R2.09 Méthodes Numériques

Thibault Godin, Lucie Naert, Anthony Ridard
IUT de Vannes Informatique

Contenu du cours

Le but du cours $\rightsquigarrow$ donner des outils mathématiques nécessaires à la compréhension et la modélisation de problèmes informatiques.

Contenu du cours

Le but du cours $\rightsquigarrow$ donner des outils mathématiques nécessaires à la compréhension et la modélisation de problèmes informatiques.

En particulier, on s'intéressera à 4 grands problèmes :

Contenu du cours

Le but du cours $\rightsquigarrow$ donner des outils mathématiques nécessaires à la compréhension et la modélisation de problèmes informatiques.

En particulier, on s'intéressera à 4 grands problèmes :

- Comment modéliser un processus discret ?
  $\rightsquigarrow$ Première notions de suite

Contenu du cours

Le but du cours $\rightsquigarrow$ donner des outils mathématiques nécessaires à la compréhension et la modélisation de problèmes informatiques.

En particulier, on s'intéressera à 4 grands problèmes :

- Comment modéliser un processus discret ?
  $\rightsquigarrow$ Première notions de suite
- Quel est le temps d'exécution d'un programme ?
  $\rightsquigarrow$ Théorie de la complexité

Contenu du cours

Le but du cours $\rightsquigarrow$ donner des outils mathématiques nécessaires à la compréhension et la modélisation de problèmes informatiques.

En particulier, on s'intéressera à 4 grands problèmes :

- Comment modéliser un processus discret ?
  $\rightsquigarrow$ Première notions de suite
- Quel est le temps d'exécution d'un programme ?
  $\rightsquigarrow$ Théorie de la complexité
- Comment évolue une valeur numérique au cours d'un programme ?
  $\rightsquigarrow$ vérification (model checking)

Contenu du cours

Le but du cours $\rightsquigarrow$ donner des outils mathématiques nécessaires à la compréhension et la modélisation de problèmes informatiques.

En particulier, on s'intéressera à 4 grands problèmes :

- Comment modéliser un processus discret ?
  $\rightsquigarrow$ Première notions de suite
- Quel est le temps d'exécution d'un programme ?
  $\rightsquigarrow$ Théorie de la complexité
- Comment évolue une valeur numérique au cours d'un programme ?
  $\rightsquigarrow$ vérification (model checking)
- Comment trouver algorithmiquement une solution à une équation ?
  $\rightsquigarrow$ analyse numérique

Modélisation : loi de Moore

En 1975, Gordon E. Moore postule une tendance globale sur le nombre de transistors dans un microprocesseur :

Modélisation : loi de Moore

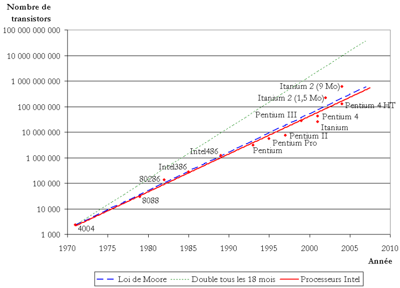
En 1975, Gordon E. Moore postule une tendance globale sur le nombre de transistors dans un microprocesseur :

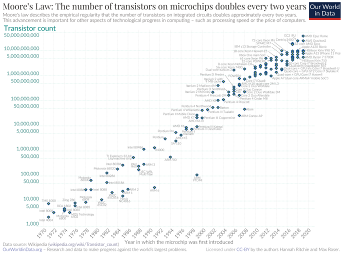
Le nombre de transistors double tous les deux ans

Modélisation : loi de Moore

En 1975, Gordon E. Moore postule une tendance globale sur le nombre de transistors dans un microprocesseur :

Le nombre de transistors double tous les deux ans

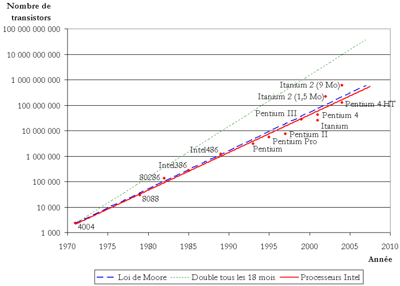


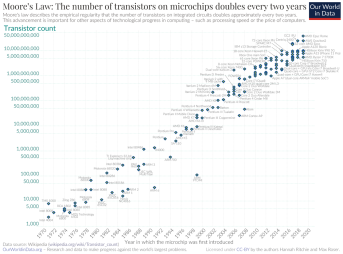


Modélisation : loi de Moore

En 1975, Gordon E. Moore postule une tendance globale sur le nombre de transistors dans un microprocesseur :

Le nombre de transistors double tous les deux ans



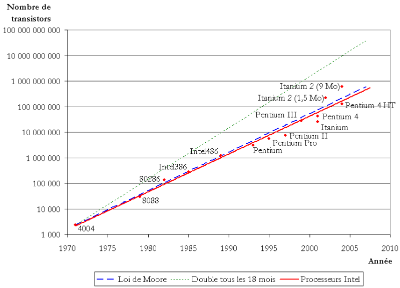


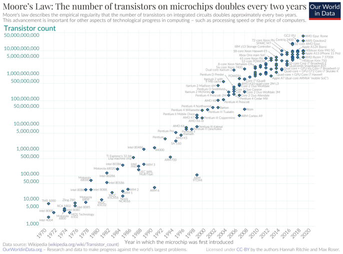
sources : wikipedia.fr QcRef87 CC BY-SA 3.0 Max Roser, Hannah Ritchie CC BY-SA 4.0 Donc si t_{1975} est le nombre de transistor en 1975, on a $t_{1977} = 2t_{1975}$

Modélisation : loi de Moore

En 1975, Gordon E. Moore postule une tendance globale sur le nombre de transistors dans un microprocesseur :

Le nombre de transistors double tous les deux ans



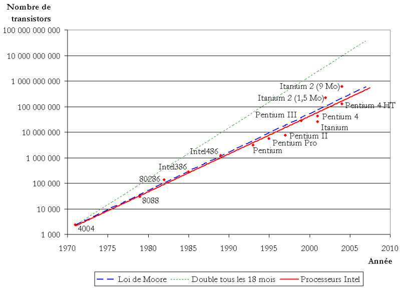


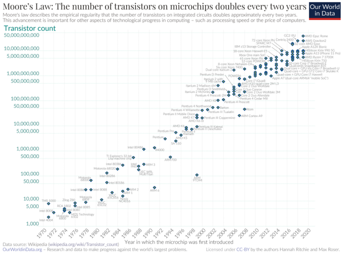
 Donc si t_{1975} est le nombre de transistor en 1975, on a $t_{1977} = 2t_{1975}$, $t_{1979} = 2t_{1977} = 2(2t_{1975})$

Modélisation : loi de Moore

En 1975, Gordon E. Moore postule une tendance globale sur le nombre de transistors dans un microprocesseur :

Le nombre de transistors double tous les deux ans





sources : wikipedia.fr QcRef87 CC BY-SA 3.0 Max Roser, Hannah Ritchie CC BY-SA 4.0

Donc si t_{1975} est le nombre de transistor en 1975, on a $t_{1977} = 2t_{1975}$, $t_{1979} = 2t_{1977} = 2(2t_{1975})$ et donc $t_{2023} = 2^{24}t_{1975} = 16\,777\,216 t_{1975}$

Analyse et Quake 3

Dans le jeu vidéo (et le traitement de l'image en général), les réflections sont calculées en fonction de la normale (perpendiculaire) à une surface.

Analyse et Quake 3

Dans le jeu vidéo (et le traitement de l'image en général), les réflections sont calculées en fonction de la normale (perpendiculaire) à une surface.

En particulier, étant donné un vecteur (x, y, z), on doit le *normaliser*, c'est à dire calculer :

$$\mathrm{v} = \left(\frac{x}{\sqrt{x^2 + y^2 + z^2}}, \frac{y}{\sqrt{x^2 + y^2 + z^2}}, \frac{z}{\sqrt{x^2 + y^2 + z^2}} \right)$$

Analyse et Quake 3

Dans le jeu vidéo (et le traitement de l'image en général), les réflections sont calculées en fonction de la normale (perpendiculaire) à une surface.

En particulier, étant donné un vecteur (x, y, z), on doit le *normaliser*, c'est à dire calculer :

$$\mathrm{v} = \left(\frac{x}{\sqrt{x^2+y^2+z^2}}, \frac{y}{\sqrt{x^2+y^2+z^2}}, \frac{z}{\sqrt{x^2+y^2+z^2}} \right)$$

Analysons ce calcul :

- $x^2 + y^2 + z^2 \rightsquigarrow$ `x*x + y*y + z*z` $\rightsquigarrow$ Ok et efficace

Analyse et Quake 3

Dans le jeu vidéo (et le traitement de l'image en général), les réflections sont calculées en fonction de la normale (perpendiculaire) à une surface.

