Mathématiques 1
Récurrence et récursivité

Institut Paul Lambin

9 octobre 2021

Introduction : Enigme des moines : énoncé

Dans un camp de bouddhistes, on apprend qu'il y a au moins un malade.

- Cette maladie n'est pas contagieuse et le nombre de malades n'évoluera plus.
- Un bouddhiste qui se sait malade doit partir pour ne pas gêner les autres.
- Cette maladie se caractérise uniquement par une tâche rouge sur le front.
- Ce symptôme leur permet de reconnaître une personne malade.
- Le problème est qu'il n'y a aucun moyen pour un bouddhiste de se voir.
- Il n'y a aucun miroir ou autre moyen permettant de se voir.
- Les moines bouddhistes ont fait le voeux de silence et ne parlent pas.
- Ils ne font que méditer, lire et sont d'excellents logiciens.
- Ils se réunissent tous, en cercle, une seule fois par jour au lever du soleil pour une méditation commune de 3 heures.
- Pendant ces trois heures, il n'ont pas le droit de communiquer entre eux ni de partir avant la fin de la séance commune.
- Au bout de 5 jours, tous les malades sont partis.

Combien y avait-il de moines malades ?

Introduction : Enigme des moines : Résolution

1) S'il n'y a qu'un seul moine malade :
   - → il ne voit aucun moine portant une tache rouge lors de la méditation
   - → il se dit "*comme il y au moins un moine malade ce ne peut être que moi*".
   - → le lendemain, il est parti.
2) S'il y a deux moines malades :
   - → ceux-ci ne voient qu'un seul moine malade lors de la méditation.
   - → ils se disent "*Si le moine malade que je vois est le seul moine malade alors il sera parti demain. S'il n'est parti demain c'est qu'il y a un autre moine malade qui ne peut être que moi*"
   - → après deux jours les deux moines sont partis.
3) S'il y trois moines malades :
   - → ceux-ci ne voient que deux moines malades lors de la méditation.
   - → Il se disent "*Si les 2 moines malades que je vois sont les seuls moines malades alors il feront le raisonnement du 2) et seront partis deux jours plus tard. S'il ne sont pas parti deux jours plus tard c'est qu'il y a troisième moine malade qui ne peut être que moi*"
   - → après trois jours,les trois moines sont partis.

Les moines malades sont partis après 5 jours → ils étaient 5 !.

Introduction : Enigme des moines : Principe utilisé

Pour résoudre l'énigme des moines, on a utilisé le raisonnement suivant

1) S'il n'y a qu'un seul moine malade alors il sera parti après un jour

2) **Si**, quand il y a n moines malades, ils partent après n jours
alors s'il y $n+1$ moines malades ils seront partis après $n+1$ jours.

Ce raisonnement est ce que l'on appelle un raisonnement **par récurrence**.

Introduction : Analogie de l’échelle

Peut-on monter aussi haut que l’on veut sur l’échelle suivante ?

La réponse est oui si les conditions suivantes sont vérifiées :

1) Je peux monter sur le **premier** échelon
2) **Si** je suis sur un échelon **alors** je peux monter sur le suivant

Introduction : Analogie de l'échelle

En effet,

- Par 1), je peux atteindre le premier échelon
- je suis sur le premier échelon $\to$ par 2), je peux atteindre le $2^{\text{ème}}$ échelon
- je suis sur le $2^{\text{ème}}$ échelon $\to$ par 2), je peux atteindre le $3^{\text{ème}}$échelon
- ....

$\to$ Je pourrai donc atteindre n'importe quel échelon et donc grimper aussi haut que je veux.

C'est ce que l'on appelle le **principe de récurrence**.

Démonstration par récurrence : Principe

On veut montrer que $p(n)$ est **vraie pour tout** $n \geq n_0$.
Pour cela, il faut

1) Prouver que $p(n_0)$ est vraie $\rightarrow$ **Pas initial**
2) Prouver que **si** $p(k)$ est **vraie** alors $p(k+1)$ est **vraie**

   $\rightarrow$ **Pas de récurrence**.

