Sep 2022

Alternate classification of integers.

Possibly the way to classify the integer numbers shown here has already been studied by other authors. If that is the case, the scarce popularity of the method suggests that it has not proven of great utility in finding new relations of interest between big numbers.
Just in case the method has not yet been studied by other authors it is explained here though the development of its implications is postponed.

The “whole” numbers 1,2,3,4,5,6,7,8... that are normally ordered according to their “size” (amount of unities) can also be *ordered according to their factorization in prime numbers* for example as follows:

Every whole (or integer) number can be assigned to a family and a subfamily of numbers, as follows:

A family *faml* of whole numbers is the set of all those numbers for which the sum of the “orders” of their prime factors is equal to *faml*.
The *order of a prime* number is the ordinal that corresponds to that prime in the (infinite) sequence of increasing prime numbers.
Since
the 1st prime number is the integer 2
the 2nd prime number is the integer 3,
... and so on with the next prime numbers 3, 5, 7, 11, 17, 19,...
it will be said that the “order” (or ordinal) of the prime 2 is 1 (since the integer 2 is the 1st in the list of prime numbers)
the “order” (or ordinal) of the prime 3 is 2 (since the integer 3 is the 2nd in the list of prime numbers)
the “order” (or ordinal) of the prime 5 is 3 (since the integer 5 is the 3rd in the list of prime numbers) ...

(Note: Wolfram’s Mathematica implements the function `Prime[j]` to give the j_th prime number. For example if Mathematica is asked to give the 22nd prime number (i.e. the prime of order 22) writing `Prime[22]` as the input, will give the integer 79 as the output).

Consider then an integer m whose prime factors are for example: once the 5th prime (i.e. Prime[5]), twice the 3rd prime (Prime[3]), once the 2nd prime (Prime[2]), four times the 1st prime (Prime[1])
i.e. the number m can be written as
m = Prime[5] * Prime[3] * Prime[3] * Prime[2] * Prime[1] * Prime[1] * Prime[1] * Prime[1] = or as =
= Prime[5] * Prime[3]^2 * Prime[2] * Prime[1]^4
(incidentally it will be: $m = 11 * 5^2 * 3 * 2^4 = 13200$)
then, according to the above definition of *family of a whole number* (as the sum of the “orders” of

its prime factors) this number m will belong to the family
faml = 5+3+3 +2+1+1+1+1 = 17
Calling simply `p[j]` what Mathematica calls `Prime[j]`, other examples of integers belonging to the family faml =17 are:
p[17]
p[12] * p[5]
$p[4]^2$ * p[3] * $p[2]^3$

Calling now "*number of factors of an integer*" the total number of primes (equal or different) that factorize the number (i.e. adding a unity to the count as many times as a prime factor appears) then
the 1st number of the example (i.e. p[17]) has only one factor (the 17th prime = 59)
the 2nd number has two factors
the 3rd number has six factors (i.e. p[4], p[4], p[3], p[2], p[2], p[2])
then

a <u>subfamily</u> *sfam* of an integer will be defined as *the number of its prime factors*, where again, any prime adds to such number of factors every time that it appears.
Therefore, in the above examples,
the number p[17] belongs to the subfamily sfam=1 (of the family faml=17)
the number p[12] * p[5] belongs to the subfamily sfam=2 (of the family faml=17)
the number $p[4]^2$ * p[3] * $p[2]^3$ belongs to the subfamily sfam=6 (of the family faml=17)

Given a family, its subfamilies will be ordered placing first those with a smaller number of prime factors.
Consider for example the family faml=6. Its component members (numbers) can be written and ordered as follows:
subfamily 1:
p[6]
subfamily 2:
p[5]*p[1]
p[4]*p[2]
p[3]*p[3]
subfamily 3:
p[4]*p[1]*p[1]
p[3]*p[2]*p[1]
p[2]*p[2]*p[2]
subfamily 4:
p[3]*p[1]*p[1]*p[1]
p[2]*p[2]*p[1]*p[1]
subfamily 5:
p[2]*p[1]*p[1]*p[1]*p[1]
subfamily 6:
p[1]*p[1]*p[1]*p[1]*p[1]*p[1]
where, as said above, `p[j]` is the name given here to what Mathematica calls `Prime[j]`, i.e. the jth prime.

