Efficient Computation of Restricted Partitions

Benjamin H. Smith

May 17, 2015

Have you ever wondered how many ways the total of 5 dollars can be made with coins having cent values \{1, 2, 5, 10, 20, 50, 100, 200\}? Would you like to take a guess at how many ways there are? Would you be surprised if I told you there were 6295434? Maybe we should tone it down a bit and ask how many ways there are to make 50 cents. There are still 451! Lets step back once more and determine how many ways we can make 10 cents. In fact, we can write them down:

1. 10
2. 5, 5
3. 5, 2, 2, 1
4. 5, 2, 1, 1, 1
5. 5, 1, 1, 1, 1
6. 2, 2, 2, 2
7. 2, 2, 2, 1, 1
8. 2, 2, 2, 1, 1
9. 2, 2, 1, 1, 1, 1, 1
10. 2, 1, 1, 1, 1, 1, 1, 1
11. 1, 1, 1, 1, 1, 1, 1, 1, 1, 1

1 Reducing the problem

Let \(nP(N, A)\) denote the number of partitions of \(N\) into parts \(A = \{1, 2, 5, 10, 20, 50, 100, 200\}\). A first attempt at reducing the complexity of this problem is to divide the workload in the following sense:

Write \(N = 100 \cdot d + c\) where \(d\) denotes the number of dollars in \(N\) and \(c\), the number of cents. For example, if \(N = 573\) which is $5.73 so \(d = 5\) and \(c = 73\). we could then proceed in determining how many dollars can be made from the 100 and 200 cent coins by determining the number of partitions of 5 into \{1, 2\} and then multiply by the number of ways to make 73 cents from \{1, 2, 5, 10, 20, 50\}. However, counting only these will result in a severe lack of possibilities! We must consider the fact that maybe, we only make 4 dollars with big coins and 173 cents with small coins. Luckily, the following table of possibilities will cover all of scenarios:

<table>
<thead>
<tr>
<th>dollars</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>cents</td>
<td>73</td>
<td>173</td>
<td>273</td>
<td>373</td>
<td>473</td>
<td>573</td>
</tr>
</tbody>
</table>

This leads to the first step of the problem’s reduction.

**Lemma 1.** For any positive integer \(N\),

\[
nP(N, A) = \sum_{i=0}^{d} nP(d - i, D) \cdot nP(c + 100i, C)
\]

where \(N = 100 \cdot d + c, D = \{1, 2\}, C = \{1, 2, 5, 10, 20, 50\}\) and \(A = 100 \cdot D + C\) as specified above.

**Proof.** Combinatorially speaking, we have expressed the partitions of \(N\) into parts from \(A\) as the disjoint union of pairs of partitions of \(d - i\) into \{1, 2\} and \(c + 100i\) into \{1, 2, 5, 10, 20, 50\}. \(\Box\)
Luckily, the number of partitions of an integer into two parts has a nice closed formula.

**Lemma 2.**

\[ nP(n, \{1, 2\}) = \left\lfloor \frac{n}{2} \right\rfloor + 1 \]  \hspace{1cm} (2)

where \( \lfloor x \rfloor \) denotes the smallest integer less than \( x \).

Furthermore, although not for our purposes, this generalizes to partitions with parts \( \{1, b\} \) for any \( b > 1 \) as

\[ nP(n, \{1, b\}) = \left\lfloor \frac{n}{b} \right\rfloor + 1. \]

**Proof.** The floor of \( n/b \) is precisely the maximum number of times that \( b \) goes into \( n \) without exceeding. Thus, by padding the remainder with 1’s, and starting with

\[ \frac{b, b, b, \ldots, b, 1, 1, \ldots, 1}{\lfloor n/b \rfloor, n - \lfloor n/b \rfloor} \]

there are \( \left\lfloor \frac{n}{2} \right\rfloor \) partitions containing \( b \)'s, the \( k \)th appearing as

\[ \frac{b, \ldots, b, 1, 1, \ldots, 1}{\lfloor n/b \rfloor - k, n - (n/b + k)} \]

and finally, one last partition containing all 1’s \( \square \)

With this in mind then, we have determined the value to the left factor in our sum from Equation (1) and it remains to compute \( nP(n_i, \mathcal{C}) \) where, for notational simplicity we have written \( n_i = c + 100i \).

