Computation of $P(n, m)$, the Number of Integer Partitions of $n$ into Exactly $m$ Parts

M.J. Kronenburg

Abstract

Two algorithms for computing $P(n, m)$, the number of integer partitions of $n$ into exactly $m$ parts, are described, and using a combination of these two algorithms, the resulting algorithm is $O(n^{3/2})$. The second algorithm uses a list of $P(n)$, the number of integer partitions of $n$, which is cached and therefore needs to be computed only once. Computing this list is also $O(n^{3/2})$. With these algorithms also $Q(n, m)$, the number of integer partitions of $n$ into exactly $m$ distinct parts, and a list of $Q(n)$, the number of integer partitions of $n$ into distinct parts, can be computed in $O(n^{3/2})$. A list of $P(n, 1)$, $P(n, n)$ and $P(m, m)$. $P(n, m)$ can be computed in $O(n^2)$. A computer algebra program is listed implementing these algorithms, and some timings of this program are provided.

Keywords: integer partition function.
MSC 2010: 05A17 11P81

1 Definitions and Basic Identities

Let the coefficient of a power series be defined as:

$$[q^n] \sum_{k=0}^\infty a_k q^k = a_n$$  \hspace{1cm} (1.1)$$

Let $P(n)$ be the number of integer partitions of $n$, and let $P(n, m)$ be the number of integer partitions of $n$ into exactly $m$ parts.

Theorem 1.1.

$$P(n) = [q^n] \frac{1}{\prod_{j=1}^n (1 - q^j)}$$ \hspace{1cm} (1.2)$$

Proof. Using the geometric series:

$$[q^n] \frac{1}{\prod_{j=1}^n (1 - q^j)} = [q^n] \prod_{j=1}^n \sum_{k_j=0}^\infty q^{k_j} = [q^n] \sum_{k_1=0}^\infty \cdots \sum_{k_n=0}^\infty q^{\sum_{j=1}^n k_j} = P(n)$$  \hspace{1cm} (1.3)$$

Theorem 1.2.

$$P(n, m) = [q^n] \frac{q^m}{\prod_{j=1}^m (1 - q^j)}$$ \hspace{1cm} (1.4)$$
Proof.

\[
[q^n] \prod_{j=1}^m q^m \prod_{k_j=0}^\infty q^{i_jk_j} = [q^n] \sum_{k_1=0}^\infty \cdots \sum_{k_m=0}^\infty q^{m+\sum_{j=1}^m jk_j} = P(n, m)
\] (1.5)

where \(P(n, m)\) is the number of integer partitions of \(n\) with greatest part equal to \(m\). By conjugation of Ferrer diagrams this is also the number of integer partitions of \(n\) into exactly \(m\) parts [3].

**Theorem 1.3.** [13] \(P(n, m) = P(n - m, m) + P(n - 1, m - 1)\) (1.6)

Proof.

\[
P(n - m, m) + P(n - 1, m - 1) = [q^{n-m}] \prod_{j=1}^m q^m \prod_{j=1}^{m-1} (1 - q^j) + [q^{n-1}] \prod_{j=1}^{m-1} (1 - q^j)
\]

\[
= [q^n] \prod_{j=1}^m q^{2m} + [q^n] \prod_{j=1}^m q^{m(1 - q^m)} = [q^n] \prod_{j=1}^m (1 - q^j) = P(n, m)
\] (1.7)

**Lemma 1.1.**

\[
\sum_{k=0}^m q^k \prod_{j=k+1}^m (1 - q^j) = 1
\] (1.8)

Proof. The lemma is true for \(m = 0\), and using induction on \(m\), when it is true for \(m\), then for \(m + 1\):

\[
\sum_{k=0}^{m+1} q^k \prod_{j=k+1}^{m+1} (1 - q^j) = q^{m+1} + \sum_{k=0}^m q^k \prod_{j=k+1}^{m+1} (1 - q^j)
\]

\[
= q^{m+1} + (1 - q^{m+1}) \sum_{k=0}^m q^k \prod_{j=k+1}^m (1 - q^j) = q^{m+1} + 1 - q^{m+1} = 1
\] (1.9)

A similar lemma can be found as (5) in [1]. Let \(P(n|\text{at most } m \text{ parts})\) be the number of integer partitions of \(n\) into at most \(m\) parts [3].

