# Éléments de Combinatoire

## Combinatoire

Atlas des Mathématiques :

> “L’analyse combinatoire s’emploie à étudier et à dénombrer divers types de groupements que l’on peut faire à partir d’ensembles finis”

Ell’est popularisée en Occident par Pascal[1] et Fermat, dans l’étude des jeux d’hasard (17ème siècle).
Elle était déjà utilisée avant dans le monde Arabe (al-Karaji[2], *circa* 1100) et en Chine (Shen Kua[3], 1031-1095).

## Arrangements et Permutations

**Définition :** Un arrangement de longueur $k$ d’un ensemble $E$ de $n$ éléments est une *suite ordonnée* de $k$ éléments *distincts* de $E$.
$\mathcal{A}_n^k$ est le nombre des arrangements de longueur $k$ sur un ensemble de $n$ éléments.

**Proposition :**

$$\mathcal{A}_n^k = \frac{n!}{(n-k)!}$$

**Définition :** Une permutation d’un ensemble à $n$ éléments est un arrangement de longueur $n$.
On écrit $\mathcal{P}_n$ pour $\mathcal{A}_n^n$. On a alors $\mathcal{P}_n = n!$

## Exemple : le voyageur de commerce

Un voyageur de commerce doit visiter $n$ villes, une et une seule fois, en commençant d’une ville donnée.
Combien de parcours possibles peut-il emprunter pour s’acquitter de sa tâche ?

[1] 1623-1662"Usage du triangle arithmétiquepour déterminer les parties"
[2] Coefficients du binôme
[3] Configurations de l’échiquier

## Combinaisons

**Définition :** Une combinaison de longueur $k$ d'un ensemble $E$ de $n$ éléments est un sous-ensemble de $k$ éléments de $E$.
On note $\mathcal{C}_n^k$ le nombre des combinaisons de longueur $k$ sur un ensemble de $n$ éléments.

**Proposition :** Chaque combinaison d'ordre $k$ donne lieu à $k!$ arrangements, donc

$$\mathcal{C}_n^k = \frac{\mathcal{A}_n^k}{k!} = \frac{n!}{(n-k)!k!} \tag{1}$$

## La formule du binôme

Les $\mathcal{C}_n^k$ sont aussi appelés coefficients binomiaux en raison du résultat suivant

**Proposition :**

$$(a+b)^n = \sum_{i=0}^{n} \mathcal{C}_n^i a^i b^{n-i} \tag{2}$$

**Corollaire :** on peut déduire de la formule précédente l'équation

$$\sum_{i=0}^{n} (-1)^i \mathcal{C}_n^i = 0 \tag{3}$$

## Application : la formule de Sylvester

Notons $|A|$ la cardinalité d'un ensemble $A$.
Nous avons la formule remarquable suivante :

$$\begin{aligned}
&|A_1 \cup A_2 \cup \ldots A_n| \\
&\quad = \sum_{i=1}^{n} |A_i| - \sum_{1 \leq i < j \leq n} |A_i \cap A_j| + \sum_{1 \leq i < j < k \leq n} |A_i \cap A_j \cap A_k| \ldots \\
&\quad\qquad + (-1)^{n-1} |A_1 \cap \ldots \cap A_n| \\
&\quad = \sum_{p=1}^{n} (-1)^{p-1} \left( \sum_{1 \leq i_1 < \ldots < i_p \leq n} |A_{i_1} \cap \ldots \cap A_{i_p}| \right)
\end{aligned}$$

### Exemple

Les étudiants du cours de Math-Info suivent tous au moins un cours de langues parmi Anglais, Espagnol et Italien. 43 suivent Anglais, 21 Espagnol et 16 Italien. Il y en a 8 qui suivent Espagnol et Italien, 10 qui suivent Anglais et Italien et 15 qui suivent Anglais et Espagnol. Enfin, seulement 3 suivent les trois cours de langues. Combien d'étudiants y a-t-il dans le cours de Math-Info ?