En particulier, étant donné un vecteur (x, y, z), on doit le *normaliser*, c'est à dire calculer :

$$\mathrm{v} = \left(\frac{x}{\sqrt{x^2 + y^2 + z^2}}, \frac{y}{\sqrt{x^2 + y^2 + z^2}}, \frac{z}{\sqrt{x^2 + y^2 + z^2}} \right)$$

Analysons ce calcul :

- $x^2 + y^2 + z^2 \rightsquigarrow$ `x*x + y*y + z*z` $\rightsquigarrow$ Ok et efficace
- $x/R \rightsquigarrow$ `x/R` $\rightsquigarrow$ Ok, mais coûteux

Analyse et Quake 3

Dans le jeu vidéo (et le traitement de l'image en général), les réflections sont calculées en fonction de la normale (perpendiculaire) à une surface.

En particulier, étant donné un vecteur (x, y, z), on doit le *normaliser*, c'est à dire calculer :

$$\mathrm{v} = \left(\frac{x}{\sqrt{x^2 + y^2 + z^2}}, \frac{y}{\sqrt{x^2 + y^2 + z^2}}, \frac{z}{\sqrt{x^2 + y^2 + z^2}} \right)$$

Analysons ce calcul :

- $x^2 + y^2 + z^2 \rightsquigarrow$ `x*x + y*y + z*z` $\rightsquigarrow$ Ok et efficace
- $x/R \rightsquigarrow$ `x/R` $\rightsquigarrow$ Ok, mais coûteux
- $\sqrt{N} \rightsquigarrow ???$

Analyse et Quake 3

Dans le jeu vidéo (et le traitement de l'image en général), les réflections sont calculées en fonction de la normale (perpendiculaire) à une surface.

En particulier, étant donné un vecteur (x, y, z), on doit le *normaliser*, c'est à dire calculer :

$$\mathrm{v} = \left(\frac{x}{\sqrt{x^2+y^2+z^2}}, \frac{y}{\sqrt{x^2+y^2+z^2}}, \frac{z}{\sqrt{x^2+y^2+z^2}} \right)$$

Analysons ce calcul :

- $x^2 + y^2 + z^2 \rightsquigarrow$ `x*x + y*y + z*z` $\rightsquigarrow$ Ok et efficace
- $x/R \rightsquigarrow$ `x/R` $\rightsquigarrow$ Ok, mais coûteux
- $\sqrt{N} \rightsquigarrow ???$

Analyse et Quake 3

Dans le jeu vidéo (et le traitement de l'image en général), les réflections sont calculées en fonction de la normale (perpendiculaire) à une surface.

En particulier, étant donné un vecteur (x, y, z), on doit le *normaliser*, c'est à dire calculer :

$$\mathrm{v} = \left(\frac{x}{\sqrt{x^2 + y^2 + z^2}}, \frac{y}{\sqrt{x^2 + y^2 + z^2}}, \frac{z}{\sqrt{x^2 + y^2 + z^2}} \right)$$

Analysons ce calcul :

- $x^2 + y^2 + z^2 \rightsquigarrow$ `x*x + y*y + z*z` $\rightsquigarrow$ Ok et efficace
- $x/R \rightsquigarrow$ `x/R` $\rightsquigarrow$ Ok, mais coûteux
- $\sqrt{N} \rightsquigarrow ???$

L'ordinateur manie des entiers.

Analyse et Quake 3

Dans le jeu vidéo (et le traitement de l'image en général), les réflections sont calculées en fonction de la normale (perpendiculaire) à une surface.

En particulier, étant donné un vecteur (x, y, z), on doit le *normaliser*, c'est à dire calculer :

$$\mathrm{v} = \left(\frac{x}{\sqrt{x^2+y^2+z^2}}, \frac{y}{\sqrt{x^2+y^2+z^2}}, \frac{z}{\sqrt{x^2+y^2+z^2}} \right)$$

Analysons ce calcul :

- $x^2 + y^2 + z^2 \rightsquigarrow$ `x*x + y*y + z*z` $\rightsquigarrow$ Ok et efficace
- $x/R \rightsquigarrow$ `x/R` $\rightsquigarrow$ Ok, mais coûteux
- $\sqrt{N} \rightsquigarrow ???$

L'ordinateur manie des entiers. On sait représenter certains réels (flottant IEEE754)

Analyse et Quake 3

Dans le jeu vidéo (et le traitement de l'image en général), les réflections sont calculées en fonction de la normale (perpendiculaire) à une surface.

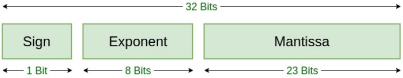
En particulier, étant donné un vecteur (x, y, z), on doit le *normaliser*, c'est à dire calculer :

$$\mathrm{v} = \left(\frac{x}{\sqrt{x^2+y^2+z^2}}, \frac{y}{\sqrt{x^2+y^2+z^2}}, \frac{z}{\sqrt{x^2+y^2+z^2}} \right)$$

Analysons ce calcul :

- $x^2 + y^2 + z^2 \rightsquigarrow$ `x*x + y*y + z*z` $\rightsquigarrow$ Ok et efficace
- $x/R \rightsquigarrow$ `x/R` $\rightsquigarrow$ Ok, mais coûteux
- $\sqrt{N} \rightsquigarrow ???$

L'ordinateur manie des entiers. On sait représenter certains réels (flottant IEEE754)



$$f = (-1)^{sign_2} \times 2^{exponent_2 - 127} \times (1.mantissa_2)$$

Analyse et Quake 3

Dans le jeu vidéo (et le traitement de l'image en général), les réflections sont calculées en fonction de la normale (perpendiculaire) à une surface.

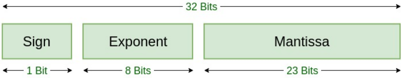
En particulier, étant donné un vecteur (x, y, z), on doit le *normaliser*, c'est à dire calculer :

$$\mathrm{v} = \left(\frac{x}{\sqrt{x^2+y^2+z^2}}, \frac{y}{\sqrt{x^2+y^2+z^2}}, \frac{z}{\sqrt{x^2+y^2+z^2}} \right)$$

Analysons ce calcul :

- $x^2 + y^2 + z^2 \rightsquigarrow$ `x*x + y*y + z*z` $\rightsquigarrow$ Ok et efficace
- $x/R \rightsquigarrow$ `x/R` $\rightsquigarrow$ Ok, mais coûteux
- $\sqrt{N} \rightsquigarrow ???$

L'ordinateur manie des entiers. On sait représenter certains réels (flottant IEEE754)



$$f = (-1)^{sign_2} \times 2^{exponent_2 - 127} \times (1.mantissa_2)$$

- ~~Comment calculer~~ $\cancel{\sqrt{f}}$?

Analyse et Quake 3

Dans le jeu vidéo (et le traitement de l'image en général), les réflections sont calculées en fonction de la normale (perpendiculaire) à une surface.

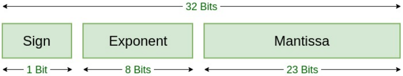
En particulier, étant donné un vecteur (x, y, z), on doit le *normaliser*, c'est à dire calculer :

$$\mathrm{v} = \left(\frac{x}{\sqrt{x^2+y^2+z^2}}, \frac{y}{\sqrt{x^2+y^2+z^2}}, \frac{z}{\sqrt{x^2+y^2+z^2}} \right)$$

Analysons ce calcul :

- $x^2 + y^2 + z^2 \rightsquigarrow$ `x*x + y*y + z*z` $\rightsquigarrow$ Ok et efficace
- $x/R \rightsquigarrow$ `x/R` $\rightsquigarrow$ Ok, mais coûteux
- $\sqrt{N} \rightsquigarrow ???$

L'ordinateur manie des entiers. On sait représenter certains réels (flottant IEEE754)



$$f = (-1)^{sign_2} \times 2^{exponent_2 - 127} \times (1.mantissa_2)$$

- ~~Comment calculer $\sqrt{f}$~~ ?
- On a seulement un nombre fini de `float`, comment donner une bonne approximation de $\sqrt{f}$?

