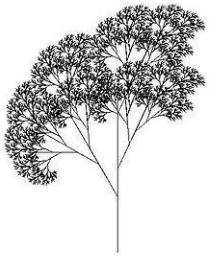
Récursivité

I. Introduction



En mathématiques, en informatique, en biologie, mais aussi dans notre quotidien, un problème doit souvent être résolu en utilisant une méthode de résolution qui est répétée plusieurs fois.
Dans l'**itération**, cette méthode est appliquée par paliers de façon séquentielle.
Dans la **récursion**, la méthode s'appelle elle-même.
La récursion est si fondamentale qu'il n'est pas possible de l'éviter : l'auto-reproduction, qui constitue le fondement de toute vie, est un processus récursif.

II. Suites numériques

1) Définition explicite

Une suite numérique peut dans certains cas être définie de manière explicite : $u_n = f(n)$.
La détermination du nième terme est alors aisée. Il suffit d'évaluer $f(n)$.

Exemple 1 : Puissances de 2
Problème : évaluer le nombre 2 à la puissance n de manière explicite, $n \in \mathbb{N}$.
On définit de manière explicite la suite : $u_n = 2^n$.

Algorithme 1 Puissances de 2, méthode explicite
Données : n : un nombre entier
Résultat : un entier égal à la nième puissance de 2

Malheureusement, toutes les suites numériques ne peuvent pas être définies de manière explicite…

2) Relation de récurrence

Dans une suite définie de manière récurrente, il est possible de calculer le terme u_n de la suite en connaissant les termes précédents.
Les égalités de la forme $u_n = f(u_{n-1})$, $u_n = f(u_{n-1}, u_{n-2})$, etc. s'appellent des ….
Les égalités qui définissent les premiers termes d'une suite sont appelées des conditions de départ.
Une suite recursive est donc définie par une **relation de récurrence** et une(des) **condition**(s) **de départ**.

On reprend l'exemple 1 : puissances de 2.
On définit de manière récursive la suite :

On propose de manière proche de la définition, un **algorithme récursif**.

Algorithme 2 : Puissances de 2, méthode récursive

Données : n : un nombre entier
Résultat : un entier égal à la nième puissance de 2

Il est aussi possible de proposer un **algorithme itératif** permettant d'aboutir au même résultat :

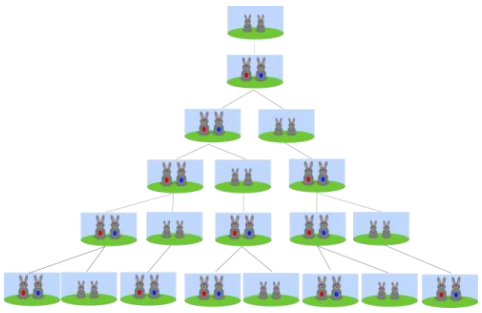
Algorithme 3 Puissances de 2, méthode itérative
Données : n : un nombre entier
Résultat : un entier égal à la nième puissance de 2

Exemple 2 : Suite de Fibonacci.
Problème : évaluer le nième terme.
On définit de manière récursive la suite :

$$fib(0)=0,\ fib(1)=1$$
$$fib(n)=fib(n-1)+fib(n-2)$$

Nota : il existe aussi une forme explicite

$$fib(n)=\frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^n-\left(\frac{1-\sqrt{5}}{2}\right)^n\right]$$

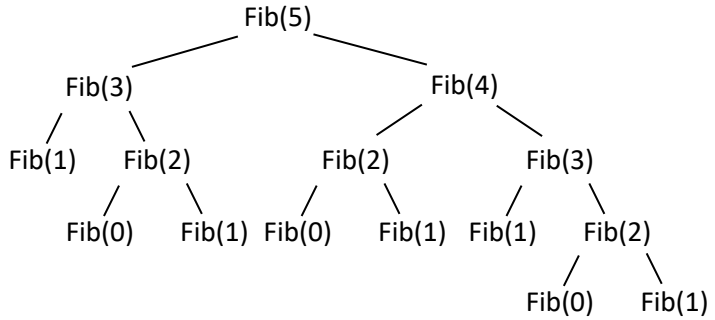


Croissance des lapins selon la suite de Fibonacci

On propose facilement un **algorithme récursif** :

Algorithme 4 Suite de Fibonacci, méthode récursive
Données : n : un nombre entier
Résultat : un entier égal au nième terme de la suite de Fibonacci

Représentons l'arbre des appels de la fonction « Fib » pour n égal à 5 :



Remarque :

Attention donc au fait qu'un algorithme facile à écrire n'est pas forcément le plus performant !!!