**Theorem 1.4.** For integer \(m \leq n\):

\[
P(n|\text{at most } m \text{ parts}) = \sum_{k=0}^m P(n, k) = P(n + m, m)
\] (1.10)
Proof. Using lemma 1.1

\[ \sum_{k=0}^{m} P(n,k) = \sum_{k=0}^{m} \left[ q^n \prod_{j=1}^{k} (1-q^j) \right] = \left[ q^n \prod_{j=1}^{m} (1-q^j) \right] \]

(1.11)

Taking \( m = n \) it follows that \( P(n) = P(2n,n) \).

2 First Algorithm for Computing \( P(n,m) \)

From the recurrence relation in theorem 1.3 a simple algorithm can be derived for computing \( P(n,m) \). Using that \( P(n,m) = 0 \) when \( n < m \), let for each \( m' \) between 1 and \( m \) an array represent \( P(n',m') \) for all \( n' \) between \( m' \) and \( n - m + m' \), starting with \( P(n',1) = 1 \). Then repeated application of the recurrence relation for all \( n' \) transforms the array from \( m' - 1 \) to \( m' \), where the position in the array of \( P(n',m') \) is the same as the position of \( P(n' - 1, m' - 1) \). When \( m' = m \) is reached the array contains \( P(n',m) \) for all \( n' \) between \( m \) and \( n \), and then the last element in the array contains \( P(n,m) \).

Algorithm 1 Computation of \( P(n,m) \)

1: procedure \( P(n,m) \)
2: for \( p \leftarrow 0 \) to \( n - m \) do
3: \( a_p \leftarrow 1 \)
4: end for
5: for \( i \leftarrow 2 \) to \( \min(m,n-m) \) do
6: for \( p \leftarrow i \) to \( n - m \) do
7: \( a_p \leftarrow a_p + a_{p-i} \)
8: end for
9: end for
10: return \( a_{n-m} \)
11: end procedure

The number of steps \( S_1(n,m) \) in this algorithm, excluding the initialization steps, is:

\[ S_1(n,m) = \frac{1}{2} (\min(m,n-m) - 1)(2(n-m) - \min(m,n-m)) \]

(2.1)

3 An Expression for \( P(n,m) \) using \( P(n) \)

An expression for \( P(n,m) \) using \( P(n) \) is derived, leading to a second algorithm for computing \( P(n,m) \). Starting with theorem 1.2

\[ P(n,m) = \left[ q^n \right] \frac{q^m \prod_{j=m+1}^{n-m} (1-q^j)}{\prod_{j=1}^{n} (1-q^j)} = \left[ q^n \right] \frac{q^m \prod_{j=1}^{n-2m} (1-q^{m+j})}{\prod_{j=1}^{n} (1-q^j)} \]

(3.1)
The following identity is taken from formula [3h] on page 99 in [5]:

\[
\prod_{j=1}^{\infty} (1 - q^{m+j}) = \sum_{k=0}^{\infty} \sum_{i=0}^{k} (-1)^i q^{k+mi} Q(k,i)
\]  

(3.2)

where \(Q(n, m)\) is the number of partitions of \(n\) into exactly \(m\) distinct parts. Because only the coefficient of \(q^n\) is needed, the sum is only needed up to \(k = n - m\), and using theorem 1.1:

\[
P(n, m) = [q^n] \sum_{k=0}^{n-m} \sum_{i=0}^{k} (-1)^i q^{k+mi} Q(k,i) \frac{1}{\prod_{j=1}^{n} (1 - q^j)}
\]

(3.3)

\[
P(n, m) = \sum_{k=0}^{n-m} \sum_{i=0}^{k} (-1)^i Q(k,i) P(n - k - m(i + 1))
\]

\(Q(n, m)\) can be expressed using \(P(n, m)\), taken from page 116 in [5], [14]:

\[
Q(n, m) = P(n - \frac{1}{2} m(m - 1), m)
\]

(3.4)

which results in:

\[
P(n, m) = \sum_{k=0}^{n-m} \sum_{i=0}^{k} (-1)^i P(k - \frac{1}{2} i(i - 1), i) P(n - k - m(i + 1))
\]

(3.5)