Analyse et Quake 3

Solution 1 : `x/ sqrt(x*x + y*y + z*z)`

Analyse et Quake 3

Solution 1 : `x/ sqrt(x*x + y*y + z*z)` $\rightsquigarrow$ ok mais division et boite noire

Analyse et Quake 3

Solution 1 : `x/ sqrt(x*x + y*y + z*z)` $\rightsquigarrow$ ok mais division et boite noire $\rightsquigarrow$ moyennement efficace et satisfaisant

Analyse et Quake 3

Solution 1 : `x/ sqrt(x*x + y*y + z*z)` $\rightsquigarrow$ ok mais division et boite noire $\rightsquigarrow$ moyennement efficace et satisfaisant

Solution 2 : calcul approché de $1/\sqrt{x^2 + y^2 + z^2}$

```
float Q_rsqrt( float number )
{
long i;
float x2, y;
const float threehalfs = 1.5F;

x2 = number * 0.5F;
y  = number;
i  = * ( long * ) &y;                    // evil floating point
    bit level hacking
i  = 0x5f3759df - ( i >> 1 );            // what the fuck?
y  = * ( float * ) &i;
y  = y * ( threehalfs - ( x2 * y * y ) );    // 1st iteration

//      y  = y * ( threehalfs - ( x2 * y * y ) );
// 2nd iteration, this can be removed

return y;
}
```

subtilité du langage C

approximation grossière de $1/\sqrt{f}$

amélioration de l'approximation $1/\sqrt{f}$

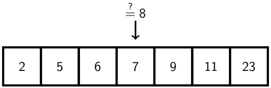
Efficacité des algorithmes

Recherches naïve et dichotomique dans une liste de n entiers triés

Efficacité des algorithmes

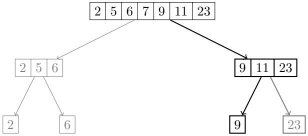
Recherches naïve et dichotomique dans une liste de n entiers triés

recherche naïve



au pire $\rightsquigarrow$ n tests d'égalité/comparaisons

recherche dichotomique

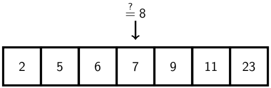


au pire $\rightsquigarrow$ $\log_2(n+1)$ tests

Efficacité des algorithmes

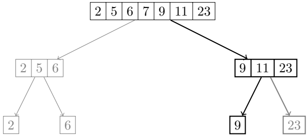
Recherches naïve et dichotomique dans une liste de n entiers triés

recherche naïve



au pire $\rightsquigarrow$ n tests d'égalité/comparaisons

recherche dichotomique



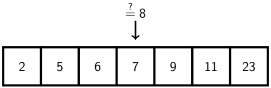
au pire $\rightsquigarrow$ $\log_2(n+1)$ tests

Quel algorithme est le plus rapide ?

Efficacité des algorithmes

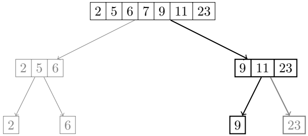
Recherches naïve et dichotomique dans une liste de n entiers triés

recherche naïve



au pire $\rightsquigarrow$ n tests d'égalité/comparaisons

recherche dichotomique



au pire $\rightsquigarrow$ $\log_2(n+1)$ tests

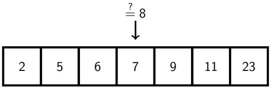
Quel algorithme est le plus rapide ?

n	10	100	1000	10 000	100 000	1 000 000	10 000 000
$\log_2(n+1)$	3.46	6.66	9.97	13.29	16.61	19.93	23.25

Efficacité des algorithmes

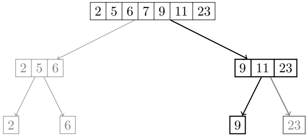
Recherches naïve et dichotomique dans une liste de n entiers triés

recherche naïve



au pire $\rightsquigarrow$ n tests d'égalité/comparaisons

recherche dichotomique



au pire $\rightsquigarrow$ $\log_2(n+1)$ tests

Quel algorithme est le plus rapide ?

n	10	100	1000	10 000	100 000	1 000 000	10 000 000
$\log_2(n+1)$	3.46	6.66	9.97	13.29	16.61	19.93	23.25

On peut étudier $u_n = \frac{\log_2(n+1)}{n}$

$u_n \to_{n\to\infty} 0$, la recherche dichotomique est bien plus efficace que la recherche naïve

Suite

- Une ***suite*** est une application $u : \mathbb{N} \to \mathbb{R}$.
- Pour $n \in \mathbb{N}$, on note $u(n)$ par u_n et on l'appelle n-ième ***terme*** ou ***terme général*** de la suite.

Suite

- Une ***suite*** est une application $u : \mathbb{N} \to \mathbb{R}$.
- Pour $n \in \mathbb{N}$, on note $u(n)$ par u_n et on l'appelle n-ième ***terme*** ou ***terme général*** de la suite.

- $u_n = n$
- $u_n = \frac{1}{n^2} + 5$
- $u_n =$ le n-ième nombre premier

Comment définit-on une suite ?

Il y a plusieurs façons de définir une suite :

Comment définit-on une suite ?

Il y a plusieurs façons de définir une suite :

- ***Par une formule explicite***,
  en fonction de l'entier n ;
  - $u_n = \sin \frac{1}{n}$, définie pour $n \in \mathbb{N}^*$
    $u_1 = \sin 1$, $u_2 = \sin \frac{1}{2}$ ...

Comment définit-on une suite ?

Il y a plusieurs façons de définir une suite :

- ***Par une formule explicite***,
  en fonction de l'entier n ;
  - $u_n = \sin \frac{1}{n}$, définie pour
    $n \in \mathbb{N}^*$
    $u_1 = \sin 1,\ u_2 = \sin \frac{1}{2}$ ...
  - $v_n = \pi 2^{-n}$
    $v_0 = \pi,\ v_1 = \frac{\pi}{2},\ v_2 = \frac{\pi}{4}$

Comment définit-on une suite ?

Il y a plusieurs façons de définir une suite :

- ***Par une formule explicite***,
  en fonction de l'entier n ;
  - $u_n = \sin \frac{1}{n}$, définie pour
    $n \in \mathbb{N}^*$
    $u_1 = \sin 1,\ u_2 = \sin \frac{1}{2}$ ...
  - $v_n = \pi 2^{-n}$
    $v_0 = \pi,\ v_1 = \frac{\pi}{2},\ v_2 = \frac{\pi}{4}$
  - suite géométrique
    $u_n = aq^n$

Comment définit-on une suite ?

Il y a plusieurs façons de définir une suite :

- ***Par une formule explicite***,
  en fonction de l'entier n ;
  - $u_n = \sin \frac{1}{n}$, définie pour $n \in \mathbb{N}^*$
    $u_1 = \sin 1,\ u_2 = \sin \frac{1}{2}\ \ldots$
  - $v_n = \pi 2^{-n}$
    $v_0 = \pi,\ v_1 = \frac{\pi}{2},\ v_2 = \frac{\pi}{4}$
  - suite géométrique
    $u_n = aq^n$

Comment définit-on une suite ?

Il y a plusieurs façons de définir une suite :

- ***Par une formule explicite***, en fonction de l'entier n ;
  - $u_n = \sin \frac{1}{n}$, définie pour $n \in \mathbb{N}^*$
    $u_1 = \sin 1,\ u_2 = \sin \frac{1}{2}\ \ldots$
  - $v_n = \pi 2^{-n}$
    $v_0 = \pi,\ v_1 = \frac{\pi}{2},\ v_2 = \frac{\pi}{4}$
  - suite géométrique
    $u_n = aq^n$

```
def explicit(n):

  un = a*(q**n)
  return un
```

Comment définit-on une suite ?

Il y a plusieurs façons de définir une suite :

- ***Par une formule itérative***
  (Formule de récurrence)

Comment définit-on une suite ?

Il y a plusieurs façons de définir une suite :

- ***Par une formule itérative***

  (Formule de récurrence)

  suite de Fibonacci.

$$\begin{cases} F_{n+2} = & F_{n+1} + F_n \\ F_0 = & 1 \\ F_1 = & 1 \end{cases}$$

Comment définit-on une suite ?

Il y a plusieurs façons de définir une suite :

- ***Par une formule itérative***

  (Formule de récurrence)

  suite de Fibonacci.

$$\begin{cases} F_{n+2} = & F_{n+1} + F_n \\ F_0 = & 1 \\ F_1 = & 1 \end{cases}$$

```
def FibonacciRec(n):

  if n==0 or n==1:
    return 1
  else:
    return  FibonacciRec(n-1)+
                 FibonacciRec(n-2)
```

Comment définit-on une suite ?

Il y a plusieurs façons de définir une suite :

- ***Par une formule itérative***

  (Formule de récurrence)

  suite de Fibonacci.

$$\begin{cases} F_{n+2} = & F_{n+1} + F_n \\ F_0 = & 1 \\ F_1 = & 1 \end{cases}$$

```
def FibonacciRec(n):

  if n==0 or n==1:
    return 1
  else:
    return  FibonacciRec(n-1)+
                  FibonacciRec(n-2)
```

Calculer les 5 premiers termes de la suite.

Comment définit-on une suite ?

Il y a plusieurs façons de définir
une suite :

- ***à l'aide d'une formule itérative utilisant une fonction***

  par exemple la suite :

  $u_0 = a$ et

  $u_{n+1} = f(u_n)$

Comment définit-on une suite ?

Il y a plusieurs façons de définir une suite :

- ***à l'aide d'une formule itérative utilisant une fonction***

  par exemple la suite :

  $u_0 = a$ et

  $u_{n+1} = f(u_n)$

  Exemple :

  $f : x \mapsto \frac{1}{2}\left(x + \frac{3}{x}\right)$

$$\begin{cases} u_0 = 5 \\ u_{n+1} = \dfrac{1}{2}\left(u_n + \dfrac{3}{u_n}\right) \end{cases}$$

Comment définit-on une suite ?

