CORRIGÉ DU DEVOIR LIBRE N°5

Complexité du crible d'Eratosthène

A. Mise en place du problème

1. Si un nombre inférieur à n est composé, il a nécessairement un diviseur inférieur à $\sqrt{n}$, car si $k > \sqrt{n}$ et $\ell > \sqrt{n}$, alors $k\ell > n$. On peut donc s'arrêter à $\sqrt{n}$.

2. Lorsqu'on en est à supprimer les multiples de p, on a déjà supprimé tous les multiples des nombres premier strictement inférieurs à p. Or les multiples kp pour $k \in [\![2, p-1]\!]$ ont un diviseur strictement inférieur à p, donc ont un diviseur premier strictement inférieur à p : ils ont déjà été supprimés.

   Le premier multiple à supprimer est donc $p \times p = p^2$.

3.
```
def crible(n):
    """Implémentation du crible d'Eratosthène au rang n"""
    estPremier = [False for _ in range(n + 1)]
    estPremier[0:2] = [False, False]
    # A la fin de l'algorithme, on veut que
    # k soit premier ssi estPremier[k] contient True
    k = 2
    while k ** 2 <= n:
        if estPremier[k]:  # Si k est un nombre premier
            m = k * k
            # premier multiple potentiellement non encore rencontré
            while m <= n:    # Suppression des multiples de k
                estPremier[m] = False
                m += k
        k += 1
    return estPremier
```

   ⚠ Surtout pas de `int(sqrt(n))` et autre `floor(n ** 0.5)` car l'utilisation des nombres flottants apporte des approximations qui fausse l'algorithme pour n grand.
   On vérifie en pratique, sur une architecture 64 bits, qu'à partir de

$$n = 20\,000\,000\,264\,013\,300\,877\,534\,066\,376\,705$$

   le résultat devra être incorrect.
   Bon, avec la complexité qu'on va obtenir et un ordre de grandeur de 1 G opérations par seconde, il faudrait près de $3 \cdot 10^{15}$ années pour aller au bout du crible avec ces valeurs de n (rappel : âge de l'univers = $13{,}6 \cdot 10^9$ années).
   Et comme la complexité spatiale est facile en $O(n)$ (avec le liste `estPremier`), il faudrait une mémoire de près de $3 \cdot 10^{30}$ octets soit 3 billiards de péta-octets.
   Mais quand même.

4. On n'effectue la deuxième boucle que si p est premier et à l'intérieur de cette boucle, on a un nombre constant α d'opérations élémentaires. Cette boucle est exécutée pour chaque multiple $m = p\ell$ de m entre p^2 et n.
   On a donc, en négligeant les opérations lorsque p n'est pas premier,

$$C(n) = \sum_{\substack{p \in \mathscr{P} \\ p \leqslant \sqrt{n}}} \left(\sum_{p^2 \leqslant m = p\ell \leqslant n} \alpha \right) = \sum_{\substack{p \in \mathscr{P} \\ p \leqslant \sqrt{n}}} \left(\sum_{\substack{p^2 \leqslant m \leqslant n \\ p \mid m}} \alpha \right)$$

5. Comme α ne dépend pas de m, on a alors $C(n) = \sum_{\substack{p \in \mathscr{P} \\ p \leqslant \sqrt{n}}} (\alpha |E_p|) = \alpha \sum_{\substack{p \in \mathscr{P} \\ p \leqslant \sqrt{n}}} |E_p|$ où $E_p = \{m \in [\![p^2, n]\!], p \mid m\}$. Or $m \in E_p$ si et seulement si m s'écrit $p\ell$ avec $p^2 \leqslant p\ell \leqslant n$ soit $p \leqslant \ell \leqslant \frac{n}{p}$ donc $|E_p| \leqslant \frac{n}{p}$ (en fait $|E_p| = \left\lfloor \frac{n}{p} \right\rfloor - p + 1$).
   Finalement, $C(n) \leqslant \alpha n \sum_{\substack{p \in \mathscr{P} \\ p \leqslant \sqrt{n}}} \frac{1}{p}$.

6. Or $\sum_{\substack{p \in \mathscr{P} \\ p \leqslant \sqrt{n}}} \frac{1}{p} = \sum_{k=1}^{\pi(\sqrt{n})} \frac{1}{p_k} \leqslant \sum_{k=1}^{n} \frac{1}{p_k}$ car $\pi\left(\sqrt{n}\right) \leqslant \pi(n) \leqslant n$.

B. Équivalent de p_n

7. Par définition, $\pi(p_n) = n$. Donc, par le théorème des nombres premiers, comme $p_n \to +\infty$, $n \sim \frac{p_n}{\ln p_n}$. Donc $p_n \sim n \ln p_n$.

8. On a $u_n = v_n + o(v_n)$ donc $\ln u_n = \ln\left(v_n + o(v_n)\right) = \ln v_n + \ln(1 + o(1))$ avec $\ln(1 + o(1)) \to 0$ et $\ln v_n \to +\infty$ donc $\ln(1 + o(1)) = o(\ln v_n)$. Donc $\ln u_n \sim \ln v_n$.

9. Comme $n \to +\infty$, on peut prendre le ln dans l'équivalent $n \sim \frac{p_n}{\ln p_n}$, ce qui donne $\ln n \sim \ln p_n - \ln(\ln p_n)$.

10. Comme $\ln p_n \to +\infty$, $\ln(\ln p_n) = o\left(\ln p_n\right)$ donc $\ln p_n - \ln(\ln p_n) \sim \ln p_n$ et donc $\ln p_n \sim \ln n$ en utilisant la question précédente.