When \(k = 0\), only \(i = 0\) gives a nonzero summand because \(P(0, 0) = 1\), leading to:

\[
P(n, m) = P(n - m) + \sum_{k=1}^{n-2m} \sum_{i=1}^{k} (-1)^i P(k - \frac{1}{2} i(i - 1), i) P(n - k - m(i + 1))
\]

(3.6)

From this formula follows:

\[
P(n, m) = P(n - m) \text{ if } m \geq \lceil n/2 \rceil
\]

(3.7)

Changing the order of summation gives:

\[
P(n, m) = \sum_{i=1}^{i_{\text{max}}(n,m)} \sum_{k=\frac{1}{2} i(i+1)}^{n-m(i+1)} (-1)^i P(k - \frac{1}{2} i(i - 1), i) P(n - k - m(i + 1))
\]

(3.8)

where \(i_{\text{max}}(n,m)\) is given by solving \(n - m(i + 1) = \frac{1}{2} i(i + 1)\):

\[
i_{\text{max}}(n,m) = \left\lfloor \frac{1}{2} \left( \sqrt{8n + (2m - 1)^2} - 2m - 1 \right) \right\rfloor
\]

(3.9)

The two last identities are the basis for a second algorithm.
4 Second Algorithm for Computing $P(n, m)$

The equations (3.8) and (3.9) are used in the second algorithm, where $P(0)\ldots P(n)$ need to be computed only once, which means they are cached in a permanent array. In the first algorithm for each $m'$ the $P(n', m')$ were computed for $n'$ between $m'$ and $n - m + m'$, but now they only are needed between $m'$ and $m' + k_{\text{max}} - k_{\text{min}}$.

**Algorithm 2** Computation of $P(n, m)$

```plaintext
1: procedure $P(n, m)$
2:     for $p \leftarrow 0$ to $n - m$ do
3:         $a_p \leftarrow 1$
4:     end for
5:     $x \leftarrow P(n - m)$
6:     for $k \leftarrow 1$ to $n - 2m$ do
7:         $x \leftarrow x - P(n - 2m - k)$
8:     end for
9:     for $i \leftarrow 2$ to $\lfloor \frac{1}{2} (\sqrt{8n + (2m - 1)^2} - 2m - 1) \rfloor$ do
10:        $k_{\text{min}} \leftarrow \frac{1}{2} i(i + 1)$
11:        $k_{\text{max}} \leftarrow n - m(i + 1)$
12:        for $p \leftarrow i$ to $k_{\text{max}} - k_{\text{min}}$ do
13:            $a_p \leftarrow a_p + a_{p-i}$
14:        end for
15:        for $k \leftarrow k_{\text{min}}$ to $k_{\text{max}}$ do
16:            $x \leftarrow x + (-1)^{i} a_{k-k_{\text{min}}} P(k_{\text{max}} - k)$
17:        end for
18:    end for
19: return $x$
20: end procedure
```

The number of steps $S_2(n, m)$ in this algorithm, when compared to the first algorithm, and not including the computation of the $P(n)$ which need to be computed only once, is about:

$$S_2(n, m) \simeq \frac{1}{2} i_{\text{max}}(n, m)(2(n - m) - i_{\text{max}}(n, m))$$  \hfill (4.1)

In figure $S_1(n, m)$ and $S_2(n, m)$ are shown for $n = 400$, from which it is clear that for small $m$ the first algorithm and for large $m$ the second algorithm is faster, where the cross over point is when $i_{\text{max}}(n, m) = m$, which has the solution $m = m_{\text{worst}}$:

$$m_{\text{worst}} = \frac{1}{6} (\sqrt{24n + 9} - 3)$$  \hfill (4.2)

Because in the second algorithm additional steps are needed in the loops over $k$, in practice $m_{\text{worst}}$ is a constant factor larger, about $m_{\text{worst}} \simeq 2.7 \sqrt{n}$. Substituting this $m$ in (2.1) it is clear that this combination of these two algorithms is $O(n^{3/2})$. Sometimes updating the cached $P(0)\ldots P(n)$ is necessary, but this is also $O(n^{3/2})$, see section 6 so the total complexity remains $O(n^{3/2})$. 

