A NEW FUNCTION RELATED TO INTEGER PARTITIONS

Edoardo GUEGLIO
egueglio@gmail.com

It is possible to define a chain of recurrence relations starting from partition function involving another partition function whose parameters vary along with the recurrence relations

All this research has been done using Mathematica®, a symbolic computation environment that uses a programming language called Wolfram Language.

I start from the standard definition of partitions function which define for each natural number n all the different sums to natural number n. For example:

\[
\text{IntegerPartitions}[13] = \{\{13\}, \{12, 1\}, \{11, 2\}, \{11, 1, 1\}, \{10, 3\}, \{10, 2, 1\}, \{10, 1, 1, 1\}, \{9, 4\}, \{9, 3, 1\}, \{9, 2, 2\}, \{9, 2, 1, 1\}, \{9, 1, 1, 1, 1\}, \{8, 5\}, \{8, 4, 1\}, \{8, 3, 2\}, \{8, 3, 1, 1\}, \{8, 2, 2, 1\}, \{8, 2, 1, 1, 1\}, \{8, 1, 1, 1, 1, 1\}, \{7, 6\}, \{7, 5, 1\}, \{7, 4, 2\}, \{7, 4, 1, 1\}, \{7, 3, 3\}, \{7, 3, 2, 1\}, \{7, 3, 1, 1, 1\}, \{7, 2, 2, 2\}, \{7, 2, 2, 1, 1\}, \{7, 2, 1, 1, 1, 1\}, \{6, 6\}, \{6, 5, 2\}, \{6, 5, 1, 1\}, \{6, 4, 3\}, \{6, 4, 2, 1\}, \{6, 4, 1, 1, 1\}, \{6, 3, 3, 1\}, \{6, 3, 2, 2\}, \{6, 3, 2, 1, 1\}, \{6, 3, 1, 1, 1, 1\}, \{6, 2, 2, 2, 1\}, \{6, 2, 2, 1, 1, 1\}, \{6, 2, 1, 1, 1, 1, 1\}, \{5, 5\}, \{5, 5, 2\}, \{5, 5, 1, 1, 1\}, \{5, 4, 4\}, \{5, 4, 3\}, \{5, 4, 2\}, \{5, 4, 2, 1\}, \{5, 4, 1, 1, 1\}, \{5, 4, 1, 1, 1, 1\}, \{5, 3, 3\}, \{5, 3, 2, 2\}, \{5, 3, 2, 1, 1\}, \{5, 3, 2, 1, 1, 1\}, \{5, 3, 1, 1, 1, 1, 1\}, \{5, 2, 2, 2, 2\}, \{5, 2, 2, 2, 1, 1\}, \{5, 2, 2, 1, 1, 1, 1, 1\}, \{5, 2, 1, 1, 1, 1, 1, 1, 1\}, \{4, 4, 4, 1\}, \{4, 4, 3, 2\}, \{4, 4, 3, 1, 1\}, \{4, 4, 2, 2, 1\}, \{4, 4, 2, 1, 1, 1\}, \{4, 4, 1, 1, 1, 1, 1\}, \{4, 3, 3, 3\}, \{4, 3, 3, 2, 1\}, \{4, 3, 3, 1, 1, 1\}, \{4, 3, 2, 2, 2\}, \{4, 3, 2, 2, 1, 1\}, \{4, 3, 2, 1, 1, 1, 1\}, \{4, 3, 1, 1, 1, 1, 1, 1\}, \{4, 2, 2, 2, 2, 1\}, \{4, 2, 2, 2, 1, 1, 1\}, \{4, 2, 2, 1, 1, 1, 1, 1\}, \{4, 2, 1, 1, 1, 1, 1, 1, 1\}, \{4, 1, 1, 1, 1, 1, 1, 1, 1, 1\}, \{3, 3, 3, 3, 3\}, \{3, 3, 3, 3, 2, 1\}, \{3, 3, 3, 2, 2, 1\}, \{3, 3, 1, 1, 1, 1, 1, 1\}, \{3, 3, 1, 1, 1, 1, 1, 1, 1\}, \{3, 2, 2, 2, 2, 2, 1\}, \{3, 2, 2, 2, 2, 1, 1, 1\}, \{3, 2, 2, 1, 1, 1, 1, 1, 1\}, \{3, 2, 1, 1, 1, 1, 1, 1, 1, 1\}, \{3, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1\}, \{2, 2, 2, 2, 2, 2, 2, 1\}, \{2, 2, 2, 2, 2, 1, 1, 1, 1\}, \{2, 2, 2, 1, 1, 1, 1, 1, 1, 1\}, \{2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1\}, \{2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1\}, \{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1\}\}