### Compter les surjections

La formule de Sylvester nous permet de prouver le résultat suivant

**Théorème :** Le nombre de surjections d'un ensemble à $m$ éléments vers un ensemble à $n$ éléments est

$$\begin{aligned} S_n^m &= n^m - \mathcal{C}_n^1(n-1)^m + \mathcal{C}_n^2(n-2)^m - \ldots + (-1)^{n-1}\mathcal{C}_n^{n-1}1^m \\ &= \sum_{p=0}^{n}(-1)^p\mathcal{C}_n^p(n-p)^m \end{aligned}$$

### Toujours sur les ensembles

L'analogie avec la cardinalité des ensembles permet d'établir facilement

$$\sum_{i=0}^{n}\mathcal{C}_n^i = 2^n \tag{4}$$

### Convolution de Vandermonde

**Théorème :** Si $n, m \geq k$, alors

$$\mathcal{C}_{n+m}^k = \sum_{i=0}^{k}\mathcal{C}_n^i\mathcal{C}_m^{k-i} \tag{5}$$

### Autres égalités remarquables

$$\mathcal{C}_n^n = 1 \tag{6}$$
$$\mathcal{C}_n^0 = 1 \tag{7}$$
$$\mathcal{C}_n^k = \mathcal{C}_n^{n-k} \tag{8}$$
$$\mathcal{C}_n^k = \mathcal{C}_{n-1}^k + \mathcal{C}_{n-1}^{k-1} \tag{9}$$

### Calculer les valeurs des combinaisons

Ces égalités sont la clé du *triangle de Pascal*[4], une méthode itérative de construction des valeurs des $\mathcal{C}_n^k$ :

| $n$ \ $k$ | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 1 | | | | |
| 1 | 1 | 1 | | | |
| 2 | 1 | 2 | 1 | | |
| 3 | 1 | 3 | 3 | 1 | |
| 4 | 1 | 4 | 6 | 4 | 1 |
| 5 | ... | | | | |

### Comment calculer sur ordinateur

Comparons les programmes suivants :

```
(* compter avec la forme close *)
let rec fact = function 0 -> 1  | n-> n*(fact (n-1));;
let c (n,k) = fact(n)/(fact(n-k)*fact(k));;
(* compter avec la forme recurrente *)
let rec c = function
      (n,0) -> 1
    | (n,k) when n=k -> 1
    | (n,k) -> (c (n-1,k-1))+ (c (n-1,k));;
```

Lequels choisiriez-vous ? Pourquoi ?

### Comment calculer vite

```
 (* a la Pascal, jusqu'à C_100^100 *)
let rec c =
    let triangle = Array.create_matrix 100 100 0 in
    let check e (n,k) =
        if triangle.(n).(k) = 0 then
          let r = e (n,k)
          in triangle.(n).(k) <- r; r
        else triangle.(n).(k) in
  function
    (n,0) -> 1
```

[4]En Italie, on l'appellele *le triangle de Tartaglia*

```
  | (n,k) when n=k -> 1
  | (n,k) -> (check c (n-1,k-1)) + (check c (n-1,k));;

(* compter avec la forme recurrente, memoisee *)
let rec c =
    let memo = Hashtbl.create 16 in
    let memocheck e v =
        try let r = Hashtbl.find memo v in  print_string "*"; r
        with _ -> let r = e v in Hashtbl.add memo v r; r in
  function
    (n,0) -> 1
  | (n,k) when n=k -> 1
  | (n,k) -> (memocheck c (n-1,k-1)) + (memocheck c (n-1,k));;
```

## Approximation de la factorielle

On peut approximer la factorielle comme suit :

$$\begin{aligned} \ln n! &= ln1 + ln2 + \ldots \ln n = \textstyle\sum_{i=1}^{n} \ln i \\ &\approx \textstyle\int_1^n \ln x dx = [xlnx - x]_1^n = n \ln n - n + 1 \\ &\approx n \ln n - n \end{aligned}$$

d'où

$$n! \approx e^{n \ln n - n} = \frac{n^n}{e^n}$$

Stirling nous fournit une approximation plus précise

$$\sqrt{2\pi n}\left(\frac{n}{e}\right)^n$$

## Calculer avec Stirling

```
let stfact n = sqrt(2*.3.1428*.n)*.(n/(exp 1))**n;;
let cst (n,k) =  let nf = float n and kf = float k in
    truncate ((stfact nf)/.((stfact kf) *. (stfact (nf -.kf))));;
 (* comparons *)
tabulate (fun n -> c (n, n/2)) 1 20;;
- : int list =
 [1; 2; 3; 6; 10; 20; 35; 70; 126; 252; 462; 924; 1716; 3432; 6435; 12870;
 24310; 48620; 92378; 184756]
tabulate (fun n -> cst (n, n/2)) 1 20;;
```

```
- : int list =
 [0; 2; 3; 6; 10; 20; 36; 72; 129; 258; 472; 943; 1749; 3493; 6542; 13070;
 24666; 49290; 93587; 187043]
```

