TP du 05-11-2019 : décomposition en facteurs premiers

Décomposition d'un entier en facteurs premiers

Dans cet exercice, on se propose d'écrire un algorithme pour décomposer un entier en produit de nombres premiers. Les algorithmes demandés doivent être écrits en langage Python. On sera très attentif à la rédaction et notamment à l'indentation du code.

On rappelle la définition de valuation p-adique : soit p un nombre premier et $n \in \mathbb{N}^*$. Si p divise n, on note $v_p(n)$ le plus grand entier k tel que p^k divise n. Si p ne divise pas n, on pose $v_p(n) = 0$. L'entier $v_p(n)$ s'appelle la valuation p-adique de n.

0.1 Une première méthode (extrait de CCP 2019)

On considère la fonction suivante `est_premier(n)` qui teste la primalité d'un entier $n \in \mathbb{N}^*$.

```
def est_premier(n):
    if n < 2:
        return False
    d = 2
    while d**2 <= n:
        if n % d == 0:
            return False
        d = d + 1
    return True
```

1. Tracer la fonction avec $n = 55$ et $n = 31$. On pourra compléter un tableau du type :

d	2	3	...	...
$d^2 \leqslant n$	V	...	...	...
$n\%d == 0$	F	...	...	...

d	2	3	...	...
$d^2 \leqslant n$	V	...	...	...
$n\%d == 0$	...	...	...	...

2. Justifier la terminaison de cette fonction, puis indiquer le nombre de divisions nécessaires dans le pire des cas pour tester la primalité d'un entier naturel n ?

3. Écrire une fonction `premier_suivant(n)` qui prend en argument un entier naturel $n \in \mathbb{N}^*$ et qui renvoie le plus petit nombre premier $p \geqslant n$.

4. Écrire une fonction `liste_premiers(n)` qui prend en argument un entier $n \in \mathbb{N}^*$ et qui renvoie la liste des nombres premiers inférieurs ou égaux à n.

5. Écrire une fonction `valuation_p_adique(n, p)` qui prend en argument un entier $n \in \mathbb{N}^*$, p un nombre premier et renvoie la valuation p-adique de n. Par exemple, puisque $40 = 2^3 \times 5$, `valuation_p_adique(40, 2)` renverra 3, `valuation_p_adique(40, 5)` renverra 1 et `valuation_p_adique(40, 7)` renverra 0. On pourra utiliser la stratégie suivante : par exemple pour calculer la valuation 2-adique de 40, on divise 40 par 2, on trouve 20, puis on divise 20 par 2, on trouve 10, puis on divise 10 par 2, on trouve 5 et enfin 5 n'est pas divisible par 2. On s'arrête.

6. Bonus pour ceux qui connaissent la récursivité (au programme en MP) : écrire une deuxième fonction cette fois-ci **récursive**, `val(n, p)` qui renvoie la valuation p-adique de n

7. En déduire une fonction `decomposition_facteurs_premiers(n)` qui calcule la décomposition en facteurs premiers d'un entier $n \geqslant 2$.

   Cette fonction renverra la liste des couples $(p, v_p(n))$ pour tous les nombres premiers p qui divisent n. Par exemple, `decomposition_facteurs_premiers(40)` renverra la liste `[[2, 3], [5,1]]`.

8. Déterminer une majoration du nombre de divisions effectuées lorsque l'on exécute `liste_premiers(n)`. En déduire, la complexité de `liste_premiers`. On pourra utiliser librement que pour tout $\alpha > 0$, on a pour n au voisinage de $+\infty$,

$$\sum_{k=1}^{n} k^\alpha \sim \int_1^n t^\alpha \, \mathrm{d}t.$$

0.2 Avec le crible d'Eratosthène

Nous allons découvrir l'algorithme du crible d'Eratosthène qui renvoie la liste des nombres premiers inférieurs à n, qui est bien plus performant que la fonction `liste_premiers`.

L'algorithme procède par élimination : il s'agit de supprimer d'une table des entiers de 2 à n tous les multiples stricts d'un entier. En supprimant tous les multiples stricts, à la fin il ne restera que les entiers qui ne sont multiples d'aucun entier, et qui sont donc les nombres premiers. On commence par rayer les multiples stricts de 2, puis à chaque fois on raye les multiples stricts du plus petit entier restant. À la fin du processus, tous les entiers qui n'ont pas été rayés sont les nombres premiers inférieurs à n.

Exemple avec $n = 10$: s = [2,3,4,5,6,7,8,9,10]

On élimine les multiples (stricts) de $p = 2$: s→ [2,3,5,7,9]

Le suivant est $p = 3$: on élimine les multiples de $p = 3$: s→ [2,3,5,7]

Le suivant est $p = 5$ on élimine les multiples de $p = 5$: s→ [2,3,5,7]

Le suivant est $p = 7$ on élimine les multiples de $p = 5$: s→ [2,3,5,7]

etc...

9. Implémenter le crible d'Eratosthène. On pourra commencer par créer un tableau t de $n+1$ booléens tous égaux à `True`. Le booléen $t[k]$ attestera de la primalité ou non de l'entier k. Si à la fin de l'agorithme, `t[k]` vaut `True`, cela signifiera que l'entier k est premier. Comme 0 et 1 ne sont pas premiers, on commence par effectuer les affectations `t[0] = False` et `t[1] = False`

   Par exemple, pour $n = 10$, après l'exécution du crible, le tableau t vaudra `[False, False, True, True, False, True, False, True, False, False, False]`.

10. Déterminer la complexité de cet algorithme.

11. Un raffinement : on peut arrêter de «rayer» lorsque le carré du plus petit entier restant est supérieur au plus grand entier restant, car dans ce cas, tous les non-premiers ont déjà été rayés précédemment. Donner la complexité.