   Ce pas de récurrence se décompose comme suit

   a) On suppose $p(k)$ **vraie** $\rightarrow$ **Hypothèse**
   b) On définit ce que l'on veut démontrer : $p(k+1)$ est **vraie** $\rightarrow$ **Thèse**
   c) En supposant $p(k)$ **vraie**, on prouve que $p(k+1)$ est **vraie**

   $\rightarrow$ **Démonstration**

Démonstration par récurrence : Exemple 1

On souhaite prouver que, quel que soit le naturel n,

le nombre $2n^3 + 3n^2 + n$ est divisible par 6.

Rappel :
Un entier a est divisible par entier b s'il existe un entier p tel que $a = b \cdot p$.

$\rightarrow$ on doit montrer que, pour tout entier $n \geq 0$,

il existe un entier p tel que $2n^3 + 3n^2 + n = 6 \cdot p$.

$\rightarrow$ démonstration par récurrence :

Démonstration par récurrence : Exemple 1

On doit montrer que, pour tout entier $n \geq 0$,

il existe un entier p tel que $2n^3 + 3n^2 + n = 6 \cdot p$.

1) **Pas initial :** $n_0 = 0 : 2 \cdot 0^3 + 3 \cdot 0^2 + 0 = 0 = 6 \cdot 0 \rightarrow p(n_0)$ est vraie
2) **Pas de récurrence** :
   a) **Hypothèse** : Pour $n = k$ fixé, il existe un entier p tel que $2k^3 + 3k^2 + k = 6p$
   b) **Thèse** : Pour $n = k+1$, il existe un entier q tel que $2(k+1)^3 + 3(k+1)^2 + (k+1) = 6q$
   c) **Démonstration :**

$$
\begin{aligned}
2(k+1)^3 + 3(k+1)^2 + (k+1) &= 2(k^3 + 3k^2 + 3k + 1) + 3(k^2 + 2k + 1) + k + 1 \\
&= 2k^3 + 6k^2 + 6k + 2 + 3k^2 + 6k + 3 + k + 1 \\
&= 2k^3 + 3k^2 + k + 6k^2 + 6k + 2 + 6k + 3 + 1 \\
&= 2k^3 + 3k^2 + k + 6k^2 + 12k + 6 \\
&= 6p + 6(k^2 + 2k + 1) \qquad \text{(Par l'}\textbf{hypothèse}\text{)} \\
&= 6(p + k^2 + 2k + 1) \\
&= 6q \qquad \text{(avec l'entier } q = p + k^2 + 2k + 1\text{)}
\end{aligned}
$$

Démonstration par récurrence : Exemple 2

On veut montrer que pour tout naturel $n \geq 1$: $1+2+3+\cdots+n = \frac{n(n+1)}{2}$.

1) **Pas initial :** $n_0 = 1$:$1 = \frac{1\cdot(1+1)}{2} = \frac{2}{2} = 1$ OK
2) **Pas de récurrence :**
   a) **Hypothèse :** pour $n = k$ fixé : $1+2+3+\cdots+k = \frac{k(k+1)}{2}$.
   b) **Thèse :** pour $n = k+1$:
   $$1+2+3+\cdots+(k+1) = \frac{(k+1)((k+1)+1)}{2} = \frac{(k+1)(k+2)}{2}.$$
   c) **Démonstration :**

$$\begin{aligned}
1+2+3+\cdots+(k+1) &= 1+2+3+\cdots+k+(k+1) \\
&= \frac{k(k+1)}{2}+(k+1) \qquad \text{(Par l'}\textbf{hypothèse}\text{)} \\
&= \frac{k(k+1)+2(k+1)}{2} \\
&= \frac{(k+1)(k+2)}{2}
\end{aligned}$$

(En mettant $(k+1)$ en évidence)

Démonstration par récurrence : Remarques

- La structure de la preuve par récurrence est très importante pour sa lisibilité $\rightarrow$ **il faut la respecter**

- Dans la partie **Démonstration** du pas de récurrence :

  toujours essayer de faire apparaître l'hypothèse afin de pouvoir l'utiliser

Démonstration par récurrence : Importance des 2 pas

Les 2 pas sont importants !