Notice that in the list of numbers composing the family, the subfamily sfam=1 (i.e. of only one prime factor) appears first.
Next appears the subfamily sfam=2 (i.e. whose component numbers have two prime factors) that in this example has three members (p[5]*p[1], p[4]*p[2] and p[3]*p[3])
Next appears the subfamily sfam=3 (i.e. whose component numbers have 3 prime factors) that in this case has three members.
And so on...

The component members (numbers) within each subfamily will be ordered putting first the members (of the subfamily) whose prime factors are of the higher order.
In case that two or more members of the subfamily have the same prime as the higher of its factors then a comparison is made of the immediately less higher factor of each member, and so on. For example, the subfamily sfam=7 of the family faml =15 ordered according to such criterion (of the "higher orders of their prime factors") is:
p9 p1 p1 p1 p1 p1 p1
p8 p2 p1 p1 p1 p1 p1
p7 p3 p1 p1 p1 p1 p1
p7 p2 p2 p1 p1 p1 p1
p6 p4 p1 p1 p1 p1 p1
p6 p3 p2 p1 p1 p1 p1
p6 p2 p2 p2 p1 p1 p1
p5 p5 p1 p1 p1 p1 p1
p5 p4 p2 p1 p1 p1 p1
p5 p3 p3 p1 p1 p1 p1
p5 p3 p2 p2 p1 p1 p1
p5 p2 p2 p2 p2 p1 p1
p4 p4 p3 p1 p1 p1 p1
p4 p4 p2 p2 p1 p1 p1
p4 p3 p3 p2 p1 p1 p1
p4 p3 p2 p2 p2 p1 p1
p4 p2 p2 p2 p2 p2 p1
p3 p3 p3 p3 p1 p1 p1
p3 p3 p3 p2 p2 p1 p1
p3 p3 p2 p2 p2 p2 p1
p3 p2 p2 p2 p2 p2 p2
where, for further economy of characters a prime of order j has been written as pj (instead of as p[j])

Notice that each new family introduces a new prime, i.e. a prime that has not yet appeared in the factorization of the numbers of the earlier families.
More precisely, the prime of order n (i.e p[n]) appears for the first time in the family faml=n.
For example,
the first member of the family faml=4 is p[4] (i.e. the 4th prime p[4]=7)
the first member of the family faml=5 is p[5] (i.e. the 5th prime p[5]=11)

...
the first member of the family faml=n is p[n] (i.e. the n_th prime)

The algorithm/program written below finds the component members (numbers) of any given subfamily of any given family and presents the results ordered according to the criteria chosen above.

The purpose of this alternate ordering (or classification) of the integers is to try to find relations between the members of the different families that might give clues to facilitate the factorization of big integer numbers.

For further clarification, the first 6 families of whole numbers are now written with their full members:
Family faml=1 (only one member):
p[1]

Family faml=2:
p[2]
p[1]*p[1]

Family faml=3:
p[3]
p[2] p[1]
p[1] p[1] p[1]

Family faml=4:
p[4]
p[3] p[1]
p[2] p[2]
p[2] p[1] p[1]
p[1] p[1] p[1] p[1]

Family faml=5:
p[5]
p[4] p[1]
p[3] p[2]
p[3] p[1] p[1]
p[2] p[2] p[1]
p[2] p[1] p[1] p[1]
p[1] p[1] p[1] p[1] p[1]

Family faml=6:
p[6]
p[5] p[1]
p[4] p[2]

p[3] p[3]
p[4] p[1] p[1]
p[3] p[2] p[1]
p[2] p[2] p[2]
p[3] p[1] p[1] p[1]
p[2] p[2] p[1] p[1]
p[2] p[1] p[1] p[1] p[1]
p[1] p[1] p[1] p[1] p[1] p[1]

For the purpose of writing the algorithm (see below) that deduces the members of a given subfamily of a given family of integers,
the prime factors of a member of the subfamily will be considered ordered from left to right in a corresponding position labeled respectively
pos[1], pos[2], pos[3],...,pos[sfam].
In what follows the total number *sfam* of prime factors (and hence positions) of the members of the subfamily will be abbreviated calling it *sf* (instead of *sfam*)

