Ordered and Unordered Factorizations of Integers

Arnold Knopfmacher
Michael Mays

We study the number of ways of writing a positive integer *n* as a product of integer factors greater than one. We survey methods from the literature for enumerating and also generating lists of such factorizations for a given number *n*. In addition, we consider the same questions with respect to factorizations that satisfy constraints, such as having all factors distinct. We implement all these methods in *Mathematica* and compare the speeds of various approaches to generating these factorizations in practice.

■ Introduction

To study the number of ways of writing a positive integer n as a product of integer factors greater than one, there are two basic cases to consider. First, we can regard rearrangements of factors as different. In the case of $n = 12$, this gives the following eight ordered factorizations.

```
{{2, 2, 3}, {2, 3, 2}, {2, 6}, {3, 2, 2}, {3, 4}, {4, 3}, {6, 2}, {12}}
```

Alternatively we can ignore the order of the factors, which then gives the following four unordered factorizations.

```
{{3, 2, 2}, {4, 3}, {6, 2}, {12}}
```

These two functions, which we denote by $H(n)$ and $P(n)$, respectively, can be considered multiplicative analogs of compositions and partitions of integers. A composition is an ordered set of positive integers that sum to n. For example, we have eight compositions of $n = 4$, namely {{4}, {3,1}, {1,3}, {2,2}, {2,2,1}, {1,2,1}, {1,1,2}, {1,1,1,1}}. In general the number of compositions of n, $C(n)$, is equal to 2^{n-1}. A partition is a set that sums to n in which order is disregarded. There are five partitions of four. To count them we can use the function `PartitionsP`, and to list them we can use `Partitions` from the *Combinatorica* package.

```
In[1]:= << DiscreteMath`Combinatorica`

In[2]:= Partitions[4]

Out[2]= {{4}, {3, 1}, {2, 2}, {2, 1, 1}, {1, 1, 1, 1}}
```



```
In[3]:= PartitionsP[Range[10]]

Out[3]= {1, 2, 3, 5, 7, 11, 15, 22, 30, 42}
```

In fact the number of compositions and partitions arise as special cases of our factorization functions, in the sense that if p is a prime number, then $H(p^r) = C(r) = 2^{r-1}$ and $P(p^r) =$ `PartitionsP`$[r]$. For general numbers n with prime factorizations $n = p_1^{r_1} p_2^{r_2} \ldots p_k^{r_k}$, the enumeration of $H(n)$ and $P(n)$ is more complicated, as we shall discuss later.

We will also discuss factorizations into distinct parts. In the ordered case, $H_d(12) = 5$ since we have the factorizations {{2,6}, {3,4}, {4,3}, {6,2}, {12}}. In the unordered case, $P_d(12) = 3$ since there are three cases, {{4,3}, {6,2}, {12}}. If p is a prime number, then as special cases we have $H_d(p^r) = C_d(r)$ and $P(p^r) =$ `PartitionsQ`$[r]$, where $C_d(r)$ denotes the number of compositions of r into distinct parts (see Richmond and Knopfmacher [1]), and `PartitionsQ` is the *Mathematica* function for counting partitions into distinct parts.

```
In[4]:= PartitionsQ[Range[10]]

Out[4]= {1, 1, 2, 2, 3, 4, 5, 6, 8, 10}
```

The enumeration and generation of integer partitions and compositions are problems discussed in standard books on combinatorics. A definitive reference is Andrews [2]. However, the corresponding problems for ordered and unordered factorizations have not until now received a comprehensive treatment.

■ Ordered Factorizations

For historical reasons, we will discuss formulas to enumerate factorizations before we discuss methods to generate the corresponding factorizations. In addition, some of the recursive methods to generate factorizations are extensions of the corresponding recursions to enumerate factorizations.

□ Recursions for Enumerating Ordered Factorizations

We begin with two recurrence formulas given by Hille [3]. The first element of an ordered factorization of $n > 1$ can be any number d such that d divides n. By appending to d all possible ordered factorizations of n/d, we obtain the recursion $H(1) = 1$; $H(n) = \sum_{d|n} H(d)$ for $n \geq 2$. We implement this as follows.

```
In[5]:= H1[1] := 1;
        H1[n_] := H1[n] = Total[H1 /@ Drop[Divisors[n], -1]]

In[7]:= Table[H1[n], {n, 1, 12}]

Out[7]= {1, 1, 1, 2, 1, 3, 1, 4, 2, 3, 1, 8}
```

Using the Möbius inversion formula, Hille also derived a second recursion,

$$H(n) = 2\left(\sum_{p_i} H\left(\frac{n}{p_i}\right) - \sum_{p_i,\, p_j} H\left(\frac{n}{p_i\, p_j}\right) + \cdots + (-1)^{k-1} H\left(\frac{n}{p_1\, p_2 \cdots p_k}\right)\right),$$

where $n = p_1^{r_1}\, p_1^{r_1}\, p_2^{r_2} \cdots p_k^{r_k}$, which holds for $n \geq 2$.