Algorithme 5 Suite de Fibonacci, méthode itérative

Données : n : un nombre entier
Résultat : un entier égal au nième terme de la suite de Fibonacci

Nota : *on montre que* $\lim_{n\to\infty} \frac{fib(n)}{fib(n-1)} = \frac{1+\sqrt{5}}{2} \approx 1{,}618$ *, le nombre d'or.*

III. Diviser pour régner

Exemple 3 : Téléphone en chaîne
Problème : Les 15 joueuses d'une équipe de volleyball ont la liste des joueuses de l'équipe avec leur numéro de téléphone. La capitaine reçoit l'information que le prochain match a été déplacé.
Il faut prévenir toutes les autres joueuses.

La première solution serait que la capitaine se charge d'appeler toutes les autres joueuses. Si elle passe 5 minutes au téléphone avec chacune d'entre-elles, elle en a pour plus d'une heure.
Un solution plus efficace et plus confortable pour la capitaine est qu'elle divise la liste de joueuses en deux moitiés. Elle appelle alors la première joueuse de chacune des deux listes obtenues. Elle leur donne l'information de report de match et leur demande à leur tour de faire la même chose : diviser en deux la demi-liste à laquelle elles appartiennent, appeler la première joueuse de chacune des parties et ainsi de suite … jusqu'à ce qu'il n'y ait plus personne à prévenir.

On peut modéliser la résolution du problème posé par l'algorithme récursif suivant :

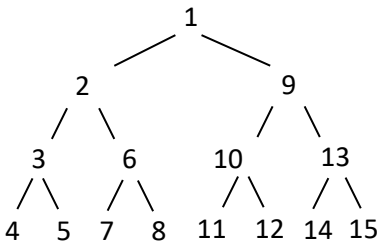
Algorithme 6 Téléphone en chaîne, méthode récursive

Données : L : la liste de joueuses de l'équipe

Le principe du « diviser pour régner », est toujours le suivant :

- le problème principal est **divisé en petits problèmes** (généralement deux),
- chaque petit problème est identique au problème principal. Il peut donc être résolu de la même façon.
- **ce procédé est répété** jusqu'à ce que le petit problème soit si simple qu'il puisse être résolu directement. Il faut donc toujours une **condition d'arrêt** des appels récursifs, sans quoi l'algorithme ne se terminerait jamais.

Représentons l'arbre des appels pour une liste de 15 joueuses numérotées de 1 à 15 :



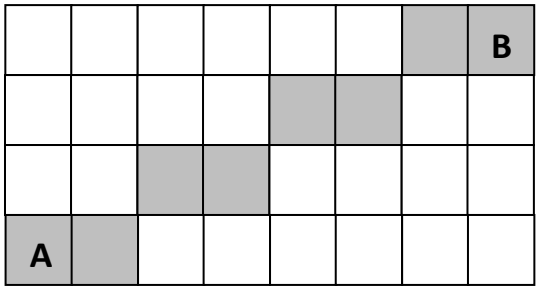
Chacune des joueuses ne passe qu'au plus deux appels téléphoniques (ici exactement deux).
Si chacun d'entre-eux dure 5 minutes, alors le problème de départ est résolu en ….
Ce temps est à comparer au temps d'1 heure et 10 minutes pour la méthode « naïve » qui consiste à effectuer 14 appels successifs.

Exemple 4 : Affichage d'une ligne

Problème : Un écran est constitué de pixels. Chacun d'entre-eux est coloré afin de représenter une image, un dessin, une photo, etc. Comment tracer une ligne sur un écran à l'aide de pixels ?

Il est possible d'utiliser une méthode récursive pour tracer une ligne d'épaisseur « 2 » entre les pixels « A » et « B » :

```
Si les pixels A et B sont voisins alors
    Marquer ces deux points
Sinon
    Chercher le milieu M de la ligne
    Tracer la ligne(A,M)
    Tracer la ligne(M,B)
```



On aboutit alors à l'algorithme suivant :

Algorithme 7 Tracer d'une ligne, méthode récursive

Données : A : pixel de départ, de coordonnées A[1],A[2]
B : pixel d'arrivée, de coordonnées B[1],B[2]

NOTA : On pourra s'intéresser à l'algorithme de Bresenham qui permet de tracer de manière plus efficace une ligne entre deux pixels.