If the factor in the i_th position of a member of the subfamily is the j_th prime number, i.e. the prime of order j, then the element pos[i] will be assigned the value j (i.e. the algorithm will assign `pos[i]=j`). For example:
the integer m = Prime[3]*Prime[2]*Prime[2]*Prime[1] = 90 belongs to the subfamily whose members have sf=4 prime factors
then the positions of the factors of this number will be assigned the respective values:
pos[1]=3, pos[2]=2, pos[3]=2, pos[4]=1

The algorithm that deduces the members of the subfamily *sfam* (i.e. whose numbers have *sfam* prime factors) of the family *faml* (i.e. for which the sum of the "orders" of the prime factors of its members is equal to *faml*) proceeds as follows:
- The first member of the subfamily is straightforwardly given by:
Prime[faml-(sfam-1)] * Prime[1] * Prime[1] *...
(and so on multiplying by Prime[1] factors until the total of factors, included the first, is equal to *sfam* . Of course if the subfamily is *sfam=1,* i.e. has only one prime factor, then the first and only member of the subfamily will be Prime[faml-(sfam-1)] = Prime[faml])
Therefore the algorith assigns to the positions of those *sfam* factors the corresponding values (order of the prime factor):
pos[1]=faml-(sfam-1), pos[2]=1, pos[3]=1,...,pos[sfam]=1

- The next members of the subfamily are constructed as follows:
Let `Nprevious` be the member of the subfamily just deduced by the algorithm in its earlier pass.
Nprevious will be an integer that should be imagined to be written as:
`Nprevious = Prime[a]*Prime[b]*Prime[c] *...sfam factors...* Prime[x]`

where a ≥ b ≥ c...≥ x
so that the position assignments of its factors are pos[1]=a, pos[2]=b, pos[3]=c, ... , pos[sfam]=x
where a ≥ b ≥ c...≥ x
In that number Nprevious, the algorithm looks, starting from the last position factor pos[sfam] and advancing towards the first positions, a position pos[j] whose content is bigger than 2 (i.e. such that pos[j]>2) . As soon as such pos[j] is found, the algorithm constructs a reference number Nref as follows:
Suppose for example that j=3 (i.e. that the prime factor in the 3rd position of Nprevious, that has been supposed to be the c_th prime, is such that c>2) . Then the reference number that the algorithm constructs will be:
`Nref = Prime[a]*Prime[b]*Prime[c-1] * Prime[c-1]*...* Prime[c-1]`
i.e. the algorithm assigns the following new set of position-functions
pos[1]=a, pos[2]=b, pos[3]=c-1, ... , pos[sfam]=c-1
and it is checked:
If $\sum_{i=1}^{sfam} pos[i] = faml$ then a new member of the subfamily has been found and is added to the pertinent array Members[faml,sfam,elem] of outputs
but if $\sum_{i=1}^{sfam} pos[i] > faml$ then the content (order of prime) of the pos[k] container is decreased by a unity (doing pos[k]=pos[k]-1) *starting with the higher position* k=sfam and repeating (such unity decrease) for that k_th position while pos[k]>1
and then jumping to the preceding position pos[k-1] and repeating the unity decrease (now pos[k-1]=pos[k-1]-1) for that (k-1)_th position while pos[k-1]>1
and so on jumping to the preceding positions
but evaluating $\sum_{i=1}^{sfam} pos[i]$ after each unity decrease and
stopping (the unity decrease) as soon as $\sum_{i=1}^{sfam} pos[i] = faml$ which would mean that a new member of the subfamily has been found (which is then assigned to the pertinent element of the array Members[faml,sfam,elem]).

This new member just found is considered the new *Nprevious* number from which the procedure described above is then repeated.

