# Comptage, suite et série

**Objectifs**

- Comprendre comment les suites servent à modéliser la progression des variables de boucles et à établir les conditions d'arrêt.
- Utiliser les séries pour compter le nombre d'opérations dans les boucles.
- Apprendre à utiliser la récurrence pour démontrer la validité d'un algorithme.

## Pourquoi compter ?

Nous avons vu que pour qu'un algorithme soit fonctionnel, il faut qu'il s'arrête et produise le résultat escompté, quelque soit l'instance donnée. Mais on peut aussi se poser la question du temps que va mettre un algorithme pour résoudre un problème. En effet, s'il faut trop de temps, l'algorithme ne servira à rien.

Afin d'étudier le temps que va mettre un algorithme en fonction de la taille des données en entrée, on utilise la **machine RAM**, qui est un modèle apparu dans les années 1960-1970, et qui représente un ordinateur simplifié où toutes les opérations élémentaires (addition, lecture, écriture) se font en un temps constant. Ainsi en comptant le nombre d'opérations élémentaires, nous avons une idée de la vitesse à laquelle va se dérouler l'algorithme.

Bien entendu, le temps d'exécution va aussi dépendre du matériel utilisé (vitesse du processeur, quantité et vitesse de la mémoire, …).

Par exemple, avec un core I9-13900K à 3GHz (c'est à dire 3 milliards d'opérations par seconde), on a la correspondance suivante :

| Nombre d'opération | Temps |
|---|---|
| $n = 10^{10}$ | 3 à 4 secondes |
| $n^2 = 10^{20}$ | 1057 ans |
| $n! = 10^{10^{11}}$ | irréaliste |
| $\log(n) = 10$ | instantané |

Il est donc important de compter le nombre d'opérations pour comparer l'efficacité de nos algorithmes et pour les améliorer.

## Les suites

**Définition**

Une suite numérique est une liste ordonnée de nombres réels où chaque nombre correspond à une position dans la liste.
Autrement dit : Une suite numérique est une fonction définie sur l'ensemble des entiers naturels (ou une partie de ceux-ci) qui associe à chaque entier un nombre réel appelé terme de la suite.
On note généralement une suite par $(u_n)$, où $n$ est l'indice appelé rang, et $u_n$ est le terme de rang $n$.

### Suites arithmétiques

**Définition**

Une suite est arithmétique si elle peut être définie par récurrence par :

$$\begin{cases} u_0 = \text{valeur} \\ u_{k+1} = u_k + \text{raison} \quad \text{pour } k > 0 \end{cases}$$

ou explicitement : $u_k = u_0 + k \times \text{raison}$ pour $k \geq 0$

**Exemple**

```
i ← 1
TQ i   n FAIRE
  i ← i + 2
FTQ
```

$i$ va prendre les valeurs de la suite arithmétique $(u_k)$ définie par :

$$\begin{cases} u_0 = 1 \\ u_{k+1} = u_k + 2 \quad \text{pour } k > 0 \end{cases}$$

La raison étant strictement positive, il s'agit d'une suite croissante non majorée, c'est à dire que quelque soit le seuil choisi $N$, il existera un indice $k_0$ tel que tous les termes de la suite à partir de $k_0$ seront supérieur à $N$.

Notre boucle demande à ce que $i$ soit inférieur à $n$, d'après les propriétés de la suite, cela arrivera forcément et donc la boucle se finit.
Pour connaître la valeur de fin de boucle (quand ce n'est pas évident) et le nombre d'opérations faites, on passe par l'expression explicite de la suite et on résout une inéquation :

$$\begin{aligned} u_k = u_0 + k \times 2 = 1 + 2k > n &\Leftrightarrow 2k > n - 1 \\ &\Leftrightarrow k > \frac{n-1}{2} \end{aligned}$$

