

## La factorielle

1. Programmer une fonction récursive `factorielle(n)` non terminale qui prend en argument un entier naturel  $n$  et qui renvoie la factorielle de  $n$ . Evaluer la complexité de cette fonction.
2. Modifier la fonction précédente pour qu'elle devienne une fonction récursive terminale. Evaluer la complexité de cette dernière fonction et commenter le résultat.
3. **Intérêt de la dichotomie.**

Le 31 décembre 2009, Fabrice BELLARD publie un nouveau record avec presque 2 700 milliards de décimales du nombre  $\pi$ . Et ceci avec un ordinateur personnel classique!

Pour optimiser la complexité des calculs, il utilise le principe de dichotomie.

Voici le principe : pour calculer le produit de tous les entiers compris entre  $a$  et  $b$ , on effectue les produits des entiers entre  $a$  et  $c$ , puis entre  $c$  et  $b$ , où  $c$  est situé au milieu; puis on effectue le produit des deux résultats. Par exemple,  $\text{prod}(6, 18) = \text{prod}(6, 12) \times \text{prod}(13, 18)$ .

Programmer la fonction récursive `prod(a, b)` qui effectue le produit de tous les entiers compris entre  $a$  et  $b$ . La factorielle de  $n$  s'obtiendra alors par l'appel `prod(1, n)`.

## Motifs

1. Qu'affiche l'algorithme suivant :

```
def fonc1(n):
    if n>0:
        print('*'*n)
        fonc1(n-1)
```

2. Modifier la fonction précédente pour qu'elle affiche :

```
>>> fonc2(5)
*
**
***
****
*****
```

3. Utiliser le principe précédent pour afficher les motifs suivants :

```
*****
****
***
**
*
*
**
***
****
*****
```

```
*****
*** **
**  **
*   **
*   **
**  **
*** **
**** **
*****
```

```

      *
     **
    ***
   ****
  *****
 *****
*****
*****
*****
*****
*****
*****
*****
*****
*****
```