5
Figure 1: Comparison of $S_1(n, m)$ and $S_2(n, m)$ for $n = 400$

5 Formulas for $P(n, m)$ when $m \leq 6$

When $m \leq 6$ there are formulas for $P(n, m)$, such as $P(n, 1) = 1$, $P(n, 2) = \lfloor n/2 \rfloor$ [3], and, where $[x]$ is the nearest integer to $x$ [3, 4, 7, 8]:

\[ P(n, 3) = \lfloor n^2/12 \rfloor \] (5.1)

\[ P(n, 4) = \lfloor n(2n^2 + 6n + 9((-1)^n - 1))/288 \rfloor \] (5.2)

\[ P(n, 5) = \lfloor n(n^3 + 10n(n + 1) - 15((-1)^n + 5))/2880 \rfloor \] (5.3)

\[ P(n, 6) = \lfloor n(6n^4 + 135n^3 + 760n^2 + 675((-1)^n - 1)n - 30F(n \mod 6))/518400 \rfloor \] (5.4)

where $F(0) = -96$, $F(1) = F(5) = 629$, $F(2) = F(4) = 224$ and $F(3) = 309$. For practical purposes these formulas are correct up to at least $n = 10^9$. 
6 Computing a List of $P(n)$ and $Q(n)$

For the second algorithm sometimes a list of the cached $P(0) \ldots P(n)$ must be computed. The first algorithm to do this is with Euler’s pentagonal number theorem \[1, 6, 11\]:

$$P(n) = \delta_{n,0} - \sum_{k=1}^{\infty} (-1)^k \left( P(n - k(3k - 1)/2) + P(n - k(3k + 1)/2) \right)$$ \hspace{1cm} (6.1)

where the sum is over all $k$ for which the argument of $P(n)$ is nonnegative. The second algorithm with a formula of J.A. Ewell has less terms \[6, 11\]:

$$P(n) = \sum_{k=0}^{\infty} P\left( \frac{n - k(k+1)/2}{4} \right) - 2 \sum_{k=1}^{\infty} (-1)^k P(n - 2k^2)$$ \hspace{1cm} (6.2)

where the sum is over all $k$ for which the argument of $P(n)$ is a nonnegative integer. In the computer program below this formula is about 20\% faster than the previous formula. In the first sum the argument is an integer when $n - k(k+1)/2 \equiv 0 \pmod{4}$, which occurs if and only if $n \mod 4 = k(k+1)/2 \mod 4$. The values of $k(k+1)/2 \mod 4$ as a function of $k$ are a repeating pattern of \{0, 1, 3, 2\}, so the $k$ begins with $k_1 = \{0, 1, 3\}$ and $k_2 = \{7, 6, 4, 5\}$ depending on the value of $n \mod 4$, and for each term both $k_1$ and $k_2$ are increased with 8. Using:

$$\frac{1}{2}(k + 8)(k + 9) - \frac{1}{2}k(k + 1) = 8k + 36$$ \hspace{1cm} (6.3)

and substituting in the right side $k = k_1 + 8k$ and $k = k_2 + 8k$ and dividing by 4 gives $2k_1 + 16k + 9$ and $2k_2 + 16k + 9$, which are the decrements of the two $P(n)$ arguments. For efficiency the first decrement can start at $2k_1 + 9$ instead of 0, but then for the second decrement $2k_2 + 9$ must be replaced by $2(k_2 - k_1)$.

When the list $P(0) \ldots P(n)$ is computed, a list of $Q(0) \ldots Q(n)$ can be computed with another formula of J.A. Ewell \[6\]:

$$Q(n) = P(n) + \sum_{k=1}^{\infty} (-1)^k \left( P(n - k(3k - 1)) + P(n - k(3k + 1)) \right)$$ \hspace{1cm} (6.4)

A formula of M. Merca \[12\] computes a list of $Q(0) \ldots Q(n)$ without needing $P(0) \ldots P(n)$:

$$Q(n) = s(n) - 2 \sum_{k=1}^{\infty} (-1)^k Q(n - 3k^2)$$ \hspace{1cm} (6.5)

where

$$s(n) = \begin{cases} 1 & \text{if } n = m(3m \pm 1)/2 \text{ for some nonnegative integer } m \\ 0 & \text{otherwise} \end{cases}$$ \hspace{1cm} (6.6)

In the computer program below this formula is about 35\% faster than the previous formula when all $P(0) \ldots P(n)$ are known. These four algorithms for computing a list of $P(n)$ and $Q(n)$ are all $O(n^{3/2})$, and are faster than repeatedly computing isolated values of $P(n)$ and $Q(n)$ with the Hardy-Ramanujan-Rademacher formula \[11\].
7 Computer Algebra Program

A Mathematica® program is given below implementing the algorithm for $P(n, m)$ and some other related functions, which are listed below.

