MPI4 - Cours 4

Séries génératrices exponentielles

La série formelle

$$A(x) = \sum_{n \geq 0} a_n \frac{x^n}{n!}$$

est appelée *série génératrice exponentielle* (SGE) de la suite (a_n) . C'est donc aussi la série génératrice ordinaire (SGO) de la suite $\left(\frac{a_n}{n!}\right)$

L'exponentielle e^x est donc la SGE de la suite constante $a_n = 1$ ($n \geq 0$). La SGE de la suite $(n!)$ est la série géométrique $\frac{1}{1-x}$.

Ces séries apparaissent souvent lorsque l'on cherche à compter des permutations. Il y a $n!$ permutations de n objets. Si on note a_n le nombre de celles qui vérifient une certaine contrainte, $\frac{a_n}{n!}$ est la probabilité qu'une permutation aléatoire vérifie la contrainte en question.

Les dérangements

Le premier exemple d'une telle série est donné par le [problème des rencontres](https://fr.wikipedia.org/wiki/Probl%C3%A8me_des_rencontres) (https://fr.wikipedia.org/wiki/Probl%C3%A8me_des_rencontres), ou *problème du chevalier de Montmort* (1708).

On peut le formuler de nombreuses manières. Par exemple, n personnes se rendent à une soirée et déposent leur chapeau au vestiaire. En repartant, chacun reprend un chapeau au hasard. Quelle est la probabilité pour que personne n'ait son propre chapeau sur la tête ?

Si on numérote les invités et les chapeaux de 1 à n , et qu'on note $\sigma(i)$ le numéro du chapeau récupéré par l'invité i , $\sigma: \{1, 2, \dots, n\} \rightarrow \{1, 2, \dots, n\}$ est une permutation de l'ensemble $\{1, 2, \dots, n\}$.

Un *point fixe* d'une permutation est un élément i tel que $\sigma(i) = i$.

On cherche donc à dénombrer les permutations sans point fixe, également appelées *dérangements*.

```
In [80]: from itertools import permutations
```

```
In [81]: permutations((1,2,3))
```

```
Out[81]: <itertools.permutations at 0x7f1bc9b0e5c8>
```

```
In [82]: list(_) # c'est un itérateur, on convertit en liste
```

```
Out[82]: [(1, 2, 3), (1, 3, 2), (2, 1, 3), (2, 3, 1), (3, 1, 2), (3, 2, 1)]
```

```
In [83]: def perms(n):
         return list(permutations(range(1,n+1)))
```

```
In [84]: perms(3)
```

```
Out[84]: [(1, 2, 3), (1, 3, 2), (2, 1, 3), (2, 3, 1), (3, 1, 2), (3, 2, 1)]
```

```
In [85]: def is_derangement(p):
         return not [i for i in range(1,len(p)+1) if p[i-1]==i] # attention aux indices qui commencent à 0
```

```
In [86]: [p for p in perms(3) if is_derangement(p)]
```

```
Out[86]: [(2, 3, 1), (3, 1, 2)]
```

```
In [87]: def num_der(n):
         return len([p for p in perms(n) if is_derangement(p)])
```

```
In [88]: [num_der(n) for n in range(8)]
```

```
Out[88]: [1, 0, 1, 2, 9, 44, 265, 1854]
```

Soit d_n le nombre de dérangements de n objets.

On connaît le nombre total de permutations ($n!$). Parmi celles-ci, il y en a d_n qui n'ont pas de point fixe.

Et les autres ? Et bien, si on suppose connus $d_{n-1}, d_{n-2}, \dots, d_1 = 0, d_0 = 1$ (convention), on peut calculer le nombre de celles qui ont 1 point fixe, 2 points fixes, etc.

En effet, pour fabriquer une permutation avec exactement un point fixe, il y a n manières de choisir le point fixe, et d_{n-1} de permuter ce qui reste sans introduire de nouveau point fixe, soit au total nd_{n-1} permutations.