11. Avec la question 7, on tire directement $p_n \sim n \ln p_n \sim n \ln n$.

C. Nature de la série de terme général $\frac{1}{p_n}$ et équivalent des sommes partielles

12. On effectue une comparaison série-intégrale sur la fonction décroissante et continue $f : x \mapsto \dfrac{1}{x\ln x}$ sur $[2, +\infty[$.

$$\frac{1}{n\ln n} + \int_2^n \frac{1}{x\ln x}\,\mathrm{d}x \leqslant S_n = \sum_{k=2}^{n} \frac{1}{k\ln k} \leqslant \int_2^n \frac{1}{x\ln x}\,\mathrm{d}x + \frac{1}{2\ln 2}$$

avec $\displaystyle\int_2^n \frac{1}{x\ln x}\,\mathrm{d}x = \ln(\ln n) - \ln(\ln 2)$ et donc $\boxed{S_n \sim \ln(\ln n)}$ et $\boxed{\text{la série } \sum \frac{1}{n\ln n} \text{ diverge.}}$

13. Comme $\dfrac{1}{p_n} \sim \dfrac{1}{n\ln n}$ vu la partie précédente, par comparaison des séries à termes positifs dans le cas de divergence d'après la question précédente, $\boxed{\sum \frac{1}{p_n} \text{ diverge}}$ et par théorème de sommation des équivalents de séries à termes positifs dans le cas de divergence, $\displaystyle\sum_{k=2}^{n} \frac{1}{p_k} \sim S_n$. Or $\displaystyle\sum_{k=2}^{n} \frac{1}{p_k} \to +\infty$ donc $\dfrac{1}{p_1} = \dfrac{1}{2} = \mathrm{o}\left(\displaystyle\sum_{k=2}^{n} \frac{1}{p_k}\right)$ donc $\displaystyle\sum_{k=1}^{n} \frac{1}{p_k} \sim \sum_{k=2}^{n} \frac{1}{p_k} \sim S_n = \sum_{k=2}^{n} \frac{1}{n\ln n}$.
Donc, d'après la question précédente toujours $\boxed{S_n \sim \ln(\ln n).}$

D. Conclusion

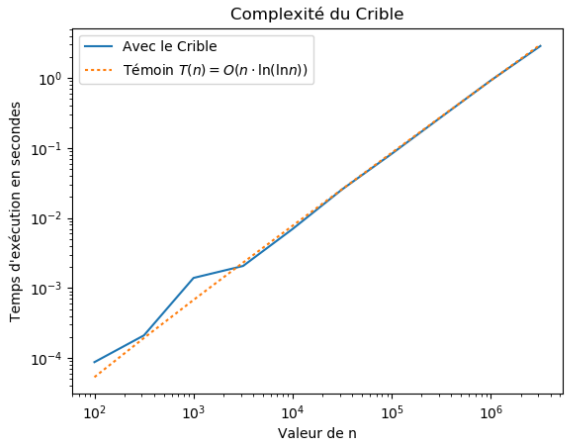
14. On a donc à partir d'un certain rang $C(n) \leqslant \alpha n \displaystyle\sum_{k=1}^{n} \frac{1}{p_k} \leqslant \frac{3\alpha}{2} n\ln(\ln n)$ donc $\boxed{C(n) = \mathrm{O}\left(n\ln(\ln n)\right).}$

Remarque : a même que

$$\sum_{k=1}^{\pi(\sqrt{n})} \frac{1}{p_k} \sim \ln\left(\ln\left(\pi\left(\sqrt{n}\right)\right)\right) \sim \ln\left(\ln\left(\frac{\sqrt{n}}{\ln\sqrt{n}}\right)\right) = \ln\left(\underbrace{\ln\sqrt{n} - \ln\left(\ln\sqrt{n}\right)}_{\sim \ln\sqrt{n} \to +\infty \neq 1}\right) \sim \ln\left(\ln\sqrt{n}\right) \sim \ln(\ln n)$$

donc la majoration $\displaystyle\sum_{\substack{p \in \mathscr{P} \\ p \leqslant \sqrt{n}}} \frac{1}{p} = \sum_{k=1}^{\pi(\sqrt{n})} \frac{1}{p_k} \leqslant \sum_{k=1}^{n} \frac{1}{p_k}$ (avec $\pi\left(\sqrt{n}\right) \leqslant \pi(n) \leqslant n$) n'était pas trop grossière.

Visualisation pratique :



obtenu avec le code suivant :

```
from time import time
import numpy as np
import matplotlib.pyplot as plt

def complexite_empirique(K=np.arange(2, 7, .5)):
    """renvoie
    - N : tableau des n = 10^k pour k dans K
    - le tableau des temps d'exécution du crible pour chaque n dans N"""
    complexite = []
    N = (10 ** K).astype(int)
    for n in N:
        print("traitement de n = {}".format(n))
        t = time()
        crible(n)
        complexite.append(time() - t)
    return N, complexite

def trace(N, complexite):
    """Tracé logarithmique des temps d'exécution du crible en
fonction de n"""
    plt.plot(N, complexite, label="Avec le Crible")
    plt.plot(N, N * np.log(np.log(N)) * 3.5e-7, linestyle=":",
             label="Témoin $T(n) = O(n\cdot\ln(\ln n))$")
    plt.xlabel("Valeur de n")
    plt.ylabel("Temps d'exécution en secondes")
    plt.title("Complexité du Crible")
    plt.legend(loc="best")
    plt.loglog()
    plt.show()
```

Fin