IV. Courbes définies récursivement

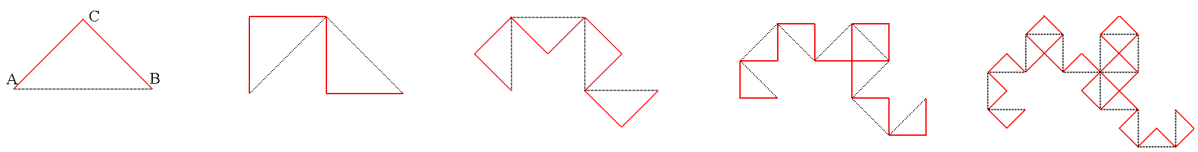
Exemple 5 : Courbe du dragon
La courbe du dragon (ou « fractale du dragon » ou « courbe de Heighway ») a été pour la première fois étudiée par les physiciens de la NASA John Heighway, Bruce Banks, et William Harter en 1960.
On peut l'obtenir par pliage en utilisant la méthode suivante :

Prenez une bande de papier
Répéter
    Pliez-la en son milieu par la droite
Jusqu'à fin
Dépliez la bande en conservant les pliures à 90°

Problème : Comment tracer une courbe du dragon (ou fractale du dragon) à l'aide d'un algorithme récursif ?

La méthode itérative est la suivante :

Partir d'un segment de base
Répéter
    En suivant la courbe, remplacer chaque segment par deux segments à angle droit
Jusqu'à fin



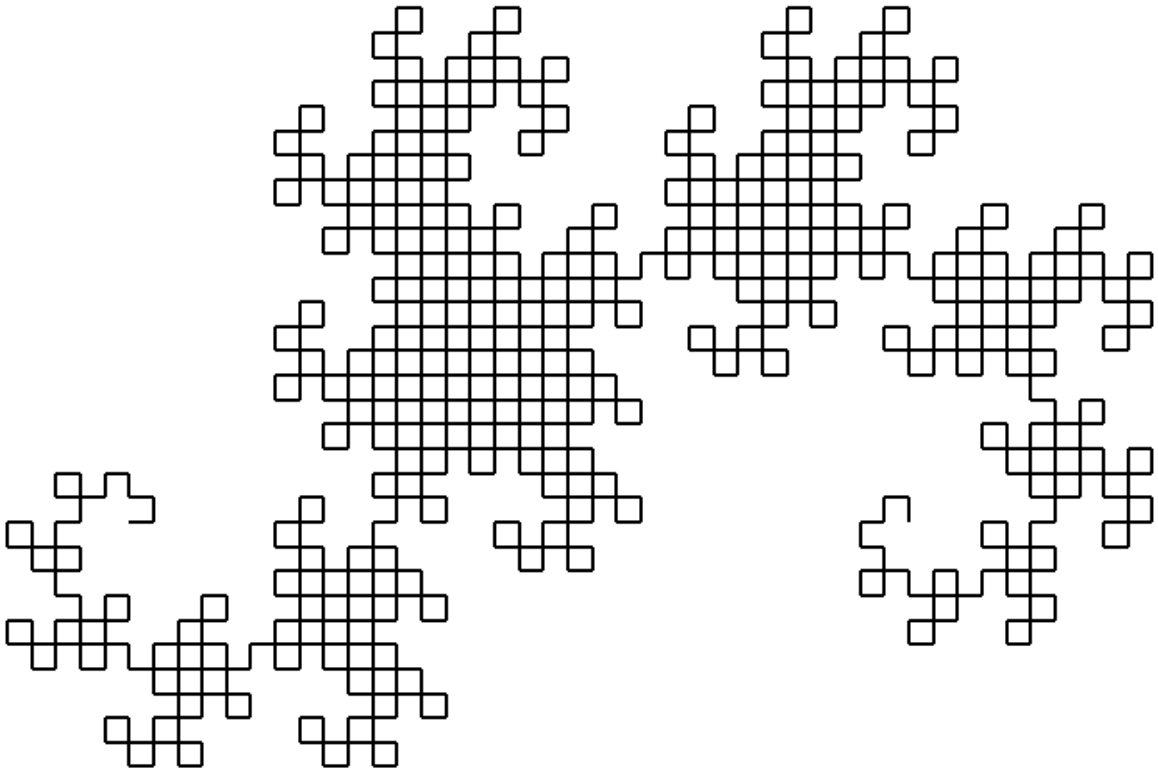
Itération 1 Itération 2 Itération 3 Itération 4 Itération 5

L'algorithme récursif est le suivant :

Algorithme 8 Courbe du dragon, méthode récursive

Données : n : niveau de récursivité
s : sens de rotation (“1” : + 90°, “-1” : - 90°)

NOTA : Dans cet algorithme, on ne précise pas les paramètres nécessaires au bon fonctionnement de la fonction « Tracer_segment » : point de départ, longueur de segment, orientation.
Par ailleurs, le nombre d’appels à cette fonction devient rapidement important avec le niveau de récursivité qui augmente (2^{12} pour un niveau n=12). En outre, certains segments sont tracés plusieurs fois. Cet algorithme récursif n’est donc pas très efficace.



Courbe du dragon avec 10 niveaux de récursivité