Projet #1 — Conjecture de Goldbach

La conjecture de Goldbach, énoncée en 1742 par Christian Goldbach, est l'affirmation mathématique suivante :

> Tout nombre entier pair supérieur à 3 peut s'écrire comme la somme de deux nombres premiers.

En apparence très simple, c'est **l'un des plus vieux problèmes non résolus** de la théorie des nombres ! Il fait l'objet, avec l'hypothèse de Riemann et la conjecture des nombres premiers jumeaux, du problème numéro 8 de la liste de problèmes dressée par David Hilbert en 1900.

Dans ce genre de problème mathématique, l'outil informatique peut aider dans deux cas :
(1) aide à la démonstration mathématique (par exemple, avec Coq[1]) ;
(2) découverte d'un contre-exemple.

Nous allons nous placer dans le deuxième cas et chercher un contre-exemple. Si nous trouvons un contre-exemple, nécessairement très grand[2], alors nous avons prouvé que la conjecture est fausse. Sinon, nous n'avons rien prouvé !

Les réponses aux questions ci-dessous seront déposées sous forme de fichiers python pour le code et pdf pour les questions sur Github Classroom : **https://classroom.github.com/a/J92z4RbA**

Pour programmer, travailler directement sur mon site web en téléchargeant régulièrement votre travail ou télécharger Thonny : **https://thonny.org** . S'il est normal et nécessaire de réaliser des recherches internet, aucun plagiat ne sera toléré. Je serais seul juge en la matière.

Contexte :

1) Chercher ce qu'est la liste de problèmes de David Hilbert et expliquer brièvement en quoi consiste cette liste de problèmes (ne pas les lister, juste expliquer si on parle de géométrie, de physique, de théorie des nombres…)

2) Vérifier que la conjecture de Goldbach est vraie pour les entiers pairs N compris entre 3 et 10. On donnera la valeur des deux entiers premiers p_1 et p_2 tels que $\mathrm{N} = p_1 + p_2$.

On rappelle qu'un nombre premier est un nombre qui n'admet aucun diviseur autre que 1 et lui-même.

Programmation :

3) Compléter une fonction `vérifier_nombre` qui prend pour paramètre un entier nommé `entier`. Cette fonction renvoie :
   - False si `entier` est négatif, nul ou égal à 1 ;
   - True sinon.

---

[1] https://fr.wikipedia.org/wiki/Coq_(logiciel)

[2] Si un petit contre-exemple existe, alors nous l'aurions sans doute déjà trouvé… cela fait 3 siècles que nous essayons de tester la conjecture de Goldbach !

4) Compléter la fonction `est_premier_lent` qui prend pour paramètre un entier `entier`.
Cette fonction renvoie :
   - False si `entier` est négatif, nul ou égal à 1 ;
   - False si `entier` n'est pas un nombre premier
   - True sinon.

   Aide : une bonne stratégie consiste à tester tous les diviseurs possibles de `entier`. Si un diviseur est trouvé, on renvoie False.

   Exemple :
   `est_premier_lent(11)` renvoie True
   `est_premier_lent(1)` renvoie False
   `est_premier_lent(81)` renvoie False

```
def est_premier_lent(entier):
    diviseur = 2
    if vérifier_nombre(entier) == True:
        while diviseur <= entier:



        return
    else:
        return
```

5) Écrire une fonction `est_nombre_goldbach` qui prend pour paramètre un entier `entier_pair`.
Cette fonction renvoie :

   - **True** ainsi que le **plus petit nombre premier** `n_1` tel que `entier_pair = n_1 + n_2` si c'est un nombre de Goldbach.
   - **False** ainsi que `entier_pair` sinon

   Aide : une bonne stratégie consiste à passer en revue tous les entiers `n_1` entre 0 et `entier_pair`−1. Si un entier `n_1` est premier, alors on va devoir tester si `n_2` est premier…

   Exemple :
   `est_nombre_goldbach(12)` renvoie True, 5
   `est_nombre_goldbach(98)` renvoie True, 19

6) Compléter la fonction `tester_conjecture_goldbach` qui prend pour paramètre un entier nommé `entier_maximum` correspondant à l'entier pair maximum que l'on souhaite tester.
Cette fonction renvoie la division du **nombre de nombres de Goldbach** par le **nombre de passage dans la boucle while**. Cette fonction fera également un affichage de chaque nombre de Goldbach et de sa décomposition en deux nombres premiers.

   Aide : attention à ne tester que les nombres entiers pairs !
   Quand une fonction nommée blablabla renvoie deux quantités a et b (comme la fonction `est_nombre_goldbach`), on peut écrire : `a, b = blablabla(truc)`.

```
def tester_conjecture_goldbach(entier_maximum):
    entier_pair = 4
    nombre_de_nombre_goldbach = 0
    nombre_de_passage_boucle = 0
    while entier <= nombre_maximum:
```

   Exemple :
   `tester_conjecture_goldbach(100)` renvoie 1.0 et affiche :

```
4 2 2
6 3 3
8 3 5
…
96 7 89
98 19 79
100 3 97
```

Optimisation :

7) Dans la fonction `est_premier_lent`, qu'est-ce qui est incorrect dans la conditionnelle `vérifier_nombre(entier) == True` ? Corriger votre programme.

8) La fonction `est_premier_lent` est très lente.
   Prenez le nombre premier 97. Combien de diviseurs possibles sont testés ?
   En fait, nous avons besoin de faire notre test jusqu'à $\sqrt{97}$ inclus. Combien de diviseurs possibles seront alors testés ?

9) À partir de la fonction `est_premier_lent`, créer une fonction `est_premier` faisant exactement la même chose que `est_premier_lent` mais de manière optimisée. On aura besoin d'importer le module math (voir les TP).

10) Modifier `tester_conjecture_goldbach` afin de le rendre plus rapide et tester la conjecture de Goldbach jusqu'à de très grandes valeurs : donner ici la décomposition du nombre le plus grand que vous avez réussi à faire !

    Exemple :
    `tester_conjecture_goldbach(10000)` renvoie 1.0 et affiche : 10000 59 9941