcours non marqué" est égale à

$$P(A) = \sum_{i \geq 1} P_{X=i}(A) P(X = i) = \sum_{i \geq 1} t^i p_i$$

par indépendance des lancers.
Notons que le conditionnement par un événement $(X = i)$ suppose qu'il ne soit pas de probabilité nulle, ce dont nous n'avons aucune idée pour $i \geq 1$. Le raisonnement est donc purement formel. D'autre part, étant donné que l'état initial est intérieur, on a $p_0 = 0$. On peut poser sans perte de généralité $\sum_{i \geq 0} p_i t^i$, expression que nous noterons $\phi_X(t)$.

**Remarque :** Dans certains cas, $X$ ne comptera que certaines transitions particulières (comme avec la loi binomiale). Plus généralement, donnons la méthode suivante :
on considère une variable aléatoire $X$ qui prend les valeurs 0, 1, 2, ... avec les probabilités $p_0$, $p_1, p_2, \ldots$Ceci se modélise par une roulette comme celle ci-dessous que l'on fait tourner une fois.




Si $X$ prend la valeur $i$, on fait ensuite tourner $i$ fois la roulette marquée de Van Dantzig. La probabilité de ne pas être marqué au cours de ces $i$ lancers est égale à $p_i t^i$. Comme il nous faut considérer toutes les valeurs de $i$, la probabilité de ne pas être marqué au cours de cette expérience en deux temps est égale à $\sum_{i\geq 0} p_i t^i$. Ce qui nous amène à la :

**Définition :** Soit $(p_i)_{i\geq 0}$ la distribution de probabilité d'une variable aléatoire $X$ prenant $\mathbb{N}$ comme valeurs. La série entière $\phi_X(t) := \sum_{i\geq 0} p_i t^i$ s'appelle la **fonction génératrice** de $X$. On a d'après le théorème de transfert, $\phi_X(t) = \mathbf{E}(t^X)$.

**Remarque :** Le rayon de convergence de $\phi_X$ est supérieur ou égal à 1.

**Propriété :** Si $R_X > 1$, on a

1. $E(X) = \phi'_X(1)$
2. $V(X) = \phi''_X(1) + \phi'_X(1) - \phi'_X(1)^2$

**Remarque importante :** Développer en série entière la fonction génératrice d'une variable aléatoire $X$ à valeurs entières nous permet ainsi de connaître précisément sa loi de probabilité. Il s'agit cependant souvent d'un exercice technique qui peut s'avérer ardu. C'est la contrepartie du gain obtenu !