This finds the list of distinct prime factors of n.

```
In[8]:= PrimeFactorList[n_] := First /@ FactorInteger[n]
```

This recursion requires the initial value $\frac{1}{2}$.

```
In[9]:= H2[1] = 1 / 2;
```

The recursion can be rendered elegantly as a oneliner.

```
In[10]:= H2[n_] := H2[n] = -2 Total[(-1)^(Length[#]) H2[n / Times @@ #] & /@
           Rest[Subsets[PrimeFactorList[n]]]]
In[11]:= Table[H2[n], {n, 2, 12}]
Out[11]= {1, 1, 2, 1, 3, 1, 4, 2, 3, 1, 8}
```

□ MacMahon's Formula for *H*

MacMahon [4] derived an explicit formula for the value of $H(n)$ as a double sum over a product. Given $n = p_1^{r_1}\, p_1^{r_1}\, p_2^{r_2} \cdots p_k^{r_k}$,

$$H(n) = \sum_{j=1}^{q} \sum_{i=0}^{j-1} (-1)^i \binom{j}{i} \prod_{b=1}^{k} \binom{r_b + j - i - 1}{r_b},$$

where $q = \sum_{i=1}^{k} r_i$, is the sum of the prime exponents of n. We program this by

```
In[12]:= H3[n_] :=
           Sum[Sum[(-1)^i Binomial[j, i] Apply[Times,
             Binomial[# + j - i - 1, #] & /@ #] &[Last /@ FactorInteger[n]],
             {i, 0, j}], {j, 1, Total[##]}]
In[13]:= H3[2^5 3^4 5] // Timing
Out[13]= {0. Second, 102576}
```

□ A Recursion for Factorizations with Distinct Parts

Warlimont [5] derived a Dirichlet series generating function for the function $H_d(k, n)$, which denotes the number of ordered factorizations of n into k distinct parts. Although Warlimont was only interested in this for asymptotic purposes, his generating function can be used to derive the following recurrence:

$$H_d(k+1, n) = k! \sum_{j=0}^{k} \frac{(-1)^j}{(k-1-j)!} \sum H_d\left(k - j,\, \frac{n}{d}\right),$$

where the inside sum is taken over all d such that $d \mid n$ and for $d \geq 2$, d is a $(j+1)$-st power.

There does not seem to be a simple combinatorial interpretation for this formula. We implement this starting with the appropriate boundary conditions.

```
In[14]:= Hd[1, n_] := 1;
         Hd[k_, 1] := 0;
         Hd[0, n_] := 0

In[17]:= Hd[0, 1] := 1;
         Hd[1, 1] := 0;
```

When n is a prime number we have the following.

```
In[19]:= Hd[k_, n_?PrimeQ] := KroneckerDelta[k, 1]
```

Also observe that the number of parts k must satisfy $2^k \leq n$.

```
In[20]:= Hd[k_, n_ /; 2^k > n] := 0
```

Now we implement the general formula.

```
In[21]:= Hd[k_, n_] :=
           Hd[k, n] = (k - 1)! Sum[((-1)^j / (k - 1 - j)!) Total[Hd[k - 1 - j, n / #] & /@
               Select[Rest[Divisors[n]], IntegerQ[#^(1/(j+1))] &]], {j, 0, k-1}]
```

```
In[22]:= Hd[#, 24] & /@ Range[0, 4]

Out[22]= {0, 1, 6, 6, 0}
```

We verify these counts by using the function `DistinctOrderedFactorizations`, which is defined in the next subsection.

```
In[23]:= DistinctOrderedFactorizations[24]

Out[23]= {{2, 3, 4}, {2, 4, 3}, {2, 12}, {3, 2, 4}, {3, 4, 2}, {3, 8},
          {4, 2, 3}, {4, 3, 2}, {4, 6}, {6, 4}, {8, 3}, {12, 2}, {24}}
```

To obtain the total number of distinct ordered factorizations, we must sum $H_d[k, n]$ over all permissible values of the number of parts k.

```
In[24]:= Hd[n_] := Total[Hd[#, n] & /@ Range[Floor[Log[2, n]]]]

In[25]:= Hd[36]

Out[25]= 13
```

In practice this recursion is slow. A faster counting method is discussed in a later section.

□ Generating Ordered Factorizations

The first recursion for $H(n)$ suggests a natural recursive approach to generate all the ordered factorizations of n.

We append to the first factor d in a factorization of $n > 1$, all possible ordered factorizations of $\frac{n}{d}$, where d can be any divisor of n.

```
In[26]:= OrderedFactorizations[1] = {{}};

In[27]:= OrderedFactorizations[n_?PrimeQ] := {{n}}

In[28]:= OrderedFactorizations[n_] :=
          OrderedFactorizations[n] = Flatten[Function[d, Prepend[#, d] & /@
               OrderedFactorizations[n / d]] /@ Rest[Divisors[n]], 1]

In[29]:= OrderedFactorizations[24]

Out[29]= {{2, 2, 2, 3}, {2, 2, 3, 2}, {2, 2, 6}, {2, 3, 2, 2}, {2, 3, 4}, {2, 4, 3},
          {2, 6, 2}, {2, 12}, {3, 2, 2, 2}, {3, 2, 4}, {3, 4, 2}, {3, 8},
          {4, 2, 3}, {4, 3, 2}, {4, 6}, {6, 2, 2}, {6, 4}, {8, 3}, {12, 2}, {24}}
```

