Décomposer en facteurs premiers

Le problème

On veut décomposer l'entier naturel N en un produit de nombres premiers (voir en page 14).
Le principe est simple : on essaie de le diviser successivement, jusqu'à « épuisement » (jusqu'à ce que le produit des facteurs mis en évidence soit N), par les nombres premiers pris dans l'ordre croissant.
Par exemple, pour 252, on obtient : $252 = 2^2 \times 3^2 \times 7$.

252	2
126	2
63	3
21	3
7	7
1	

Complément culturel

Les nombres premiers forment l'alphabet des entiers naturels. Ils sont aux entiers ce que les lettres sont aux mots. De même qu'un mot est une suite de lettres, un entier naturel (à partir de 2) est un produit de nombres premiers.

Les programmes

Python

Le programme « dec » teste la divisibilité par 2 du nombre initial, puis de son éventuel quotient par 2, et ainsi de suite jusqu'à obtenir un nombre impair. Il recommence ensuite avec 3, puis avec tous les impairs, de deux en deux (en dehors de 2, tous les nombres premiers sont impairs). Seuls les nombres premiers fourniront une possibilité de réponse positive au test de divisibilité, les facteurs des nombres non premiers ayant déjà été « purgés ». Chaque fois que *d* divise l'entier N, il est ajouté à la liste et N est remplacé par N / *d*. À la fin du programme, la liste des diviseurs premiers est affichée.
Langage : le reste de la division euclidienne de N par *d* est obtenu en tapant N%*d* ; pour obtenir le quotient, on écrit int(N/*d*).

le programme

```
>>> def dec(N):
        Résultat=[]
        d=2
        while N%d==0:
                Résultat.append(d)
                q=int(N/d)
                N=q
        d=3
        while d<=N:
                while N%d==0:
                        Résultat.append(d)
                        q=int(N/d)
                        N=q
                d=d+2
        return Résultat
```

le résultat

```
>>> dec(1000)
[2, 2, 2, 5, 5, 5]
>>> dec(28121964)
[2, 2, 3, 13, 71, 2539]
```

Scratch

```
quand [drapeau] pressé
répéter (longueur de [List]) fois
    supprimer (1) de [List]
demander [N=?] et attendre
à [N] attribuer (réponse)
à [d] attribuer (2)
à [q] attribuer ((N) - ((N) mod (2))) / (2)
répéter jusqu'à ((N) mod (d)) > (0)
    ajouter (d) à [List]
    à [N] attribuer (q)
    à [q] attribuer ((N) - ((N) mod (2))) / (2)
à [d] attribuer (3)
à [q] attribuer ((N) - ((N) mod (d))) / (d)
répéter jusqu'à (d) > (N)
    répéter jusqu'à ((N) mod (d)) > (0)
        ajouter (d) à [List]
        à [q] attribuer ((N) - ((N) mod (d))) / (d)
        à [N] attribuer (q)
    changer [d] par (2)
dire (List) pendant (2) secondes
```

AlgoBox

```
VARIABLES
  N EST_DU_TYPE NOMBRE
  d EST_DU_TYPE NOMBRE
  q EST_DU_TYPE NOMBRE
  List EST_DU_TYPE LISTE
  i EST_DU_TYPE NOMBRE
  k EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
  LIRE N
  d PREND_LA_VALEUR 2
  i PREND_LA_VALEUR 1
  TANT_QUE (N-d*floor(N/d)==0) FAIRE
    DEBUT_TANT_QUE
    List[i] PREND_LA_VALEUR d
    i PREND_LA_VALEUR i+1
    q PREND_LA_VALEUR floor(N/d)
    N PREND_LA_VALEUR q
    FIN_TANT_QUE
  d PREND_LA_VALEUR 3
  TANT_QUE (d<=N) FAIRE
    DEBUT_TANT_QUE
    TANT_QUE (N-d*floor(N/d)==0) FAIRE
      DEBUT_TANT_QUE
      List[i] PREND_LA_VALEUR d
      i PREND_LA_VALEUR i+1
      q PREND_LA_VALEUR floor(N/d)
      N PREND_LA_VALEUR q
      FIN_TANT_QUE
    d PREND_LA_VALEUR d+2
    FIN_TANT_QUE
  k PREND_LA_VALEUR 1
  POUR k ALLANT_DE 1 A i-1
    DEBUT_POUR
    AFFICHER List[k]
    PAUSE
    FIN_POUR
FIN_ALGORITHME
```

Prolongements

Si la décomposition en facteurs premiers permet d'écrire N sous la forme N = $p_1^{\alpha_1}.p_2^{\alpha_2}.....p_k^{\alpha_k}$ où $p_1, p_2,..., p_k$ sont des nombres premiers et $\alpha_1, \alpha_2,..., \alpha_k$ sont des entiers naturels non nuls, **le nombre de diviseurs positifs de N est $(\alpha_1 + 1)(\alpha_2 + 1)...(\alpha_k + 1)$.**

le programme

```
>>> def nombrededivisuers(Liste):
        i=0
        P=1
        Q=2
        while i<(len(Liste)-1):
            if Liste[i]==Liste[i+1]:
                Q=Q+1
            else:
                P=P*Q
                Q=2
            i=i+1
        P=P*Q
        return P
```

Le programme ci-contre donne le nombre de diviseurs positifs à partir de la décomposition en produit de facteurs premiers écrite *in extenso* (sans utiliser d'exposants).

On pourra utiliser la composée « nombredediviseurs(dec(N)) », où « dec » est définie à la page précédente ; nous obtiendrons ainsi directement le nombre de diviseurs positifs de N.