Mathématiques appliquées à l’informatique

HELMO Haute Ecole Libre Mosane TECHNIQUE

Chapitre 7

Récurrence, récursivité, induction

CH07 – Récurrence, récursivité, induction

HELMO Haute Ecole Libre Mosane TECHNIQUE

7.1.1 – Suites et récurrence
7.1.1.1 – Définitions et vocabulaire

Rappel (voir Ch5)

Une suite est

- une liste ordonnée d'éléments
- pas nécessairement différents
- numérotés par les entiers 1, 2, 3, …

Haute Ecole Libre Mosane
TECHNIQUE

7.1.1 – Suites et récurrence
7.1.1.1 – Définitions et vocabulaire

Définitions

- Une suite est dite non numérique lorsqu’il s’agit d’une liste d'éléments autres que des nombres (une suite de formes géométriques, de gestes, …)
  - Signe de reconnaissance : un certain motif se répète tout au long de la suite
  - **Exemple**

    🔔 🖃 🖃 ☎ 🔔 🖃 🖃 ☎ 🔔 🖃 🖃 ☎ 🔔 🖃 🖃 ☎

    (= suite de motifs)

7.1.1 – Suites et récurrence
7.1.1.1 – Définitions et vocabulaire

- Si elle ne fait intervenir que des nombres,
  la suite est alors numérique
- Une suite numérique est généralement infinie
- On parle de suite numérique finie pour désigner une liste ordonnée et finie de nombres réels

7.1.1 – Suites et récurrence
7.1.1.1 – Définitions et vocabulaire

Vocabulaire

- Les différents nombres de cette liste sont les termes de la suite
- Le premier terme d'une suite est souvent noté $\boldsymbol{u_1}$ (notation indicée)
- Son n^ième terme, ou terme d'indice n ou terme général, est noté $\boldsymbol{u_n}$ avec $\boldsymbol{u_n = f(n)}$
  - la lettre u peut être remplacée par n'importe quelle autre lettre qui identifie la suite
  - écrire $\boldsymbol{u_n}$ nécessite de trouver une formule !

HELMo Haute Ecole Libre Mosane TECHNIQUE

7.1.1 – Suites et récurrence
7.1.1.1 – Définitions et vocabulaire

- La position précise occupée par chaque terme dans une suite est appelée rang
- Le nombre qui permet de passer d'un élément au suivant est appelé raison de la suite
- Une suite est notée (u_n) avec $n \in \mathbb{N}_0$

En algèbre financière, le premier terme est généralement un terme d'indice 0 (t_0, C_0 , ...)

HELMO Haute Ecole Libre Mosane TECHNIQUE

7.1.1 – Suites et récurrence

7.1.1.1 – Définitions et vocabulaire

Exemples

- Soit la suite $(3, 6, 9, 12, 15, 18, \ldots)$

  $+3 \quad +3 \quad +3 \quad +3 \quad +3$

  - le premier terme de la suite est 3 $(u_1 = 3)$
  - au deuxième rang, le terme est 6
    (le deuxième terme est 6 ; $u_2 = 6$)
  - au cinquième rang, le terme est 15
    (le cinquième terme est 15 ; $u_5 = 15$)
  - la raison de cette suite est $+3$
  - le terme u_n s'écrit $3n$ $(u_n = 3n)$
  - la suite se note alors $(3n)_{(n \in \mathbb{N}_0)}$

7.1.1 – Suites et récurrence
7.1.1.1 – Définitions et vocabulaire

- Soit la suite $\left(1, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \frac{1}{5}, \frac{1}{6}, \ldots \right)$
  - le premier terme de la suite est 1 ($u_1 = 1$)
  - au cinquième rang, le terme est $\frac{1}{5}$ ($u_5 = \frac{1}{5}$)
  - le terme u_n s'écrit $\frac{1}{n}$
  - la suite se note alors $\left(\frac{1}{n}\right)_{(n \in \mathbb{N}_0)}$

Remarque

une suite peut n'être définie que sur une partie de $\mathbb{N}$

$\Rightarrow$ *la suite est définie à partir du rang* x

HELMO Haute Ecole Libre Mosane TECHNIQUE

7.1.1 – Suites et récurrence
7.1.1.2 – Détermination d'une suite

- Une suite est déterminée quand on connaît son premier terme et son terme général
- On décrit une suite
  - par une **FORMULE EXPLICITE** : le terme général de la suite est exprimé en fonction de n ; on parle aussi de suite explicite
  - par **RÉCURRENCE** : on donne le premier terme, ou les deux - trois premiers termes, ainsi qu'une formule qui exprime le terme général en fonction du (ou des) terme(s) précédent(s) ;
    on parlera alors de suite récursive

HELMo Haute Ecole Libre Mosane TECHNIQUE

7.1.1 – Suites et récurrence
7.1.1.2 – Détermination d'une suite

Exemples

- Soit la suite $(u_n)_{(n \in \mathbb{N}_0)}$, définie par la formule $u_n = 7 + n$:
  - le premier terme de cette suite est $u_1 = 8$
  - la suite est donc ($8, 9, 10, 11,$ ...),
  - = la suite des nombres naturels supérieurs ou égaux à 8
- La suite $(a_n)_{(n \in \mathbb{N}_0)}$, de terme général $a_n = \frac{n+1}{n}$ est la suite (2, $\frac{3}{2}, \frac{4}{3}, \frac{5}{4}, \frac{6}{5},$ ...)

7.1.1 – Suites et récurrence
7.1.1.2 – Détermination d'une suite

- La suite $(b_n)_{(n \in \mathbb{N}_0)}$, de premier terme 1, exprimée par la formule de récurrence $b_{n+1} = \dfrac{b_n + 1}{b_n}$ est la suite $(1, 2, \frac{3}{2}, \frac{5}{3}, \frac{8}{5}, \frac{13}{8}, \ldots)$
- La suite $(c_n)_{(n \in \mathbb{N}_0)}$

  définie par $\begin{cases} c_1 = 7 \\ c_n = c_{n-1} + 1 \end{cases} \quad (n \in \mathbb{N}\backslash\{0, 1\})$

  est une autre manière de définir la suite du premier exemple !