Il y a plusieurs façons de définir une suite :

- ***à l'aide d'une formule itérative utilisant une fonction***

  par exemple la suite :

  $u_0 = a$ et

  $u_{n+1} = f(u_n)$

  Exemple :

  $f : x \mapsto \frac{1}{2}\left(x + \frac{3}{x}\right)$

$$\begin{cases} u_0 = 5 \\ u_{n+1} = \dfrac{1}{2}\left(u_n + \dfrac{3}{u_n}\right) \end{cases}$$

```
def f(x):

  return 1/2*(x + 3/x)

def un(n):
  if n==0:
    return 5
  else:
    return f(un(n-1))
```

Comment définit-on une suite ?

Il y a plusieurs façons de définir une suite :

- ***à l'aide d'une formule itérative utilisant une fonction***

  par exemple la suite :

  $u_0 = a$ et

  $u_{n+1} = f\left(u_n\right)$

  Exemple :

  $f : x \mapsto \frac{1}{2}\left(x + \frac{3}{x}\right)$

  $$\begin{cases} u_0 = 5 \\ u_{n+1} = \dfrac{1}{2}\left(u_n + \dfrac{3}{u_n}\right) \end{cases}$$

```
def f(x):

  return 1/2*(x + 3/x)

def un(n):
  if n==0:
    return 5
  else:
    return f(un(n-1))
```

✎ Calculer les 3 premiers termes de la suite, puis donner à la calculatrice une valeur approchée de u_5.

Premières propriétés

Définition

Soit $(u_n)_{n\in\mathbb{N}}$ une suite.

- $(u_n)_{n\in\mathbb{N}}$ est ***majorée*** si $\quad \exists M \in \mathbb{R} \quad \forall n \in \mathbb{N} \quad u_n \leq M.$

Premières propriétés

Définition

Soit $(u_n)_{n\in\mathbb{N}}$ une suite.

- $(u_n)_{n\in\mathbb{N}}$ est ***majorée*** si $\quad \exists M \in \mathbb{R} \quad \forall n \in \mathbb{N} \quad u_n \leq M.$
- $(u_n)_{n\in\mathbb{N}}$ est ***minorée*** si $\quad \exists m \in \mathbb{R} \quad \forall n \in \mathbb{N} \quad u_n \geq m.$

Premières propriétés

Définition

Soit $(u_n)_{n\in\mathbb{N}}$ une suite.

- $(u_n)_{n\in\mathbb{N}}$ est ***majorée*** si $\quad \exists M \in \mathbb{R} \quad \forall n \in \mathbb{N} \quad u_n \leq M.$
- $(u_n)_{n\in\mathbb{N}}$ est ***minorée*** si $\quad \exists m \in \mathbb{R} \quad \forall n \in \mathbb{N} \quad u_n \geq m.$
- $(u_n)_{n\in\mathbb{N}}$ est ***bornée*** si elle est majorée et minorée, ce qui revient à dire :

$$\exists B \in \mathbb{R} \quad \forall n \in \mathbb{N} \quad |u_n| \leq B.$$

Premières propriétés

Définition

Soit $(u_n)_{n\in\mathbb{N}}$ une suite.

- $(u_n)_{n\in\mathbb{N}}$ est ***majorée*** si $\quad \exists M \in \mathbb{R} \quad \forall n \in \mathbb{N} \quad u_n \leq M.$
- $(u_n)_{n\in\mathbb{N}}$ est ***minorée*** si $\quad \exists m \in \mathbb{R} \quad \forall n \in \mathbb{N} \quad u_n \geq m.$
- $(u_n)_{n\in\mathbb{N}}$ est ***bornée*** si elle est majorée et minorée, ce qui revient à dire :

$$\exists B \in \mathbb{R} \quad \forall n \in \mathbb{N} \quad |u_n| \leq B.$$

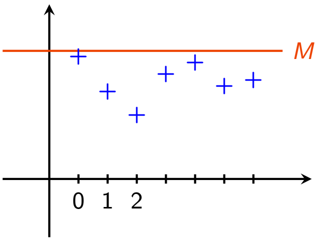
Premières propriétés

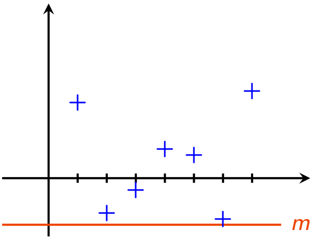
Définition

Soit $(u_n)_{n\in\mathbb{N}}$ une suite.

- $(u_n)_{n\in\mathbb{N}}$ est ***majorée*** si $\quad \exists M \in \mathbb{R} \quad \forall n \in \mathbb{N} \quad u_n \leq M.$
- $(u_n)_{n\in\mathbb{N}}$ est ***minorée*** si $\quad \exists m \in \mathbb{R} \quad \forall n \in \mathbb{N} \quad u_n \geq m.$
- $(u_n)_{n\in\mathbb{N}}$ est ***bornée*** si elle est majorée et minorée, ce qui revient à dire :

$$\exists B \in \mathbb{R} \quad \forall n \in \mathbb{N} \quad |u_n| \leq B.$$





Premières propriétés

Montrer que la suite $u_n = -n$ est majorée.

Donner un exemple de suite bornée et un autre de suite ni minorée ni majorée.

Montrer que la suite de Fibonacci est minorée par 1 mais n'est pas majorée.

Démonstration par récurrence

On utilise l'implication suivante[a] :

$$\Big(\underbrace{\mathcal{P}(0)}_{\text{Initialisation}} \quad \text{et} \quad \big(\underbrace{\forall n \in \mathbb{N},\ \mathcal{P}(n) \Longrightarrow \mathcal{P}(n+1)}_{\text{Hérédité}}\big)\Big) \Longrightarrow \Big(\forall n \in \mathbb{N},\ \mathcal{P}(n)\Big)$$

a. Il s'agit du principe de récurrence

- Cette technique est à privilégier lorsque la démonstration *directe*[a] n'aboutit pas
- On peut généraliser :
  - en initialisant à un entier $n_0 > 0$
  - en considérant une récurrence « forte »

a. Considérer un n quelconque de $\mathbb{N}$ et montrer $\mathcal{P}(n)$

Récurrence

<u>Initialisation</u> :

Vérifions $\mathcal{P}(0)$

$\left.\begin{array}{c}\vdots\end{array}\right\}$ Vérification de $\mathcal{P}(0)$

<u>Hérédité</u> :

Soit $n \in \mathbb{N}$

Supposons[a] $\mathcal{P}(n)$

Montrons $\mathcal{P}(n+1)$

$\left.\begin{array}{c}\vdots\end{array}\right\}$ Preuve de $\mathcal{P}(n+1)$

a. Il s'agit de l'Hypothèse de Récurrence (HR)

Récurrence

On considère la suite $\left(u_n\right)_{n\in\mathbb{N}}$ définie par : $u_0 = 1$ et $\forall n \in \mathbb{N},\ u_{n+1} = 2u_n$.
Démontrer « $\forall n \in \mathbb{N},\ u_n = 2^n$ ».

Récurrence

On considère la suite $\left(u_n\right)_{n\in\mathbb{N}}$ définie par : $u_0 = 1$ et $\forall n \in \mathbb{N},\ u_{n+1} = 2u_n$.
Démontrer « $\forall n \in \mathbb{N},\ u_n = 2^n$ ».

Montrer que la suite de Fibonacci est minorée par 1 mais n'est pas majorée.
Donner un exemple de suite bornée et un autre de suite ni minorée ni majorée.

Premières propriétés

Une suite est dite :

1. ***croissante*** si, $\forall n \in \mathbb{N}\ u_{n+1} \geqslant u_n$

Premières propriétés

Une suite est dite :

1. ***croissante*** si, $\forall n \in \mathbb{N}\ u_{n+1} \geqslant u_n$
2. ***décroissante*** si, $\forall n \in \mathbb{N}\ u_{n+1} \leqslant u_n$

Premières propriétés

Une suite est dite :

1. ***croissante*** si, $\forall n \in \mathbb{N}\ u_{n+1} \geqslant u_n$
2. ***décroissante*** si, $\forall n \in \mathbb{N}\ u_{n+1} \leqslant u_n$
3. ***monotone*** si elle est croissante ou décroissante

Premières propriétés

Une suite est dite :

1. ***croissante*** si, $\forall n \in \mathbb{N}\ u_{n+1} \geqslant u_n$
2. ***décroissante*** si, $\forall n \in \mathbb{N}\ u_{n+1} \leqslant u_n$
3. ***monotone*** si elle est croissante ou décroissante

Premières propriétés

Une suite est dite :

1. ***croissante*** si, $\forall n \in \mathbb{N}$ $u_{n+1} \geqslant u_n$
2. ***décroissante*** si, $\forall n \in \mathbb{N}$ $u_{n+1} \leqslant u_n$
3. ***monotone*** si elle est croissante ou décroissante

Pour étudier la monotonie on étudie les suites :

1. u_{n+1} - u_n (en la comparant à 0)
   si $(u_n)_{n \in \mathbb{N}}$ est une suite positive :
2. $\frac{u_{n+1}}{u_n}$ (en la comparant à 1).

