Après tout, l'idée initiale d'Euler..., Denise Vella-Chemla, novembre 2024

On a trouvé, par hasard pur, des formules pour la somme des inverses des racines carrées, la somme des inverses des racines cubiques, la somme des inverses des racines quatrièmes, la somme des inverses des racines cinquièmes, la somme des inverses des racines sixièmes. On se dit qu'après tout, c'était un peu le but d'Euler, quand il inventa la fonction zeta (appelée maintenant fonction zeta de Riemann) et qu'il lia sa somme sur tous les entiers et son produit sur les seuls nombres premiers, selon ce qu'on appelle je crois une formule du produit eulérien : qu'on sache calculer aisément de telles sommes, pour des nombres entiers, plutôt que pour des nombres complexes, comme le fit si brillamment Riemann.

Voici les formules qu'on a trouvées :

$$\sum_{k=1}^{n} \frac{1}{\sqrt{k}} \simeq 2\sqrt{n}$$

$$\sum_{k=1}^{n} \frac{1}{\sqrt[3]{k}} \simeq \frac{3}{2} \left(\sqrt[3]{n}\right)^2$$

$$\sum_{k=1}^{n} \frac{1}{\sqrt[4]{k}} \simeq \frac{4}{3} \left(\sqrt[4]{n}\right)^3$$

$$\sum_{k=1}^{n} \frac{1}{\sqrt[5]{k}} \simeq \frac{5}{4} \left(\sqrt[5]{n}\right)^4$$

et donc il serait possible que $\displaystyle\sum_{k=1}^{n} \frac{1}{\sqrt[p]{k}} \simeq \frac{p}{p-1} \left(\sqrt[p]{n}\right)^{p-1}$

On fournit ci-dessous le programme python qui code les formules et son résultat sur quelques exemples. C'est suffisamment (très) proche pour qu'on pense que les formules (à prouver) sont correctes.

```
import math
from math import sqrt,log,floor

somme = 1
nmax = 5900000
for n in range(1,nmax):
      somme = somme+(1/(n**(1/2)))
print(somme)
print(nmax)
print((2)*((nmax**(1/2))))
print('')
```

```
somme = 1
nmax = 30000000
for n in range(1,nmax):
      somme = somme+(1/(n**(1/3)))
print(somme)
print(nmax)
print((3/2)*((nmax**(1/3))**2))
print('')
somme = 1
nmax = 9000000
for n in range(1,nmax):
      somme = somme+(1/(n**(1/4)))
print(somme)
print(nmax)
print((4/3)*((nmax**(1/4))**3))
print('')
somme = 1
nmax = 30000000
for n in range(1,nmax):
      somme = somme+(1/(n**(1/5)))
print(somme)
print(nmax)
print((5/4)*((nmax**(1/5))**4))
```

Voici son résultat :

```
4857.522560240212
5900000
4857.983120596447

144823.43272145392
30000000
144823.40769084435

219089.20059498394
9000000
219089.02300206647

1198414.644493366
30000000
1198414.3943927297
```

Cela fait vraiment plaisir !

On parvient à vérifier notre formule à partir d'une autre formule trouvée sur la toile, mais bien plus complexe à calculer : voici la copie d'écran qu'on implémente en python

To answer a question asked in private, considering the more general case of

$$S_n = \sum_{k=1}^{n} \frac{1}{\sqrt[a]{k}} = H_n^{\left(\frac{1}{a}\right)}$$

using asymptotics

$$n^{\frac{1}{a}} \left(S_n - \zeta\left(\frac{1}{a}\right) \right) =$$

$$\frac{an}{a-1} + \frac{1}{2} - \frac{1}{12an} + \frac{1}{720} \sum_{k=1}^{\infty} (-1)^{k+1} \frac{\prod_{m=1}^{2k} (am+1)}{\alpha_k \, (an)^{2k+1}}$$

where the α_k form the sequence

$$\{1, 42, 1680, 66528, 1816214400, 103783680, 14820309504000, \cdots\}$$

Multiplied by 720, you will find them in sequence $A060055$ in $OEIS$.

par le programme suivant (les dénominateurs croissant dangereusement, on se contente des 7 premiers nombres en lien avec les nombres de Bernoulli) :

```
import scipy
from scipy.special import zeta

def alpha(k):
      if k == 1:
            return(1)
      else:
            if k == 2:
                  return(42)
            else:
                  if k == 3:
                        return(1680)
                  else:
                        if k == 4:
                              return(66528)
                        else:
                              if k == 5:
                                    return(1816214400)
                              else:
                                    if k == 6:
                                          return(103783680)
                                    else:
                                          if k == 7:
                                                return(14820309504000)
```

```
def calcprod(a,k):
     prod = 1
     for m in range(1,2*k+1):
          prod = prod*(a*m+1)
     return(prod)

def sommeinfinie(n,a):
     somme = 0
     for k in range(1,8):
          somme = somme+((-1)**(k+1))*calcprod(a,k)/(alpha(k)*((a*n)**(2*k+1)))
     return(somme)

def bla(n,a):
     e1=a*n/(a-1)
     e2=0.5
     e3 = 1/(12*a*n)
     e4 = (1/720)*sommeinfinie(n,a)
     return(e1+e2-e3+e4)

n = 5900000
a = 2
print('resultat 1 = ',(bla(n,a)+n**(1/a)*zeta(1/a))/(n**(1/a)))
n=30000000
a = 3
print('resultat 2 = ',(bla(n,a)+n**(1/a)*zeta(1/a))/(n**(1/a)))
n=9000000
a = 4
print('resultat 2 = ',(bla(n,a)+n**(1/a)*zeta(1/a))/(n**(1/a)))
n=30000000
a = 5
print('resultat 2 = ',(bla(n,a)+n**(1/a)*zeta(1/a))/(n**(1/a)))
```

On obtient des résultats quasi-identiques (à 1 près car on avait initialisé notre somme à 1) à ceux qui avaient été obtenus par nos formules ultra-simples.

```
C:\Users\Denise_Vella\Desktop>python mercr
4857.522560240212
5900000
4857.983120596447

144823.43272145392
30000000
144823.40769084435

219089.20059498394
9000000
219089.02300206645

1198414.644493366
30000000
1198414.3943927293

C:\Users\Denise_Vella\Desktop>
```

```
C:\Users\Denise_Vella\Desktop>python essai-capture.py
resultat 1 =  4856.522971934378
resultat 2 =  144822.43593974513
resultat 2 =  219088.21885237042
resultat 2 =  1198413.6764506618

C:\Users\Denise_Vella\Desktop>
C:\Users\Denise_Vella\Desktop>
C:\Users\Denise_Vella\Desktop>
C:\Users\Denise_Vella\Desktop>
C:\Users\Denise_Vella\Desktop>
C:\Users\Denise_Vella\Desktop>
C:\Users\Denise_Vella\Desktop>
C:\Users\Denise_Vella\Desktop>
C:\Users\Denise_Vella\Desktop>
C:\Users\Denise_Vella\Desktop>
C:\Users\Denise_Vella\Desktop>
C:\Users\Denise_Vella\Desktop>
```