JOUR 1

PUISSANCES

solutions

Question 1

On énumère d'abord par catégories :

- les carrés : $2^2 = 4$, 9, 16, 25, 36, 49, 64, 81, 100, etc.,
- les cubes : $2^3 = 8$, 27, 64, 125, etc.,
- les puissances quatrièmes : $2^4 = 16$, 81, 256, etc.,
- les puissances cinquièmes : $2^5 = 32$, 243, etc.,
- les puissances sixièmes : $2^6 = 64$, 729, etc.,
- les puissances septièmes : $2^7 = 128$, etc.,

et là on peut s'arrêter puisqu'on a trouvé tout ce qui ne dépasse pas 100. On remet dans l'ordre :

$$4 ; 8 ; 9 ; 16 ; 25 ; 27 ; 32 ; 36 ; 49 ; 64 ; 81 ; 100.$$

Il y en a 12.

Question 2

On y retourne par catégorie. Notre point de repère est $2^10 = 4^5 = 32^2 = 1\,024$. Donc :

- chez les carrés : $31^2 = 961$, $32^2 = 1\,024$,
- chez les cubes : $10^3 = 1\,000$, $11^3 = 1\,331$,
- chez les puissances quatrièmes : $5^4 = 625$, $6^4 = 1\,296$,

(on voit que la base baisse très vite)

- chez les puissances cinquièmes : $3^5 = 243$, $4^5 = 1\,024$,
- chez les puissances sixièmes : $3^6 = 729$, $4^6 = 4\,096$,
- chez les puissances septièmes : $2^7 = 128$, $3^7 = 2\,187$,
- chez les puissances huitièmes : $2^8 = 256$, $3^8 > 3^7$,
- chez les puissances neuvièmes : $2^9 = 512$, $3^9 > 3^7$,
- ches les puissances dixièmes : $2^{10} = 1\,024$,

et on s'arrête là. La plus petite qu'on ait trouvé est donc 1 024.

Question 3

Soit $x \in \mathbf{N}$ qui est à la fois une puissance de 2 (disons 2^m) et une puissance de 3 (disons 3^n). D'après le théorème fondamental de l'arithmétique, si l'on dispose de deux décompositions de x en produit de facteurs premiers, alors les nombres premiers qui y apparaissent sont les mêmes, avec les mêmes exposants. Donc de

$$2^m = 3^n$$

on déduit que $m = n = 0$, autrement dit que $x = 1$. En particulier, on peut dire que $x = 1^6$, mais ce n'est pas très intéressant...

Question 4

Même argument que dans la question précédente. Si x est un entier pair, sa décomposition en produit de facteurs premiers comporte un facteur 2. Ceci interdit à x d'être une puissance de 3, puisqu'alors il aurait une décomposition de la forme 3^n, où le facteur 2 est absent.

Question 5

Pour avoir à la fois un carré, un cube, et une puissance cinquième, on cherche le PPCM de ces exposants. C'est 30 (le plus petit multiple commun à 2, à 3 et à 5). Prenons un nombre de la forme t^{30}, avec $t \in \mathbf{N} - \{0\,;\,1\}$. On a donc

$$t^{30} = (t^{15})^2 = (t^{10})^3 = (t^6)^5,$$

c'est donc à la fois un carré, un cube et une puissance cinquième.

Montrons maintenant que *toutes* les solutions sont de cette forme-là. Soit x un nombre qui est un carré. On écrit sa décomposition en produit de facteurs premiers :

$$x = \prod_p p^{v_p(x)},$$

le produit étant pris sur tous les nombres premiers. Supposons que x est une puissance n-ième : disons $x = b^n$. On écrit la déomposition de b

$$b = \prod_p p^{v_p(b)},$$

puis celle de b^n

$$b^n = \left(\prod_p p^{v_p(b)}\right)^n = \prod_p \left(p^{v_p(b)}\right)^n = \prod_p p^{n\times v_p(b)},$$

puis on identifie avec la décomposition de x (puisque, répétons-le, si l'on dispose de deux décompositions d'un nombre en produit de facteurs premiers, alors les nombres premiers qui y apparaissent sont les mêmes, et avec les mêmes exposants). On en déduit que pour tout nombre premier p on a

$$v_p(x) = n \times v_p(b),$$

c'est-à-dire que $v_p(x)$ est un multiple de n.

Revenons au problème : si x est à la fois un carré, un cube et une puissance sixième, alors les $v_p(x)$ sont à la fois multiples de 2, de 3 et de 5, donc multiples de 30. On peut donc écrire $v_p(x) = 30 \times \alpha_p$ pour un certain $\alpha_p \in \mathbf{N}$, et ainsi

$$x = \prod_p p^{30\times\alpha_p} = \left(\prod_p p^{\alpha_p}\right)^{30}$$

et donc x est une puissance 30-ième.

La plus petite solution à la question est $2^{30} = 1\,073\,741\,824 = 32\,768^2 = 1\,024^3 = 64^5$.