7.1.1 – Suites et récurrence
7.1.1.2 – Détermination d’une suite

Remarque

En programmation, on calculera les termes
d’une suite explicite ou récursive
à l’aide de boucles

7.1.1 – Suites et récurrence
7.1.1.3 – Exercices

1. Pour chacune des suites suivantes,
écrire les trois termes suivants,
exprimer le $n^{\text{ième}}$ terme en fonction de n et
calculer u_{100} :

   1) $1, 3, 5, 7, \ldots$

   2) $3\ ;\ 1{,}5\ ;\ 0{,}75\ ;\ 0{,}375\ ;\ \ldots$

   3) $0,\ \frac{1}{4},\ \frac{2}{5},\ \frac{3}{6}, \ldots$

   4) $1, 4, 9, 16, \ldots$

   5) $2,\ 3,\ 5,\ 8, \ldots$

HEL MO Haute Ecole Libre Mosane TECHNIQUE

7.1.1 – Suites et récurrence
7.1.1.3 – Exercices

1. Pour chacune des suites suivantes, écrire les trois termes suivants, exprimer le $n^{\text{ième}}$ terme en fonction de n et calculer u_{100} :

   1) $1, 3, 5, 7, \ldots$ $9, 11, 13$

   $u_n = 2n - 1$ $u_{100} = 199$

   2) $3\ ;\ 1{,}5\ ;\ 0{,}75\ ;\ 0{,}375\ ;\ \ldots$

   $0{,}1875\ ;\ 0{,}09375 ;\ 0{,}046875$

   $$u_n = \frac{3}{2^{n-1}}$$

   $u_{100} = 4{,}73\text{x}10^{-30}$

HEL MO Haute Ecole Libre Mosane TECHNIQUE

7.1.1 – Suites et récurrence
7.1.1.3 – Exercices

1. Pour chacune des suites suivantes, écrire les trois termes suivants, exprimer le $n^{\text{ième}}$ terme en fonction de n et calculer u_{100} :

3) $0, \frac{1}{4}, \frac{2}{5}, \frac{3}{6}, \ldots$ $\frac{4}{7}, \frac{5}{8}, \frac{6}{9}$

$$u_n = \frac{n-1}{n+2} \qquad u_{100} = \frac{99}{102}$$

4) $1, 4, 9, 16, \ldots$ $25, 36, 49$

$$u_n = n^2 \qquad u_{100} = 10\,000$$

HELMO Haute Ecole Libre Mosane TECHNIQUE

7.1.1 – Suites et récurrence
7.1.1.3 – Exercices

1. 5) 2, 3, 5, 8, ... $12, 17, 23$

$+1 \quad +2 \quad +3 \quad +4 \quad +5$

$$u_1 = 2$$
$$u_2 = 2 + 1 = 3$$
$$u_3 = 2 + 1 + 2 = 5$$
$$u_4 = 2 + 1 + 2 + 3 = 8$$
$$u_5 = 2 + 1 + 2 + 3 + 4 = 12$$
$$u_6 = 2 + 1 + 2 + 3 + 4 + 5 = 17$$
$$u_n = 2 + 1 + 2 + 3 + 4 + (n-1)$$
$$= 2 + somme\ des\ (n-1)\ premiers\ entiers$$

7.1.1 – Suites et récurrence
7.1.1.3 – Exercices

1. 5) $2,\ 3,\ 5,\ 8, \ldots 12, 17, 23$

Calcul de la somme des n premiers entiers à partir de 1 :

- Soit à l'aide des formules des suites arithmétiques (SA),

  puisqu'il s'agit d'une SA pour laquelle $v_1 = 1$ et $r = +1$:

  $$v_n = v_1 + (n-1)\,.\,r = 1 + n - 1 = n \text{ et}$$

  $$S_n = \frac{n}{2}(v_1 + v_n) = \frac{n}{2}(1+n) = \frac{n(n+1)}{2}$$

  (somme des n premiers termes d'une suite arithmétique)

7.1.1 – Suites et récurrence
7.1.1.3 – Exercices

1. 5) 2, 3, 5, 8, … $12, 17, 23$

Calcul de la somme des n premiers entiers (suite) :

➢ Soit en suivant le raisonnement suivant (« échelle ») :

$$1 + 2 + 3 + \cdots + (n-2) + (n-1) + n$$

$$= \underbrace{(1+n) + \big(2 + (n-1)\big) + \big(3 + (n-2)\big) + \cdots + \left(\frac{n}{2} + \left(n - \frac{n}{2} + 1\right)\right)}_{\frac{n}{2}\ termes}$$

$$= \frac{n}{2}(1+n) = \frac{n(n+1)}{2}$$

7.1.1 – Suites et récurrence
7.1.1.3 – Exercices

1. 5) $2,\ 3,\ 5,\ 8, \dots$ $12, 17, 23$

   Calcul de la somme des n premiers entiers à partir de 1 :

➢ D'autres variantes !

- ✓ Calcul de la somme des n premiers entiers non nuls (méthode de Gauss) (vidéo)
- ✓ Calcul de la somme des n premiers entiers non nuls (méthode des triangles) (vidéo)
- ✓ Somme des n premiers entiers (ac-caen.fr)

7.1.1 – Suites et récurrence
7.1.1.3 – Exercices

1. 5) $2,\ 3,\ 5,\ 8, \ldots$ $12, 17, 23$

Ce qui donne, pour la suite qui nous intéresse :

$$u_n = 2 + \frac{(n-1)(n-1+1)}{2} = 2 + \frac{n(n-1)}{2}$$

et donc $u_{100} = 4\,952$

HELMO Haute Ecole Libre Mosane TECHNIQUE

7.1.1 – Suites et récurrence
7.1.1.3 – Exercices

2. On donne les suites (u_n) telles que :

1) $u_n = -\frac{n}{2} + 1 \quad (n \in \mathbb{N}_0)$

2) $u_n = n^2 - 5n + 4 \quad (n \in \mathbb{N}_0)$

3) $u_n = \frac{1}{n-1} \quad (n \in \mathbb{N}\backslash\{0,1\})$

4) $u_n = \frac{3}{2^n} \quad (n \in \mathbb{N}_0)$

5) $\begin{cases} u_1 = 3 \\ u_n = 2u_{n-1} - 1 \end{cases} \quad (n \in \mathbb{N}\backslash\{0,1\})$

Pour chacune de ces suites, calculer, si cela est possible, u_1, u_2, u_3, u_4, u_{n-1}, u_n et u_{n+1}

HELMo Haute Ecole Libre Mosane
TECHNIQUE

7.1.1 – Suites et récurrence
7.1.1.3 – Exercices

2. On donne les suites (u_n) telles que :

1) $u_n = -\frac{n}{2} + 1 \quad (n \in \mathbb{N}_0)$

$u_1 = \frac{1}{2}\,;\, u_2 = 0\,;\, u_3 = \frac{-1}{2}\,;\, u_4 = -1$

$u_{n-1} = -\frac{n}{2} + \frac{3}{2}\,;\, u_{n+1} = -\frac{n}{2} + \frac{1}{2}$

2) $u_n = n^2 - 5n + 4 \quad (n \in \mathbb{N}_0)$

$u_1 = 0;\, u_2 = -2\,;\, u_3 = -2;\, u_4 = 0$

$u_{n-1} = n^2 - 7n + 10\,;\, u_{n+1} = n^2 - 3n$

HELMO Haute Ecole Libre Mosane TECHNIQUE

7.1.1 – Suites et récurrence
7.1.1.3 – Exercices

2. On donne les suites (u_n) telles que :

3) $u_n = \frac{1}{n-1} \quad (n \in \mathbb{N}\backslash\{0,1\})$

u_1 / ; $u_2 = 1$; $u_3 = \frac{1}{2}$; $u_4 = \frac{1}{3}$

$u_{n-1} = \frac{1}{n-2}$; $u_{n+1} = \frac{1}{n}$

4) $u_n = \frac{3}{2^n} \quad (n \in \mathbb{N}_0)$

$u_1 = \frac{3}{2}$; $u_2 = \frac{3}{4}$; $u_3 = \frac{3}{8}$; $u_4 = \frac{3}{16}$