Ainsi, nous ferons $\lfloor \frac{n-1}{2} \rfloor + 1$ opérations et la valeur de sortie de la variable $i$ sera :

$$i = u_k = 1 + 2 \times (\lfloor \frac{n-1}{2} \rfloor + 1) = \begin{cases} 1 + n - 1 + 2 = n + 2 & \text{si } n \text{ est impair} \\ 1 + n - 2 + 2 = n + 1 & \text{sinon} \end{cases}$$

### Suites géométriques

**Définition**

Une suite est géométrique si elle peut être définie par récurrence par :

$$\begin{cases} u_0 = \text{valeur} \\ u_{k+1} = u_k \times \text{raison} \quad \text{pour } k > 0 \end{cases}$$

ou explicitement : $u_k = u_0 \times \text{raison}^k$ pour $k \geq 0$

**Exemple**

```
x ← 2
q ← 3
TQ x   seuil FAIRE
  x ← x * q
FTQ
```

$x$ va prendre les valeurs de la suite géométrique $(u_k)$ définie par :

$$\begin{cases} u_0 = 2 \\ u_{k+1} = u_k \times 3 \quad \text{pour } k > 0 \end{cases}$$

Si la raison est strictement supérieure à 1 et que $a$ est positif, il s'agit d'une suite croissante non majorée.

Si la raison est comprise strictement entre -1 et 1, la suite convergera vers 0.
Notre boucle demande à ce que $x$ soit inférieur à seuil, d'après les propriétés de la suite, cela arrivera forcément et donc la boucle se finit.
Nombre d'opérations à faire :

$$\begin{aligned} u_k = u_0 \times q^k = 2 \times 3^k > \text{seuil} &\Leftrightarrow 3^k > \frac{\text{seuil}}{2} \\ &\Leftrightarrow k\log(3) > \log(\frac{\text{seuil}}{2}) \\ &\Leftrightarrow k > \frac{\log(\frac{\text{seuil}}{2})}{\log(3)} \end{aligned}$$

### Suites arithmético-géométriques

Il s'agit d'un mélange des deux suites précédentes pour la version récurrente.

**Définition**

Une suite est arithmético-géométrique si elle peut être définie par récurrence par :

$$\begin{cases} u_0 = \text{valeur} \\ u_{k+1} = u_k \times a + b \quad \text{pour } k > 0 \end{cases}$$

ou explicitement : $u_k = l + (u_0 - l) \times a^k$ pour $k \geq 0$ avec $l = \dfrac{b}{1-a}$ le point fixe de la suite (c'est à dire que $l = l \times a + b$)

**Exemple**

Soit la suite $(u_n)$ définie sur $\mathbb{N}$ par :

$$\begin{cases} u_0 = 5 \\ u_{n+1} = u_n \times 2 + 4 \quad \text{pour } n > 0 \end{cases}$$

Donner l'expression explicite de $(u_n)$.
*Réponse :* $u_n = -4 + 9 \times 2^n$

## Les sommes partielles

**Définition**

Etant donnée une suite $(u_n)$, la somme partielle de ses termes souvent notée $S_n$ correspond à :

$$S_n = \sum_{k=0}^{n} u_k = u_0 + u_1 + u_2 + ... + u_n$$

Etudier les sommes partielles va nous permettre de compter le nombre total d'opérations qu'il y a dans des boucles.

**Propriétés**

- **Linéarité :** $\sum_{k=0}^{n} \alpha u_k + v_k = \alpha \sum_{k=0}^{n} u_k + \sum_{k=0}^{n} v_k$
- Cela ne marche pas avec la multiplication !
- **Changement d'indice :** $\sum_{k=0}^{n} u_k = \sum_{j=1}^{n+1} u_{j-1}$ en posant $j = k + 1$

**Exemple**

```
i ← 1
s ← 0
TQ i   n FAIRE
  s ← s + i
  i ← i + 1
FTQ
```

Notre variable de boucle $i$ va prendre toutes les valeurs entières de 1 à $n$. Pour chaque passage dans la boucle, il va y avoir 2 opérations élémentaires. Ainsi, le nombre total d'opérations sera :

$$S_n = \sum_{k=1}^{n} 2 = 2 \sum_{k=1}^{n} 1 = 2n$$

Il y aura $2n$ opérations faites.