PartitionsPList[n]
Gives a list of the $n$ numbers $P(1)\ldots P(n)$, where $P(n)$ is the number of partitions of $n$, using the algorithm of J.A. Ewell [6][11]. The list is cached, and the algorithm is $O(n^{3/2})$.

PartitionsQList[n]
Gives a list of the $n$ numbers $Q(1)\ldots Q(n)$, where $Q(n)$ is the number of partitions of $n$ into distinct parts, using the algorithm of M. Merca [12]. The list is cached, and the algorithm is $O(n^{3/2})$.

PartitionsInPartsP[n,m]
Gives the number $P(n,m)$ of partitions of $n$ into exactly $m$ parts, with the combination of the two algorithms described in this paper, which is $O(n^{3/2})$.

PartitionsInPartsQ[n,m]
Gives the number $Q(n,m)$ of partitions of $n$ into exactly $m$ distinct parts, using the formula $Q(n,m) = P(n-m(m-1)/2,m), which is O(n^{3/2})$.

PartitionsInPartsPList[n]
Gives a list of the $n$ numbers $P(n,1)\ldots P(n,n)$, using the algorithm for $P(n,m)$. This algorithm is optimized with ListConvolve, and is therefore $O(n^2)$.

PartitionsInPartsPList[n,m]
Gives a list of the $n-m+1$ numbers $P(m,m)\ldots P(n,m)$, using the algorithm for $P(n,m)$. This algorithm is optimized with ListConvolve, and is therefore $O(n^2)$.

PartitionsInPartsQList[n]
Gives a list of the $m_{\text{max}} = \lfloor (\sqrt{8n+1} - 1)/2 \rfloor$ numbers $Q(n,1)\ldots Q(n,m_{\text{max}})$, using the algorithm for $P(n,m)$. This algorithm is $O(n^{3/2})$.

PartitionsInPartsQList[n,m]
Gives a list of the $n-m(m+1)/2+1$ numbers $Q(m(m+1)/2,m)\ldots Q(n,m)$, using the formula $Q(n,m) = P(n-m(m-1)/2,m). This algorithm is $O(n^2)$.

Below is the listing of a Mathematica® package which can be copied into a PartitionsInParts.m package file.
(* ::Package:: *)

BeginPackage["PartitionsInParts'"];

PartitionsPList::usage = "PartitionsPList[n] gives a list of the n numbers P(1)...P(n), the number of unrestricted partitions of 1..n."

PartitionsQList::usage = "PartitionsQList[n] gives a list of the n numbers Q(1)...Q(n), the number of partitions of 1..n into distinct parts."

PartitionsInPartsP::usage = "PartitionsInPartsP[n,m] gives the number P(n,m), the number of partitions of n into exactly m parts."

PartitionsInPartsQ::usage = "PartitionsInPartsQ[n,m] gives the number Q(n,m), the number of partitions of n into exactly m distinct parts."

PartitionsInPartsPList::usage = "PartitionsInPartsPList[n] gives a list of the n numbers P(n,1)..P(n,n), the number of partitions of n into exactly 1..n parts. PartitionsInPartsPList[n,m] gives a list of the n-m+1 numbers P(m,m)..P(n,m), the number of partitions of m..n into exactly m parts."

PartitionsInPartsQList::usage = "PartitionsInPartsQList[n] gives a list of the numbers Q(n,1)..Q(n,mmax), the number of partitions of n into exactly 1..mmax distinct parts. PartitionsInPartsQList[n,m] gives a list of the numbers Q(m(m+1)/2,m)..Q(n,m), the number of partitions of m(m+1)/2..n into exactly m distinct parts."