$u_{n-1} = \frac{3}{2^{n-1}}$; $u_{n+1} = \frac{3}{2^{n+1}}$

HELMo Haute Ecole Libre Mosane TECHNIQUE

7.1.1 – Suites et récurrence
7.1.1.3 – Exercices

2. On donne les suites (u_n) telles que :

$$5)\ \begin{cases} u_1 = 3 \\ u_n = 2u_{n-1} - 1 \end{cases} \quad (n \in \mathbb{N}\backslash\{0,1\})$$

$$u_1 = 3\ ;\ u_2 = 5\ ;\ u_3 = 9;\ u_4 = 17$$

$$u_n = 1 + 2^n$$

$$u_{n-1} = 1 + 2^{n-1};\ u_{n+1} = 1 + 2^{n+1}$$

HELMO Haute Ecole Libre Mosane TECHNIQUE

7.1.1 – Suites et récurrence
7.1.1.3 – Exercices

3. Soient les suites suivantes :

   1) $7, 9, 12, 16, 21 \ \ldots$

   2) $1, 3, 15, 105, 945, \ldots$

   3) $1, 2, 6, 42, 1806 \ldots$

   Ecrire le terme suivant ainsi que
   la définition récursive de chaque suite

7.1.1 – Suites et récurrence
7.1.1.3 – Exercices

3. Soient les suites suivantes :

1) $7, 9, 12, 16, 21$ ... 27

$$\begin{cases} u_1 = 7 \\ u_n = u_{n-1} + n \end{cases} \quad (n \in \mathbb{N}\backslash\{0,1\})$$

2) $1, 3, 15, 105, 945, \ldots$ $10\ 395$

$$\begin{cases} u_1 = 1 \\ u_n = u_{n-1} \,.\, (2n-1) \end{cases} \quad (n \in \mathbb{N}\backslash\{0,1\})$$

HELMo Haute Ecole Libre Mosane TECHNIQUE

7.1.1 – Suites et récurrence
7.1.1.3 – Exercices

3. Soient les suites suivantes :

3) $1, 2, 6, 42, 1806 \ldots$ $3\,263\,442$

$$\begin{cases} u_1 = 1 \\ u_n = u_{n-1} \,.\, (u_{n-1} + 1) \end{cases} \quad (n \in \mathbb{N}\backslash\{0,1\})$$

$$= \begin{cases} u_1 = 1 \\ u_n = u_{n-1}^2 + u_{n-1} \end{cases} \quad (n \in \mathbb{N}\backslash\{0,1\})$$

HELMo Haute Ecole Libre Mosane TECHNIQUE

7.1.1 – Suites et récurrence
7.1.1.3 – Exercices

4. Calculer les 5 premiers termes des suites récursives suivantes :

   1) $u_1 = 1$ et $u_{n+1} = u_n \, . (\, u_n + 2)$

   2) $u_1 = 1 \, , \, u_2 = 1$ et $u_{n+2} = u_{n+1} + u_n$
   (suite de Fibonacci)

   3) $u_1 : \, entier \; plus \; grand \; que \; zéro \; au \; choix$
   et $u_{n+1} = \; u_n \, pair \, ? \; u_n/2 \; : \; 3u_n + 1$
   (suite de Syracuse)

HEL MO Haute Ecole Libre Mosane TECHNIQUE

7.1.1 – Suites et récurrence
7.1.1.3 – Exercices

4. Calculer les 5 premiers termes des suites récursives suivantes :

   1) $u_1 = 1$ et $u_{n+1} = u_n \,.\, (u_n + 2)$

      $u_1 = 1$; $u_2 = 3$; $u_3 = 15$; $u_4 = 255$ et $u_5 = 65\,535$

   2) $u_1 = 1$, $u_2 = 1$ et $u_{n+2} = u_{n+1} + u_n$ (suite de Fibonacci)

      $u_1 = 1$; $u_2 = 1$; $u_3 = 2$; $u_4 = 3$ et $u_5 = 5$

7.1.1 – Suites et récurrence
7.1.1.3 – Exercices

4. Calculer les 5 premiers termes des suites récursives suivantes :

3) u_1: $entier\ plus\ grand\ que\ zéro\ au\ choix$
et $u_{n+1} = u_n\ pair\,?\,u_n/2\ :\ 3u_n + 1$
(suite de Syracuse)

Par exemple, $u_1 = 14$
$u_2 = 7$; $u_3 = 22$; $u_4 = 11$ et $u_5 = 34$
= *suite de Syracuse du nombre 14 !*

7.1.1 – Suites et récurrence
7.1.1.3 – Exercices

5. Pour chacune des suites suivantes définies par leur terme général, indiquer à partir de quel rang elles sont définies, puis calculer la somme des trois premiers termes

1) $u_n = (-1)^n \sqrt{n}$

2) $u_n = \sqrt{n^2 - 4n}$

3) $u_n = \dfrac{2^n}{n^2 - 1}$

HELMO Haute Ecole Libre Mosane TECHNIQUE

7.1.1 – Suites et récurrence
7.1.1.3 – Exercices

5. Pour chacune des suites suivantes définies par leur terme général, indiquer à partir de quel rang elles sont définies, puis calculer la somme des trois premiers termes

1) $u_n = (-1)^n \sqrt{n}$

(u_n) est définie SSI $\sqrt{n}$ est définie,
c'est-à-dire SSI $n \geq 0$;
ce qui est vrai $\forall n \in \mathbb{N}$ et donc vrai $\forall n \in \mathbb{N}_0$

$u_1 = -1$; $u_2 = \sqrt{2}$; $u_3 = -\sqrt{3}$

$u_1 + u_2 + u_3 = -1 + \sqrt{2} - \sqrt{3}$

7.1.1 – Suites et récurrence
7.1.1.3 – Exercices

5. Pour chacune des suites suivantes définies par leur terme général, indiquer à partir de quel rang elles sont définies, puis calculer la somme des trois premiers termes

2) $u_n = \sqrt{n^2 - 4n}$

(u_n) est définie SSI $\sqrt{n^2 - 4n}$ est définie,
c'est-à-dire SSI $n^2 - 4n \geq 0$;
ce qui est vrai $\forall n \geq 4$

(u_n) est donc définie $\forall n \geq 4$

$u_4 = 0$; $u_5 = \sqrt{5}$; $u_6 = 2\sqrt{3}$

7.1.1 – Suites et récurrence
7.1.1.3 – Exercices

5. Pour chacune des suites suivantes définies par leur terme général, indiquer à partir de quel rang elles sont définies, puis calculer la somme des trois premiers termes

3) $u_n = \frac{2^n}{n^2 - 1}$

(u_n) est définie SSI $n^2 - 1 \neq 0$;
or $n^2 - 1 = (n-1)(n+1)$,
dans $\mathbb{N}$, seul le facteur $(n-1)$ peut s'annuler
(u_n) est donc définie $\forall n \geq 2$

$u_2 = \frac{4}{3}$; $u_3 = 1$; $u_4 = \frac{16}{15}$ $\quad$ $u_2 + u_3 + u_4 = \frac{17}{5}$