One way to list all the ordered factorizations with distinct parts is to simply select these from the list of all ordered factorizations.

```
In[30]:= DistinctOrderedFactorizations[n_] :=
          Select[OrderedFactorizations[n], Unequal @@ # &]

In[31]:= DistinctOrderedFactorizations[24]

Out[31]= {{2, 3, 4}, {2, 4, 3}, {2, 12}, {3, 2, 4}, {3, 4, 2}, {3, 8},
          {4, 2, 3}, {4, 3, 2}, {4, 6}, {6, 4}, {8, 3}, {12, 2}, {24}}
```

However, there are faster methods for doing this, which we discuss in a later section.

■ Unordered Factorizations

There do not appear to be any explicit formulas for $P(n)$ in the literature. Also, the recurrence relations that are known tend to lack simple combinatorial interpretations.

□ A Product Recursion

Harris and Subbarao [6] give the following product recursion for $P(n)$. For any positive integer a, let $d_i = a^{1/i}$ if a is an ith power and $d_i = 1$ otherwise. Let $\overline{d} = \prod_{i=1}^{\infty} d_i$. This gives $\prod_{d|n} \overline{d}^{\,p(n/d)} = n^{p(n)}$. To make use of this, we take logs of the recurrence. First, we define the $\overline{d}$ values in terms of a given positive integer a. One approach is to use `Product`.

```
In[32]:= d1[a_] := Product[If[IntegerQ[c = a^(1/i)], c, 1], {i, 1, Ceiling[Log[2, a]]}]
```

Alternatively, here is a more elegant construction.

```
In[33]:= d2[a_] := Apply[Times, Select[a^(1/Range[Ceiling[Log[2, a]]]), IntegerQ]]
```

Then we can define `P2`.

```
In[34]:= P2[1] = 1;

In[35]:= P2[n_] := P2[n] = Round[1 / (Log[n])
           Simplify[Total[ Log[d2[#]] P2[n / #] & /@ Rest[Divisors[n]]]]]
```

The use of `Simplify` in `P2` is not required, but speeds up the overall computation. `Round` produces the correct integer value for `P2` much faster than by using additional simplification methods to remove the logarithms.

```
In[36]:= P2[3^3 2^3] // Timing
Out[36]= {0.078 Second, 31}
```

□ A Recursion for Unordered Factorizations with Largest Part *m*

Let $g(m, n)$ denote the number of unordered factorizations of n with largest part at most m. Hughes and Shallit [7] gave the recursion

$$g(m, n) = \sum_{\substack{d \mid n \\ d \le m}} g\left(d, \frac{n}{d}\right).$$

This particular recursion is easy to explain: Let $n = a_1 a_2 \ldots a_n$ be an unordered factorization of n with parts at most m and parts arranged in decreasing order, so that the largest part is a_1. The number of ways to choose $a_2, \ldots, a_k$ is then $g(a_1, n/a_1)$. For a_1 we can choose any divisor d of n such that $d \le m$. Summing over all such d gives the result. We implement this as follows.

```
In[37]:= g[m_, 1] := 1;
         g[1, n_] := 0;
         g[1, 1] = 1;
         g[m_, n_] :=
          g[m, n] = Total[g[#, n / #] & /@ Select[Divisors[n], # ≤ m &]]
```

All unordered factorizations are counted by $g(n, n)$.

```
In[41]:= P1[n_] := g[n, n]

In[42]:= P1 /@ Range[12]
Out[42]= {1, 1, 1, 2, 1, 2, 1, 3, 2, 2, 1, 4}
```

This gives a much faster recursion than `P2`.

```
In[43]:= P1[3^6 2^4] // Timing
Out[43]= {0.016 Second, 323}

In[44]:= P2[3^6 2^4] // Timing
Out[44]= {0.422 Second, 323}
```

Canfield, Erdös, and Pomerance [8] remarked in their paper that it is not particularly easy to compute $P(n)$. They mention that even showing $P(1800) = 137$ takes some work. Their approach was based on a tree traversal algorithm. With our recursion this computation presents no problem.

```
In[45]:= P1[1800] // Timing
Out[45]= {0.015 Second, 137}
```

□ Generating Unordered Factorizations

One inefficient approach is simply to sort the larger list of ordered factorizations and remove duplicates. However, a better approach is to use the Hughes–Shallit idea to recursively build up a list of ordered factorizations with largest part at most m.

```
In[46]:= UnorderedFactorizations[m_, 1] = {{}};
         UnorderedFactorizations[1, n_] = {{}};

In[48]:= UnorderedFactorizations[m_, n_ /; PrimeQ[n]] := If[m < n, {}, {{n}}]

In[49]:= UnorderedFactorizations[m_, n_] :=
          UnorderedFactorizations[m, n] = Flatten[
            Function[d, Prepend[#, d] & /@ UnorderedFactorizations[d, n / d]] /@
             Rest[Select[Divisors[n], # ≤ m &]], 1]

In[50]:= UnorderedFactorizations[n_] := UnorderedFactorizations[n, n]
```