Pour exactement 2 points fixes, on a $\binom{n}{2}$ choix pour les points fixes, et d_{n-2} possibilités pour permuter le reste. Et ainsi de suite : pour k points fixes, on a $\binom{n}{k}$ choix pour les points fixes, et d_{n-k} manières de permuter le reste.

Donc,

$$n! = d_n + nd_{n-1} + \binom{n}{2}d_{n-2} + \dots + \binom{n}{n-1}d_1 + \binom{n}{n}d_0 = \sum_{k=0}^n \binom{n}{k}d_{n-k}.$$

On peut alors écrire

$$n! = \sum_{k=0}^n \frac{n!}{k!(n-k)!} d_{n-k}$$

et en simplifiant les $n!$

$$1 = \sum_{k=0}^n \frac{1}{k!} \frac{d_{n-k}}{(n-k)!}$$

On reconnaît la formule pour le produit des séries !

$$C(x) = A(x)B(x), \quad c_n = \sum_{k=0}^n a_k b_{n-k},$$

avec

$$c_n = 1, \quad a_k = \frac{1}{k!}, \quad b_{n-k} = \frac{d_{n-k}}{(n-k)!}$$

autrement dit,

$$C(x) = \frac{1}{1-x}, \quad A(x) = e^x, \quad B(x) = D(x) = \sum_{n \geq 0} \frac{d_n}{n!} x^n$$

et ainsi

$$e^x D(x) = \frac{1}{1-x}, \quad \text{soit finalement } D(x) = \frac{e^{-x}}{1-x}.$$

On peut maintenant développer le produit

$$D(x) = \sum_{p \geq 0} \frac{(-x)^p}{p!} \sum_{q \geq 0} 1 \cdot x^q = \sum_{n \geq 0} \left(\sum_{p=0}^n \frac{(-1)^p}{p!} \right) x^n$$

et donc

$$d_n = \sum_{p=0}^n (-1)^p \frac{n!}{p!}$$

```
In [89]: # test
def fact(n):
    if n<2: return 1
    return n*fact(n-1)

def d(n):
    return sum([(-1)**p*fact(n)//fact(p) for p in range(n+1)])
```

```
In [90]: [d(n) for n in range(8)]
```

```
Out[90]: [1, 0, 1, 2, 9, 44, 265, 1854]
```

Cette formule pourrait sembler décevante, dans la mesure où elle se présente sous forme d'une somme que l'on n'arrive pas à simplifier, et qui ne permet pas un calcul beaucoup plus efficace que la récurrence.

Mais si on s'intéresse à la *probabilité*

$$p_n = \frac{d_n}{n!} = \sum_{p=0}^n \frac{(-1)^p}{p!}$$

on voit que c'est une somme partielle de la série (convergente) pour e^{-1}

$$e^{-1} = \frac{1}{e} = \sum_{p=0}^{\infty} \frac{(-1)^p}{p!}$$

```
In [91]: from math import *
```

```
In [92]: exp(-1)
```

```
Out[92]: 0.36787944117144233
```

```
In [93]: [d(n)/fact(n) for n in range(1,11)]
```

```
Out[93]: [0.0,
          0.5,
          0.3333333333333333,
          0.375,
          0.36666666666666664,
          0.3680555555555556,
          0.3678571428571429,
          0.36788194444444444,
          0.36787918871252206,
          0.3678794642857143]
```

On voit que pour $n = 10$, la probabilité coïncide déjà avec e^{-1} sur les 7 premières décimales.

En fait, $|p_n - e^{-1}| < \frac{1}{(n+1)!}$ (dans une série alternée dont le terme général tend vers 0 en décroissant, l'erreur commise en approchant la somme par une somme partielle est inférieure au premier terme négligé).

Partitions d'ensembles et nombres de Bell

Un autre problème classique qui peut se traiter par les séries exponentielles est celui du *nombre de partitions* d'un ensemble de n éléments.

Une *partition* (https://fr.wikipedia.org/wiki/Partition_d'un_ensemble) π d'un ensemble E est un ensemble de parties non vides (appelées *blocs*) et deux à deux disjointes de E , dont la réunion est E .