Premières propriétés (suite)

✍ Donner un exemple de suite non monotone bornée et un autre de suite décroissante non minorée.

✍ Montrer que la suite de Fibonacci est croissante (étudier $F_n - F_{n-1}$), puis que la suite $(u_n)_{n\in\mathbb{N}} = \left(\frac{1}{2^n}\right)_{n\in\mathbb{N}}$ est décroissante

✍ Le produit de deux suites croissantes est-il une suite croissante ? *(Que pensez vous des suites* $u_n = -\dfrac{1}{n^3}$ *ou* $v_n = -1 - \dfrac{1}{n}$ *et du produit* $u_n v_n$ *?)*

Suites de référence

On dit qu'une suite $(u_n)_{n\in\mathbb{N}}$ est une suite ***arithmétique*** si $u_n = u_{n-1} + r$ et $u_0 = a$

(TP) Étudier la suite arithmétique définie par $u_n = u_{n-1} + r$ et $u_0 = a$

Donner une formule explicite en fonction de n.

Illustrer les comportements possibles de la suite selon les paramètre a et r

Suites de référence

On dit qu'une suite $(u_n)_{n\in\mathbb{N}}$ est une suite ***géométrique*** si $u_n = q \cdot u_{n-1}$ et $u_0 = a$

(TP) Étudier la suite géométrique définie par $u_n = qu_{n-1}$ et $u_0 = a$

Donner une formule explicite en fonction de n.

Illustrer les comportements possibles de la suite selon les paramètre a et q

Limite

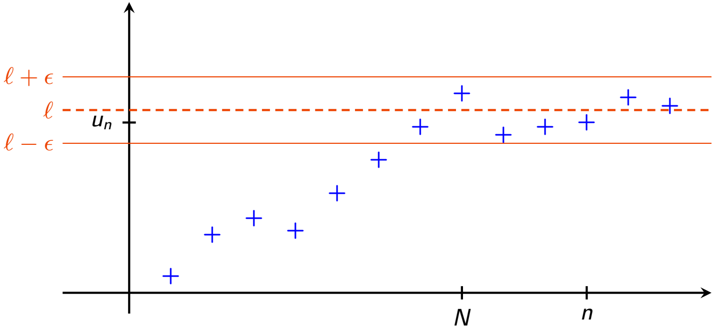
La suite $(u_n)_{n\in\mathbb{N}}$ a pour ***limite*** $\ell \in \mathbb{R}$ si : pour tout $\varepsilon > 0$, il existe un entier naturel N tel que si $n \geq N$ alors $|u_n - \ell| \leq \varepsilon$:

$$\forall \varepsilon > 0 \quad \exists N \in \mathbb{N} \quad \forall n \in \mathbb{N} \qquad (n \geq N \implies |u_n - \ell| \leq \varepsilon)$$

Limite

La suite $(u_n)_{n\in\mathbb{N}}$ a pour ***limite*** $\ell \in \mathbb{R}$ si : pour tout $\varepsilon > 0$, il existe un entier naturel N tel que si $n \geq N$ alors $|u_n - \ell| \leq \varepsilon$:

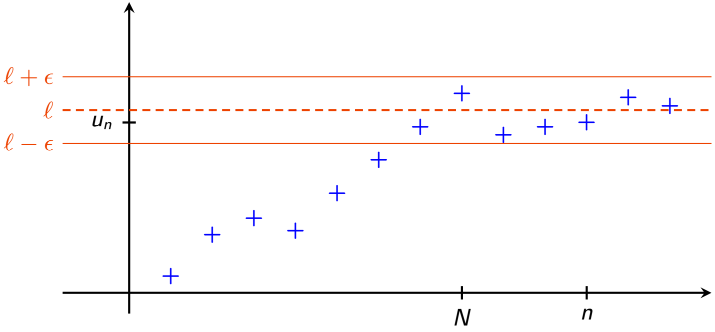
$$\forall \varepsilon > 0 \quad \exists N \in \mathbb{N} \quad \forall n \in \mathbb{N} \qquad (n \geq N \implies |u_n - \ell| \leq \varepsilon)$$



Limite

La suite $(u_n)_{n\in\mathbb{N}}$ a pour ***limite*** $\ell \in \mathbb{R}$ si : pour tout $\varepsilon > 0$, il existe un entier naturel N tel que si $n \geq N$ alors $|u_n - \ell| \leq \varepsilon$:

$$\forall \varepsilon > 0 \quad \exists N \in \mathbb{N} \quad \forall n \in \mathbb{N} \qquad (n \geq N \implies |u_n - \ell| \leq \varepsilon)$$



On dit aussi que la suite $(u_n)_{n\in\mathbb{N}}$ ***tend vers*** ℓ. Autrement dit : u_n est proche d'aussi près que l'on veut de ℓ, à partir d'un certain rang.

Limite

1. La suite $(u_n)_{n\in\mathbb{N}}$ ***tend vers*** $+\infty$ si :

$$\forall A > 0 \quad \exists N \in \mathbb{N} \quad \forall n \in \mathbb{N} \qquad (n \geq N \implies u_n \geq A)$$

2. La suite $(u_n)_{n\in\mathbb{N}}$ ***tend vers*** $-\infty$ si :

$$\forall A > 0 \quad \exists N \in \mathbb{N} \quad \forall n \in \mathbb{N} \qquad (n \geq N \implies u_n \leq -A)$$

Limite

1. La suite $(u_n)_{n\in\mathbb{N}}$ ***tend vers*** $+\infty$ si :

$$\forall A>0 \quad \exists N\in\mathbb{N} \quad \forall n\in\mathbb{N} \qquad (n\geq N \implies u_n\geq A)$$

2. La suite $(u_n)_{n\in\mathbb{N}}$ ***tend vers*** $-\infty$ si :

$$\forall A>0 \quad \exists N\in\mathbb{N} \quad \forall n\in\mathbb{N} \qquad (n\geq N \implies u_n\leq -A)$$

On note $\lim_{n\to+\infty} u_n=\ell$ ou parfois $u_n \xrightarrow[n\to+\infty]{} \ell$, et de même pour une limite $\pm\infty$.

Limite

1. La suite $(u_n)_{n\in\mathbb{N}}$ ***tend vers*** $+\infty$ si :

$$\forall A > 0 \quad \exists N \in \mathbb{N} \quad \forall n \in \mathbb{N} \qquad (n \geq N \implies u_n \geq A)$$

2. La suite $(u_n)_{n\in\mathbb{N}}$ ***tend vers*** $-\infty$ si :

$$\forall A > 0 \quad \exists N \in \mathbb{N} \quad \forall n \in \mathbb{N} \qquad (n \geq N \implies u_n \leq -A)$$

On note $\lim_{n\to+\infty} u_n = \ell$ ou parfois $u_n \xrightarrow[n\to+\infty]{} \ell$, et de même pour une limite $\pm\infty$.

Une suite $(u_n)_{n\in\mathbb{N}}$ est ***convergente*** si elle admet une limite ***finie***. Elle est ***divergente*** sinon (c'est-à-dire soit la suite tend vers $\pm\infty$, soit elle n'admet pas de limite).

Limite

Donner un exemple de suite non monotone tendant vers 0. Idem tendant vers 1.

Donner un exemple de suite n'admettant pas de limite. Donner un exemple de suite bornée n'admettant pas de limite

Propriétés des limites

1. $\lim_{n\to+\infty} u_n = \ell \iff \lim_{n\to+\infty}(u_n - \ell) = 0 \iff \lim_{n\to+\infty}|u_n - \ell| = 0$,
2. $\lim_{n\to+\infty} u_n = \ell \implies \lim_{n\to+\infty}|u_n| = |\ell|$.

Propriétés des limites

1. $\lim_{n\to+\infty} u_n = \ell \iff \lim_{n\to+\infty}(u_n-\ell) = 0 \iff \lim_{n\to+\infty}|u_n-\ell| = 0$,
2. $\lim_{n\to+\infty} u_n = \ell \implies \lim_{n\to+\infty}|u_n| = |\ell|$.

Soient $(u_n)_{n\in\mathbb{N}}$ et $(v_n)_{n\in\mathbb{N}}$ deux suites convergentes.