Now we test this out.

```
In[51]:= UnorderedFactorizations[24]

Out[51]= {{3, 2, 2, 2}, {4, 3, 2}, {6, 2, 2}, {6, 4}, {8, 3}, {12, 2}, {24}}

In[52]:= Length[UnorderedFactorizations[3^4 2^4]] == P1[3^4 2^4]

Out[52]= True
```

□ Unordered Factorizations with Distinct Parts

A modification of Hughes–Shallit reasoning gives a recursion for unordered factorizations with distinct parts and largest part at most m. We merely observe that the part added to d should have largest part less than or equal to $d - 1$:

$$\mathrm{gd}(m, n) = \sum_{\substack{d \mid n \\ d \le m}} \mathrm{gd}\left(d - 1, \frac{n}{d}\right).$$

We program this with necessary boundary conditions to start the recursion.

```
In[53]:= gd[m_, 1] := 1;
         gd[1, n_] := 0;
         gd[1, 1] = 1;
         gd[0, n_] := 0;
         gd[m_, n_] :=
          gd[m, n] = Total[gd[# - 1, n / #] & /@ Select[Divisors[n], # ≤ m &]]

In[58]:= P_d[n_] := gd[n, n]
```

Here is an example.

```
In[59]:= P_d[3^4 2^2 5 7] // Timing

Out[59]= {0.016 Second, 253}
```

Generating Unordered Factorizations with Distinct Parts

Again we can simply select the permissible factorizations from the larger list of all unordered factorizations, but a better approach is to recursively generate them using the idea for the recursion for gd(m, n).

```
In[60]:= DistinctUnorderedFactorizations[m_, 1] = {{}};
         DistinctUnorderedFactorizations[1, n_] = {};
         DistinctUnorderedFactorizations[0, n_] = {};

In[63]:= DistinctUnorderedFactorizations[m_, n_ /; PrimeQ[n]] :=
          If[m < n, {}, {{n}}]

In[64]:= DistinctUnorderedFactorizations[m_, n_] :=
          DistinctUnorderedFactorizations[m, n] = Flatten[Function[d,
              Prepend[#, d] & /@ DistinctUnorderedFactorizations[d - 1, n / d]] /@
             Rest[Select[Divisors[n], # ≤ m &]], 1]

In[65]:= DistinctUnorderedFactorizations[n_] :=
          DistinctUnorderedFactorizations[n, n]
```

Here are the distinct unordered factorizations of 24.

```
In[66]:= DistinctUnorderedFactorizations[24]

Out[66]= {{4, 3, 2}, {6, 4}, {8, 3}, {12, 2}, {24}}
```

Here are the nondistinct factorizations.

```
In[67]:= Complement[UnorderedFactorizations[24], %]

Out[67]= {{6, 2, 2}, {3, 2, 2, 2}}
```

We check that we are generating the right number of cases.

```
In[68]:= P_d[3^4 2^4] == Length[DistinctUnorderedFactorizations[3^4 2^4]]

Out[68]= True
```

□ Faster Generation of Ordered Factorization Lists

The lists of unordered factorizations constructed earlier lead to a much faster way of generating the corresponding lists of ordered factorizations. We merely observe that all ordered cases arise as permutations of unordered cases. This gives a different ordering of the factorizations to the earlier method. However, the lists are easily checked to be the same.

```
In[69]:= OrderedFactorizations2[n_] :=
          Flatten[Permutations /@ UnorderedFactorizations[n], 1]

In[70]:= OrderedFactorizations2[24]

Out[70]= {{3, 2, 2, 2}, {2, 3, 2, 2}, {2, 2, 3, 2},
          {2, 2, 2, 3}, {4, 3, 2}, {4, 2, 3}, {3, 4, 2}, {3, 2, 4},
          {2, 4, 3}, {2, 3, 4}, {6, 2, 2}, {2, 6, 2}, {2, 2, 6},
          {6, 4}, {4, 6}, {8, 3}, {3, 8}, {12, 2}, {2, 12}, {24}}
```

```
In[71]:= Sort[%] == Sort[OrderedFactorizations[24]]

Out[71]= True
```

The new approach gives a faster method for generating the factorizations list.

```
In[72]:= Length[OrderedFactorizations2[30^3]] // Timing

Out[72]= {0.078 Second, 64324}

In[73]:= Length[OrderedFactorizations[30^3]] // Timing

Out[73]= {0.172 Second, 64324}
```

Factorizations with Distinct Parts

Again all ordered cases arise as permutations of unordered distinct cases. This also leads to a large speedup in computation time.

```
In[74]:= DistinctOrderedFactorizationsNew[n_] :=
          Flatten[Permutations /@ DistinctUnorderedFactorizations[n], 1]

In[75]:= Sort[DistinctOrderedFactorizationsNew[60]] ==
          Sort[DistinctOrderedFactorizations[60]]

Out[75]= True
```

□ Faster Count for Ordered Factorizations with Distinct Parts

Let $P_d(k, n)$ denote the number of unordered factorizations into k distinct parts. We observe that $H_d(n) = \sum_k k!\, P_d(k, n)$. We do not have a formula for $P_d(k, n)$, but we can compute its values by sorting the lists of distinct unordered factorizations according to length. Although we generate (generally short) lists of factorizations as part of the counting process, this turns out to be the fastest method we have found to compute the usually large values of H_d.