$\rightarrow$ Si **l'un des deux n'est pas vérifié** alors **la preuve ne fonctionne pas !**

Importance du pas initial

On voudrait montrer que pour tout naturel n : 10^n+1 est divisible par 9.
$\rightarrow$ pour tout entier $n \geq 0$, il existe un entier p tel que $10^n+1=9p$.

Regardons le **pas de récurrence** :

a) **Hypothèse** : pour $n=k$ fixé : il existe un entier p tel que $10^k+1=9p$
b) **Thèse :** pour $n=k+1$: il existe un entier q tel que $10^{k+1}+1=9q$
c) **Démonstration :**

$$\begin{aligned} 10^{k+1}+1 &= 10\cdot 10^k+1 \\ &= 9\cdot 10^k+10^k+1 \\ &= 9\cdot 10^k+9p \quad \text{(Par l'}\textbf{hypothèse}\text{)} \\ &= 9\left(10^k+p\right) \\ &= 9q \quad \text{(avec l'entier } q=10^k+p\text{)} \end{aligned}$$

Le pas de récurrence est donc vrai.
Mais **pas initial** :$n_0=0$: $10^0+1=1+1=2$ n'est pas divisible par 9 !
$\rightarrow$ Le pas initial est **faux** ! $\rightarrow$ **La preuve ne fonctionne pas !**
En effet, 10^n+1 n'est jamais divisible par 9 quelle que soit la valeur de n !

Importance du pas de récurrence

On veut démontrer que, pour tout entier $n \geq 1$, si je prends au hasard n billes de billard alors elles sont toutes de la même couleur.

1) **Pas initial :** $n_0 = 1$: Si je prends 1 bille alors toutes les billes que j'ai prises seront bien de la même couleur puisqu'il n'y en a qu'une.
2) **Pas de récurrence :**
   a) **Hypothèse** : pour $n = k$ fixé :
   si je prends k billes au hasard alors elle sont toutes de la même couleur
   b) **Thèse** :pour $n = k+1$:
   si je prends $k+1$ billes au hasard alors elles sont toutes de la même couleur
   c) **Démonstration :**
      - Soit $k+1$ billes de billard prises au hasard : $b_1, b_2, \ldots, b_k, b_{k+1}$
      - Si on considère les billes 1 à k, comme il s'agit de k billes prises au hasard, elles sont, par l'hypothèse, de la même couleur.
      - Si on considère les billes 2 à $k+1$, comme il s'agit de k billes prises au hasard, elles sont, par l'hypothèse, de la même couleur.
      - Donc les billes 1 et $k+1$ sont de la même couleur que les billes 2 à k.
      - Donc elles sont toutes de la même couleur.

      $\rightarrow$ les $k+1$ billes sont de la même couleur.

Importance du pas de récurrence

La preuve du transparent précédent paraît correcte.

$\rightarrow$ si elle l'est alors toutes les billes de billard du monde sont de la même couleur !

$\rightarrow$ Il y a une erreur quelque part.

$\rightarrow$ le pas de récurrence ne fonctionne pas pour passer du cas de 1 bille au cas de 2 billes !

Quand on a $k = 1$, alors $k + 1 = 2$.

$\rightarrow$ le groupe des billes de 1 à k se réduit à la bille b_1

$\rightarrow$ le groupe des billes de 2 à $k + 1$ se réduit à la bille b_2

$\rightarrow$ ces 2 groupes n'ont pas de billes en commun !

$\rightarrow$ l'argument qui dit "les billes 1 et $k + 1$ sont de la même couleur que les billes 2 à k" ne fonctionne pas !