1. Si $\lim_{n\to+\infty} u_n = \ell$, où $\ell \in \mathbb{R}$, alors pour $\lambda \in \mathbb{R}$ on a $\lim_{n\to+\infty} \lambda u_n = \lambda\ell$.
2. Si $\lim_{n\to+\infty} u_n = \ell$ et $\lim_{n\to+\infty} v_n = \ell'$, où $\ell, \ell' \in \mathbb{R}$, alors
$$\lim_{n\to+\infty}(u_n+v_n) = \ell+\ell'$$
$$\lim_{n\to+\infty}(u_n\times v_n) = \ell\times\ell'$$
3. Si $\lim_{n\to+\infty} u_n = \ell$ où $\ell \in \mathbb{R}^* = \mathbb{R}\setminus\{0\}$ alors $u_n \neq 0$ pour n assez grand et $\lim_{n\to+\infty} \frac{1}{u_n} = \frac{1}{\ell}$.

Propriétés des limites

1. $\lim_{n\to+\infty} u_n = \ell \iff \lim_{n\to+\infty}(u_n - \ell) = 0 \iff \lim_{n\to+\infty}|u_n - \ell| = 0$,
2. $\lim_{n\to+\infty} u_n = \ell \implies \lim_{n\to+\infty}|u_n| = |\ell|$.

Soient $(u_n)_{n\in\mathbb{N}}$ et $(v_n)_{n\in\mathbb{N}}$ deux suites convergentes.

1. Si $\lim_{n\to+\infty} u_n = \ell$, où $\ell \in \mathbb{R}$, alors pour $\lambda \in \mathbb{R}$ on a $\lim_{n\to+\infty} \lambda u_n = \lambda\ell$.
2. Si $\lim_{n\to+\infty} u_n = \ell$ et $\lim_{n\to+\infty} v_n = \ell'$, où $\ell, \ell' \in \mathbb{R}$, alors
$$\lim_{n\to+\infty} (u_n + v_n) = \ell + \ell'$$
$$\lim_{n\to+\infty} (u_n \times v_n) = \ell \times \ell'$$
3. Si $\lim_{n\to+\infty} u_n = \ell$ où $\ell \in \mathbb{R}^* = \mathbb{R}\setminus\{0\}$ alors $u_n \neq 0$ pour n assez grand et $\lim_{n\to+\infty} \frac{1}{u_n} = \frac{1}{\ell}$.

✎ Donner un exemple de suite telle que $\lim u_n^2$ existe et soit finie mais $\lim u_n$ n'existe pas.

Théorème des gendarmes (limite par encadrement)

Soient $(u_n)_{n\in\mathbb{N}}$, $(v_n)_{n\in\mathbb{N}}$ et $(w_n)_{n\in\mathbb{N}}$, 3 suites telles que $u_n \leqslant v_n \leqslant w_n$ à partir d'un certain rang N_0

On suppose de plus que $\lim\limits_{n\to+\infty} u_n = \lim\limits_{n\to+\infty} w_n = \ell$

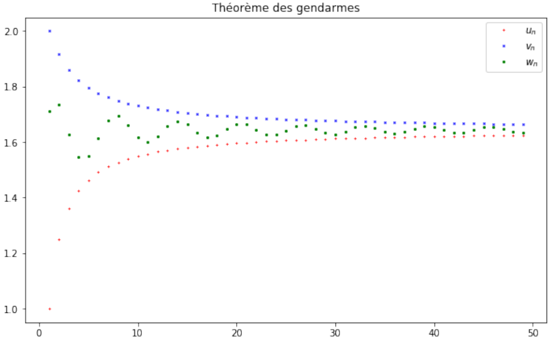
Alors, la suite $(v_n)_{n\in\mathbb{N}}$ converge et $\boxed{\lim\limits_{n\to+\infty} v_n = \ell}$

Théorème des gendarmes (limite par encadrement)

Soient $(u_n)_{n\in\mathbb{N}}$, $(v_n)_{n\in\mathbb{N}}$ et $(w_n)_{n\in\mathbb{N}}$, 3 suites telles que $u_n \leqslant v_n \leqslant w_n$ à partir d'un certain rang N_0

On suppose de plus que $\lim\limits_{n\to+\infty} u_n = \lim\limits_{n\to+\infty} w_n = \ell$

Alors, la suite $(v_n)_{n\in\mathbb{N}}$ converge et $\boxed{\lim\limits_{n\to+\infty} v_n = \ell}$

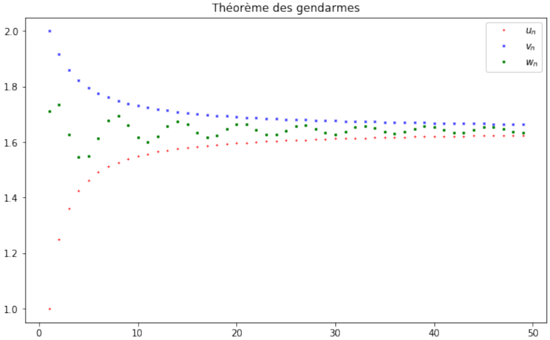


Théorème des gendarmes (limite par encadrement)

Soient $(u_n)_{n\in\mathbb{N}}$, $(v_n)_{n\in\mathbb{N}}$ et $(w_n)_{n\in\mathbb{N}}$, 3 suites telles que $u_n \leqslant v_n \leqslant w_n$ à partir d'un certain rang N_0

On suppose de plus que $\lim\limits_{n\to+\infty} u_n = \lim\limits_{n\to+\infty} w_n = \ell$

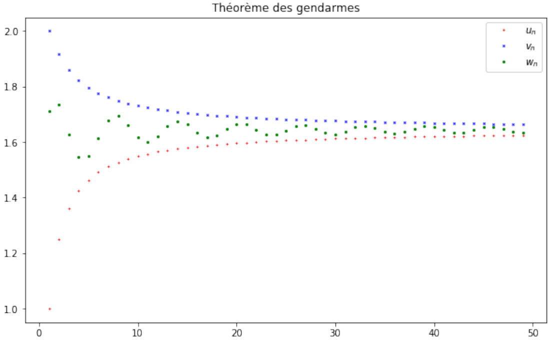
Alors, la suite $(v_n)_{n\in\mathbb{N}}$ converge et $\boxed{\lim\limits_{n\to+\infty} v_n = \ell}$



rmq : rien n'empêche de prendre une suite constante pour u_n ou w_n

Théorème des gendarmes (limite par encadrement)

Soient $(u_n)_{n\in\mathbb{N}}$, $(v_n)_{n\in\mathbb{N}}$ et $(w_n)_{n\in\mathbb{N}}$, 3 suites telles que $u_n \leqslant v_n \leqslant w_n$
à partir d'un certain rang N_0
On suppose de plus que $\lim\limits_{n\to+\infty} u_n = \lim\limits_{n\to+\infty} w_n = \ell$
Alors, la suite $(v_n)_{n\in\mathbb{N}}$ converge et $\boxed{\lim\limits_{n\to+\infty} v_n = \ell}$



rmq : rien n'empêche de prendre une suite constante pour u_n ou w_n

Montrer que la suite $\frac{\sin(\frac{1}{n})}{n}$ est convergente et donner sa limite.

Limites monotones

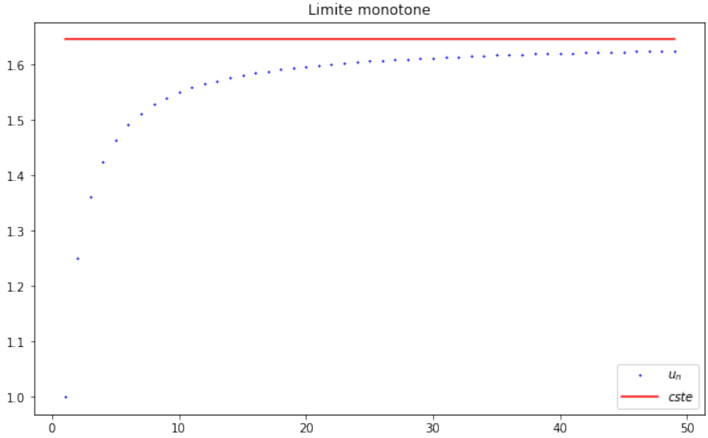
Théorème

Toute suite croissante et majorée est convergente.

Limites monotones

Théorème

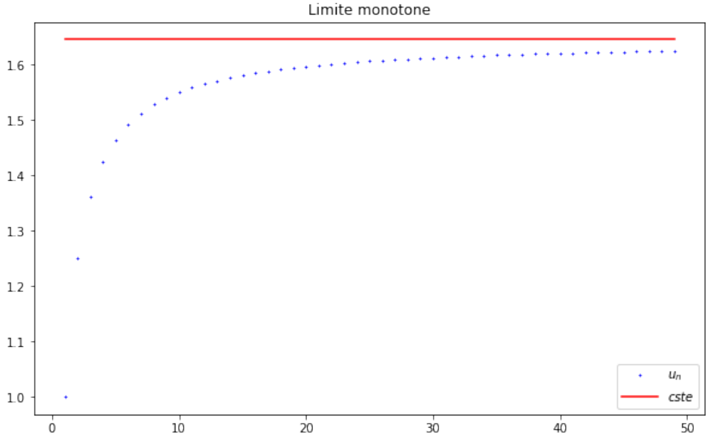
Toute suite croissante et majorée est convergente.