7.1.1 – Suites et récurrence
7.1.1.3 – Exercices

6. Montrer que la suite $(u_n)_{(n \in \mathbb{N}_0)}$ définie par

$$u_n = n \,.\, 2^n$$

vérifie la relation de récurrence

$$u_{n+2} = 4\,(u_{n+1} - u_n)$$

$$\begin{aligned}
u_{n+1} - u_n &= (n+1).\,2^{n+1} - n\,.\,2^n \\
&= n.\,2^{n+1} + 2^{n+1} - n\,.\,2^n \\
&= 2.n.\,2^n + 2.2^n - n\,.\,2^n \\
&= 2^n(2n + 2 - n) = 2^n(\mathrm{n}+2)
\end{aligned}$$

$$\begin{aligned}
u_{n+2} &= (n+2).\,2^{n+2} = (n+2).\,2^n\,.\,2^2 \\
&= 2^n(\mathrm{n}+2).\,4
\end{aligned}$$

CH07 – Récurrence, récursivité, induction

HELMO Haute Ecole Libre Mosane TECHNIQUE

7.1.2 – Le principe de récurrence

- Le principe de récurrence repose sur deux étapes : l'initialisation et l'hérédité
- Prenons une analogie : le « domino cascade »

➢ Après avoir passé beaucoup de temps à disposer soigneusement les dominos, nous sommes prêts !

7.1.2 – Le principe de récurrence

- Que faut-il faire pour commencer la cascade ?

  Pousser le premier domino

  = **initialisation**

- Quelle est la condition pour que tous les dominos tombent, jusqu'au dernier ?

  Que l'espacement entre deux dominos successifs soit correct ; que chaque domino, à son tour, se comporte comme le premier !

  = **hérédité** !

HELMO Haute Ecole Libre Mosane TECHNIQUE

7.1.2 – Le principe de récurrence

En résumé

le principe de récurrence affirme que :

- Si le premier domino fait tomber le deuxième,

ET

- Si n'importe quel autre domino est capable de faire tomber son suivant,
- Alors tous les dominos tomberont !

🔔 Le but d'une récurrence est de montrer qu'une propriété $P(n)$ est vraie pour tout n

HEL MO Haute Ecole Libre Mosane TECHNIQUE

CH07 – Récurrence, récursivité, induction

7.1 Récurrence

7.1.1 Les suites et la récurrence

7.1.2 Le principe de récurrence

7.1.3 Démonstration par récurrence

7.2 Récursivité

7.3 Induction

HEL MO Haute Ecole Libre Mosane TECHNIQUE

7.1.3 – Démonstration par récurrence

7.1.3.1 – Principe[1]

- Soit $p(n)$ un prédicat dépendant de la variable entière n et soit n_0 un entier fixé
- Supposons que nous soyons capable de :
  1. prouver que $p(n_0)$ est vrai
  2. montrer que si les énoncés $p(n_0), p(n_0 + 1), \dots,$ $p(n_0 + k)$ sont vrais,
     alors $p(n_o + k + 1)$ est encore vrai

➢ Nous pouvons conclure que $p(n)$ est vrai, pour tout $n \geq n_0$

[1] *Tiré de [Marchand, ch. 2.6]*

7.1.3 – Démonstration par récurrence

7.1.3.1 – Principe

Deux éléments indispensables

- **Pas initial**

  « prouver que $p(n_0)$ est vrai »

- **Pas récurrent**

  « montrer que si les énoncés $p(n_0), p(n_0+1), \ldots$,
  $p(n_0+k)$ sont vrais,
  alors $p(n_o+k+1)$ est encore vrai »

➢ Il se réduit souvent à montrer que si $p(n)$ est vrai, alors $p(n+1)$ est vrai

HELMO Haute Ecole Libre Mosane TECHNIQUE

7.1.3 – Démonstration par récurrence
7.1.3.1 – Principe

Exemple : « $1 + 2 + \cdots + n = \frac{1}{2} n(n+1), \forall n \geq 1$ »

- Pas initial : montrons que $p(1)$ est vrai

  $1 = \frac{1}{2} . 1 . (1+1) \Leftrightarrow 1 = 1 \rightarrow$ vrai
- Pas récurrent
  - ✓ On suppose que $p(n)$ est vrai, c'est-à-dire :

    $1 + 2 + \cdots + n = \frac{1}{2} . n . (n+1)$
  - ✓ Montrons que $p(n+1)$ est vrai, c'est-à-dire :

    $1 + 2 + \cdots + n + (n+1) = \frac{1}{2} . (n+1) . (n+2)$

HELMo Haute Ecole Libre Mosane TECHNIQUE

7.1.3 – Démonstration par récurrence

7.1.3.1 – Principe

Hypothèse de récurrence

$$\boxed{1+2+\cdots+n} = \boxed{\frac{1}{2}.n.(n+1)}$$

$$1+2+\cdots+n+(n+1) = \frac{1}{2}.n.(n+1)+(n+1)$$

$$= \frac{1}{2}.[n.(n+1)+2(n+1)]$$

$$= \frac{1}{2}(n+1)(n+2) \quad \text{OK !}$$

- Si $p(n)$ est vrai, alors $p(n+1)$ est vrai
- La proposition $p(n)$ est établie pour tout $n \geq 1$

7.1.3 – Démonstration par récurrence

7.1.3.2 – Proposition héréditaire

- Une proposition $p(n)$ pour laquelle le pas récurrent est vérifié s'appelle une **proposition héréditaire**
- Elle peut n'être vraie pour aucune valeur de n

HELMo Haute Ecole Libre Mosane TECHNIQUE

7.1.3 – Démonstration par récurrence

7.1.3.2 – Proposition héréditaire

Exemple : « $n(n+1)$ *est impair* »

- Manifestement faux, car $2\,.\,3 = 6$ qui est pair !
- Mais c'est bien une propriété héréditaire :
- Si $n(n+1)$ *est impair*, alors $n(n+1) = 2k+1$
- $(n+1)(n+2) = n^2 + 2n + n + 2$

$$= n(n+1) + 2(n+1)$$

$$= \underline{2}k + 1 + \underline{2}(n+1)$$

$$= 2(k+n+1) + 1 = 2k' + 1$$

$\rightarrow$ « $(n+1)(n+2)$ *est impair* »

7.1.3 – Démonstration par récurrence

7.1.3.3 – Récurrences emboîtées

Preuve par « récurrences emboîtées »

$$« \; n^2.(n^2-1) \; \textit{est divisible par 12}, \forall n \geq 1 \; »$$

- Vrai pour $n = 1$: $0 \; \boldsymbol{mod} \; 12 = 0$
- Supposons que $n^2.(n^2-1) \; \boldsymbol{mod} \; 12 = 0$
- Démontrons que $(n+1)^2.\left((n+1)^2-1\right) \boldsymbol{mod} \; 12 = 0$

$$(n+1)^2.\left((n+1)^2-1\right) \boldsymbol{mod} \; 12$$
$$= (n^4+4n^3+5n^2+2n) \; \boldsymbol{mod} \; 12$$
$$= (n^2.(n^2-1) + 4n^3+6n^2+2n) \; \boldsymbol{mod} \; 12$$
$$= (n^2.(n^2-1)) \; \boldsymbol{mod} \; 12 + (4n^3+6n^2+2n) \; \boldsymbol{mod} \; 12$$

