PROJET EULER
Problème 119

Table des matières

1 Rappels de nos objectifs

Suite à notre dernier entretien, nos objectifs sont les suivants :
- Etudier la piste du logarithme afin de déterminer à l'avance nos bornes de recherche
- Déterminer les limites de notre fontion qui calcule la somme des chiffres d'un nombre
- Essayer de supprimer ce phénomène de chance dans notre programme

2 Comment trouver les bornes de recherche ?

Ici, notre objectif est de borner a et b de manière à ce que a^b soit inférieur à un certain nombre défini à l'avance. Ainsi, on considère que lorsque l'on cherche le n-ième terme de notre liste de nombres, on connait approximativement son ordre de grandeur et il existe une valeur connue, supérieure à cette dernière. Si on appelle $vMax$ cette valeur maximum, le but est donc de trouver pour quelles valeurs de a et de b, $a^b < vMax$.

2.1 Comment borner a ?

Nous devons trouver la valeur maximum de a telle que $a^b < vMax$. Pour cela, il faut prendre la valeur minimum de b, car $\forall a1, a2, vMax, b1, b2 \in \mathbb{R}, b1 > b2 \Rightarrow \exists a1 > a2, a2^{b1} > a1^{b2}$.

Ainsi, on obtient l'inéquation suivante :

$$a^2 < vMax$$

$$\Leftrightarrow \ln a^2 < \ln vMax$$

$$\Leftrightarrow 2 * \ln(a) < \ln(vMax)$$

$$\Leftrightarrow \ln(a) < \frac{\ln(vMax)}{2}$$

$$\Leftrightarrow a < e^{\frac{\ln(vMax)}{2}}$$

Ainsi, si l'on connaît $vMax$, on sait à l'avance que a ne devra pas dépasser $e^{\frac{\ln(vMax)}{2}}$

2.2 Comment borner b ?

Une fois que l'on connaît la valeur maximum de a, on peut exprimer la valeur limite de b en fonction de a et de $vMax$. On a donc l'inéquation suivante :

$$a^b < vMax$$

$$\Leftrightarrow \ln a^b < \ln vMax$$

$$\Leftrightarrow b * \ln(a) < \ln(vMax)$$

$$\Leftrightarrow b < \frac{\ln(vMax)}{\ln a}$$

Ainsi, si l'on connaît $vMax$ et a , on sait à l'avance que b ne devra pas dépasser $\frac{\ln(vMax)}{\ln a}$

3 Optimisation de la fonction qui calcule la somme des chiffres d'un nombre ?

Lorsque l'on borne a et b, le temps de calcul augmente largement car en supprimant la chance de notre programme on augmente largement le nombre d'appels à la fonction sommeChiffresNombre afin d'être sûr de trouver le bon résultat. Nous avons donc essayé de réduire le temps de calcul de cette fonction, d'autant plus que celle-ci ne fonctionne plus à partir d'une certaine valeur. En effet, nous n'avons pas réussi à en connaître la raison mais à partir de 10^{17} notre fonction ne retourne pas le bon résultat. Nous avons donc essayé une différente méthode pour calculer la somme des chiffres d'un nombre et cette dernière s'est retrouvé beaucoup plus efficace, pas tellement au niveau du temps de calcul mais plus au niveau de la complexité et du fait qu'elle fonctionne pour toutes les valeurs. Elle consiste à transformer le nombre en chaîne de caractères et calculer la somme des caractères de la chaîne obtenue. En python, la fonction correspond donc à :

```
1 def sommeChiffreNombre(nombre):
2     return sum([ int(c) for c in str(nombre) ])
```

4 Comment déterminer vMax ?

Pour pouvoir borner a et b, il faut d'abord déterminer $vMax$. Le temps d'exécution dépendra de $vMax$ car si il est trop éloigné de la valeur recherchée, le temps d'exécution du programme sera trop élevé. Cependant, on ne peut connaître à l'avance la valeur recherchée. C'est pourquoi nous avons fait en sorte d'augmenter progressivement $vMax$: si $vMax$ est atteint sans que l'on ait trouvé le n^{eme} terme (où le n^{eme} terme est celui que l'on recherche), on augmente $vMax$. La formule que nous avons choisit pour augmenter $vMax$ est :

$$vMax = vMax * 5^{(n^{ème}terme - \text{la quantité déjà trouvé})/2}$$

Cette formule n'est pas admise mais résulte de nos expériences. Par exemple : on cherche le 20ème terme, et on en a déja trouvé 10 avec le $vMax$ actuel. Ici, $n^{eme}terme - $la quantité déjà trouvé $= 10$. On multiplie donc $vMax$ par5^5 ($5^{10/2}$). Ainsi, peu importe le $vMax$ d'origine, le programme finira toujours pas trouvé la solution. Néanmoins il existe des $vMax$ plus efficaces pour certaines valeurs. Par exemple : la valeur recherchée est 1000000 et le $vMax$ d'origine est 2. Admettons que $vMax$ atteigne 999999, le programme n'aura pas trouvé la bonne valeur mais $vMax$ augmentera

de beaucoup trop pour rien. Alors que si le $vMax$ d'origine avait été égal à 3, on aurait atteint plus rapidement et simplement 1000000. On remarque donc que le programme est plus rapide pour $vMax = 2$ que pour $vMax = 3$, même s'il fonctionne dans les deux cas. Ce n'est ici qu'un exemple, et en suivant les contraintes de ce problème, on peut déterminer un palier : si $n < 10$, il faut choisir $vmax = 10$, sinon il faut commencer avec $vMax = 700000$.