First, we sort our lists of factorizations according to length. For example, here $n = 36$.

```
In[76]:= lis =
          Sort[DistinctUnorderedFactorizations[36], Length[#1] < Length[#2] &]

Out[76]= {{36}, {18, 2}, {12, 3}, {9, 4}, {6, 3, 2}}
```

Then, we split up the different classes with respect to length, count how many of length k occur, multiply by $k!$, and sum.

```
In[77]:= Split[lis, Length[#1] == Length[#2] &]

Out[77]= {{{36}}, {{18, 2}, {12, 3}, {9, 4}}, {{6, 3, 2}}}

In[78]:= a = Length /@ %;
         Plus @@ (a Range[Length[a]]!)

Out[79]= 13
```

This agrees with our previous computation. We put this method together as a oneliner.

```
In[80]:= Ĥd[n_] := Plus @@ (# Range[Length[#]]!) &[
           Length /@ Split[Sort[DistinctUnorderedFactorizations[n],
              Length[#1] < Length[#2] &], Length[#1] == Length[#2] &]]

In[81]:= Ĥd[30^3] // Timing

Out[81]= {0.047 Second, 11655}
```

In general, $\hat{H}_d$ seems to provide a considerable speedup over H_d.

```
In[82]:= Hd[30^3] // Timing

Out[82]= {4.375 Second, 11655}
```

■ Highly Factorable Numbers

Now that we have implemented various methods to count ordered and unordered factorizations, we will put them to use to produce lists of numbers that have a greater number of factorizations than any smaller positive integer.

We say that a natural number n is highly factorable with respect to the function f, if $f(m) < f(n)$ for all m, $1 \le m < n$. In [8], Canfield, Erdös, and Pomerance computed a list of highly factorable numbers with respect to the function P. Since the functions P and H depend on the prime exponents but not the primes themselves, it is clear that a highly factorable number must be of the form $n = p_1^{r_1}\, p_2^{r_2} \cdots p_k^{r_k}$ with $r_1 \ge r_2 \ge \cdots r_k \ge 1$ and where p_i denotes the ith prime number. We use `Partitions` to generate a list of acceptable exponents and then define a function `ExponentsToNumber` to produce a natural number n using the exponents from the partition and the corresponding first few primes.

```
In[83]:= ExponentsToNumber[exponentList_] := Times @@@
           (Prime[Range[#]] & /@ (Length /@ exponentList)^exponentList)

In[84]:= Partitions[4]

Out[84]= {{4}, {3, 1}, {2, 2}, {2, 1, 1}, {1, 1, 1, 1}}

In[85]:= ExponentsToNumber[%]

Out[85]= {16, 24, 36, 60, 210}
```

To produce all numbers of this form less than a given value *bound*, we must consider all partitions of numbers 1 to `Log[2,bound]`, as the smallest number arising from a partition of n is 2^n.

```
In[86]:= lst[bound_] := Flatten[Table[Partitions[k], {k, Log[2, bound]}], 1];
```

For example, let us find all the highly factorable numbers less than 1000.

```
In[87]:= a = Sort[ExponentsToNumber[lst[1000]]];
```

Now eliminate numbers greater than *bound* in our list, compute the value of the P function for each number in the list, and eliminate the values that are not highly factorable using a replacement rule.

```
In[88]:= b = Select[a, # < 1000 &];

In[89]:= valsP = {#, P1[#]} & /@ b;

In[90]:= LargePlist = valsP //.
           {x___, {y1_, y2_}, {z1_, z2_}, w___} :> {x, {y1, y2}, w} /; y2 ≥ z2;
```

Now we use `GridBox` to display our table of highly factorable P numbers.

```
In[91]:= StyleBox[FormBox[GridBox[Prepend[LargePlist, {"n", "P(n)"}],
           GridFrame → 3, ColumnLines → 1, RowLines → {2, 1}, ColumnSpacings →
            0.5, ColumnAlignments → "."], "TraditionalForm"],
          Background → GrayLevel[0.85], FontSize → 8] // DisplayForm
```

Out[91]//DisplayForm=

n	$P(n)$
2	1
4	2
8	3
12	4
16	5
24	7
36	9
48	12
72	16
96	19
120	21
144	29
192	30
216	31
240	38
288	47
360	52
432	57
480	64
576	77
720	98
960	105