Limites monotones

Théorème

Toute suite croissante et majorée est convergente.



Attention $(u_n)_n$ croissante et $u_n \leq M$ ne signifie pas $u_n \to M$ (mais $\lim u_n \leq M$

Limites monotones

Et aussi :

- Toute suite décroissante et minorée est convergente.

Limites monotones

Et aussi :

- Toute suite décroissante et minorée est convergente.
- Une suite croissante et qui n'est pas majorée tend vers $+\infty$.

Limites monotones

Et aussi :

- Toute suite décroissante et minorée est convergente.
- Une suite croissante et qui n'est pas majorée tend vers $+\infty$.
- Une suite décroissante et qui n'est pas minorée tend vers $-\infty$.

Limites monotones

Et aussi :

- Toute suite décroissante et minorée est convergente.
- Une suite croissante et qui n'est pas majorée tend vers $+\infty$.
- Une suite décroissante et qui n'est pas minorée tend vers $-\infty$.

Limites monotones

Et aussi :

- Toute suite décroissante et minorée est convergente.
- Une suite croissante et qui n'est pas majorée tend vers $+\infty$.
- Une suite décroissante et qui n'est pas minorée tend vers $-\infty$.

Soit $(u_n)_{n\in\mathbb{N}}$ la suite définie par $u_0 = 1$ et pour $n \geq 1$, $u_n = \sqrt{2 + u_{n-1}}$.
Montrer que cette suite est croissante et majorée par 2. Que peut-on en conclure ?

Une suite positive, qui converge vers 0 est-elle décroissante ? *(Que pensez vous de la suite* $\dfrac{2+(-1)^n}{n}$ *?)*

Une suite croissante, négative, converge-t-elle vers 0 ?

Comparaison de suites

En général, on cherche un algorithme le plus efficace possible

Comparaison de suites

En général, on cherche un algorithme le plus efficace possible

- en pire cas ou en moyenne
- sur des entrées de tailles variées

idéalement, on compte exactement le nombre d'opération en fonction de la taille de l'entrée (c'est donc une suite), mais c'est généralement trop compliqué $\rightsquigarrow$ passage à la limite.

Comparaison de suites

En général, on cherche un algorithme le plus efficace possible

- en pire cas ou en moyenne
- sur des entrées de tailles variées

idéalement, on compte exactement le nombre d'opération en fonction de la taille de l'entrée (c'est donc une suite), mais c'est généralement trop compliqué $\rightsquigarrow$ passage à la limite.

- $u_n = o(v_n) \iff \lim_{n\to\infty} \frac{u_n}{v_n} = 0$

Comparaison de suites

En général, on cherche un algorithme le plus efficace possible

- en pire cas ou en moyenne
- sur des entrées de tailles variées

idéalement, on compte exactement le nombre d'opération en fonction de la taille de l'entrée (c'est donc une suite), mais c'est généralement trop compliqué $\rightsquigarrow$ passage à la limite.

- $u_n = o(v_n) \iff \lim_{n\to\infty} \frac{u_n}{v_n} = 0$
- $u_n = \mathcal{O}(v_n) \iff \frac{u_n}{v_n}$ est bornée

Comparaison de suites

En général, on cherche un algorithme le plus efficace possible

- en pire cas ou en moyenne
- sur des entrées de tailles variées

idéalement, on compte exactement le nombre d'opération en fonction de la taille de l'entrée (c'est donc une suite), mais c'est généralement trop compliqué $\rightsquigarrow$ passage à la limite.

- $u_n = o(v_n) \iff \lim_{n\to\infty} \frac{u_n}{v_n} = 0$
- $u_n = \mathcal{O}(v_n) \iff \frac{u_n}{v_n}$ est bornée
- $u_n \sim v_n \iff \lim_{n\to\infty} \frac{u_n}{v_n} = 0$

Comparaison de suites

En général, on cherche un algorithme le plus efficace possible

- en pire cas ou en moyenne
- sur des entrées de tailles variées

idéalement, on compte exactement le nombre d'opération en fonction de la taille de l'entrée (c'est donc une suite), mais c'est généralement trop compliqué $\rightsquigarrow$ passage à la limite.

- $u_n = o(v_n) \iff \lim_{n\to\infty} \frac{u_n}{v_n} = 0$
- $u_n = \mathcal{O}(v_n) \iff \frac{u_n}{v_n}$ est bornée
- $u_n \sim v_n \iff \lim_{n\to\infty} \frac{u_n}{v_n} = 0$

Comparaison de suites

En général, on cherche un algorithme le plus efficace possible

- en pire cas ou en moyenne
- sur des entrées de tailles variées

idéalement, on compte exactement le nombre d'opération en fonction de la taille de l'entrée (c'est donc une suite), mais c'est généralement trop compliqué $\rightsquigarrow$ passage à la limite.

- $u_n = o(v_n) \iff \lim_{n\to\infty} \frac{u_n}{v_n} = 0$
- $u_n = \mathcal{O}(v_n) \iff \frac{u_n}{v_n}$ est bornée
- $u_n \sim v_n \iff \lim_{n\to\infty} \frac{u_n}{v_n} = 0$

En informatique on utilise principalement le $\mathcal{O}$, plus facile à manipuler que les autres

Simplification en pratique I

En cas de forme indéterminée $\frac{\infty}{\infty}$ dans un calcul de limite de fonction, on se rapporte très souvent au tableau suivant :

$n \to +\infty$	log	poly.	exp
log	$\frac{\text{log}}{\text{log}} \rightsquigarrow$ factor.	$\frac{\text{poly}}{\text{log}} \to \infty$	$\frac{\text{exp}}{\text{log}} \to \infty$
poly	$\frac{\text{log}}{\text{poly}} \to 0$	$\frac{\text{poly}}{\text{poly}} \rightsquigarrow$ factor.	$\frac{\text{exp}}{\text{poly}} \to \infty$
exp	$\frac{\text{log}}{\text{exp}} \to 0$	$\frac{\text{poly}}{\text{exp}} \to 0$	$\frac{\text{exp}}{\text{exp}} \rightsquigarrow$ factor.

Par exemple, pour $\lim_{n\to+\infty} \frac{-2n^3}{3^n}$ est de la forme $\frac{\text{poly}}{\text{exp}}$, et tend donc vers l'infini. Ici $\lim_{n\to+\infty} \frac{-2n^3}{3^n} = -\infty$ à cause du coefficient -2.

Simplification en pratique II

Les cases diagonales du tableau ne nous permettent pas de conclure directement, mais on peut trouver la réponse en *factorisant* par le terme dominant.

polynôme sur polynôme : On factorise par le terme de plus grande puissance.

$\lim_{n\to+\infty} \frac{-2n^3+n}{3n^2+1}$:

Simplification en pratique II

Les cases diagonales du tableau ne nous permettent pas de conclure directement, mais on peut trouver la réponse en *factorisant* par le terme dominant.

polynôme sur polynôme : On factorise par le terme de plus grande puissance.

$\lim_{n\to+\infty} \frac{-2n^3+n}{3n^2+1}$:

$$\begin{aligned}
\lim_{n\to+\infty} \frac{-2n^3+n}{3n^2+1} &= \lim_{n\to+\infty} \frac{n^3}{n^2}\left(\frac{-2+\frac{1}{n^2}}{3+\frac{1}{n^2}}\right) \\
&= \lim_{n\to+\infty} \frac{n}{1}\left(\frac{\overbrace{-2+\frac{1}{n^2}}^{\to -2}}{\underbrace{3+\frac{1}{n^2}}_{\to 3}}\right) \\
&= \lim_{n\to+\infty} n\frac{-2}{3} \\
&= -\infty
\end{aligned}$$

$\lim_{n\to+\infty} \frac{n^5-2n^3+n}{7n^5+3n^2+1}$

$\lim_{n\to+\infty} \frac{n^5-2n^3+n}{7n^5+3n^2+1}$

$$\lim_{n\to+\infty} \frac{n^5-2n^3+n}{7n^5+3n^2+1} = \lim_{n\to+\infty} \frac{n^5}{n^5}\left(\frac{1-\frac{2}{n^2}+\frac{1}{n^4}}{7+\frac{3}{n^3}+\frac{1}{n^5}}\right)$$

$$= \lim_{n\to+\infty} 1\cdot\frac{1}{7}$$

$$= \frac{1}{7}$$

$\lim_{n\to+\infty} \frac{2n^3+n}{7n^5+3n^2+1}$:

$$\lim_{n\to+\infty} \frac{2n^3+n}{7n^5+3n^2+1} = \lim_{n\to+\infty} \frac{n^3}{n^5}\left(\frac{2+\frac{1}{n^2}}{7+\frac{3}{n^3}+\frac{1}{n^5}}\right)$$

$$= \lim_{n\to+\infty} \frac{2}{7}\frac{1}{n^2}$$

$$= 0$$

Application en informatique : boucles `for`

Lemme

$$\sum_{k\geq 1}^{n} k^{\alpha}(\ln k)^{\beta} = \mathcal{O}(n^{\alpha+1}(\ln n)^{\beta})$$

Data: n

for $1 \leq i \leq n$ **do**
  for $i \leq j \leq n$ **do**
    print(i)

Zoologie

$\mathcal{O}(1)$	accès variable
$\mathcal{O}(\ln n)$	recherche dichotomique
$\mathcal{O}(n)$	recherche naïve, somme des éléments d'un tableau
$\mathcal{O}(n \ln n)$	tri d'un tableau (optimal)
$\mathcal{O}(n^2)$	tri naïf
$\mathcal{O}(n^3)$	produit matriciel (naïf)
$\mathcal{O}(2^n)$	sudoku (backtracking)
$\mathcal{O}(n!)$	sudoku (brute force)

`https://www.irif.fr/~sperifel/complexite.pdf`

Zoologie

Attention : la complexité dépend de l'algorithme/implémentation, pas du problème !