On s'intéresse au nombre b_n de partitions de l'ensemble $[n] := \{1, 2, \dots, n\}$, appelé *nième nombre de Bell*.

Structure récursive : une partition de $[n]$, c'est soit une partition de $[n-1]$ à laquelle on a ajouté un bloc $\{n\}$, soit une partition de $[n-1]$ où on a inséré n sans un bloc existant.

```
In [94]: p = ((1,3,6),(2,),(4,5)) # une partition de [6], représentée par un tuple de tuples
```

```
In [95]: def insert_all(x,p): # p partition ensembliste, on insère x dans un bloc existant de toutes les manières possibles
          return [p[:i] + ((x,)+p[i,]) + p[i+1:] for i in range(len(p))]
```

```
In [96]: insert_all(7,p)
```

```
Out[96]: [(7, 1, 3, 6), (2,), (4, 5)],
          ((1, 3, 6), (7, 2), (4, 5)),
          ((1, 3, 6), (2,), (7, 4, 5))]
```

```
In [97]: def add_block(x,p): # ajout d'un nouveau bloc {x}
          return ((x,)+p)
```

```
In [98]: add_block(7,p)
```

```
Out[98]: ((7,), (1, 3, 6), (2,), (4, 5))
```

```
In [99]: from functools import reduce
          def set_partitions(ll):
              if not ll: return [tuple()]
              else:
                  mm = set_partitions(ll[1:]); x=ll[0]
                  return [add_block(x,p) for p in mm] + reduce(lambda a,b:a+b, [insert_all(x,p) for p in mm])
```

```
In [100]: set_partitions(range(1,5)) # Et voilà le travail ...
```

```
Out[100]: [(1,), (2,), (3,), (4,)],
           ((1,), (2,), (3, 4)),
           ((1,), (2, 3), (4,)),
           ((1,), (3,), (2, 4)),
           ((1,), (2, 3, 4)),
           ((1, 2), (3,), (4,)),
           ((2,), (1, 3), (4,)),
           ((2,), (3,), (1, 4)),
           ((1, 2), (3, 4)),
           ((2,), (1, 3, 4)),
           ((1, 2, 3), (4,)),
           ((2, 3), (1, 4)),
           ((1, 3), (2, 4)),
           ((3,), (1, 2, 4)),
           ((1, 2, 3, 4),)]
```

```
In [101]: [len(set_partitions(range(1,n))) for n in range(2,8)]
```

```
Out[101]: [1, 2, 5, 15, 52, 203]
```

Passons à la théorie. Pour fabriquer l'une des b_n partitions de $[n]$, on peut

- ajouter un bloc $\{n\}$ à l'une des b_{n-1} partitions de $[n-1]$: b_{n-1} possibilités
- insérer n dans un bloc d'un élément $\{i\}$ ($n-1$ choix pour i) et prendre une des b_{n-2} partitions des éléments restants
- insérer n dans un bloc de deux éléments $\{i, j\}$ ($\binom{n-1}{2}$ choix pour i, j) et prendre une des b_{n-3} partitions des éléments restants
- ...
- insérer n dans un bloc de k éléments $\{i_1, \dots, i_k\}$ ($\binom{n-1}{k}$ choix) et prendre une des b_{n-1-k} partitions des éléments restants, etc.

Au total,

$$b_n = b_{n-1} + (n-1)b_{n-2} + \binom{n-1}{2}b_{n-3} + \dots = \sum_{k=0}^{n-1} \binom{n-1}{k} b_{n-1-k}.$$

Comme précédemment, on écrit les binomiaux avec les factorielles

$$b_n = \sum_{k=0}^{n-1} \frac{(n-1)!}{k!(n-1-k)!} b_{n-1-k}$$

c'est à dire

$$\frac{b_n}{(n-1)!} = \sum_{k=0}^{n-1} \frac{1}{k!} \frac{b_{n-1-k}}{(n-1-k)!}$$

Ça ressemble encore à la formule du produit. En remplaçant n par $n+1$, et en arrangeant un peu à gauche, il vient

$$(n+1) \frac{b_{n+1}}{(n+1)!} = \sum_{k=0}^n \frac{1}{k!} \frac{b_{n-k}}{(n-k)!}$$

Si l'on pose

$$B(x) = \sum_{n \geq 0} \frac{b_n}{n!} x^n$$

on voit que le membre droit est le coefficient de x^n dans $e^x B(x)$.