**Formules à retenir**

- $\sum_{k=a}^{b} 1 = b - a + 1$ *(C'est le nombre de termes dans la somme)*
- $\sum_{k=a}^{b} k = \frac{(b-a+1)(a+b)}{2}$
- $\sum_{k=a}^{b} q^k = q^a \times \frac{1-q^{b-a+1}}{1-q}$

**Exemple : Boucles imbriquées**

```
x ← 0
i ← 1
TQ i   n FAIRE
  j ← 1
  TQ j   i FAIRE
    x ← x+1
    j ← j+1
  FTQ
  i ← i+1
FTQ
```

**Boucle intérieure :**
Notre variable de boucle $j$ va prendre toutes les valeurs entières de 1 à $i$. Pour chaque passage dans la boucle, il va y avoir 2 opérations élémentaires. Ainsi, le nombre d'opérations sera :

$$B_i = \sum_{k=1}^{i} 2 = 2\sum_{k=1}^{i} 1 = 2i$$

Il y aura $2i$ opérations faites dans la boucle intérieure.
**Boucle extérieure :**
Notre variable de boucle $i$ va prendre toutes les valeurs entières de 1 à $n$. Pour chaque passage dans la boucle, il va y avoir $2i + 2$ opérations élémentaires. Ainsi, le nombre total d'opérations sera :

$$S_n = \sum_{i=1}^{n}(2i+2) = 2\sum_{i=1}^{n} i + 2\sum_{i=1}^{n} 1 = 2 \times \frac{n(n+1)}{2} + 2n = n^2 + 3n$$

Il y aura donc $n^2 + 3n$ opérations faites.

## Récurrence - Rappels

La propriété $\mathcal{P}(n)$ choisie correspond à un invariant de boucle qui permettra, une fois la boucle finie, de répondre au problème donné.

**Méthode de la preuve par récurrence**

Pour montrer qu'une propriété $\mathcal{P}(n)$ est vraie pour tout $n \geq n_0$, il faut montrer 2 points :
- la propriété $\mathcal{P}(n_0)$ est vraie (**initialisation**)
- si la propriété $\mathcal{P}(n)$ est vraie alors la propriété $\mathcal{P}(n+1)$ l'est aussi (**hérédité**)

**Exemple**

Montrer que pour tout entier $n \geq 0$, $\sum_{i=0}^{n} i = \frac{n(n+1)}{2}$

*(Exercice à faire en TD)*

**Exemple : Validité de l'algorithme factorielle**

Montrer la validité de cet algorithme qui calcule la factorielle d'un nombre $n$ :

```
ALGORITHME Factorielle(n)
DONNEES:
  n : entier
VARIABLES:
  f,k : entiers
DEBUT
  f ← 1
  k ← 1
  TQ k   n FAIRE
    f ← f*k
    k ← k+1
  FTQ
  RENVOYER f
FIN
```

**Invariant proposé :** à la fin de chaque boucle, $f = k!$
Soit, pour $k$ entier, la propriété $P(k)$ : "$f = k!$ à la fin de $k$ itérations".
**Initialisation :** $P(1)$ : À la première itération, $f \leftarrow f * k$ or $f = 1$ et $k = 1$ d'où $f = 1 \times 1 = 1$ et $1! = 1$ donc $P(1)$ est vraie.
**Hérédité :** Supposons que $P(k)$ soit vraie pour un certain entier $k$. À la fin de la $k^{\text{ième}}$ itération, la variable $k$ contient $k+1$, donc lors de la $(k+1)^{\text{ième}}$ itération, $f \leftarrow f * k$ or

$f = k!$ et $k = k + 1$, donc $f = k! \times (k + 1) = (k + 1)!$
$P(k + 1)$ est donc vraie.
**Conclusion :** Ainsi, en sortant de la nième itération, $P(n)$ est vraie et donc on renvoie $f = n!$. Notre algorithme est validé.