Replacing $P1$ by $H3$ leads to the following list of highly factorable numbers with respect to H.

```
In[92]:= Off[General::spell1]
```

```
In[93]:= LargeHlist = {#, H3[#]} & /@ b //.
           {x___, {y1_, y2_}, {z1_, z2_}, w___} :> {x, {y1, y2}, w} /; y2 ≥ z2;
         StyleBox[FormBox[GridBox[Prepend[LargeHlist, {"n", "H(n)"}],
             GridFrame → 3, ColumnLines → 1, RowLines → {2, 1}, ColumnSpacings →
              0.5, ColumnAlignments → "."], "TraditionalForm"],
           Background → GrayLevel[0.85], FontSize → 8] // DisplayForm
```

Out[94]//DisplayForm=

n	$H(n)$
2	1
4	2
6	3
8	4
12	8
24	20
36	26
48	48
72	76
96	112
120	132
144	208
192	256
240	368
288	544
360	604
432	768
480	976
576	1376
720	1888
864	2208
960	2496

Erdös, Canfield, and Pomerance were able to compute a table of all highly factorable numbers less than 10^9 with respect to P in their paper [8]. The approach just used gives a much faster method to find the 118 highly factorable numbers less than 10^9 with respect to P, as well as the 124 highly factorable numbers less than 10^9 with respect to H.

□ Numbers Highly Factorable with Respect to Both *P* and *H*

There appear to be many numbers common to both of the displayed lists. To find these common numbers, join the two lists and extract the first elements (the common values of n). Find the pairs by using `Split`. Then extract the common numbers as the first element of the sublists of length 2.

```
In[95]:= commonPositions = First /@ (Join[LargePlist, LargeHlist] // Sort)

Out[95]= {2, 2, 4, 4, 6, 8, 8, 12, 12, 16, 24, 24, 36, 36, 48, 48, 72, 72,
          96, 96, 120, 120, 144, 144, 192, 192, 216, 240, 240, 288, 288,
          360, 360, 432, 432, 480, 480, 576, 576, 720, 720, 864, 960, 960}
```

```
In[96]:= First /@ Select[Split[commonPositions, #1 == #2 &], Length[#] == 2 &]

Out[96]= {2, 4, 8, 12, 24, 36, 48, 72, 96, 120,
  144, 192, 240, 288, 360, 432, 480, 576, 720, 960}
```

So up to 1000, most highly factorable numbers appear in both lists. However, common numbers seem to become less frequent as we increase our bound. For example, we find that there are 55 common highly factorable numbers less than 10^9, the largest of these being 43545600.

■ Factorizations with Relatively Prime Parts

In this final section we investigate an interesting class of restricted factorizations, namely the class of factorizations in which the factors must be relatively prime to each other. Clearly this is a stronger restriction than requiring distinct factors. The asymptotic growth of such factorizations has been studied by Warlimont [5]. We note that in the special case of *squarefree* integers, all factorizations are necessarily relatively prime and distinct. Thus, for squarefree integers the values of the three functions that count unrestricted or distinct or relatively prime factorizations, all coincide for the ordered and unordered cases respectively.

We will discuss and compare several different approaches to generate the corresponding lists of factorizations.

□ Ordered Factorizations with Relatively Prime Parts

In the ordered case, factorizations of $n = p_1^{r_1} p_2^{r_2} \ldots p_k^{r_k}$ into relatively prime parts have a natural correspondence to ordered partitions of a set with the k elements $p_1^{r_1}, p_2^{r_2}, \ldots, p_k^{r_k}$. The exponential generating function for the number of ordered set partitions is $\frac{1}{2-E^x}$. (These and further results on set partitions that follow can be found in Wilf's book [9].) From this exponential generating function we can easily compute the first few values.

```
In[97]:= CoefficientList[Series[1/(2 - E^x), {x, 0, 10}], x] Range[0, 10] !

Out[97]= {1, 1, 3, 13, 75, 541, 4683, 47293, 545835, 7087261, 102247563}
```

Here is another expression for the number of ordered set partitions (sometimes called ordered Bell numbers) due to Carlitz [10].

```
In[98]:= OrderedBell[0] := 1;

In[99]:= OrderedBell[r_] := Sum_{k=0}^{r} Sum_{j=0}^{k} (-1)^(k-j) Binomial[k, j] j^r

In[100]:= Table[OrderedBell[r], {r, 0, 10}]

Out[100]= {1, 1, 3, 13, 75, 541, 4683, 47293, 545835, 7087261, 102247563}
```

In addition there is also this pretty expression as an infinite sum.

```
In[101]:= OrderedBell2[n_] := 1/2 Sum[m^n 2^-m, {m, 0, Infinity}]
```

Using either of these it is easy to count the number of ordered relatively prime factorizations.

```
In[102]:= H_r[n_] := OrderedBell[Length[FactorInteger[n]]]

In[103]:= H_r[30^4]

Out[103]= 13
```

□ Unordered Factorizations with Relatively Prime Parts

In the unordered case, factorizations of $n = p_1^{r_1} p_2^{r_2} \ldots p_k^{r_k}$ into relatively prime parts now have a natural correspondence to unordered partitions of a set with the k elements $p_1^{r_1}, p_2^{r_2}, \ldots, p_k^{r_k}$. The exponential generating function for the number of ordered set partitions is E^{E^x-1}. We use this to easily compute the first few values.

```
In[104]:= CoefficientList[Series[E^(E^x - 1), {x, 0, 10}], x] Range[0, 10]!

Out[104]= {1, 1, 2, 5, 15, 52, 203, 877, 4140, 21147, 115975}
```