Le membre gauche est quant à lui le coefficient de x^n dans $B'(x)$.

Ainsi, $B(x)$ vérifie l'équation différentielle

$$B'(x) = e^x B(x), \text{ avec la condition initiale } B(0) = b_0 = 1.$$

Elle est facile à intégrer :

$$\frac{B'(x)}{B(x)} = \frac{d}{dx} \ln B(x) = e^x$$

donc

$$\ln B(x) = \int e^x dx + C = e^x + C$$

et $B(0) = 1 \Rightarrow \ln B(0) = \ln 1 = 0 = e^0 + C = 1 + C$, donc $C = -1$ et finalement,

$$B(x) = e^{e^x - 1}.$$

Là encore, cette formule ne donne pas un moyen de calculer b_n beaucoup plus efficace que la récurrence, mais elle conduit à une expression curieuse.

On peut écrire

$$\begin{aligned} B(x) = e^{e^x - 1} &= \frac{1}{e} \sum_{k \geq 0} \frac{(e^x)^k}{k!} = \frac{1}{e} \sum_{k \geq 0} \frac{e^{kx}}{k!} = \frac{1}{e} \sum_{k \geq 0} \frac{1}{k!} \sum_{n \geq 0} \frac{(kx)^n}{n!} \\ &= \frac{1}{e} \sum_{n \geq 0} \frac{1}{n!} \left(\sum_{k \geq 0} \frac{k^n}{k!} \right) x^n \end{aligned}$$

et donc

$$b_n = \frac{1}{e} \sum_{k \geq 0} \frac{k^n}{k!}.$$

C'est la formule de Dobinski ([https://en.wikipedia.org/wiki/Dobinski's formula](https://en.wikipedia.org/wiki/Dobinski's_formula)).

Elle est étrange, car elle exprime un nombre entier comme quotient de deux nombres irrationnels.

```
In [102]: def dobin(n,N): # N = nombre de termes
          return sum([(k**n)/fact(k) for k in range(N)])/exp(1)
```

```
In [103]: [dobin(n,10) for n in range(1,10)]
```

```
Out[103]: [0.9999988747974021,
1.9999886256007275,
4.99988488605809,
14.99883349870737,
51.988162413111915,
202.87968157383543,
875.7749016560189,
4127.501208768477,
21019.202460187804]
```

```
In [104]: [round(dobin(n,10)) for n in range(1,10)]
```

```
Out[104]: [1, 2, 5, 15, 52, 203, 876, 4128, 21019]
```

Les polynômes de Touchard

On peut facilement raffiner les calculs précédents, pour compter les partitions de $[n]$ par nombre de blocs.

Si on note $S(n, k)$ le nombre de partitions de $[n]$ en k blocs, et

$$B_n(t) = \sum_{k=1}^n S(n, k)t^k,$$

on voit que la récurrence pour $b_n = B_n(1)$ devient

$$B_n(t) = t \sum_{k=0}^{n-1} \binom{n-1}{k} B_{n-1-k}(t),$$

puisque le k ème terme décrit une partition obtenue en ajoutant un bloc à une partition d'un ensemble de $n-1-k$ éléments. L'équation différentielle devient

$$\frac{d}{dx} \mathcal{B}(x; t) = t e^x \mathcal{B}(x; t), \quad \mathcal{B}(0; t) = 1$$

et la solution est

$$\mathcal{B}(x; t) = e^{t(e^x - 1)}.$$

Les polynômes $B_n(t)$ sont appelés polynômes de Touchard (https://en.wikipedia.org/wiki/Touchard_polynomials), ou de Bell. Les $S(n, k)$ sont les nombre de Stirling de seconde espèce (https://en.wikipedia.org/wiki/Touchard_polynomials), dont on verra plus loin une autre définition.

```
In [ ]:
```