**Remark.** Although, it may not yet seem like we have reduced the complexity of the problem, as now we have to compute a sum of products of the function we have already deemed difficult to evaluate. Furthermore, the second factor still involves a term with the original number (when \( i = d \)). However, we have reduced the problem to a smaller set of parts which, as observed in Lemma 2, yields more manageable calculations.

Using a similar reduction technique as before we may write

\[ n_i = 10 \cdot x_i + r_i \]

where \( x_i = \left\lfloor \frac{n_i}{10} \right\rfloor \) and \( 0 \leq r_i = n_i - 10x_i < 10 \) and as above, we have

\[ nP(n_i, \mathcal{C}) = \sum_{j=0}^{x_i} nP(x_i - j, \mathcal{F}) \cdot nP(r_i + 10j, \mathcal{F}) \]  \hspace{1cm} (3)

where \( \mathcal{F} = \{1, 2, 5\} \).

We now recognize the familiar pattern and seek to make a final refinement on our set of parts. For this, let us first consider the following more general result.

\footnote{This time, we split our cents into things that can be made with 10, 20 or 50 cents and things made with 1,2,5}
Lemma 3. For any positive integer \( n \), \( F = \{1,2,5\} \) write \( n = 5m + l \) with \( m = \lfloor n/5 \rfloor > 0 \) and \( 0 \leq l < 5 \). Then
\[
nP(n, F) = \sum_{k=0}^{m} nP(5k + l, \{1,2\}) = \sum_{k=0}^{m} \left( \left\lfloor \frac{5k+l}{2} \right\rfloor + 1 \right).
\]

Proof. Using the reduction technique as in previous Lemmas since \( n = 5m + l \), we may write
\[
nP(n, F) = \sum_{k=0}^{m} nP(m - k, \{1\}) \cdot nP(5k + l, \{1,2\})
\]
where, on the left, we have reduced the parts involving 5’s to be labeled as 1’s (i.e. in the same way we wrote dollars as 1’s when the coins are actually valued at 100).

The number of partitions of any number into only 1’s is always 1. That is to say \( nP(m - k, \{1\}) = 1 \) and validates the first equality in the statement. Finally, invoking Equation (2) completes the proof. \( \square \)

1.1 Recap

At this stage we might be ready to collect all of the facts we have uncovered and get ready to start evaluating. Let’s take a look.

Beginning from Equation (1) and making all simplifying substitutions, we have shown
\[
nP(N, A) = \sum_{i=0}^{d} nP(d - i, D) \cdot nP(c + 100i, C)
\]
\[
= \sum_{i=0}^{d} \left( \left\lfloor \frac{d-i}{2} \right\rfloor + 1 \right) \cdot \sum_{j=0}^{x_i} nP(y_{ij}, F) \cdot nP(z_{ij}, F)
\]
\[
= \sum_{i=0}^{d} \left( \left\lfloor \frac{d-i}{2} \right\rfloor + 1 \right) \cdot \sum_{j=0}^{x_i} \left[ \sum_{k=0}^{m_y} \left( \left\lfloor \frac{5k+\ell_y}{2} \right\rfloor + 1 \right) \cdot \sum_{k=0}^{m_z} \left( \left\lfloor \frac{5k+\ell_z}{2} \right\rfloor + 1 \right) \right]
\]
(4)

where we have again suppressed notation and written \( y_{ij} = x_i - j, z_{ij} = r_i + 10j \) and \( m_y, \ell_y, m_z, \ell_z \) are defined in terms of these as
\[
y_{ij} = 5m_y + \ell_y
\]
\[
z_{ij} = 4m_z + \ell_z
\]
satisfying the usual \( m_\omega = \left\lfloor \frac{\omega_{ij}}{5} \right\rfloor \) and \( 0 \leq \ell_\omega < 5 \) for \( \omega \in \{y, z\} \).

Unfortunately, such a deeply nested chain of summations, although faster than the naive approach discussed later, remains computationally infeasible and further simplifications must be made. A great place to start would be to see if any of these interior summations can be evaluated more efficiently.