- Vu l'hypothèse de récurrence, il nous reste à établir que $(4n^3+6n^2+2n) \; \boldsymbol{mod} \; 12 = 0$

HELMo Haute Ecole Libre Mosane TECHNIQUE

7.1.3 – Démonstration par récurrence

7.1.3.3 – Récurrences emboîtées

$$\text{«} \forall n \geq 1, (4n^3 + 6n^2 + 2n)\ \boldsymbol{mod}\ 12 = 0 \text{ »}$$

- Vrai pour $n = 1$: $12\ \boldsymbol{mod}\ 12 = 0$
- Supposons que $(4n^3 + 6n^2 + 2n)\ \boldsymbol{mod}\ 12 = 0$
- Démontrons que

$$\left(4(n+1)^3 + 6(n+1)^2 + 2(n+1)\right)\ \boldsymbol{mod}\ 12 = 0$$

$$\left(4(n+1)^3 + 6(n+1)^2 + 2(n+1)\right)\ \boldsymbol{mod}\ 12$$
$$= \left(4n^3 + 6n^2 + 2n + 12(n^2 + 2n + 1)\right)\ \boldsymbol{mod}\ 12$$
$$= (4n^3 + 6n^2 + 2n)\ \boldsymbol{mod}\ 12 + (12(n^2 + 2n + 1))\ \boldsymbol{mod}\ 12$$

- Vrai : hypothèse de récurrence + $(12 \times q)\ \boldsymbol{mod}\ 12 = 0$

$\Rightarrow$ Ceci achève la démonstration !

HELMO Haute Ecole Libre Mosane TECHNIQUE

7.1.3 – Démonstration par récurrence
7.1.3.4 – Utilité en informatique

- Preuve de programmes
  - Démontrer mathématiquement que des (sous-) programmes informatiques sont corrects
- Prouver qu'un programme fournit le résultat correct dans tous les cas de figure
- Plus général que le meilleur des plans de test !
- Faire une démonstration peut prendre beaucoup moins de temps que préparer et exécuter une batterie de tests …

7.1.3 – Démonstration par récurrence

7.1.3.3 – Exercices

Démontrer par récurrence que :

1. $\forall n > 0$: la somme des n premiers nombres impairs positifs est égale à n^2
2. $\forall n \geq 1 : 1^2 + 2^2 + \cdots + n^2 = \frac{1}{6}n(n+1)(2n+1)$
3. $\forall n \geq 1 : 1^3 + 2^3 + \cdots + n^3 = \frac{1}{4}n^2(n+1)^2$
4. $\forall n \in \mathbb{N} : 3^{3n+2} + 2^{n+4}$ est divisible par 5
5. $\forall n \in \mathbb{N} : 6 \, . \, 7^n - 2 \, . \, 3^n$ est divisible par 4

HELMO Haute Ecole Libre Mosane TECHNIQUE

7.1.3 – Démonstration par récurrence

7.1.3.3 – Exercices

1. $\forall n > 0$: la somme des n premiers nombres impairs positifs est égale à n^2, c'est-à-dire :

$$1 + 3 + \cdots + (2n - 1) = n^2 \text{ (*)}$$

Pas initial : vrai pour $n = 1$, car $1 = 1^2$

Pas récurrent : supposons que l'égalité (*) est vraie

$$1 + 3 + \cdots + (2n - 1) + (2(n + 1) - 1)$$
$$= n^2 + 2n + 2 - 1$$
$$= n^2 + 2n + 1$$
$$= (n + 1)^2 \; \mathit{CQFD}$$

7.1.3 – Démonstration par récurrence

7.1.3.3 – Exercices

2. $\forall n \geq 1 : 1^2 + 2^2 + \cdots + n^2 = \frac{1}{6} n(n+1)(2n+1)$ (*)

Pas initial : vrai pour $n = 1$, car $1^2 = \frac{1}{6} \cdot 1 \cdot 2 \cdot 3 = 1$

Pas récurrent : supposons que l'égalité (*) est vraie

$$1^2 + 2^2 + \cdots + n^2 + (n+1)^2$$
$$= \frac{1}{6} n\underline{(n+1)}(2n+1) + \underline{(n+1)}^2$$
$$= (n+1)\frac{1}{6}\big(n(2n+1) + 6(n+1)\big)$$
$$= \frac{1}{6}(n+1)(2n^2 + 7n + 6)$$
$$= \frac{1}{6}(n+1)(n+2)(2n+3) \qquad \mathit{CQFD}$$

7.1.3 – Démonstration par récurrence

7.1.3.3 – Exercices

3. $\forall n \geq 1 : 1^3 + 2^3 + \cdots + n^3 = \frac{1}{4} n^2 (n+1)^2$ (*)

Pas initial : vrai pour $n = 1$, car $1^3 = \frac{1}{4} \cdot 1^2 \cdot 2^2 = 1$

Pas récurrent : supposons que l'égalité (*) est vraie

$$1^3 + 2^3 + \cdots + n^3 + (n+1)^3$$

$$= \frac{1}{4} n^2 (n+1)^2 + (n+1)^3$$

$$= \frac{1}{4} (n+1)^2 \big(n^2 + 4(n+1)\big)$$

$$= \frac{1}{4} (n+1)^2 (n+2)^2 \quad \textit{CQFD}$$

HELMo Haute Ecole Libre Mosane TECHNIQUE

7.1.3 – Démonstration par récurrence

7.1.3.3 – Exercices

4. $\forall n \in \mathbb{N} : 3^{3n+2} + 2^{n+4}$ est divisible par 5

Pas initial : vrai pour $n = 0$, car $3^2 + 2^4 = 9 + 16 = 25$

Pas récurrent : supposons que $3^{3n+2} + 2^{n+4}\ mod\ 5 = 0$

$$\begin{aligned}
&(3^{3(n+1)+2} + 2^{(n+1)+4})\ mod\ 5 \\
&= (3^3 \cdot 3^{3n+2} + 2 \cdot 2^{n+4})\ mod\ 5 \\
&= (3^3(3^{3n+2} + 2^{n+4}) - 3^3 \cdot 2^{n+4} + 2 \cdot 2^{n+4})\ mod\ 5 \\
&= (3^3(3^{3n+2} + 2^{n+4}) - 25 \cdot 2^{n+4})\ mod\ 5
\end{aligned}$$

Vu l'hypothèse de récurrence et le fait que 25 est divisible par 5, la propriété est démontrée.