$\rightarrow$ pour en revenir à l'analogie de l'échelle, on ne peut pas grimper de l'échelon 1 à l'échelon 2 !

$\rightarrow$ Le **pas de récurrence** est **faux** car $\exists k (= 1)$ pour lequel il n'est **pas vrai**.

$\rightarrow$ Donc **la preuve ne fonctionne pas** !

Définition récursive

Une **définition récursive** d'un concept est une définition du concept **qui s'invoque lui-même**

Exemple :

Définissons une partie P de $\mathbb{N}$ de la manière suivante

(1) 1 est un élément de P
(2) si x est un élément de P alors $5 \cdot x$ est un élément de P
(3) si y est un élément de P alors $7 \cdot y$ est un élément de P

Quelle est cette partie P ?

a) Par (1) : 1 est élément de P
b) Par (2) et a) : $5 \cdot 1 = 5$ est un élément de P
c) Par (3) et a) : $7 \cdot 1 = 7$ est un élément de P
d) Par (2) et b) : $5 \cdot 5 = 5^2 = 25$ est un élément de P
e) Par (3) et b) : $7 \cdot 5 = 35$ est un élément de P
f) Par (3) et d) : $7 \cdot 5^2 = 175$ est un élément de P

...

$\rightarrow$ P est l'ensemble des naturels de la forme $5^m \cdot 7^n$ avec m et n des naturels.

Opérateur ternaire

L'opérateur ternaire est une manière abrégée pour

$$\begin{array}{|l|}\hline \textit{si (condition) alors} \\ \quad res = expression1 \\ \textit{sinon} \\ \quad res = expression2 \\ \hline \end{array}$$

En effet, en Java le code

```
if (condition)
   res=expression1 ;
else
   res=expression2 ;
```

peut s'écrire de manière **plus compacte** :

```
res = (condition) ? expression1 : expression2;
```

Méthode récursive : définition

Une méthode **récursive** est une méthode qui **s'appelle elle-même**.

Exemple 1 :

Méthode qui calcule $n!$: la factorielle d'un nombre entier n.

La factorielle est définie comme suit : Si n est un naturel alors

$$n! = \begin{cases} 1 & \text{si} \quad n = 0 \\ 1 * 2 * 3 \cdots * n & \text{si} \quad n > 0 \end{cases}$$

Implémentation : de manière itérative en $O(n)$ $\rightarrow$ avec une boucle :

```
public static int factorielle(int n){
    int fact = 1;
    for(int i = 1 ; i <= n ; i++){
        fact = fact*i;
    }
    return fact;
}
```

Remarque :

Si `n`=0 alors on n'entre pas dans la boucle et la méthode renvoie `1` **OK**

Méthode récursive : exemple 1

Cependant la factorielle peut aussi se définir de manière récursive :

$$n! = \begin{cases} 1 & \text{si} \quad n = 0 \\ n*(n-1)! & \text{si} \quad n > 0 \end{cases}$$

$\rightarrow$ Méthode récursive en $O(n)$:

```
public static int factorielle(int n){
    if (n==0)
        return 1;
    else
        return n*factorielle(n-1);
}
```

Méthode récursive : exemple 1 : remarques

1) La méthode `factorielle` s'appelle elle-même → une méthode récursive
2) On peut réécrire cette méthode de manière plus compacte grâce à l'opérateur ternaire :

```
public static int factorielle(int n){
    return ((n==0)? 1 : n*factorielle(n-1));
}
```

3) Schéma des appels succeccifs pour calculer 3! :

```
factorielle(3)                                                          = 6
      ↘                                                                ↗
        3*factorielle(2)                                       = 3 * 2 = 6
              ↘                                               ↗
                2*factorielle(1)                        =   2 * 1 = 2
                       ↘                               ↗
                         1*factorielle(0)=1*1  =  1
```

Méthode récursive : exemple 2

Méthode qui calcule la somme cumulée : $1 + 2 + \cdots n$ pour tout naturel n.