For a positive integer \( n \) and \( b \in \{0,1,2,3,4\} \), define \( f(n, b) \) as the sum \( \sum_{k=0}^{n} \left\lfloor \frac{5k+b}{2} \right\rfloor \). This resembles a portion of the interior terms found in our above calculation and evaluates as;
Proposition 4. For any positive integer $n$ and $b \in \{0, 1, 2, 3, 4\}$ we have

$$ f(n, b) := \sum_{k=0}^{n} \left\lfloor \frac{5k+b}{2} \right\rfloor = \begin{cases} (5m + b + 2)(m + 1), & n = 2m + 1 \\ 5m^2 + (b + 2)m + \left\lfloor \frac{b}{2} \right\rfloor, & n = 2m \end{cases} $$

which also admits the closed-form expression

$$ f(n, b) = (5m + b + 2)(m + 1) - (1 - \{ \frac{n}{2} \}) (5m + b + 2 - \lfloor b/2 \rfloor) $$

where $\{x\}$ denotes the fractional part$^2$ of $x$ but turns out to be more computationally demanding.

Proof. This is proven by first observing the first few values of the sequence $\left\lfloor \frac{5k+b}{2} \right\rfloor$ and recognizing the pattern:

<table>
<thead>
<tr>
<th>b</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>$\cdots$</th>
<th>$2i + 1$</th>
<th>$2i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>2</td>
<td>5</td>
<td>7</td>
<td>10</td>
<td>12</td>
<td>15</td>
<td>17</td>
<td>20</td>
<td>$2 + 5i$</td>
<td>$5i$</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>3</td>
<td>5</td>
<td>8</td>
<td>10</td>
<td>13</td>
<td>15</td>
<td>18</td>
<td>20</td>
<td>$3 + 5i$</td>
<td>$5i$</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
<td>6</td>
<td>8</td>
<td>11</td>
<td>13</td>
<td>16</td>
<td>18</td>
<td>21</td>
<td>$3 + 5i$</td>
<td>$1 + 5i$</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>4</td>
<td>6</td>
<td>9</td>
<td>11</td>
<td>14</td>
<td>16</td>
<td>19</td>
<td>21</td>
<td>$4 + 5i$</td>
<td>$1 + 5i$</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>4</td>
<td>7</td>
<td>9</td>
<td>12</td>
<td>14</td>
<td>17</td>
<td>19</td>
<td>22</td>
<td>$4 + 5i$</td>
<td>$2 + 5i$</td>
<td></td>
</tr>
</tbody>
</table>

Notice first that the varying constant terms appearing in two right-most columns of our table, denoted by $\eta_b^o, \eta_b^e$ (for even and odd), satisfy $\eta_b^o + \eta_b^e = b + 2$.

With this pattern in mind then, the summation is split into the two cases provided above. That is,

1. When $n$ is odd; say $n = 2m + 1$, then by splitting into even and odd terms respectively,

$$ \sum_{k=0}^{2m+1} \left\lfloor \frac{5k+b}{2} \right\rfloor = \sum_{i=0}^{m} (5i + \eta_1^b) + \sum_{i=0}^{m} (5i + \eta_2^b) $$

$$ = 2 \sum_{i=0}^{m} 5i + \sum_{i=0}^{m} (\eta_1^b + \eta_2^b) $$

$$ = 5m(m+1) + (m+1)(b+2) = (5m + b + 2)(m + 1) $$

as claimed.

$^2$The fractional part is uniquely defined in terms of the floor function as $x = \lfloor x \rfloor + \{x\}$. 


2. When $n$ is even; say $n = 2m$, we find one less odd term in our summation and find
\[
\sum_{k=0}^{2m} \left\lfloor \frac{5k+b}{2} \right\rfloor = \sum_{i=0}^{m} (5i + \eta^b_1) + \sum_{i=0}^{m-1} (5i + \eta^b_2)
\]
\[
= \left[ \sum_{i=0}^{m} (5i + \eta^b_1) + \sum_{i=0}^{m} (5i + \eta^b_2) \right] - (5m + \eta^b_o)
\]
\[
= (5m + b + 2)(m + 1) - (5m + \eta^b_o).
\]

A closer examination of the column for even terms in our table shows that $\eta^b_o = \left\lfloor \frac{b}{2} \right\rfloor$ so that $\eta^b_o + \eta^b_b = b + 2$ implies $\eta^b_o = b + 2 - \left\lfloor \frac{b}{2} \right\rfloor$. Substituting this and a few simplifications to the above expression reveals the desired result.

