Stay Home April - Démonstration du problème 11

par tuzepoito

18 avril 2020

Présentation

Le problème [1] a été énoncé ainsi : il faut récupérer un mot de passe, ou « flag » à partir du code JavaScript suivant :

```
const jsSHA = require("jssha");

function calc_flag(a, b, c) {
    var s = b / a;
    var d = 0;
    while (d < c) {
        s += b / (a * a - (1 + 2 * d) * a + d * d + d);
        d += 1;
    }
    return s;
}

const a = 298974293284332591;
const b = 4329782349832498723498723498 78;
const c = 2783324234832341;
var sha256 = new jsSHA('SHA-256', "TEXT");
sha256.update(calc_flag(a, b, c).toFixed(5).toString());
var hash = sha256.getHash("HEX");

if (hash === prompt("mot de passe !")) {
    login_admin();
}
```

En somme, on a une fonction `calc_flag` qui retourne un nombre à virgule flottante défini dans la variable `s`. Ce nombre est ensuite converti en chaîne de caractères pour générer un hash SHA-256 qui devra être fourni comme solution du problème.

Le premier problème est que la boucle principale dans la fonction `calc_flag` est répétée `c` fois, ce qui est très long. Même avec un ordinateur particulièrement puissant, sur la plupart des machines cela prendrait des heures ; or le but était de fournir la solution le plus vite possible, de préférence en moins de 24 heures (le temps qu'il fallait pour résoudre l'exercice).

1. initialement énoncé sur le Stay Home April Challenge de h25 : `http://sha25.h25.io`

Première approche

Une première remarque que l'on peut faire est que la variable `s` est initialisée à `b / a`, ce qui représente eviron $1{,}448 \cdot 10^{12}$. Or le nombre qu'on lui ajoute dans la boucle,

```
b / (a * a - (1 + 2 * d) * a + d * d + d)
```

prend des valeurs entre $4{,}844 \cdot 10^{-6}$ et $4{,}935 \cdot 10^{-6}$ lorsque `d` varie entre 0 et `c - 1`. Cet écart de 10^{18} en ordre de grandeur, et l'arithmétique en virgule flottante utilisée en JavaScript, cause une erreur d'arrondi qui fait que l'addition n'a aucun effet [2].

On est alors tenté de supprimer la boucle puisqu'elle est apparemment sans effet. Mais en fait ce n'est pas la réponse attendue par l'exercice.

Solution

La valeur théorique finale de `s` peut être exprimée ainsi :

$$s = \frac{b}{a} + \sum_{d=0}^{c-1} \frac{b}{a^2 - (1+2d)a + d^2 + d}$$

Il nous faudrait trouver un moyen d'exprimer cette équation qui nous débarrasse de cette somme, qui boucle de 0 à `c-1`.

Le site Wolfram Alpha peut nous venir en aide si on lui donne l'équation d'après ce lien :
`https://www.wolframalpha.com/input/?i=b%2Fa++%2B+%E2%88%91+b+%2F+%28a+*+a+-+%281+%2B+2+*+d%29+*+a+%2B+d+*+d+%2B+d%29%2C+d+%3D0+to+%28c-1%29`

(cette formulation est donnée par egaetan)

Il nous donnera alors l'égalité suivante :

$$s = \frac{b}{a - c}$$

2. Lien vers Wikipédia (en anglais) :`https://en.wikipedia.org/wiki/Double-precision_floating-point_format`

C'est bien joli tout ça, c'est un beau cas de « l'ordi fait brr » :

(source : `https://www.reddit.com/r/mathmemes/comments/fqww75/the_four_color_map_theorem_was_pretty/`)

Mais on peut aussi avoir envie de comprendre le raisonnement derrière. Alors définissons :

$$S = \sum_{d=0}^{c-1} \frac{b}{a^2 - (1+2d)a + d^2 + d}$$

On prouve :

$$\begin{aligned}
S &= \sum_{d=0}^{c-1} \frac{b}{(a-d)(a-d-1)} \\
&= \sum_{d=0}^{c-1} \left(-\frac{b}{a-d} + \frac{b}{a-d-1} \right) \\
&= -\frac{b}{a} + \frac{b}{a-1} - \frac{b}{a-1} + \frac{b}{a-2} - \dots - \frac{b}{a-c+1} + \frac{b}{a-c} \\
&= \frac{b}{a-c} - \frac{b}{a} \\
S &= \frac{bc}{a(a-c)}
\end{aligned}$$

Ainsi :

$$\begin{aligned}
s &= \frac{b}{a} + \frac{bc}{a(a-c)} \\
s &= \frac{b}{a-c}
\end{aligned}$$

On peut ainsi remplacer la fonction `calc_flag` :

```
function calc_flag(a, b, c) {
    return b / (a - c);
}
```

Alors `calc_flag(a,b,c).toFixed(5)` vaut `"1461821190472.85791"`. Le hash SHA-256 qui en résulte, et donc la solution au problème, est 14d1e9cc71949f4407f7b07de2075c276cf2fccf2619bafdc1ef0e240670c30d.