TD 12 Récursivité

Lycée Louis Thuillier - Physique-Chimie - PCSIB

Notion de récursivité

Principe de la récursivité

La récursivité est un moyen de répéter un bloc d'instruction sans utiliser une boucle *while* ou *for*.

Propriété. Récursivité
Une fonction récursive est une fonction qui, pour s'exécuter, va s'appeler elle même.

> ***Exemple 1 :*** *On note $S(n)$ la somme des entiers entre 1 et n. On peut définir $S(n)$ par :*
> ▷ $S(n) = n + S(n-1)$ *pour* $n \neq 0$
> ▷ *et* $S(n=0) = 0$

> ***Application 1 :*** *Coder en Python la fonction S par récursivité.*
> *Si vous trouvez que votre code est bizarre, c'est normal !*

Attention ! Toute fonction récursive doit présenter
▷ l'appel récursif : on appelle la fonction au sein de sa propre définition
▷ condition d'arrêt : on fixe la valeur de la fonction en un point

> ***Application 2 :*** *Ecrire une fonction récursive factorielle qui prend en argument un entier naturel et renvoie la factorielle de cet entier.*

Preuve de terminaison

Propriété.
Pour se terminer, un algorithme récursif doit toujours revenir à la condition d'arrêt.

Comme pour les autres algorithme, pour prouver la terminaison d'un algorithme récursif on construira une suite d'entier strictement décroissante.

> ***Exemple 2 :*** *Pour la fonction factorielle, on étudiera (u_p), la suite des arguments de la fonction factorielle appelée.*
> *A chaque itération $u_{p+1} = u_p - 1$: la suite est strictement décroissante. Elle ne prend alors qu'un nombre fini de valeur : l'algorithme se termine.*

Applications de la récursivité

Recherche d'élément

▶ Recherche d'un minimum

1. En remarquant que :

$$Min([a_0, a_1, ..., a_n]) = Min(Min([a_0, a_1, ..., a_{n-1}]), a_n)$$

écrire une fonction récursive de paramètre une liste L et qui renvoie le minimum de cette liste L.

2. Effectuer une preuve de terminaison de cet algorithme.
On cherchera pour cela une suite d'entier qui décroit à chaque itération de la récursivité.

3. On note C_n le nombre d'opérations élémentaires effectuées pour une liste L à n.
   ▷ Donner le lien entre C_n et C_{n-1}.
   ▷ En déduire la complexité de l'algorithme récursif.

▶ Recherche dichotomique

Soit une liste $[a_0, a_1, ..., a_n]$ de nombres. On cherche le minimum de la liste. On a vu qu'une bonne façon de procéder est de chercher par dichotomie.

1. Rappeler les conditions sur la liste $[a_0, a_1, ..., a_n]$ pour que la recherche par dichotomie fonctionne.

2. Ecrire deux fonctions *recherche* qui prennent en argument une liste L et une élément e et qui renvoie $True$ si e se trouve dans la liste et $False$ sinon.
Les deux fonctions doivent fonctionner par recherche dichotomique, la première fonction ne devra pas être récursive, la seconde oui.

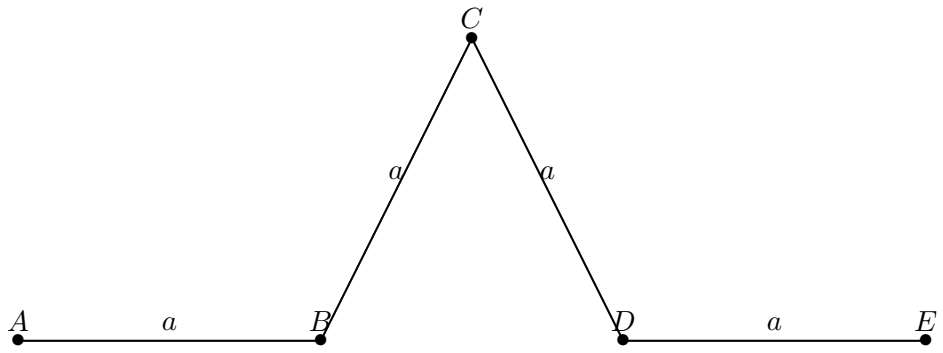
3. Donner la complexité $C(n)$ de l'algorithme de recherche dichotomique récursif. On pourra essayer de faire lien entre $C(n)$ et $C(n/2)$.

Fractal et flocon de Koch

Le mode *turtle* permet de dessiner des lignes brisées. On contrôle pour cela une petite tortue que l'on peut déplacer via
▷ l'instruction $turtle.forward(d)$ permet de la faire avancer en ligne droite sur une longueur d.
On trace ainsi un segment de longueur d dans la direction de la tortue

▷ l'instruction $turtle.left(b)$ permet de tourner la tortue d'un angle b (en degrés !!) vers sa gauche
▷ l'instruction $turtle.left(b)$ permet de tourner la tortue d'un angle b (en degrés !!) vers sa droite
▷ l'instruction $turtle.mainloop()$ permet de terminer le tracé

1. Écrire une fonction $segment_Koch$ qui génèrée le tracé suivant.



Cette ligne brisée représente l'étape 1 d'une ligne de coche de côté initial $3a$.
Le passage de l'étape k à l'étape $k+1$ consiste à remplacer chaque segment de la ligne brisée de l'étape k par la ligne brisée $segment_Koch(a_k/3)$ décrite précédemment sans changer les extrémité du segment, où a_k désigne la longueur d'un segment de la ligne brisée à l'étape k.

2. Construire à la main et schématiquement l'étape 2.
3. Proposer un schéma de la construction de la ligne de Koch d'ordre n.
4. Écrire une fonction récursive $ligne_Koch(a, n)$ qui trace la ligne de Koch de côté initial a et à l'étape n.
5. Pour générer un flocon de Koch, créer un triangle équilatéral dont chaque côté est un segment de Koch d'ordre n.