Exercices de programmation 1

Fonction indicatrice d'Euler, exponentiation rapide

Exercice 1. *Indicatrice d'Euler 1*

1. Définir une fonction `pgcd` qui prend en paramètres deux entiers a et b et renvoie leur PGCD.
2. Définir une fonction `est_premier` qui prend en paramètre deux entiers a et b, et qui renvoie 1 si a et b sont premiers entre eux, 0 sinon.
3. Définir une fonction `indic_euler1` qui prend en paramètre un entier n, et qui renvoie la valeur de $\varphi(n)$, en considérant que c'est le nombre d'entiers inférieurs à n premiers avec n.

Exercice 2. *Indicatrice d'Euler 2*

1. Écrire une fonction `decomp_prem` qui affiche la décomposition en facteurs premiers sous la forme d'une liste de couples (p_i, a_i), par exemple:

```
decomp_prem(148) -> [(2,2), (37,1)]
decomp_prem(1092) -> [(2,2), (3,1), (7,1), (13,1)]
```

   car $148 = 2^2 \times 37$ et $1092 = 2^2 \times 3 \times 7 \times 13$.

2. Définir une fonction récursive `indic_euler2` qui prend en paramètre un entier n, et qui renvoie la valeur de $\varphi(n)$, via les règles de calcul suivante :
   - si $n = p$ est premier, alors
   $$\varphi(p) = p - 1.$$
   - si $n = p^k$ est une puissance d'un nombre premier, alors
   $$\varphi(p^k) = p^k - p^{k-1}.$$
   - si $n = p_1 \times p_2$ est le produit de deux nombres **premiers entre eux**, alors
   $$\varphi(p_1 \times p_2) = \varphi(p_1) \times \varphi(p_2).$$
   - Comparer l'efficacité des deux fonctions `indic_euler1` et `indic_euler2` en temps d'exécution.

Exercice 3. *Calcul de puissance modulaire naïf*

Définir une fonction `puissance_mod` qui prend en paramètres un entier a, un entier k et un entier n, et qui calcule la quantité $a^k[n]$ par une fonction itérative naïve (dans une boucle, on multiplie par a à chaque passage).

Exercice 4. *Calcul de puissance modulaire par exponentiation rapide*

1. Définir une fonction `expo_rapide` qui prend en paramètres un entier a, un entier k et un entier n, et qui calcule la quantité $a^k[n]$ par l'algorithme d'exponentiation rapide récursif.

2. Comparer les résultats obtenus avec l'instruction `pow(a,k,n)` de Python.

3. Comparer l'efficacité des fonctions `puissance_mod` et `expo_rapide` en temps d'exécution pour calculer des puissances modulaires de plusieurs très grands entiers.

4. Vérifier les valeurs obtenues à la main de :

   - $2^{65}[53]$,
   - $7^{231}[238]$.

On rappelle que pour mesurer le temps d'exécution d'une fonction en Python, on peut utiliser le module `time` et le code suivant :

```
import time

start = time.time()
# Instructions
end = time.time()
print("Temps d'execution :", end-start)
```