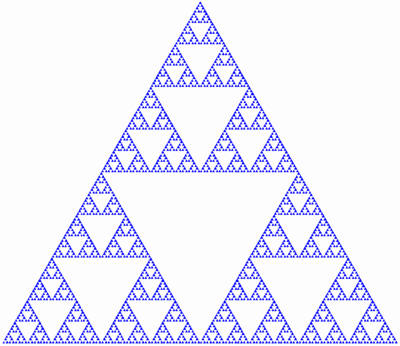
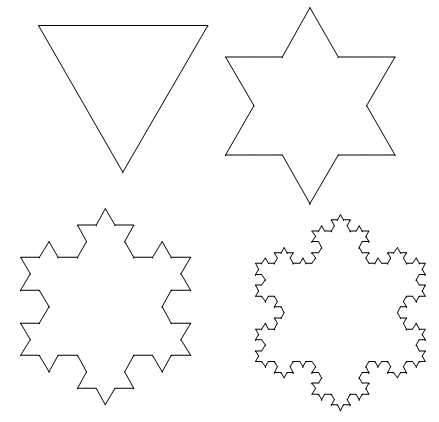
Récursivité



# Exercices divers

## Algorithme d’Euclide

L’algorithme d’Euclide permet, étant donnés deux entiers *a* et *b*, de calculer leur plus grand commun diviseur (pgcd) *d*. Cet algorithme se base sur la propriété suivante :

où *a* % *b* représente le reste de la division euclidienne de *a* par *b*.

1. Ecrire une fonction récursive pgcd(a,b)qui calcule le plus grand commun diviseur de deux entiers en utilisant l’algorithme d’Euclide.

Le théorème de Bézout nous assure également l’existence de deux entiers *u* et *v* tels que :

*a.u* + *b.v* = *d* (*u* et *v* sont des coefficients de Bézout de *a* et *b*).

Une version étendue de l’algorithme d’Euclide permet de calculer, en plus du pgcd *d* des valeurs possibles pour les coefficients de Bézout *u* et *v*.

Cet algorithme prend en entrée deux entiers *a* et *b*. Il procède de la manière suivante :

* Si *b* = 0 alors *d* = *a*, *u* = 1 et *v* = 0.
* Sinon, on applique récursivement l’algorithme sur les entiers *b* et (*a* % *b*).

On obtient ainsi *d*’, *u*’ et *v*’ tels que :

*d*’ = pgcd(*b*, *a* % b) ; et : *b.u*’ + (*a* % *b*).*v’* = *d’*

On en déduit la solution pour *a* et b grâce aux égalités :

*d* = *d’*, *u* = *v’* et *v* = *u*’ – (*a*//*b*).*v’*

1. En déduire une fonction bezoutqui étant donnés deux entiers *a* et *b* calcule le triplet (*d*,*u*,*v*) comme expliqué ci-dessus.

## PGCD Rapide

On peut écrire un PGCD très rapide, uniquement à l’aide de soustractions et de divisions par deux ou de restes modulo 2 (très rapides). L’algorithme se base sur la parité des nombres : par exemple, si *a* est pair et *b* impair, pgcd(*a*, *b*) = pgcd(*a*/2, *b*). Autre exemple : si *a* et *b* sont impairs ; pgcd(a,b)=pgcd(M-m,m) avec m=min(a,b) et M=max(a,b)

En se basant sur ce type de constatations, et en traitant tous les cas possibles, on peut à chaque fois réduire le problème.

1. Ecrire la fonction qui calcule le pgcd rapide.

## Déterminant

On souhaite calculer le déterminant d’une matrice (carrée) *A* par la formule de développement sur la première ligne (ou sur la première colonne) mais pas une autre (par exemple si on le faisait sur la troisième, la fonction ne conviendrait pas pour un déterminant 2x2).

1. Ecrire une fonction extraire(A,lig,col)qui permet de fabriquer la matrice extraite de *A* en rayant la ligne *lig* et la colonne *col* (dans l’optique d’obtenir son déterminant).
2. Ecrire une fonction récursive determinant(A)retournant le déterminant de *A*.
3. Tester l’algorithme sur une matrice (de taille au moins 3,3) et comparer le résultat à ce que retourne la fonction det du module *linalg* de la bibliothèque *numpy*.

## Récursivité mutuelle

1. Implémenter les suites définies par : *un*+1 = *un*/*vn* ; et : *vn*+1 = *vn* + *un*, avec *u*0= 1 et *v*0 = 1.

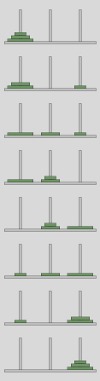
# Tours de Hanoï

Le problème des tours de Hanoï est un problème mathématique célèbre inventé par le mathématicien français Edouard Lucas 1883 qu'il devrait à un de ses amis, N. Claus de Siam. C'est également un jeu de réflexion intéressant.

## Introduction et formalisation du problème

Le problème des tours de Hanoï est un jeu faisant parti de la catégorie des casse-tête. Il est très utilisé en psychiatrie, étudié en mathématiques et en algorithmique et souvent utilisé pour montrer la puissance et l'intérêt de la récursivité.

Le problème, qui peut devenir très vite complexe, peut être décrit très simplement. On dispose de trois « tours » A, B et C formées d'un empilement plus ou moins grand de disques de telle sorte que chaque disque ait un diamètre inférieur à son prédécesseur. Ainsi, le plus grand disque de chaque tour se situe à leur base et le plus petit sur leur sommet. Au départ, tous les disques (supposons qu'il y en ait n) se trouvent empilés suivant les règles précédemment décrites sur la tour A. L'objectif est de déplacer tous les disques de la tour A vers la tour C en s'aidant uniquement de la tour B, tout en respectant les règles suivantes :

* On ne peut bouger qu'un seul disque par étape et cela doit toujours être le plus petit (celui qui se trouve au sommet) ;

A B C

* À chaque étape, la règle d'organisation des tours décrite précédemment doit être respectée.

L’illustration ci-contre montre à chaque tour de jeu l’évolution des différents disques pour un problème à 3 disques…

## Solution récursive

La résolution du problème des tours de Hanoï est très étudiée en algorithmique où elle sert notamment à montrer que l'utilisation de la récursivité pour résoudre de gros problèmes peut produire des codes à la fois logiques, puissants et concis.

Pour résoudre le problème des tours de Hanoï, il faut procéder logiquement. La résolution des tours de Hanoï avec deux disques permet de trouver la logique qui va nous permettre de résoudre des tours de Hanoï avec un nombre supérieur de disques.

1. En partant d’un raisonnement sur deux disques, trouver l’algorithme de résolution des tours de Hanoï sur un nombre *N* disques.

Nous allons maintenant programmer une fonction récursive sous Python. Cette fonction prendra en arguments d’entrée 3 chaînes de caractère symbolisant chacune des tours A, B et C et une liste constituée de *N* chaînes de caractère, représentant pour chacune d’elle un disque.

On souhaite, de plus, que lorsque l’on exécute cette fonction, elle affiche les différentes étapes permettant de résoudre le problème des tours de Hanoï.

Voici un exemple lorsque l’on a 3 disques :

>>> hanoi("A","B","C",["\_","\_\_","\_\_\_"])

\_ de A vers C

\_\_ de A vers B

\_ de C vers B

\_\_\_ de A vers C

\_ de B vers A

\_\_ de B vers C

\_ de A vers C

1. Ecrire sous Python une telle fonction hanoi et la tester.