## Enumerer les combinaisons

Plus remarquablement, elles sont la clé du programme récursif suivant, qui engendre effectivement toutes les combinaisons.

```
 (* engendrer *)
let addprefix v ll = List.map (fun l -> v::l) ll
let rec initsegment b = function 0 -> []
                        | n -> b::(initsegment (b+1) (n-1))

let rec comb b = function
    (n,0) -> [[]]
  | (n,m) when n=m -> [initsegment b n]
  | (n,m) -> (comb (b+1) (n-1,m))
             @ (addprefix b (comb (b+1) (n-1,m-1)))
```

## Exemples

**Exemple :** le LOTO

Combien de grilles simples différentes peut-on jouer ?

## Exemples

**Exemple :** les échecs

Combien de configuration différentes peut-on obtenir en plaçant les pièces sur l'échiquier ?


**Attention :** pour chaque couleur, les pions sont indistinguables entre eux, comme les tours, les cavaliers et les fous.

### L'équation des échecs : permutations avec objets indistinguables

On apprend du cas des échecs que si on dispose de $n$ objets, composés de $k$ groupes de $n_i$ objets indistinguables, alors il y a

$$\mathcal{C}_n^{n_1,\ldots,n_k} = \frac{n!}{n_1! \ldots n_k!}$$

façons de les permuter.

Ceci est aussi les nombre de façon de distribuer $n$ objets différents dans $k$ boites différentes de façon à mettre $n_i$ objets dans la $i$-ème boite.
On les appelle aussi...

### Coefficients multinomiaux

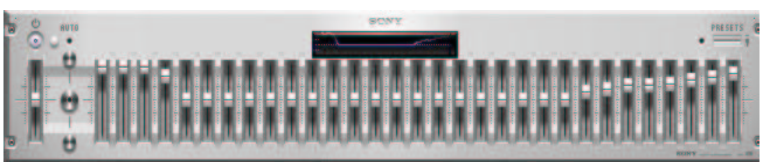
On a en effet

$$(a_1 + \ldots + a_k)^n = \sum_{n_1+\ldots+n_k=n} \mathcal{C}_n^{n_1,\ldots,n_k} a_1^{n_1} \ldots a_k^{n_k}$$

### Arrangements *avec répétition*

**Définition :** un arrangement avec répétition de longueur $k$ d'un ensemble $E$ de $n$ éléments est une *suite ordonnée* de $k$ éléments de $E$, *pas nécessairement distincts*.

**Exemple :** si chaque curseur a $n$ positions, et il y a $k$ curseurs, combien de configuration peut avoir cet égalisateur ?

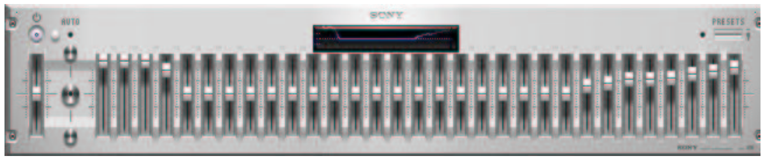



Les arrangements avec répétition de longueur $k$ sur un ensemble de $n$ éléments sont $n^k$.

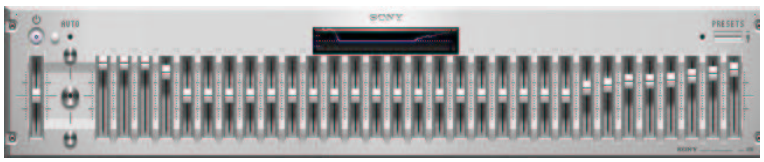
### Combinaisons *avec répétition*

**Définition :**Une combinaison avec répétition de longueur $k$ d'un ensemble $E$ de $n$ éléments est un *suite non ordonnée* de $k$ éléments de $E$, *pas nécessairement distincts*.
Les combinaisons avec répétition de longueur $k$ sur un ensemble de $n$ éléments sont $\mathcal{C}_{n+k-1}^{n-1}$.