Begin["'Private'"];

PartitionsPList[n_Integer?Positive]:=(partpupdateeuler[n];partplist[[Range[2,n+1]]])
PartitionsQList[n_Integer?Positive]:=(partqupdatemerca[n];partqlist[[Range[2,n+1]]])
PartitionsInPartsP[n_Integer?NonNegative,m_Integer?NonNegative]:=
  If[n==m==0,1,If[m==0||n<m,0,If[n==m,1,If[m<=6,partitionsinpartsp1[n,m],
    If[m<=2.7Sqrt[n],partitionsinpartsp2[n,m,False],partitionsinpartsp3[n,m]]]]]]
PartitionsInPartsQ[n_Integer?NonNegative,m_Integer?NonNegative]:=
  If[n-m(m-1)/2<m,0,PartitionsInPartsP[n-m(m-1)/2,m]]
PartitionsInPartsPList[n_Integer?Positive]:=partitionsinpartsplist1[n]
PartitionsInPartsPList[n_Integer?NonNegative,m_Integer?NonNegative]:=
  If[m==0,PadRight[{1},n+1],If[n<m,{},
    If[m<0.21Exp[0.78Log[n]],partitionsinpartsp2[n,m,True],partitionsinpartsplist1[n,m]]]]
PartitionsInPartsQList[n_Integer?Positive]:=partitionsinpartsqlist1[n]
PartitionsInPartsQList[n_Integer?NonNegative,m_Integer?NonNegative]:=
  If[n-m(m-1)/2<m,{},PartitionsInPartsPList[n-m(m-1)/2,m]]

partplist={1};partqlist={1};
partpupdateeuler[n_]:=Block[{length=Length[partplist],result,kmax,kk},
partupdateewell[n_] := Block[{length = Length[partplist], result, ks, k1, k2, kk1, kk2}, Which[n >= length, partplist = PadRight[partplist, n + 1]; ks = {{9, 11, 15, 13}, {14, 10, 2, 6}, {0, 1, 6, 3}, {28, 21, 10, 15}}; Do[ks[[3]] = (i - ks[[3]])[[kk1]]]; Do[kk1 = 4k - 2; result += (-1)^k partplist[[kk1]], {k, kk1}]; result *= -2; k1 = k2 = k; k = nsqrt = Sqrt[24i + 1]; Which[IntegerQ[(nsqrt + 1)/6] || IntegerQ[(nsqrt - 1)/6], result++]; partplist[[i + 1]] = result, {i, length, n}]]

partupdateemerc[n_] := Block[{length = Length[partqlist], result, kmax, kk}, Which[n >= length, partqlist = PadRight[partqlist, n + 1]; Do[result = partplist[[i + 1]]; kmax = Floor[Sqrt[12i + 1] - 1]/6]; kk = 1; Do[result += (-1)^k partplist[[kk]], {k, kk}]; result *= -2; tsqrt = Sqrt[24i + 1]; Which[IntegerQ[(tsqrt + 1)/6] || IntegerQ[(tsqrt - 1)/6], result++]; partqlist[[i + 1]] = result, {i, length, n}]]

partitionsinpartsp1[n_, m_] := Switch[m, 1, 1, 2, Floor[n/2], 3, Round[n^2/12], 4, Round[n(2n^2+6n+9(-1)^n-1)/288], 5, Round[n(n^3+10n(n+1)-15(3(-1)^n+5))/2880], 6, Round[n(6n^4+135n^3+760n^2+675((-1)^n-1)n-30)/518400]}

partitionsinpartsp2[n_, m_, all_] := Block[{temp = ConstantArray[1, n - m + 1], nmax = n - m, pmax}, Do[result = partplist[[nmax + 1]]; For[i = 2, kmin <= kmax, kmin += i + 1, kmax -= m; nmax = Floor[(nmax + 1)/i] - 1; Do[temp[[Range[p + 1, p + i]]]] += temp[[Range[p - i + 1, p]]], {p, i, pmax}]; If[all, temp, temp[[nmax + 1]]]]