Again there is a pretty expression for the number of partitions of a set (or Bell numbers) as an infinite sum. Now it is easy to count the number of unordered relatively prime factorizations.

```
In[105]:= Off[General::spell]

In[106]:= Bell[n_] := 1/E Sum[m^n / m!, {m, 0, Infinity}]

In[107]:= P_r[n_] := Bell[Length[FactorInteger[n]]]

In[108]:= P_r[30^4]

Out[108]= 5
```

□ Generating Lists of Relatively Prime Factorizations

There are several different approaches that can be used to find the desired lists.

Unordered Relatively Prime Factorizations by Selection

To generate a list of relatively prime factorizations, we need only search among the distinct unordered factorizations and pick out those with relatively prime parts.

Method 1. By Factorization

Our first approach to selecting the relatively prime cases is to factor the numbers in each distinct unordered factorization, and after flattening and sorting, see if this matches the factorization of n.

```
In[109]:= Sort[Flatten[FactorInteger[{15, 4}], 1]] == FactorInteger[60]

Out[109]= True

In[110]:= Sort[Flatten[FactorInteger[{6, 10}], 1]] == FactorInteger[60]

Out[110]= False
```

This shows that {15, 4} is admissible as a relatively prime factorization of 60 but {6, 10} is not. We illustrate the method in the case $n = 60$.

```
In[111]:= DistinctUnorderedFactorizations[60]

Out[111]= {{5, 4, 3}, {6, 5, 2}, {10, 3, 2},
           {10, 6}, {12, 5}, {15, 4}, {20, 3}, {30, 2}, {60}}

In[112]:= a = Flatten[#, 1] & /@ FactorInteger /@ %;
```

We determine the positions of the cases with admissible factorizations. Then we read off these factorizations from the list.

```
In[113]:= Position[Sort /@ a, FactorInteger[60]]

Out[113]= {{1}, {5}, {6}, {7}, {9}}

In[114]:= DistinctUnorderedFactorizations[60][[Flatten[%]]]

Out[114]= {{5, 4, 3}, {12, 5}, {15, 4}, {20, 3}, {60}}
```

We put this together into one function.

```
In[115]:= UnorderedRelativelyPrime1[n_] := DistinctUnorderedFactorizations[
            n][[Flatten[Position[Sort /@ (Flatten[#, 1] & /@ (FactorInteger /@
                  DistinctUnorderedFactorizations[n])), FactorInteger[n]]]]]

In[116]:= UnorderedRelativelyPrime1[240]

Out[116]= {{16, 5, 3}, {16, 15}, {48, 5}, {80, 3}, {240}}
```

Method 2. By Greatest Common Divisors

A second approach is to check that the factors are relatively prime directly, by finding the greatest common divisors of every pair of factors. However, Bressoud and Wagon [11] give a much more efficient way to test a long list for relative primality of all pairs.

```
In[117]:= RelativelyPrimeList[lst_] := LCM @@ lst == Times @@ lst
```

We rewrite our function accordingly.

```
In[118]:= UnorderedRelativelyPrime2[n_] := DistinctUnorderedFactorizations[
            n][[Flatten[Position[RelativelyPrimeList /@
               DistinctUnorderedFactorizations[n], True]]]]

In[119]:= Length[UnorderedRelativelyPrime2[240 49 11 13]] // Timing

Out[119]= {0.438 Second, 203}

In[120]:= Length[UnorderedRelativelyPrime1[240 49 11 13]] // Timing

Out[120]= {0.219 Second, 203}
```

Method 3: By Generating Set Partitions

The idea here is to generate the partitions of a set with k elements and then replace these elements with the appropriate prime powers to obtain a list of relatively prime factorizations. We can recursively compute the lists of set partitions by noting that the partitions of a set with n elements can be created by appending the *singleton* $\{n\}$ to each list of partitions of $n-1$ elements, or by appending the *element* n to each partition of $n-1$ elements. An implementation of this recursion using `ReplaceList` is given by Dickau [12] as follows.

```
In[121]:= BellRule1[n_] := {S__} → {S, {n}};
          BellRule2[n_] := {b___, {S__}, a___} → {b, {S, n}, a};

In[123]:= BellLists[1] = {{{1}}};

In[124]:= BellLists[n_Integer?Positive] := BellLists[n] =
            Flatten[
             Map[ReplaceList[#, {BellRule1[n], BellRule2[n]}] &,
              BellLists[n - 1]], 1]
```

Here is an example using Dickau's function.

```
In[125]:= BellLists[3]

Out[125]= {{{1}, {2}, {3}}, {{1, 3}, {2}},
           {{1}, {2, 3}}, {{1, 2}, {3}}, {{1, 2, 3}}}
```

