COMPUTING PARTITION NUMBERS Q(N)

EVANGELOS GEORGIADIS

Abstract. In this short note we present an algorithm that computes partition numbers $q(n)$ for moderately sized $n \leq 10^5$. The algorithm is based on a recurrence of $q(n)$ expressed in terms of $q(n-k^2)$.

Definition. The partition number $q(n)$ counts the number of restricted partitions of a positive integer n, that is, the number of distinct ways in which n can be written as a sum of distinct positive integers when order is irrelevant and repetitions are not allowed. For example, $q(6) = 4$ since $q(6) = \mid \{6, (5{+}1), (4{+}2), (3{+}2{+}1)\} \mid = 4$. The partition numbers form sequence $A000009$ in OEIS [1].

Recurrence. Our algorithm is based on the following recurrences for $q(n)$; further, it computes $q(n)$ in terms of $q(n-k^2)$.

$$q(n) + 2\sum_{k=1}^{\lfloor\sqrt{n}\rfloor}(-1)^k q(n-k^2) = \sigma(n), \tag{1}$$

where $\sigma(n)$ is given by

$$\sigma(n) = \begin{cases} (-1)^j & \text{if } n = j(3j \pm 1)/2 \text{ for some } j \in \mathbb{N}, \\ 0 & \text{otherwise,} \end{cases}$$

further we define $q(0) = 1$.

Proof. The recurrence follows from well-known identities for the restricted partition function (see [3, Chapter XIX]). For convenience, we set the scence with the identities in question.

$$\prod_{n=1}^{\infty}(1-x^n) = 1 + \sum_{n=1}^{\infty}(-1)^n \left\{x^{n(3n\pm1)/2}\right\}, \tag{2}$$

$$\begin{aligned}
\prod_{n=1}^{\infty}(1-x^n) &= \prod_{n=1}^{\infty}(1+x^n)\left\{\prod_{n=0}^{\infty}(1-x^{2n+1})^2(1-x^{2n+2})\right\} \\
&= \prod_{n=1}^{\infty}(1+x^n)\left\{\sum_{n=-\infty}^{\infty}(-1)^n x^{n^2}\right\} \\
&= \prod_{n=1}^{\infty}(1+x^n)\left\{1 + 2\sum_{n=1}^{\infty}(-1)^n x^{n^2}\right\}. \qquad (3)
\end{aligned}$$

Multiplying (2) with (3) and comparing coefficients of x^n will yield our recurrence.

(4)
$$\left(\sum_{n=0}^{\infty} q(n)x^n\right)\left(1+2\sum_{n=1}^{\infty}(-1)^n x^{n^2}\right) = \sum_{n=0}^{\infty} x^n \left(q(n) + \sum_{k=1}^{\lfloor\sqrt{n}\rfloor}(-1)^k 2q(n-k^2)\right).$$

□

Example. We would like to illustrate a sample computation of recurrence (1) for $n = 20$.

$$\begin{aligned} q(20) &= 2\{q(20-1) - q(20-4) + q(20-9) - q(20-16)\} + \sigma(20) \\ &= 2\{q(19) - q(16) + q(11) - q(4)\} \\ &= 2\{54 - 32 + 12 - 2\} \\ &= 64. \end{aligned}$$

Algorithm. We have established the validity of our recurrence and can now safely make use of it. We precompute values of $\sigma(n)$ for our table and then use recurrence (6) to compute the actual values of $q(n)$.

$$q(n) + 2\sum_{k=1}^{\lfloor\sqrt{n}\rfloor}(-1)^k q(n-k^2) = \sigma(n), \tag{5}$$

where $\sigma(n)$ is now given by

$$\sigma(n) = \begin{cases} (-1)^j & \text{if } n = j(3j \pm 1)/2 \text{ for some } j \in \mathbb{N}, \\ 0 & \text{otherwise,} \end{cases}$$

and $q(0) = 1$.

$$q(n) = 2\sum_{k=1}^{\lfloor\sqrt{n}\rfloor}(-1)^{k+1} q(n-k^2). \tag{6}$$

This recurrence yields the following algorithm to compute partition numbers. Note, that rather than using the recursion to compute $q(n)$ top-down, we instead work bottom-up to build a table of $q(m)$ for m up to n.