As you can see what you get is the set of all the different sorted lists with items from greater to lower.

I define a new function with two parameters: n stands for the number to be decomposed and m stands for the fact that last item in each list MUST BE GREATER THAN m. When we have m=0, then this condition vanishes. A further implicit condition is that the first two items MUST BE EQUAL. It
clearly follows that each list has at least two items. Here is the definition with an example:

\begin{verbatim}
In[34]:= pfist2eqlastgtm[n_Integer, m_Integer] :=
Select[IntegerPartitions[n], Length[#] > 1 &
&& Last[#] == Last[#][[2]] &
&& Last[#] > m &]
pfist2eqlastgtm[13, 0]
pfist2eqlastgtm[13, 1]
pfist2eqlastgtm[13, 2]
\end{verbatim}

\begin{verbatim}
Out[32]=
\end{verbatim}

As you can see the lists in the set are not correctly sorted so I add a list sorting function and correct
the above expression:

\begin{verbatim}
partitions.nb

the above expression:

\begin{verbatim}
As you can see the lists in the set are not correctly sorted so I add a list sorting function and correct
\end{verbatim}

Now consider to modify \texttt{IntegerPartitions(n)} adding 1 to the first item of every list. Every new list is a
possible partition of 10 and every list is different by definition. So you can complement the set
\texttt{IntegerPartitions(10)} with \texttt{IntegerPartitions(9)} modified. This is resumed with the following expres-
sion:

\begin{verbatim}
In[34]:= Complement[IntegerPartitions[10],
Prepend[Rest[#], First[#] + 1] & /@ IntegerPartitions[9]]
\end{verbatim}

As you can see the lists in the set are not correctly sorted so I add a list sorting function and correct
the above expression:
In[35]:= lorderQ[1_List, n_Integer] := False

lorderQ[n_Integer, 1_List] := True

lorderQ[n1_Integer, n2_Integer] := n1 ≥ n2

lorderQ[l1_List, l2_List] := Module[{i},
  For[i = 1, i ≤ Length[l1] && i ≤ Length[l2], i++,
    If[l1[[i]] == l2[[i]],
      Continue[];
      Return[lorderQ[l1[[i]], l2[[i]]]]
    ];
  ];

If[Length[l1] ≥ Length[l2],
  Return[True],
  Return[False]
]

Sort[Complement[IntegerPartitions[10],
  Prepend[Rest[#], First[#] + 1] & /@ IntegerPartitions[9]], lorderQ]

{3, 3, 2, 1, 1}, {3, 3, 1, 1, 1, 1}, {2, 2, 2, 2, 2}, {2, 2, 2, 2, 1, 1},
{2, 2, 2, 1, 1, 1, 1}, {2, 2, 1, 1, 1, 1, 1, 1}, {1, 1, 1, 1, 1, 1, 1, 1, 1}}

The set of lists we have are the one and only with first two items identical and this is the only
ccondition imposed by pfirst2eqlastgtm(10,0):

In[40]:= pfirst2eqlastgtm[10, 0]