7.1.3 – Démonstration par récurrence

7.1.3.3 – Exercices

5. $\forall n \in \mathbb{N} : 6 \cdot 7^n - 2 \cdot 3^n$ est divisible par 4

Pas initial : vrai pour $n = 0$, car $6 \,.\, 7^0 - 2 \,.\, 3^0 = 4$

Pas récurrent :

supposons que $6 \,.\, 7^n - 2 \,.\, 3^n \; mod \; 4 = 0$

$$6 \,.\, 7^{n+1} - 2 \,.\, 3^{n+1}$$
$$= 6 \,.\, 7^n \,.\, 7 - 2 \,.\, 3^n \,.\, 3$$
$$= 6 \,.\, 7^n \,.\, (4 + 3) - 2 \,.\, 3^n . 3$$
$$= 6 \,.\, 7^n \,.\, 4 + 6 \,.\, 7^n \,.\, 3 - 2 \,.\, 3^n . 3$$
$$= 6 \,.\, 7^n \,.\, 4 + (6 \,.\, 7^n - 2 \,.\, 3^n) \,.\, 3$$

Vu l'hypothèse de récurrence et le fait que 4 est divisible par 4, la propriété est démontrée.

Haute Ecole Libre Mosane
TECHNIQUE

CH07 – Récurrence, récursivité, induction

7.1 Récurrence

7.2 Récursivité

7.2.1 Définitions

7.2.2 Algorithme itératif – Algorithme récursif

7.2.3 Principe d'écriture d'un algorithme récursif

7.2.4 Autres exemples : différentes facettes de la récursivité

7.2.5 Terminaison d'un algorithme

7.2.6 Résolution de problèmes en utilisant la récursivité

7.2.7 Algorithme récursif vs. algorithme itératif

7.2.8 Exercices

7.2.9 D'autres domaines d'application de la récursivité

7.3 Induction

HEL MO Haute Ecole Libre Mosane TECHNIQUE

7.2 – Récursivité
7.2.1 – Définitions

Définitions

- La récursivité est l’implémentation en programmation de la notion mathématique de récurrence
- La récursivité est une démarche qui fait référence à l'objet même de la démarche à un moment du processus
- L'approche récursive est un des concepts de base en informatique

HELMo Haute Ecole Libre Mosane TECHNIQUE

7.2 – Récursivité
7.2.1 – Définitions

- Un concept est dit récursif si sa définition fait appel à ce concept lui-même
- Un processus récursif peut théoriquement être répété un nombre indéfini de fois par application de la même règle, par la voie d'un automatisme
- Un algorithme est dit récursif s'il fait appel à lui-même
- Un algorithme récursif est un algorithme qui résout un problème en calculant des solutions d'instances plus petites du même problème

7.2 – Récursivité
7.2.1 – Définitions

- On oppose généralement les algorithmes récursifs aux algorithmes itératifs qui s'exécutent sans appeler explicitement l'algorithme lui-même
- La récursivité est un moyen simple et élégant pour résoudre certains problèmes (difficiles à programmer à l'aide de boucles simples)
- La récursivité peut diminuer considérablement le temps de programmation

HEL MO Haute Ecole Libre Mosane TECHNIQUE

7.2 – Récursivité
7.2.2 – Algorithme itératif – Algorithme récursif

Exemple

- Considérons la suite des sommes des n premiers nombres entiers strictement positifs

$$(S_n) = (1, 3, 6, 10, 15, 21, 28, \dots)$$

  - $S_1 = 1$
  - $S_2 = 1 + 2 = 3$
  - $S_3 = 1 + 2 + 3 = 6$
  - $S_4 = 1 + 2 + 3 + 4 = 10$
  - …
  - $S_n = 1 + 2 + 3 + \dots + n$

7.2 – Récursivité

7.2.2 – Algorithme itératif – Algorithme récursif

- Définition explicite

$$(S_n) : S_n = \frac{n(n+1)}{2}$$ *(voir 7.1.1.3 – Ex.1.5)*

- Définition itérative

$$(S_n) : S_n = 1 + 2 + \ ... + n = \sum_{i=1}^{n} i$$

- Définition récursive

$$(S_n) : S_1 = 1;\ S_n = S_{n-1} + n$$

HELMO Haute Ecole Libre Mosane TECHNIQUE

7.2 – Récursivité

7.2.2 – Algorithme itératif – Algorithme récursif

Chacune de ces définitions donne lieu à un type d'algorithme :

- Algorithme explicite (pas de boucle)
  - à partir de la définition explicite

```
somme(n)
    SI n < 1 retourne -1
    retourne n*(n+1)/2
```

-> erreur !

HELMo Haute Ecole Libre Mosane TECHNIQUE

7.2 – Récursivité

7.2.2 – Algorithme itératif – Algorithme récursif

- Algorithme itératif (une boucle) :
  - à partir de la définition itérative

```
somme(n)
    SI n < 1 retourne -1                 -> erreur !
    réponse = 1
    POUR (i = 2 ; i <= n ; i++)
        réponse = réponse + i
    retourne réponse
```

7.2 – Récursivité
7.2.2 – Algorithme itératif – Algorithme récursif

- Algorithme récursif
  - à partir de la définition récursive

```
somme(n)
    SI n < 1 retourne -1                -> erreur !
    SI n = 1 retourne 1                 -> 1ère valeur !
    réponse = somme(n - 1) + n
    retourne réponse
```

$\Rightarrow$ c’est la récursivité qui fait office de boucle !

7.2 – Récursivité

7.2.2 – Algorithme itératif – Algorithme récursif

Simulation de l'algorithme récursif

somme(4)
    retourne somme(4 - 1) + 4
        retourne somme(3 - 1) + 3
            retourne somme(2 - 1) + 2
                retourne 1
            1 + 2
        3 + 3
    6 + 4
Fin

7.2 – Récursivité

7.2.2 – Algorithme itératif – Algorithme récursif

Représentation sous la forme d'un arbre

(1 élément / sous-niveau) :

somme(4)

|

somme(4 – 1)

|

somme(3 – 1)

|

somme(2 – 1)

|

retourne 1

7.2 – Récursivité
7.2.3 – Principe d'écriture d'un algorithme récursif

Algorithme récursif à partir de la définition récursive d'une suite :

- On commence par tester si on obtient u_1
  = condition de sortie de l'algorithme
  ou cas de base

Sans sa présence, l'algorithme ne peut pas se terminer

- On procède à l'appel récursif, en utilisant la formule générale de la définition récursive de la suite, et on retourne le résultat
  = cas de propagation

HEL MO Haute Ecole Libre Mosane TECHNIQUE

7.2 – Récursivité
7.2.4 – Autres exemples : différentes facettes de la récursivité

Exemple 1 - Factorielle

- La factorielle d'un nombre naturel est définie de la manière suivante :

$$0! = 1 \text{ et } \forall n > 0 : n! = 1 \,.\, 2 \,.\, \dots \,.\, (n-1) \,.\, n$$

- Algorithme itératif

```
public static long factorielle(long n) { // n >= 0
    long fact = 1;
    for(int i = 2; i <= n; i++) fact = fact*i;
    return fact;
}
```

HELMO Haute Ecole Libre Mosane TECHNIQUE

7.2 – Récursivité
7.2.4 – Autres exemples : différentes facettes de la récursivité