Algorithm 1. For a positive integer n compute $q(m)$ for $0 \le m \le n$:
1. Set $q_0 \leftarrow 1$
   For $k = 0$ to n, set $q_k = \sigma(k)$ (using $\sigma(0) = 1$).
2. For k from 0 to $n-1$:
3. For each integer $m = k + j^2 \le n$ set $q_m \leftarrow q_m + 2\left((-1)^{j+1}q_k\right)$.
4. Output $q(m) = q_m$ for m from 0 to n.

Algorithm 1 may also be used for the sole purpose computing $q(n)$. The execution of Algorithm 1 on input $n = 10$ is illustrated in the table below. In each row the value of q_k is underlined and modified entries are in bold.

k	q_0	q_1	q_2	q_3	q_4	q_5	q_6	q_7	q_8	q_9	q_{10}
-	1	-1	-1	0	0	1	0	1	0	0	0
0	1	**1**	-1	0	**-2**	1	0	1	0	**2**	0
1	1	1	**1**	0	-2	**-1**	0	1	0	2	**2**
2	1	1	1	**2**	-2	-1	**-2**	1	0	2	2
3	1	1	1	2	**2**	-1	-2	**-3**	0	2	2
4	1	1	1	2	2	**3**	-2	-3	**-4**	2	2
5	1	1	1	2	2	3	**4**	-3	4	**-4**	2
6	1	1	1	2	2	3	4	**5**	-4	-4	**-6**
7	1	1	1	2	2	3	4	5	**6**	-4	-6
8	1	1	1	2	2	3	4	5	6	**8**	-6
9	1	1	1	2	2	3	4	5	6	8	**10**

Complexity of Algorithm 1. Our algorithm uses $O(n^{3/2})$ arithmetic operations. In particular, in Step 3, the complexity is bounded by

$$\sum_{k=0}^{n-1} \sum_{j=1}^{\sqrt{n-k}} 0(1)\text{arithmetic operations,}$$

where the arithmetic operations are all additions or subtractions (which covers multiplications by 2). This, in turn, provides a bound of the form $3/2n^2 + O(n)$ arithmetic operations; additionally, by bounding the size of the integers (all of which are smaller than n bits) we get the upper bound $q(n) \leq p(n) \leq 2^n$, $O(n^{5/2})$ on the complexity of Algorithm 1. Finally, we note that our algorithm has space complexity $O(n^2)$, yielding it suitable only for moderately sized values of n.

Acknowledgement

I would like to thank Prof. Kiran Kedlaya and Prof. Andrew Sutherland for advising me with this problem. In particular, I am very grateful to Prof. Andrew Sutherland for providing me with a sample algorithm as well as its analysis of an analogous problem; this led to some of the ideas in this technical report, developing these ideas as well as crafting this report.
Additionally, I would like to thank Prof. Tom Apostol of Caltech as well as Prof. Philippe Flajolet of INRIA for pointing out several misprints and providing reference pointers. Last but not least, I am tremendously grateful to Prof. Mike Sipser of MIT.

References

1. N. J. A. Sloane, *The on-line encyclopedia of integer sequences*, 2009, `www.research.att.com/~njas/sequences/`.
2. George E. Andrews and Kimmo Eriksson, *Integer Partitions*, 2004, Cambridge University Press.
3. G.H. Hardy and E.M. Wright, *An Introduction to the Theory of Numbers*, 5th Edition, 1980, Oxford University Press.
4. H.S. Wilf, *generatingfunctionology*, 2nd Edition, 1994, Academic Press Inc.

E-mail address: `egeorg@mit.edu`