1)**De manière itérative** en $O(n)$ $\rightarrow$ avec une boucle :

```
public static int sommeCumulee(int n){
   int somme = 0;
   for(int i = 1 ; i <= n ; i++){
      somme = somme+i;
   }
   return somme;
}
```

Remarque :

si `n`=0 alors on n'entre pas dans la boucle et la méthode renvoie `0` **OK**

Méthode récursive : exemple 2

2) **De manière récursive** en $O(n)$:

```
public static int sommeCumulee(int n){
    if (n==0)
        return 0;
    else
        return n+sommeCumulee(n-1);
}
```

De manière plus compacte avec l'opérateur ternaire :

```
public static int sommeCumulee(int n){
    return ((n==0)? 0 : n+sommeCumulee(n-1));
}
```

Méthode récursive : exemple 2

On a démontré que $1+2+\cdots+n = \dfrac{n(n+1)}{2}$.

$\rightarrow$ On peut programmer cette somme en $O(1)$:

```
public static int sommeCumulee(int n){
    return n*(n+1)/2 ;
}
```

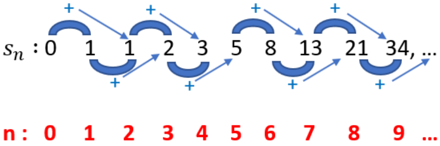
Limitation de la récursivité : Suite de Fibonacci

En l'utilisant pour modéliser l'évolution d'une population de lapin, Leonardo Fibonacci a donné son nom à la suite définie par

$$\begin{cases} s_0 & = & 0 \\ s_1 & = & 1 \\ s_n & = & s_{n-1} + s_{n-2} \quad \text{si } n \geq 2 \end{cases}$$

Donc un élément est la somme des 2 éléments le précédant dans la suite.

Voici un dessin montrant cela :



Méthode qui, pour tout entier n, calcule le $n^{\text{ème}}$ élément de la suite de Fibonacci ?

Suite de Fibonacci : Version itérative

```
public int fibo(int n) {
    if (n==0)
        return 0 ;
    if (n==1)
        return 1 ;
    int si=0 ;
    int sii=1 ;
    for (int i=2 ; i<=n ; i++) {
        int temp = si+sii ;
        si = sii ;
        sii = temp ;
    }
    return sii ;
}
```

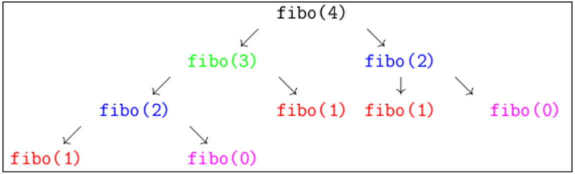
Une seule boucle que l'on va parcourir n fois $\rightarrow$ méthode en $O(n)$.

Suite de Fibonacci : Version récursive

```
public int fibo(int n) {
    if (n==0)
        return 0 ;
    if (n==1)
        return 1 ;
    return fibo(n-1)+fibo(n-2) ;
 }
```

Chaque exécution de la méthode fait **deux appels récursifs successifs** !

Voici une représentation de l'exécution de la méthode avec `n`=4 :



Suite de Fibonacci : Version récursive

Remarques :

1) On exécute plusieurs fois la même chose : 3 fois `fibo(1)` par exemple

2) Pour calculer `fibo(4)`, la méthode a fait 8 appels récursifs !

3) La récursivité dans cette méthode est une récursivité (monstrueuse) arborescente !

deux appels récursifs consécutifs $\rightarrow$ méthode est en $O(2^n)$.

Limitation de la récursivité : Conclusion

On vient de voir plusieurs méthodes qui peuvent s'écrire tant de manière itérative que de manière récursive.

