Travaux pratiques

informatique commune

Autour de la conjecture de Syracuse

La conjecture de Syracuse

On doit cette conjecture au mathématicien allemand Lothar COLLATZ qui, en 1937, proposa à la communauté mathématique le problème suivant :

> on part d'un nombre entier strictement positif ; s'il est pair on le divise par 2, s'il est impair on le multiplie par 3 et on ajoute 1. On réitère ensuite cette opération.

Par exemple, à partir de 14 on construit la suite de nombres :

$$14\,,7\,,22\,,11\,,34,\ 17,\ 52,\ 26,\ 13,\ 40,\ 20,\ 10,\ 5,\ 16,\ 8,\ 4,\ 2,\ 1,\ 4,\ 2\ \ldots$$

Après que le nombre 1 ait été atteint, la suite des valeurs (1, 4, 2, 1, 4, 2 ...) se répète indéfiniment en un cycle de longueur 3.

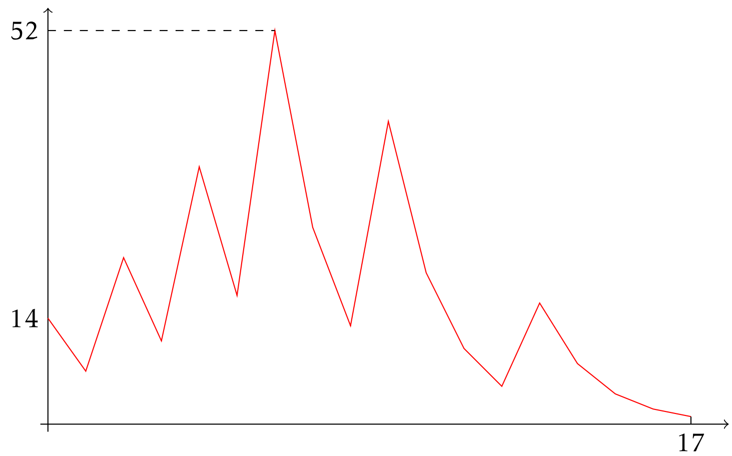

FIGURE 1 – le graphe de la suite de Collatz pour $c = 14$

La conjecture de Syracuse[1] est l'hypothèse mathématique selon laquelle n'importe quel entier de départ conduit à la valeur 1 au bout d'un certain temps.

Nous allons expérimenter cette conjecture en programmant l'évolution de la suite $(u_n)_{n\in\mathbb{N}}$ définie par les relations

$$u_0 = c \qquad \text{et} \qquad \forall n \geqslant 1, \quad u_{n+1} = \begin{cases} \dfrac{u_n}{2} & \text{si } u_n \text{ est pair} \\ 3u_n + 1 & \text{si } u_n \text{ est impair} \end{cases}$$

Temps de vol et altitude maximale

Question 1. Le *temps de vol* d'un entier c et le plus petit entier n (en admettant qu'il existe) pour lequel $u_n = 1$. Par exemple, le temps de vol pour $c = 14$ est égal à 17.
Définir une fonction nommée `tempsdevol` prenant un paramètre entier c et retournant le plus petit entier n pour lequel $u_n = 1$.

Question 2. De manière tout aussi imagée, on appelle *altitude maximale* de c la valeur maximale de la suite $(u_n)_{n\in\mathbb{N}}$. Par exemple, l'altitude maximale de $c = 14$ est égale à 52.
Modifier votre algorithme pour définir une fonction nommée `altitude` qui calcule cette fois-ci l'altitude maximale pour un entier c donné en paramètre.

1. Du nom de l'université américaine qui a popularisé ce problème.

Vérification expérimentale de la conjecture

On désire désormais vérifier la validité de la conjecture pour toute valeur $c \leqslant 1\,000\,000$. Une première solution consisterait à calculer le temps de vol pour toutes ces valeurs, mais ce calcul est long et il y a mieux à faire en observant que si la conjecture a déjà été vérifiée pour toute valeur $c' < c$, il suffit qu'il existe un rang n pour lequel $u_n < c$ pour être certain que la conjecture sera aussi vérifiée au rang c.

Question 3. On appelle *temps d'arrêt* (ou encore *temps de vol en altitude*) le premier entier n (s'il existe) pour lequel $u_n < c$.

a) Écrire une fonction `tempsdarret` prenant un paramètre entier c et retournant le temps d'arrêt de la suite de Syracuse correspondante.