- Si dans la définition précédente, on remarque que

$$n! = \underbrace{1\,.\,2\,.\,\ldots\,.\,(n-1)}_{=\,(n-1)!}\,.\,n$$

on peut définir la factorielle de manière récursive :

$$0! = 1 \text{ et } \forall n > 0 : n! = n\,.\,(n-1)!$$

- Algorithme récursif

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

HELMo Haute Ecole Libre Mosane TECHNIQUE

7.2 – Récursivité

7.2.4 – Autres exemples : différentes facettes de la récursivité

- Processus de calcul récursif pour 3!
  - `factorielle(3)`
  - `3*factorielle(2)`
  - `3*(2*factorielle(1))`
  - `3*(2*(1*factorielle(0)))`
  - `3*(2*(1*1))`
  - `3*(2*1)`
  - `3*2`
  - `6`

expansion

compression

➢ Nb d'appels et espace mémoire proportionnels à n

➢ Processus **linéairement récursif**

7.2 – Récursivité

7.2.4 – Autres exemples : différentes facettes de la récursivité

Exemple 2 – PGCD

- Définition récursive du pgcd de 2 naturels non nuls :
  - $pgcd(a, b) = b$ si $a \bmod b = 0$
  - $pgcd(a, b) = pgcd(b, a \bmod b)$ sinon
- Algorithme récursif

```
public static int pgcd(int a, int b) {  // a et b > 0
    return ((a % b == 0) ? b : pgcd(b, a % b));
}
```

7.2 – Récursivité

7.2.4 – Autres exemples : différentes facettes de la récursivité

- Processus de calcul récursif pour `pgcd(96,27)`
  - `pgcd(96,27)    96 % 27 = 15`
  - `pgcd(27,15)    27 % 15 = 12`
  - `pgcd(15,12)    15 % 12 = 3`
  - `pgcd(12,3)     12 % 3 = 0`
  - `3`

➢ **Récursivité terminale** [1] : un appel à `pgcd()` est juste remplacé par un autre appel à `pgcd()`

➢ Pas de résultat intermédiaire à stocker

[1] En anglais, *tail recursion*

7.2 – Récursivité

7.2.4 – Autres exemples : différentes facettes de la récursivité

Exemple 3 – Fibonacci

- Suite de Fibonacci : 0, 1, 1, 2, 3, 5, 8, 13, 21, …
  - Chaque terme est la somme des 2 termes qui le précédent
- Définition récursive du calcul du terme n $(n > 0)$:
  - $fibonacci(1) = 0$
  - $fibonacci(2) = 1$
  - $fibonacci(n) = fibonacci(n-1) + fibonacci(n-2)$
- Algorithme récursif

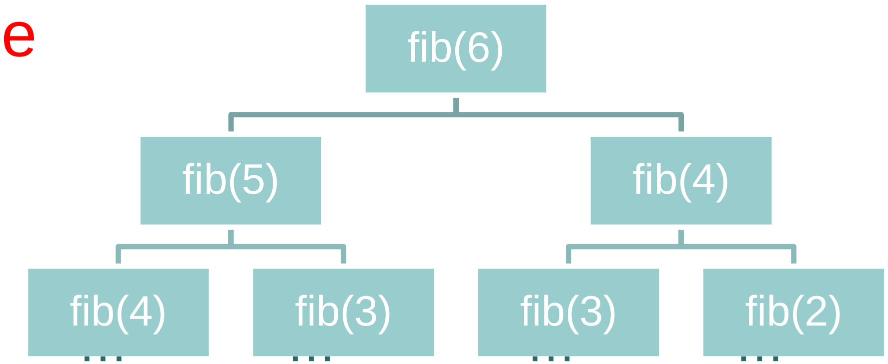
```
public static int fibonacci(int n) {   // n > 0
     if (n <= 2) return n-1;
     return fibonacci(n-1) + fibonacci(n-2);
}
```

HEL MO Haute Ecole Libre Mosane TECHNIQUE

7.2 – Récursivité

7.2.4 – Autres exemples : différentes facettes de la récursivité

- Problème



  - ➢ chaque pas récursif génère 2 appels !

- **Récursivité « arborescente »**
  - ➢ Nombre d'appels proportionnel à 2^n !

    `fibonacci(45) → 2.269.806.339` appels récursifs !
  - ➢ Complexité exponentielle → totalement inefficace !!!
  - ➢ L'algorithme itératif est de complexité linéaire ☺

7.2 – Récursivité
7.2.4 – Autres exemples : différentes facettes de la récursivité

- Algorithme itératif

```
public static int fibonacci(int n) {   // n > 0
    int fibo1 = 0, fibo2 = 1, temp;
    if (n < 2) return fibo1;
    for( int i = 2; i < n; i++) {
        temp = fibo1 + fibo2;
        fibo1 = fibo2;
        fibo2 = temp;
    }
    return fibo2;
}
```

HELMO Haute Ecole Libre Mosane TECHNIQUE

7.2 – Récursivité

7.2.4 – Autres exemples : différentes facettes de la récursivité

Récursivité imbriquée

- La complexité de certaines fonctions récursives peut être encore plus défavorable que pour Fibonacci !

Exemple : la fonction d'Ackermann

```
public static int ack(int m, int n) { // m>=0 et n>=0
    if (m == 0) return n+1;
    else if (n == 0) return ack(m-1, 1);
    else return ack(m-1, ack(m, n-1));
}
```

- `ack(4,1)=65533 → 28.629.810.010` appels récursifs !!!
- La complexité est liée aux deux appels récursifs imbriqués

HEL MO Haute Ecole Libre Mosane TECHNIQUE

7.2 – Récursivité

7.2.4 – Autres exemples : différentes facettes de la récursivité

Récursivité directe/indirecte

- Lorsqu'un algorithme fait appel à lui-même, on parle de **récursivité directe**
- Parfois, 2 algorithmes (ou plus) peuvent s'appeler l'un l'autre pour résoudre un problème, on parle alors de **récursivité indirecte**

Exemple : pair/impair

```
public static boolean pair(int n) { // n >= 0
  return (n == 0) ? true : return impair(n-1)
}
public static boolean impair(int n) { // n >= 0
  return (n == 0) ? false : return pair(n-1)
}
```

HELMo Haute Ecole Libre Mosane TECHNIQUE

7.2 – Récursivité
7.2.5 – Terminaison d'un algorithme

- Comment peut-on être certain que l'exécution d'un algorithme récursif se termine ?
- Pour aboutir à une solution, un algorithme récursif se base généralement sur :
  - ✓ un ou plusieurs **cas de base** pour lesquels la solution est connue
  - ✓ Une ou plusieurs **règle(s) de récurrence**

➢ L'algorithme se termine si et seulement si la règle de récurrence permet de se rapprocher progressivement d'un cas de base et de l'atteindre

HELMO Haute Ecole Libre Mosane TECHNIQUE

7.2 – Récursivité
7.2.5 – Terminaison d'un algorithme

Exemple 1 : factorielle

- Cas de base : $0! = 1$
- Règle de récurrence : $n! = n\,.\,(n-1)!$
  - ✓ À chaque appel récursif, la valeur de n diminue d'une unité
  - ✓ Cette valeur finira donc par atteindre 0

➢ On est donc certain que le calcul récursif de la factorielle finira par se terminer ! ☺