partitionsinpartsp3[n_, m_] := Block[{temp = ConstantArray[1, n - m + 1], nmax = n - m, pmax, kmmax, kmin, mmax, partplist = PadRight[partplist, nmax + 1]; Do[result = partplist[[nmax - k + 1]], {k, nmax - m}]; For[i = 2, kmin <= kmax, kmin += i + 1, kmax -= m; pmax = Floor[(nmax + 1)/i] - 1; Do[temp[[Range[p + 1, p + i]]]] += temp[[Range[p - i + 1, p]]], {p, i, pmax}]; Which[pmax >= 0, temp[[Range[pmax + 1, nmax + 1]]]] += temp[[Range[pmax + 1, nmax + 1]]]; Do[result += (-1)^i temp[[Range[kmmax + 1, partplist[[nmax - k + 1]]], {k, kmmax}], kmin, kmax]]]; result = ConstantArray[0, n], mmax, pmax, kmmax, kmin, kmax, conv},
\[m_{\text{max}} = \text{Ceiling}\left[\frac{\sqrt{24n+9}-3}{6}\right] + 1; \text{partpupdate} = \text{well}[n-m_{\text{max}}];\]
\[
\text{result}[\text{Range}[n, m_{\text{max}}-1]] = \text{partplist}[\text{Range}[n-m_{\text{max}}+1]]; \text{kmin} = 1;
\]
\[
\text{Do}[\text{Which}[i > 1, p_{\text{max}} = \text{Floor}\left[\frac{(n+1)}{i}\right]-1];
\]
\[
\text{Do}[\text{temp}[\text{Range}[p+1, p+i]] += \text{temp}[\text{Range}[p-i+1, p]], \{p, i, p_{\text{max}}, i\}];
\]
\[
\text{temp}[\text{Range}[p_{\text{max}}+1, n+i+1]] += \text{temp}[\text{Range}[p_{\text{max}}+1, n-i+1]];\]
\[
\text{result}[i] = \text{temp}[n-i+1]; k_{\text{min}} = n_{\text{max}}(i+1)-k_{\text{min}}+1;
\]
\[
\text{Which}[k_{\text{max}} > 1, \text{conv} = \text{ListConvolve}[\text{temp}[\text{Range}[k_{\text{max}}]], \text{partplist}[\text{Range}[k_{\text{max}}]], \{1, 1\}, 0]; p_{\text{max}} = \text{Floor}\left[\frac{(n-k_{\text{min}})}{(i+1)}\right];
\]
\[
\text{result}[\text{Range}[p_{\text{max}}, n-i+1]] += (-1)^i \text{conv}[\text{Range}[k_{\text{max}}-(p_{\text{max}}-n_{\text{max}})(i+1), k_{\text{max}}, i+1]]; k_{\text{min}} += i + 1, \{i, n_{\text{max}}-1\}]; \text{result}]
\]

\text{partitionsinpartsplist1}[n_, m_] := \text{Block}[\{\text{temp} = \text{ConstantArray}[1, n+1], \text{result} = \text{ConstantArray}[0, n-m+1], m_{\text{max}}, p_{\text{max}}, k_{\text{min}}, k_{\text{max}}, \text{conv}\},
\]
\[
\text{partpupdate} = \text{well}[n-m]; m_{\text{max}} = \text{Floor}\left[\frac{\sqrt{8n+4m(m-1)+9}-2m-1}{2}\right];
\]
\[
\text{result}[\text{Range}[n-m+1]] = \text{partplist}[\text{Range}[n-m+1]]; k_{\text{min}} = 1;
\]
\[
\text{Do}[k_{\text{max}} = n_{\text{max}}(i+1)-k_{\text{min}}+1; \text{Which}[i > 1, p_{\text{max}} = \text{Floor}\left[\frac{(k_{\text{max}}+1)}{i}\right]-1];
\]
\[
\text{Do}[\text{temp}[\text{Range}[p+1, p+i]] += \text{temp}[\text{Range}[p-i+1, p]], \{p, i, p_{\text{max}}, i\}];
\]
\[
\text{Which}[p_{\text{max}} > 0, \text{temp}[\text{Range}[p_{\text{max}}+1, k_{\text{max}}+1]] += \text{temp}[\text{Range}[p_{\text{max}}+1, k_{\text{max}}-i+1]]; \text{conv} = \text{ListConvolve}[\text{temp}[\text{Range}[k_{\text{max}}]], \text{partplist}[\text{Range}[k_{\text{max}}]], \{1, 1\}, 0]; p_{\text{max}} = k_{\text{min}} + m_{\text{max}}(i+1); \text{result}[\text{Range}[p_{\text{max}}+1, n-m+1]] += (-1)^i \text{conv}[\text{Range}[n-p_{\text{max}}+1]]; k_{\text{min}} += i + 1, \{i, n_{\text{max}}\}]; \text{result}]
\]