Nous souhaitons maintenant mesurer le temps nécessaire pour vérifier la conjecture jusqu'à un paramètre entier m. Pour cela, nous allons utiliser la fonction `time` du module `time` du même nom, sans argument, qui retourne le temps en secondes depuis une date de référence (qui dépend du système).

b) À l'aide de cette fonction écrire une fonction `verification` qui prend en argument un paramètre entier m et retourne le temps nécessaire pour vérifier que toutes les valeurs $c \in [\![2, m]\!]$ ont bien un temps d'arrêt fini.
Quelle durée d'exécution obtient-t'on pour $m = 1\,000\,000$?

c) Quel est le temps d'arrêt d'un entier pair ? et d'un entier de la forme $c = 4n + 1$? En déduire qu'on peut restreindre la recherche aux entiers de la forme $4n + 3$, et modifier en conséquence la fonction précédente. Combien de temps gagne-t'on par rapport à la version précédente pour $m = 1\,000\,000$?
Vérifier ensuite la conjecture pour $m = 10\,000\,000$.

Records

Question 4.

a) Déterminer l'altitude maximale que l'on peut atteindre lorsque $c \in [\![1, 1\,000\,000]\!]$, ainsi que la valeur minimale de c permettant d'obtenir cette altitude.

b) Déterminer le temps de vol en altitude (autrement dit le temps d'arrêt) de durée maximale lorsque $c \in [\![1, 1\,000\,000]\!]$ ainsi que la valeur de c correspondante.

Question 5. On appelle *vol en altitude de durée record* un vol dont tous les temps d'arrêt de rangs inférieurs sont plus courts. Par exemple, le vol réalisé pour $c = 7$ est un vol en altitude de durée record (égale à 11) car tous les vols débutant par $c = 1, 2, 3, 4, 5, 6$ ont des temps d'arrêt de durées inférieures à 11. Déterminez tous les vols en altitude de durée record pour $c \leqslant 1\,000\,000$.

Affichage du vol

Pour obtenir des graphes analogues à celui de la figure 1, on utilise la fonction `plot` qui appartient à un module appelé `mathplotlib.pyplot` et dédié au tracé de graphes. Vous allez donc commencer par importer celui-ci à l'aide de la commande :

```
import matplotlib.pyplot as plt
```

Désormais toutes les fonctions de ce module vous sont accessibles à condition de les préfixer par `plt`. Nous découvrirons progressivement les nombreuses possibilités qu'offre ce module, mais aujourd'hui nous n'aurons besoin que de deux fonctions : `plt.plot` et `plt.show`.

Sous sa forme la plus simple, la fonction `plt.plot` n'exige qu'une liste en paramètre : `plt.plot([a0, a1, ..., an])` crée un graphe constitué d'une ligne brisée reliant les points de coordonnées (k, a_k) pour $k \in [\![0, n]\!]$.
En Python, une liste est encadrée par des crochets et ses éléments séparés par une virgule. Nous étudierons les listes plus tard dans le cours ; pour l'instant nous n'aurons besoin que du résultat suivant : si `lst` est une liste, on ajoute un élément `x` à celle-ci à l'aide de la commande : `lst.append(x)`.
Une fois votre graphe créé par la fonction `plt.plot`, il reste à le faire apparaître dans une fenêtre annexe à l'aide de l'instruction `plt.show()`.

Question 6. Définir une fonction nommée `graphique` qui prend un entier c en paramètre et qui construit le graphe de la suite $(u_n)_{n\in\mathbb{N}}$ durant son temps de vol.

Et pour les plus rapides

Nous avons vu à la question 2.c qu'il suffit de restreindre l'étude de la conjecture aux entiers de la forme $4n+3$, soit à 25 % des entiers. On peut chercher à affiner cette démarche en s'intéressant aux entiers de la forme $8n+k$ mais ceci ne conduit pas à une amélioration des performances puisqu'on ne peut que restreindre l'étude aux entiers de la forme $8n+3$ et $8n+7$.
En revanche, il est possible de restreindre l'étude aux entiers de la forme $16n+7$, $16n+11$ et $16n+15$ soit $18,75$ % des entiers.
Si on écrit les entiers sous la forme $65536n+k$ ($65536 = 2^{16}$), à combien de valeurs de k peut-on restreindre l'étude ?