Sans complexe (Denise Vella-Chemla, 31 mars 2024)

Pour ceux qui ne se risquent pas trop dans le plan complexe, voici des résultats de calculs étonnants, qui montrent la ressemblance numérique qu'il peut y avoir entre certaines sommes d'inverses de racines d'entiers (ici des racines carrées, cubiques, quartiques (ou bicarrées ou quatrièmes), quintiques (cinquièmes)), et le produit d'une seule de telles racines (la dernière[1]) par une fraction rationnelle (en fait, l'inverse d'une fraction rationnelle plutôt). Voyons le programme, c'est plus simple, on comprendra rapidement le but visé :

```
import math
from math import sqrt

print('racine carree')
nmax = 10000000
somme = 1
for k in range(2,nmax+1):
      somme = somme+pow(k,-0.5)
print(somme)
print(2*pow(nmax,0.5))-2
print('')
print('cubique')
somme = 1
for k in range(2,nmax+1):
      somme = somme+pow(k,-1/3)
print(somme)
print(3*pow(nmax,2/3)/2-3/2)
print('')
print('quatrieme')
somme = 1
for k in range(2,nmax+1):
      somme = somme+pow(k,-1/4)
print(somme)
print(4*pow(nmax,3/4)/3-4/3)
print('')
print('cinquieme')
somme = 1
for k in range(2,nmax+1):
      somme = somme+pow(k,-1/5)
print(somme)
print(5*pow(nmax,4/5)/4)-5/4
```

Le résultat de ce programme tout simple est :

[1]En fait, il s'agit de calculer la limite d'une somme, mais comme l'infini n'existe pas en informatique, on va moins loin.

```
racine carree
6323.095123940201
6322.555320336759

racine cubique
69622.86146473711
69622.33250419164

racine quatrieme
237103.1169515448
237102.5880051897

racine cinquieme
497633.2491763715
497632.71319187194
```

On pourrait résumer ces résultats par cette formule (testée pour $2 \leq k \leq 5$) :

$$\sum_{n=1}^{N} n^{-\frac{1}{k}} = \frac{k}{k-1} N^{\frac{k-1}{k}} - \frac{k}{k-1}$$

On teste l'inversion de réels par le programme :

```
print('racine sqrt(2)-ieme')
nmax = 10000000
somme = 1
for k in range(2,nmax+1):
      somme = somme+pow(k,-1/sqrt(2))
print(somme)
print(sqrt(2)*pow(nmax,(sqrt(2)-1)/sqrt(2))/(sqrt(2)-1)-(sqrt(2)/(sqrt(2-1))))
print('')
print('racine sqrt(3)-ieme')
nmax = 10000000
somme = 1
for k in range(2,nmax+1):
      somme = somme+pow(k,-1/sqrt(3))
print(somme)
print(sqrt(3)*pow(nmax,(sqrt(3)-1)/sqrt(3))/(sqrt(3)-1)-(sqrt(3)/(sqrt(3)-1)))
print('')
print('racine sqrt(5)-ieme')
nmax = 10000000
somme = 1
for k in range(2,nmax+1):
      somme = somme+pow(k,-1/sqrt(5))
print(somme)
print(sqrt(5)*pow(nmax,(sqrt(5)-1)/sqrt(5))/(sqrt(5)-1)-(sqrt(5)/(sqrt(5)-1)))
print('')
print('racine sqrt(11)-ieme')
nmax = 10000000
somme = 1
for k in range(2,nmax+1):
      somme = somme+pow(k,-1/sqrt(11))
print(somme)
print(sqrt(11)*pow(nmax,(sqrt(11)-1)/sqrt(11))/(sqrt(11)-1)-(sqrt(11)/(sqrt(11)-1)))
```

Ça marche...

```
racine sqrt(2)-ieme
380.4451536262492
379.8896659382487

racine sqrt(3)-ieme
2148.818632882152
2148.2729838521122

racine sqrt(5)-ieme
13393.873727589036
13393.337806659993

racine sqrt(11)-ieme
110983.31556131091
110982.78755376296
```

On s'enhardit dans le programme suivant, en prenant l'inverse d'un complexe choisi au hasard :

```
print('racine sqrt(11)-ieme')
nmax = 10000000
somme = 1
for k in range(2,nmax+1):
     somme = somme+pow(k,-1/(sqrt(11)+sqrt(3)*1j))
print(somme)
print((sqrt(11)+sqrt(3)*1j)*pow(nmax,((sqrt(11)-1+sqrt(3)*1j)
          /(sqrt(11)+sqrt(3)*1j)))/(sqrt(11)-1+sqrt(3)*1j)
          -(sqrt(11)+sqrt(3)*1j)/(sqrt(11)-1+sqrt(3)*1j))
```

Et sans surprise :

```
racine sqrt(11)+sqrt(3)i -ieme
(-73744.01084409936+274363.98023297265j)
(-73744.52540501424+274363.9800489613j)
```

La formule fonctionne à nouveau. Il semblerait qu'on ait toujours, si l'on respecte la contrainte ci-dessous, que k soit entier, réel ou complexe (et à une erreur minime près à préciser) :

$$\sum_{n=1}^{N} n^{-\frac{1}{k}} = \frac{k}{k-1} N^{\frac{k-1}{k}} - \frac{k}{k-1}.$$

La contrainte est qu'il faut que k soit supérieur à 1, c'est-à-dire que si on prend $k = 1/4$ pour en fait, par le 1/1/4 retrouver 4 comme puissance, la formule (le test par programme de la formule) ne fonctionne plus.