Finally, using a little binary-arithmetic-trick, we can determine which terms $f(n, b)$ have in common and manage the additional portions as follows:

Since $n$ is either even or odd, then $\left\{ \frac{n}{2} \right\}$ is either 0 or 1 and can be interpreted as either off or on. Our calculations showed that
\[
f(n, b) = \begin{cases} 
(5m + b + 2)(m + 1) - \left( 5m + b + 2 - \left\lfloor \frac{b}{2} \right\rfloor \right), & n = 2m \\
0, & \text{otherwise}
\end{cases}
\]

so multiplying this conditional term by $1 - \left\{ \frac{n}{2} \right\}$, which is on when $n$ is even and off when $n$ is odd, we obtain a closed expression for $f$ to be
\[
f(n, b) = (5m + b + 2)(m + 1) - (1 - \left\{ \frac{n}{2} \right\}) \left( 5m + b + 2 - \left\lfloor \frac{b}{2} \right\rfloor \right).
\]

With, this we can further simplify our result from Equation (4) by removing some internal summation to obtain
\[
NP(N, A) = \sum_{i=0}^{d} \left( \left\lfloor \frac{d-i}{2} \right\rfloor + 1 \right) \sum_{j=0}^{\eta_i} (f(m_y, \ell_y) + m_y + 1)(f(m_z, \ell_z) + m_z + 1).
\]

From here, any further reductions of this formula would be unlikely to provide more efficiency or insight.
2 Comparing results to the naive solution; The code

A substantial amount of preprocessing has been done here and it is now time to see how our efforts have paid off.

Something else we need to keep in mind is the number of queries as we have not yet made any efforts to handle this additional complexity.

1. The naive algorithm;

   ```python
   def numPartitions(n,C):
       # partitions of n with parts from a list C
       if C == [1]:
           return 1
       elif n < C[-1]:
           return numPartitions(n,C[:-1])
       else:
           tot = 0
           for k in range(int(n/C[-1])+1):
               tot += numPartitions(n - k*C[-1], C[:-1])
       return tot
   ```

2. Using our formula (6);

   ```python
   def f(n,b):
       # sum the floor of (5k+b)/2 (according to Proposition 4)
       m = int(n/2)
       if n % 2 == 0:
           return 5*(m**2) + (b+2)*m + b//2
       else:
           return (5*m+b+2) * (m+1)
   
   def insideSum(c,i):
       Ni = 100*i + c
       xi, ri = divmod(Ni,10)
       mid_total = 0
       for j in range(xi+1):
           aij, lij = divmod(xi-j, 5)
           left_total = f(aij,lij) + aij + 1
           aij, lij = divmod(10*j + ri, 5)
           right_total = f(aij,lij) + aij + 1
           mid_total += left_total*right_total
       return mid_total
   ```

   ```python
   T = input("Number of queries: ")
   for _ in range(T):
       N = input()
       d, c = divmod(N, 100)
grand_total = 0
for i in range(d+1):
    grand_total += ((d-i)//2 + 1) * insideSum(c,i)
print grand_total

3. The fastest Algorithm;

C = [1,2,5,10,20,50,100,200]
A = [[0 for j in range(len(C))] for i in range(10**6 + 1)]

for i in range(10**6 + 1):
    A[i][0] = 1
for j in range(1,len(C)):
    if i >= C[j]:
        A[i][j] += A[i][j-1]
        A[i][j] += A[i-C[j]][j]
    else:
        A[i][j] = A[i][j-1]

T = input("Number of queries: ")
for _ in range(T):
    N = input()
    print A[N][-1]

3. General Solution; for partitions restricted to a finite set

N, M = map(int, raw_input().split(' '))
C = map(int, raw_input().split(' '))
C.sort()                #Precautionary
A = [[0 for j in range(M)] for i in range(N+1)]

A[0] = [1 for _ in range(M)]
for i in range(1,N+1):
    if i % C[0] == 0:
        A[i][0] = 1
    else:
        A[i][0] = 0
for j in range(1,M):
    if i >= C[j]:
        A[i][j] += A[i][j-1]
        A[i][j] += A[i-C[j]][j]
    else:
        A[i][j] = A[i][j-1]
print A[N][-1]