Out[40]= {{5, 5}, {4, 4, 2}, {4, 4, 1, 1}, {3, 3, 3, 1}, {3, 3, 2, 2},
{3, 3, 2, 1, 1}, {3, 3, 1, 1, 1, 1}, {2, 2, 2, 2, 2}, {2, 2, 2, 2, 1, 1},
{2, 2, 2, 1, 1, 1, 1}, {2, 2, 1, 1, 1, 1, 1, 1, 1}, {1, 1, 1, 1, 1, 1, 1, 1, 1, 1}}

The conclusions for n=10 can be easily extended to every n. As a further example for n=20:

In[41]:= Sort[Complement[IntegerPartitions[20], Prepend[Rest[#], First[#] + 1] & /@
  IntegerPartitions[19]], lorderQ] = pfirst2eqlastgtm[20, 0]

Out[41]= True

After this first integer relation to start the chain consider the relation between pfirst2eqlastgtm(n,0)
and pfirst2eqlastgtm(n,1). Consider the following example:
Now consider to modify `pfirst2eqlastgtm[14,0]` adding 1 as the last item of every list. Every new list is a possible partition of 15 with first two items identical and every list is different by definition. So you can complement the set `pfirst2eqlastgtm[15,0]` with `pfirst2eqlastgtm[14,0]` modified. This is resumed with the following expression:

```
In[44] := Sort[Complement[pfirst2eqlastgtm[15, 0],
  Append[#, 1] & /@ pfirst2eqlastgtm[14, 0]], lorderQ]
Out[44] :=
```

```
{{6, 6, 3}, {5, 5, 5}, {5, 5, 3, 2}, {4, 4, 4, 3},
  {4, 4, 3, 2, 2}, {3, 3, 3, 3, 3}, {3, 3, 3, 2, 2, 2}}
```

The set of lists we have are the one and only with first two items identical and last item greater than 1. This is the only condition imposed by `pfirst2eqlastgtm` (15, 1):

```
In[45] := pfirst2eqlastgtm[15, 1]
Out[45] :=
```

```
{{6, 6, 3}, {5, 5, 5}, {5, 5, 3, 2}, {4, 4, 4, 3},
  {4, 4, 3, 2, 2}, {3, 3, 3, 3, 3}, {3, 3, 3, 2, 2, 2}}
```

The conclusions for \( n = 15 \) can be easily extended to every \( n \):

\[
\text{pfirst2eqlastgtm}(n,0)=\text{pfirst2eqlastgtm}(n-1,0)+\text{pfirst2eqlastgtm}(n,1)
\]

As a further example for \( n = 30 \):

```
In[46] := Sort[Complement[pfirst2eqlastgtm[30, 0], Append[#, 1] & /@ pfirst2eqlastgtm[29, 0]],
  lorderQ] := pfirst2eqlastgtm[30, 1]
Out[46] := True
```

Now I can consider the relation between `pfirst2eqlastgtm[1,1]` and `pfirst2eqlastgtm[1,2]`. Consider the following example:
Using the same argumentation you can reach to the general integer relation:

\[ p_{\text{first2eqlastgtm}}(n, k) = p_{\text{first2eqlastgtm}}(n-k-1, k) + p_{\text{first2eqlastgtm}}(n, k+1) \]

Now consider to find a transformation from \( p_{\text{first2eqlastgtm}}(19, 1) \) to list of \( p_{\text{first2eqlastgtm}}(20, 1) \).

You cannot add 1 to first item or append 1. If you try indeed to add 1 to last item and sort you can
catch duplicates. The only safe operation is to append 2 to previous \( p_{\text{first2eqlastgtm}}(18, 1) \).

The conclusions for \( n = 20 \) can be easily extended to every \( n \):

\[ p_{\text{first2eqlastgtm}}(n, 1) = p_{\text{first2eqlastgtm}}(n-2, 1) + p_{\text{first2eqlastgtm}}(n, 2) \]

As a further example for \( n = 40 \):

\[ p_{\text{first2eqlastgtm}}(40, 1) = p_{\text{first2eqlastgtm}}(38, 1) + \]