Exercice A

On a déjà fait, dans la question précédente, la démonstration de cet exercice (et dans un cas plus compliqué en fait). Donc rien de nouveau ici.

a) Soit $r = \sqrt[3]{a}$. On a $r^2 = (\sqrt[3]{a})^2 = \sqrt[3]{a^2} = \sqrt[3]{x} = \sqrt[3]{b^3} = b$.

b) Écrivons la décomposition de x en produit de facteurs premiers :

$$x = \prod_p p^{v_p(x)},$$

le produit étant pris sur l'ensemble des nombres premiers. On écrit également celle de a :

$$a = \prod_p p^{v_p(a)},$$

et enfin celle de a^2 :

$$a^2 = \left(\prod_p p^{v_p(a)}\right)^2 = \prod_p \left(p^{v_p(a)}\right)^2 = \prod_p p^{2\times v_p(a)}.$$

En identifiant cette décomposition avec celle de x (puisque $x = a^2$), on obtient que pour tout nombre premier p, on a $v_p(x) = 2 \times v_p(a)$. En particulier, $v_p(x)$ est pair.

c) Écrivons la décomposition de b
$$b = \prod_p p^{v_p(b)}$$
et celle de b^3
$$b^3 = \left(\prod_p p^{v_p(b)}\right)^3 = \prod_p \left(p^{v_p(b)}\right)^3 = \prod_p p^{3\times v_p(b)}.$$
En identifiant avec la décomposition de x (puisque $x = b^3$) on obtient que pour tout nombre premier p on a $v_p(x) = 3 \times v_p(b)$. En particulier, $v_p(x)$ est un multiple de 3.

d) Soit n un nombre qui est divisible par 2 et par 3. On a donc $v_2(n) \geqslant 1$ et $v_3(n) \geqslant 1$ et on peut écrire
$$n = \prod_p p^{v_p(n)} = 2 \times 3 \times \left(2^{v_2(n)-1} \times 3^{v_3(n)-1} \times \prod_{p\geqslant 5} p^{v_p(n)}\right).$$
En particulier n est un multiple de 6. Si on revient à l'exercice, cela signifie que pour tout nombre premier p, $v_p(x)$ est un multiple de 6.

e) On peut donc écrire $v_p(x) = 6 \times \alpha_p$ pour un $\alpha_p \in \mathbf{N}$ (qui dépend évidemment de p). Et ainsi
$$x = \prod_p p^{6\times\alpha_p} = \left(\prod_p p^{\alpha_p}\right)^6$$
donc x est une puissance sixième.

f) On a $r^3 = a$ donc $r^6 = a^2 = x$. Ainsi $r = \sqrt[6]{x}$, et puisque x est une puissance sixième, r est entier !

Question 6

Le programme construit la liste (rangée dans l'ordre croissant) de toutes les puissances non triviales inférieures ou égales à M. Pour ce faire, il part d'une liste `B` de booléens, initialement tous faux. Puis, dans cette liste, il change en `True` toutes les cases dont l'indice est une puissance non triviale (la variable p parcourt les puissances de d à partir de d^2, et la variable d parcourt les nombres à partir de 2). De cette manière, à la fin de la double-boucle `while`, les cases de `B` contenant la valeur `True` sont exactement celle dont l'indice est une puissance non triviale. La boucle suivante `for` récupère la liste de ces indices.

```
>>> Puissances(1000)
[4, 8, 9, 16, 25, 27, 32, 36, 49, 64, 81, 100, 121, 125, 128, 144, 169,
 196, 216, 225, 243, 256, 289, 324, 343, 361, 400, 441, 484, 512, 529,
 576, 625, 676, 729, 784, 841, 900, 961, 1000]
```

Question 7

On utilise évidemment le programme fournit pour avoir la liste des puissances.

```
def PuissancesConsécutives(M) :
    P = Puissances(M)
    L = []
    for i in range(len(P) - 1) :
        if P[i + 1] == P[i] + 1 :
            L.append( (P[i], P[i + 1]) )
    return L
```

```
>>> PuissancesConsécutives(10 ** 8)
[(8, 9)]
```

Question 8

Rien de neuf par rapport au programme précédent.

```
def PuissancesDistantes(d, M) :
    P = Puissances(M)
    L = []
    for i in range(len(P) - 1) :
        if P[i + 1] == P[i] + d :
            L.append( (P[i], P[i + 1]) )
    return L
```

```
>>> PuissancesDistantes(2, 10 ** 8)
[(25, 27)]
>>> PuissancesDistantes(3, 10 ** 8)
[(125, 128)]
>>> PuissancesDistantes(4, 10 ** 8)
[(4, 8), (32, 36), (121, 125)]
```

Question 9

On écrit un programme qui calcule la première somme en prenant toutes les puissances inférieures ou égales à M.

```
def PremièreSomme(M) :
    P = Puissances(M)
    s = 0
    for n in P :
        s += 1 / n
    return s
```

Puis un autre qui calcule la deuxième somme.

```
def DeuxièmeSomme(M) :
    P = Puissances(M)
    s = 0
    for n in P :
        s += 1 / (n - 1)
    return s
```

Et on s'aperçoit que c'est le même programme, donc plutôt que d'écrire *deux* programmes, on en écrit un seul qui calcule une somme quelconque.

```
def Somme(f, M) :
    P = Puissances(M)
    s = 0
    for n in P :
        s += f(n)
    return s
```

Maintenant on teste ce programme avec la fonction $f(n) = 1/n$ et la fonction $f(n) = 1/(n-1)$. On utilise le mot-clé `lambda` pour définir une fonction « à la volée ».

```
>>> Somme(lambda n : 1 / n, 10 ** 8)
0.874361963380861
```

```
>>> Somme(lambda n : 1 / (n - 1), 10 ** 8)
0.9998975949755857
```