The lines that follow (in light pink background color) are those of an algorithm that constructs the members of the subfamily *sfam* of the family *faml*.
The input of both faml and sfam must be written by hand in the first line of the program.
The output is introduced/saved in a list/array *Members[faml,sfam,elem]* of 3 variables (family, subfamily, #element)
in which
the 1st part of the member shows the prime factors (e.g. $p[1]^2 p[2] p[7]$) of a specific member *elem* of the subfamily *sfam* of the family *faml*
the 2nd part of the member shows the result quantity of multiplying those prime factors (e.g. 204 which is the result of $Prime[1]^2 Prime[2] Prime[7]$)

```
faml = 11; sfam = 4;
elem = 0;
Members[faml, sfam, elem] = {};
pos[1] = faml - (sfam - 1); suma = pos[1];
For[i = 2, i < (sfam + 1), i++, pos[i] = 1; suma = suma + pos[i]]
elem = elem + 1;
mFactors = Product[p[pos[i]], {i, 1, sfam}];
m = Product[Prime[pos[i]], {i, 1, sfam}];
Members[faml, sfam, elem] = {mFactors, m};

fin = 0;
While[fin == 0,

 jj = sfam;
 While[((Sum[pos[i], {i, 1, jj - 1}] + (sfam - jj + 1) * (pos[jj] - 1)) < faml ||
     pos[jj] < 3) && jj > 0, jj = jj - 1];

If[jj < 1, fin = 1];

 If[fin == 0,
  pos[jj] = pos[jj] - 1;
  For[i = jj, i < (sfam + 1), i++, pos[i] = pos[jj]];
  suma = 0; For[i = 1, i < (sfam + 1), i++, suma = suma + pos[i]];

  For[jj = sfam, jj > 1, jj = jj - 1,
   While[(pos[jj] > 1) && (suma > faml), pos[jj] = pos[jj] - 1; suma = suma - 1]];

  elem = elem + 1;
  mFactors = Product[p[pos[i]], {i, 1, sfam}];
  m = Product[Prime[pos[i]], {i, 1, sfam}];
  Members[faml, sfam, elem] = {mFactors, m};

 ];
]
```

```
elem
Members[faml, sfam, 2]
```

11

$\{p[1]^2\, p[2]\, p[7],\ 204\}$

The following algorithm (based in the earlier one and very similar to it) deduces the t*otal number of elements (members) of a given family faml.*

```
faml = 18;
TotElementsFam = 0;

For[sfam = 1 , sfam < (faml + 1), sfam++,
 Members[faml, sfam, elem] = {}; elem = 0;

 pos[1] = faml - (sfam - 1); suma = pos[1];
 For[i = 2, i < (sfam + 1), i++, pos[i] = 1; suma = suma + pos[i]];

 elem = elem + 1;
 mFactors = Product[p[pos[i]], {i, 1, sfam}];
 m = Product[Prime[pos[i]], {i, 1, sfam}];
 Members[faml, sfam, elem] = {mFactors, m};
 TotElementsFam = TotElementsFam + 1;

 fin = 0;
 While[fin == 0,

  jj = sfam;
  While[((Sum[pos[i], {i, 1, jj - 1}] + (sfam - jj + 1) * (pos[jj] - 1)) < faml ||
      pos[jj] < 3) && jj > 0, jj = jj - 1];

If[jj < 1, fin = 1];

  If[fin == 0,
   pos[jj] = pos[jj] - 1;
   For[i = jj, i < (sfam + 1), i++, pos[i] = pos[jj]];

   suma = 0; For[i = 1, i < (sfam + 1), i++, suma = suma + pos[i]];


   For[jj = sfam, jj > 1,  jj = jj - 1,
    While[(pos[jj] > 1)  &&  (suma > faml),  pos[jj] = pos[jj] - 1;  suma = suma - 1]];

   elem = elem + 1;
   mFactors = Product[p[pos[i]], {i, 1, sfam}];
   m = Product[Prime[pos[i]], {i, 1, sfam}];
   Members[faml, sfam, elem] = {mFactors, m};
   TotElementsFam = TotElementsFam + 1;
  ];
 ]
]
```

```
TotElementsFam
```

```
385
```

Other information about the members of the family chosen in the input are also saved by the algorithm.
For example, asking for the 5th member of the subfamily sfam=11 gives

```
Members[faml, 11, 5]
```

$\{p[1]^9\, p[4]\, p[5],\ 39\,424\}$