Now if, for example, $n = 180 = 4 * 9 * 5$, we must substitute $1 \to 4$, $2 \to 9$, $3 \to 5$ to obtain the factors in the relatively prime factorizations. Finally, we multiply the factors together to produce the desired result.

```
In[126]:= BellLists[3] /. MapThread[Rule, {Range[3], {4, 9, 5}}]

Out[126]= {{{4}, {9}, {5}}, {{4, 5}, {9}},
           {{4}, {9, 5}}, {{4, 9}, {5}}, {{4, 9, 5}}}

In[127]:= Apply[Times, #, 1] & /@ %

Out[127]= {{4, 9, 5}, {20, 9}, {4, 45}, {36, 5}, {180}}
```

Putting this together we have a nice oneliner. This approach is unsurprisingly much faster than the previous ones!

```
In[128]:= UnorderedRelativelyPrime3[n_] := With[{f = FactorInteger[n]},
            Apply[Times, #, 1] & /@ (BellLists[Length[f]] /.
               MapThread[Rule, {Range[Length[f]], Power @@@ f}])]

In[129]:= Length[UnorderedRelativelyPrime3[240 49 11 13]] // Timing

Out[129]= {0. Second, 203}
```

Ordered Relatively Prime Factorizations

Whichever of the three approaches is used to generate the unordered cases, we need only permute the elements of each such unordered factorization to produce the lists of the ordered ones.

```
In[130]:= OrderedRelativelyPrime[n_] :=
           Flatten[Permutations /@ UnorderedRelativelyPrime1[n], 1]
```

■ Conclusion

We have studied methods to enumerate and to construct ordered and unordered factorizations of integers, subject to various constraints on the parts. Recursive descriptions of these objects, together with *Mathematica*'s functional programming techniques, were used to implement these algorithms efficiently.

■ Acknowledgments

The authors thank the referee for his careful reading of the manuscript and for enhancements to both the efficiency and elegance of the coding. In addition, the authors thank Glenn Scholebo, the compositor of *The Mathematica Journal*, for improving the typesetting of the paper.

■ References

[1] B. Richmond and A. Knopfmacher, "Compositions with Distinct Parts," *Aequationes Mathematicae*, **49**, 1995 pp. 86–97.

[2] G. E. Andrews, *The Theory of Partitions*, Encyclopedia of Mathematics and Its Applications 2, New York: Addison-Wesley, 1976.

[3] E. Hille, "A Problem in 'Factorisatio Numerorum,'" *Acta Arithmetica*, **2**, 1936 pp. 134–144.

[4] P. A. MacMahon, "Memoir on the Theory of the Compositions of Numbers," *Philosophical Transactions of the Royal Society of London (A)*, **184**, 1893 pp. 835–901.

[5] R. Warlimont, "Factorisatio Numerorum with Constraints," *Journal of Number Theory*, **45**, 1993 pp. 186–199.

[6] V. C. Harris and M. V. Subbarao, "On Product Partitions of Integers," *Canadian Mathematical Bulletin*, **34**(4), 1991 pp. 474–479.

[7] J. F. Hughes and J. O. Shallit, "On the Number of Multiplicative Partitions," *American Mathematical Monthly*, **90**(7), 1983 pp. 468–471.

[8] E. R. Canfield, P. Erdös, and C. Pomerance, "On a Problem of Oppenheim Concerning 'Factorisatio Numerorum,'" *Journal of Number Theory*, **17**, 1983 pp. 1–28.

[9] H. S. Wilf, *generatingfunctionology*, 2nd ed., New York: Academic Press, 1994.

[10] L. Carlitz, "Extended Bernoulli and Eulerian Numbers," *Duke Mathematical Journal*, **31**, 1964 pp. 667–689.

[11] D. M. Bressoud and S. Wagon, *A Course in Computational Number Theory*, New York: Springer-Verlag, 2000.

[12] R. Dickau, "Visualizing Combinatorial Enumeration," *Mathematica in Education and Research*, **8**, 1999 pp. 11–18.

About the Authors

Arnold Knopfmacher is a Professor of Mathematics at the University of the Witwatersrand, Johannesburg, where he also obtained his Ph.D. degree. His main research interests are in enumerative combinatorics and elementary number theory. He is Director of The John Knopfmacher Centre for Applicable Analysis and Number Theory, which was established in 1992 by his late father, a distinguished number theorist. Prior to John's death in 1999, Arnold and John collaborated extensively producing over thirty joint papers together.

Michael Mays is a Professor of Mathematics at West Virginia University, where he does research in combinatorics and number theory and develops software and course materials for the Institute for Math Learning. He has maintained his collaboration with Arnold Knopfmacher with five visits over the last decade to The John Knopfmacher Centre for Applicable Analysis and Number Theory at the University of the Witwatersrand.

Arnold Knopfmacher
The John Knopfmacher Centre for Applicable Analysis and Number Theory
University of the Witwatersrand
Johannesburg 2050, South Africa
arnoldk@cam.wits.ac.za
www.wits.ac.za/science/number_theory/arnold.htm

Michael Mays
Department of Mathematics
West Virginia University
Morgantown, WV 26506-6310
mays@math.wvu.edu
www.math.wvu.edu/~mays