HELMo Haute Ecole Libre Mosane TECHNIQUE

7.2 – Récursivité
7.2.5 – Terminaison d'un algorithme

Exemple 2 : fonction de Morris

- Soit la fonction récursive suivante :

```
public static int morris(int m, int n) {
    if (m == 0) return 1;
    else return morris(m - 1, morris(m, n));
}
```

- Que vaut `morris(1, 0)` ?
  - `morris(0, morris(1, 0))`
  - `morris(0, morris(0, morris(1, 0)))`
  - …

➢ Si `(m != 0)`, le calcul ne se termine pas ! ☹

7.2 – Récursivité
7.2.5 – Terminaison d'un algorithme

Exemple 3 : paradoxe de Zénon

- Comparons la terminaison des deux énoncés récursifs suivants :
  - ✓ Pour parcourir une certaine distance, *j'avance d'un mètre*, puis je dois ensuite parcourir la distance restante

```
parcourir(n): tant que n>0 avancer(1), parcourir(n-1)
```

  - ✓ Pour parcourir une certaine distance, *j'avance de la moitié de la distance*, puis je dois ensuite parcourir la distance restante

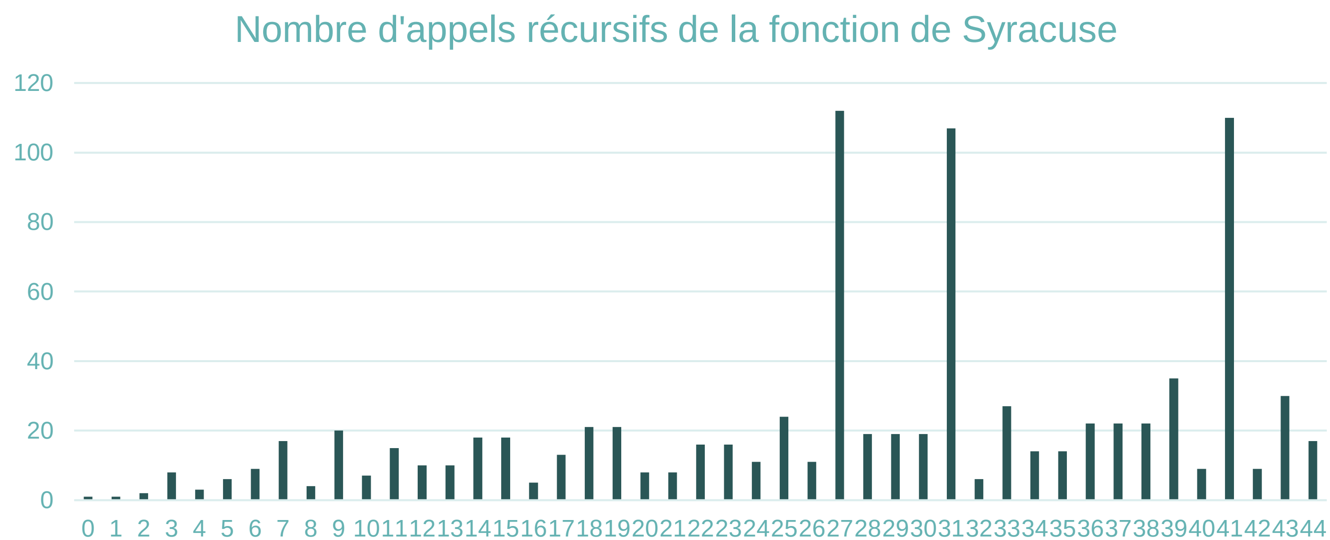
```
parcourir(n): tant que n>0 avancer(n/2), parcourir(n/2)
```

- Dans le cas 2, on n'atteindra jamais la destination finale, car dans $\mathbb{R}$ on peut diviser n indéfiniment !

7.2 – Récursivité
7.2.5 – Terminaison d’un algorithme

Exemple 4 : fonction de Syracuse

```
syracuse(n) =
  si (n = 0) ou (n = 1) alors 1
  sinon si (n mod 2 = 0) alors syracuse(n/2)
  sinon syracuse(3*n + 1)
```



7.2 – Récursivité
7.2.5 – Terminaison d'un algorithme

- Personne n'a encore réussi à démontrer la terminaison de la fonction de Syracuse pour toute valeur de $n > 1$!
- La question de déterminer si un algorithme récursif (ou non) se termine est un **problème indécidable**
  - Il n'existe aucune méthode générale permettant de répondre à coup sûr à cette question !

HELMO Haute Ecole Libre Mosane TECHNIQUE

7.2 – Récursivité

7.2.6 – Résolution de problèmes en utilisant la récursivité

- De nombreux problèmes peuvent être résolus en utilisant la récursivité
- Toutes les fonctions récursives présentent les caractéristiques suivantes :
  - La fonction est implémentée en utilisant un *if-else* (ou un *if-else if-else*) conduisant à des cas différents
  - Un (ou plusieurs) cas de base (le cas le plus simple) est (sont) utilisé(s) pour stopper la production de nouveaux appels récursifs
  - Chaque appel récursif réduit le problème de départ, de manière à ce qu’il soit de plus en plus proche du cas de base et le devienne

HEL MO Haute Ecole Libre Mosane TECHNIQUE

7.2 – Récursivité

7.2.6 – Résolution de problèmes en utilisant la récursivité

- Pour résoudre un problème par récursivité :
  - ✓ On peut le diviser en sous-problèmes
  - ✓ Chaque sous-problème est presque le même que le problème de départ, mais il est de taille plus petite
  - ✓ On peut appliquer la même approche à chaque sous-problème

HEL MO Haute Ecole Libre Mosane TECHNIQUE

7.2 – Récursivité
7.2.7 – Algorithme récursif vs. algorithme itératif

- Un algorithme récursif réalise des opérations répétitives par des appels récursifs au même sous-programme (***pas d'utilisation de boucle***); c'est une autre structure de contrôle : une instruction *if* doit être présente pour permettre un appel récursif ou non
- Dans un algorithme itératif, on utilise les boucles; la répétition du corps de la boucle est contrôlée par la structure de contrôle (*for*, *while*, …)
- Tout problème résolu de manière récursive peut l'être aussi avec des itérations non récursives

7.2 – Récursivité

7.2.7 – Algorithme récursif vs. algorithme itératif

Alors …

- Quel style de programmation choisir ?
- Quelle est la valeur ajoutée de la récursivité ?

HEL MO Haute Ecole Libre Mosane TECHNIQUE

7.2 – Récursivité

7.2.7 – Algorithme récursif vs. algorithme itératif

Exemple – Factorielle

- Version itérative

```
public static long factorielle(long n) { // n >= 0
     long fact = 1;
     for(int i = 2; i <= n; i++) fact = fact*i;
     return fact;
}
```

- On décrit les étapes successives du calcul
- Programmation **impérative** ou **procédurale**

HELMo Haute Ecole Libre Mosane TECHNIQUE

7.2 – Récursivité

7.2.7 – Algorithme récursif vs. algorithme itératif

Exemple – Factorielle

- Version récursive

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

- On construit la solution à partir d'un cas plus simple
- Niveau d'abstraction supérieur
- Programmation **déclarative**
- Programmation **fonctionnelle** (LISP)

HELMo Haute Ecole Libre Mosane TECHNIQUE