**Exemple :** Si chaque curseur a $n$ positions, et il y a $k$ curseurs, combien de configurations *de même puissance* $p \leq n$ peut avoir cet égalisateur ?



**Exemple :** Si chaque curseur a $n$ positions, et il y a $k$ curseurs, combien de configurations *de même puissance* $p \leq nk$ peut avoir cet égalisateur ?



**Preuve :** Il faut utiliser tout notre bagage d'astuces. Il faut appliquer Sylvester pour déduire la vraie valeur par différence entre le nombre total et celui des dispositions impossibles.. . . Le nombre total de dispositions possibles des curseurs, si on n'avait pas la contrainte $x_i \leq n$, serait bien $\mathcal{C}_{p+k-1}^{k-1}$. Mais il faut alors enlever les dispositions "impossibles", correspondant à la violation de la contrainte.

Donc on a

$$Dp = \mathcal{C}_{p+k-1}^{k-1} \setminus |\{dispositions\ avec\ quelques\ x_i > n\}|$$

mais ce dernier ensemble est égale à

$$\bigcup_{1 \leq i \leq k} \{dispositions\ avec\ x_i > n\}$$

et sa taille est, par Sylvester (notons $D_i$ les ensembles de la famille de tout à l'heure) :

$$\sum_{1 \leq j \leq k} ( \sum_{i_1 < \ldots < i_j} (-1)^{j-1} |D_{i_1} \cap \ldots \cap D_{i_j}|)$$

Or, le calcul de $|D_{i_1} \cap \ldots \cap D_{i_k}|$ est assez facile : on sait que $x_{i_1}, \ldots, x_{i_j}$ valent au moins $n+1$, parce-que ils violent la contrainte, donc on donne

$n+1$ aux $j$ curseurs, il nous reste $p-(n+1)j$ à distribuer, avec répétitions, parmis les $k$ courseurs, ce qui fait donc $\mathcal{C}_{p-(n+1)j+k-1}^{k-1}$. On a alors

$$Dp = \mathcal{C}_{p+k-1}^{k-1} - \sum_{1\leq j\leq k}\Big(\sum_{i_1<\ldots<i_j}(-1)^{j-1}\mathcal{C}_{p-(n+1)j+k-1}^{k-1}\Big)$$

$$Dp = \mathcal{C}_{p+k-1}^{k-1} - \sum_{1\leq j\leq k}(-1)^{j-1}\mathcal{C}_{p-(n+1)j+k-1}^{k-1}\Big(\sum_{i_1<\ldots<i_j}(-1)^{j-1}\Big)$$

$$Dp = \mathcal{C}_{p+k-1}^{k-1} - \sum_{1\leq j\leq k}(-1)^{j-1}\mathcal{C}_{p-(n+1)j+k-1}^{k-1}\mathcal{C}_k^j$$

$$Dp = \sum_{0\leq j\leq k}(-1)^{j}\mathcal{C}_{p-(n+1)j+k-1}^{k-1}\mathcal{C}_k^j$$

Mais attention, comme $p \leq nk$, il faut arreter la somme au prémier $j$ t.q. $(n+1)(j+1) > nk$

## Applications

Les combinaisons avec répétitions permettent de compter les solutions des équations de la forme

$$x_1 + \ldots + x_k = p$$

La solution du cas de l'égalisateur nous permet de compter les solutions des équations de la forme

$$x_1 + \ldots + x_k = p$$

avec des contraintes $x_i \leq n$ (ou plus généralement $x_i \leq n_i$). Dans la solution de l'égalisateur on a appris à compter aussi[5] les solutions des équations de la forme

$$x_1 + \ldots + x_k = p$$

avec des contraintes $x_i \geq n$ (ou plus généralement $x_i \geq n_i$).

[5] en réalité, on a vu que c'est plus facile