\text{partitionsinpartsqlist1}[n_] := \text{Block}[\{\text{temp} = \text{ConstantArray}[1, n+1], \text{result}, m_{\text{max}}, n_{\text{max}} = n-1, p_{\text{max}}\},
\]
\[
\text{m_{\text{max}}} = \text{Floor}\left[\frac{\sqrt{8n+1}-1}{2}\right]; \text{result} = \text{ConstantArray}[0, m_{\text{max}}]; \text{result}[1] = 1;
\]
\[
\text{Do}[n_{\text{max}} -= i; p_{\text{max}} = \text{Floor}\left[\frac{(n_{\text{max}}+1)}{i}\right]-1];
\]
\[
\text{Do}[\text{temp}[\text{Range}[p+1, p+i]] += \text{temp}[\text{Range}[p-i+1, p]], \{p, i, p_{\text{max}}, i\}];
\]
\[
\text{Which}[p_{\text{max}} > 0, \text{temp}[\text{Range}[p_{\text{max}}+1, n_{\text{max}}+1]] += \text{temp}[\text{Range}[p_{\text{max}}+1, n_{\text{max}}-i+1]]; \text{result}[i] = \text{temp}[\text{Range}[n_{\text{max}}+1]], \{i, 2, n_{\text{max}}\}]; \text{result}]
\]

End[];
EndPackage[];

In figure 2 some timings of these functions are provided using Mathematica® 12.3 on an Intel® Core i7 9700K 3.60GHz with 32GB DDR4-2133 RAM. For \text{PartitionsInPartsP} and \text{PartitionsInPartsPList} the time for updating the list of \(P(n)\), which is needed only once, is not included in the timing.
Figure 2: Timings of Mathematica® Functions

References

[1] G.E. Andrews, Euler’s Pentagonal Number Theorem, Math. Mag. 56 (1983) 279-284.

[2] G.E. Andrews, The Theory of Partitions, Cambridge University Press, 1984.

[3] G.E. Andrews, K. Eriksson, Integer Partitions, Cambridge University Press, 2004.

[4] W. Bomfim, J.E. Schoenfield, The On-Line Encyclopedia of Integer Sequences, A026810, A026811, A026812, https://oeis.org

[5] L. Comtet, Advanced Combinatorics, rev. enl. ed., Reidel Publishing Company, 1974.

[6] J.A. Ewell, Recurrences for the partition function and its relatives, Rocky Mountain J. Math. 34 (2004) 619-627.

[7] R. Honsberger, Mathematical Gems III, Math. Assoc. Amer., 1985.
[8] D.E. Knuth, *The Art of Computer Programming, Volume 4A: Combinatorial Algorithms, Part 1*, Addison-Wesley, 2011.

[9] B. Kronholm, On congruence properties of $p(n,m)$, *Proc. Amer. Math. Soc.* 133 (2005) 2891-2895.

[10] B. Kronholm, A. Larsen, Symmetry and Prime Divisibility Properties of Partitions of $n$ into Exactly $m$ Parts, *Ann. Comb.* 19 (2015) 735-747.

[11] M. Merca, Fast computation of the partition function, *J. Number Theory* 164 (2016) 405-416.

[12] M. Merca, Distinct partitions and overpartitions, *Carpathian J. Math.* 38 (2022) 149-158.

[13] E.W. Weisstein, *Partition Function P*. From Mathworld - A Wolfram Web Resource. [https://mathworld.wolfram.com/PartitionFunctionP.html](https://mathworld.wolfram.com/PartitionFunctionP.html)

[14] E.W. Weisstein, *Partition Function Q*. From Mathworld - A Wolfram Web Resource. [https://mathworld.wolfram.com/PartitionFunctionQ.html](https://mathworld.wolfram.com/PartitionFunctionQ.html)

[15] S. Wolfram, *The Mathematica Book*, 5th ed., Wolfram Media, 2003.