Zoologie

Attention : la complexité dépend de l'algorithme/implémentation, pas du problème !

calcul du n-ième nombre de Fibonacci :

réc$^{\text{ursif}}$ naïf	réc$^{\text{ursif}}$ terminal	itér$^{\text{ratif}}$ classique	matric$^{\text{iel}}$ naïf	matric$^{\text{iel}}$ (exp. rapide)
$\mathcal{O}(2^n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(\ln n)$

Zoologie

Attention : la complexité dépend de l'algorithme/implémentation, pas du problème !

calcul du n-ième nombre de Fibonacci :

réc$^{\text{ursif}}$ naïf	réc$^{\text{ursif}}$ terminal	itér$^{\text{ratif}}$ classique	matric$^{\text{iel}}$ naïf	matric$^{\text{iel}}$ (exp. rapide)
$\mathcal{O}(2^n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(\ln n)$

Attention : la notation $\mathcal{O}$ peut cacher des choses $\rightsquigarrow$ algorithmes galactiques

Zoologie

Attention : la complexité dépend de l'algorithme/implémentation, pas du problème !

calcul du n-ième nombre de Fibonacci :

réc$^{\text{ursif}}$ naïf	réc$^{\text{ursif}}$ terminal	itér$^{\text{ratif}}$ classique	matric$^{\text{iel}}$ naïf	matric$^{\text{iel}}$ (exp. rapide)
$\mathcal{O}(2^n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(\ln n)$

Attention : la notation $\mathcal{O}$ peut cacher des choses $\rightsquigarrow$ algorithmes galactiques

Multiplication matricielle :

Zoologie

Attention : la complexité dépend de l'algorithme/implémentation, pas du problème !

calcul du n-ième nombre de Fibonacci :

réc$^{\text{ursif}}$ naïf	réc$^{\text{ursif}}$ terminal	itér$^{\text{ratif}}$ classique	matric$^{\text{iel}}$ naïf	matric$^{\text{iel}}$ (exp. rapide)
$\mathcal{O}(2^n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(\ln n)$

Attention : la notation $\mathcal{O}$ peut cacher des choses $\rightsquigarrow$ algorithmes galactiques

Multiplication matricielle :

naïf	alg. de Strassen	alg. de Coppersmith–Winograd	Duan, Wu, Zhou (2022)
$\mathcal{O}(n^3)$	$\mathcal{O}(n^{2.807})$	$\mathcal{O}(n^{2.376})$	$\mathcal{O}(n^{2.37188})$

Complexité : récursivité

Pour le récursif, la complexité est aussi def. par recurrence

ex :

Algorithm 1: Dichotomie

```
Function recherche(T, x, d, f) :
    if f < d then
        return −1
    else
        m = ⌊(b+a)/2⌋
        if T[m] = x then
            return m
        else if T[m] < x then
            return recherche(T,x,m+1,f)
        else
            return recherche(T,x,m-1,m)
```

Complexité : récursivité

Master theorem

Posons $T(n)$ la suite/fonction définie par la récurrence :

$T(n) = a\ T\left(\frac{n}{b}\right) + O(n^d)$

alors :

- Si $d < \log_b a$ alors $T(n) = O(n^{\log_b a})$

Complexité : récursivité

Master theorem

Posons $T(n)$ la suite/fonction définie par la récurrence :

$T(n) = a\ T\left(\frac{n}{b}\right) + O(n^d)$

alors :

- Si $d < \log_b a$ alors $T(n) = O(n^{\log_b a})$
- Si $d = \log_b a$ alors $T(n) = O(n^d \log n)$

Complexité : récursivité

Master theorem

Posons $T(n)$ la suite/fonction définie par la récurrence :

$T(n) = a\ T\left(\frac{n}{b}\right) + O(n^d)$

alors :

- Si $d < \log_b a$ alors $T(n) = O(n^{\log_b a})$
- Si $d = \log_b a$ alors $T(n) = O(n^d \log n)$
- Si $d > \log_b a$ alors $T(n) = O(n^d)$.

Complexité : récursivité

Master theorem

Posons $T(n)$ la suite/fonction définie par la récurrence :

$T(n) = a\ T\left(\frac{n}{b}\right) + O(n^d)$

alors :

- Si $d < \log_b a$ alors $T(n) = O(n^{\log_b a})$
- Si $d = \log_b a$ alors $T(n) = O(n^d \log n)$
- Si $d > \log_b a$ alors $T(n) = O(n^d)$.

Complexité : récursivité

Master theorem

Posons $T(n)$ la suite/fonction définie par la récurrence :

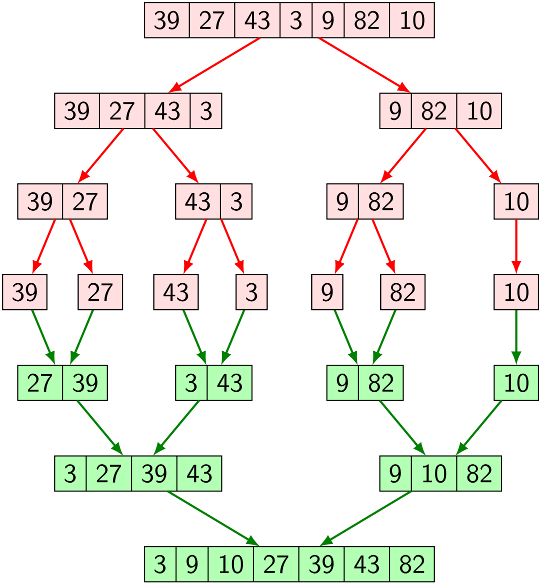
$T(n) = a\ T\left(\frac{n}{b}\right) + O(n^d)$

alors :

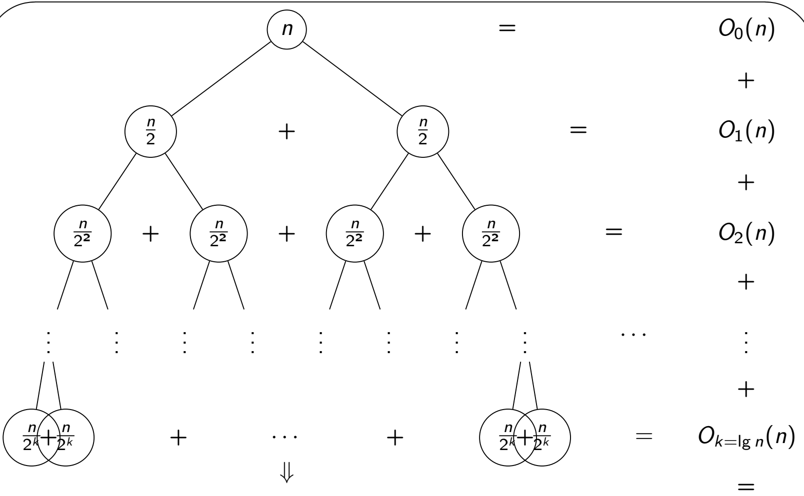
- Si $d < \log_b a$ alors $T(n) = O(n^{\log_b a})$
- Si $d = \log_b a$ alors $T(n) = O(n^d \log n)$
- Si $d > \log_b a$ alors $T(n) = O(n^d)$.

Donner la complexité du tri fusion par cette méthode

Tri-fusion


39 27 43 3 9 82 10
39 27 43 3
9 82 10
39 27
43 3
9 82
10
39
27
43
3
9
82
10
27 39
3 43
9 82
10
3 27 39 43
9 10 82
3 9 10 27 39 43 82

Tri-fusion



$$O\left(\sum_{i=0}^{k} 2^i \cdot \frac{n}{2^i}\right) = O\left(\sum_{i=0}^{k} n\right) = O(k \cdot n) \Leftrightarrow O(n \cdot \lg n)$$