Il y a d'ailleurs un théorème qui dit

a) *Toute méthode itérative peut être écrite de manière récursive.*
b) *Toute méthode récursive peut être écrite de manière itérative.*

Choix entre récursif et itératif basé sur 2 critères :

1) La facilité d'implémentation
2) La complexité (la rapidité d'exécution)

PGCD

Le **Plus Grand Commun Diviseur** de deux entiers *a* et *b* est

le plus grand entier qui divise *a* et *b* et se note $\mathrm{PGCD}(a, b)$.

Exemple :

Si $a = 980$ et $b = 1512$ alors

$$\begin{cases} a = 980 = 2^2 \cdot 5 \cdot 7^2 \\ b = 1512 = 2^3 \cdot 3^3 \cdot 7 \end{cases} \rightarrow \mathrm{PGCD}(a, b) = 2^2 \cdot 7 = 28$$

- $\longrightarrow$ recours à la décomposition en facteurs premiers.
- $\longrightarrow$ méthode très lourde et très complexe.
- $\longrightarrow$ méthode plus efficace : Algorithme d'Euclide

Algorithme d'Euclide : Principe

Soit

- deux entiers (≥ 1) a et b
- r le reste de la division entière de a par $b \rightarrow \exists\, q$ tel que $a = b \cdot q + r$

On a alors la propriété suivante :

$$\boxed{\begin{array}{l} \text{Si } r = 0 \text{ alors} \\ \quad \text{PGCD}(a, b) = b \\ \text{Sinon} \\ \quad \text{PGCD}(a, b) = \text{PGCD}(b, r) \end{array}}$$

En effet, comme $a = b \cdot q + r$, alors

- $r = a - b \cdot q$
- si un entier d divise a et b alors d divise $a - b \cdot q \rightarrow d$ divise r et b
- si un entier d divise b et r alors d divise $b \cdot q + r \rightarrow d$ divise a et b

$\rightarrow$ Tout diviseur commun à a et b est un diviseur commun à b et r et réciproquement.

$\rightarrow$ PGCD$(a, b) =$PGCD(b, r).

Algorithme d'Euclide : exemple d'exécution

Calculons $\text{PGCD}(1512, 980)$:

$$
\begin{array}{lllll}
1) & 1512 \mod 980 = 532 & \rightarrow & 1512 = 980 \cdot 1 + 532 & \rightarrow & \text{PGCD}(1512, 980) = \text{PGCD}(980, 532) \\
2) & 980 \mod 532 = 448 & \rightarrow & 980 = 532 \cdot 1 + 448 & \rightarrow & \text{PGCD}(980, 532) = \text{PGCD}(532, 448) \\
3) & 532 \mod 448 = 84 & \rightarrow & 532 = 448 \cdot 1 + 84 & \rightarrow & \text{PGCD}(532, 448) = \text{PGCD}(448, 84) \\
4) & 448 \mod 84 = 28 & \rightarrow & 448 = 84 \cdot 5 + \mathbf{28} & \rightarrow & \text{PGCD}(448, 84) = \text{PGCD}(84, 28) \\
5) & 84 \mod 28 = 0 & & & \rightarrow & \text{PGCD}(84, 28) = 28
\end{array}
$$

Donc $\text{PGCD}(1512, 980) = 28$, ce qui bien ce que nous avions trouvé précédemment.

Donc le PGCD est le **dernier reste non nul**.

Algorithme d'Euclide : remarques

1) PGCD(a,b) est multiple de tous les autres communs diviseurs de a et b.

$$\rightarrow \forall a \forall b \forall c \left(\left((c|a) \wedge (c|b) \right) \Rightarrow \left(c \middle| \mathrm{PGCD}(a,b) \right) \right)$$

2) Le PGCD peut être implémenté tant de manière itérative que récursive. Voici un pseudo-code :

$$\boxed{\begin{array}{l} pgcd = a \\ r = b \\ tant\ que\ (r\ n'est\ pas\ égal\ à\ 0)\ répéter \\ \qquad temp = r \\ \qquad r = pgcd \mod r \\ \qquad pgcd = temp \end{array}}$$