## APPENDIX

While this article describes a process of converting an integer *n* (rank) into a sequence of numbers (a lottery ticket), the reverse, going from ticket to rank, can be done in a similar manner. To make the tickets look "random" and not appear in sequential order, first run the ticket rank through a pseudo-random mapping to get a "random" ticket rank and only then apply the conversion to an actual ticket.

One example of such a random mapping is to do prime multiplication of the ticket number modulo the ticket space. To be more explicit, let *N* be the total ticket space and *A* be a randomly selected prime number smaller than *N*. $A_x$ mod *N* maps *x* to another number less than *N*, and this mapping is one-to-one. This defines a permutation, and given a random prime, *A*, can produce seemingly random numbers that look sufficiently different to a typical buyer.

As an example, consider a ticket space of 10 with the prime number 7. A random constant also can be added to the mapping to provide a random shift of the numbers as well,which avoids having 0 always map to 0.

| *x* | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| 7*x* mod 10 | 0 | 7 | 4 | 1 | 8 | 5 | 2 | 9 | 6 | 3 |

A lottery ticket consists of five integers from 1 to 69 (white balls) and a sixth integer ranging from 1 to 26 (the powerball). In this case, n must be at least 0 and less than 292,201,338. We first consider generating just the white balls in the range from 0 to 68 (simply add 1 to each ball in the end).

Our approach is to generate each number sequentially, keeping track of a lower bound to ensure a strictly increasing order. Let `GenTicket(n,l,s)` be a function where `n` is the rank in question, `l` is a lower bound for the numbers we are allowed to use, and outputs a sequence of `s` integers in strictly increasing order with values at least `0` and less than `h` (globally provided). To generate the white balls, call `GenTicket(n,0,5)`, where `n` is the rank and `h` is globally provided as 69. Define `GenTicket` as follows in Python, where `Binom(n,k)` counts the number of combinations to choose `k` from `n` objects:

```
global h # upper bound of ticket numbers
def GenTicket(n,l,s):
  if s == 1:
    return [n + l]
  else:
    i = 1
    while n >= Binom(h – l – i, s – 1):
      n -= Binom(h – l – i, s – 1)
      i += 1
    return [l + i – 1] + GenTicket(n, l + i,
    s – 1)
```

Intuitively, we use `Binom` to determine how many tickets there are starting with the given lower bound, and continuously reduce the ticket space until we know the range in which the first number should lie. Then, we can compute the rest of the ticket recursively.

For the remaining powerball, simply divide the integer n by the total number of possible white balls (11,238,513 for Powerball) to get the powerball number. As long as n is within the possible number of tickets, this will compute the appropriate "level" in the ticket space.

The first (*n* = 0) ticket would be (1,2,3,4,5,1) and the last (*n* = 292,201,337) ticket would be (65,66,67,68,69,26). The computation to find the 100,000,000th ticket can be used as an example. We generate this ticket with 0-based indexing, so at the end, we increase each value by 1 to get the standard ticket values.

First divide *n* = 100,000,000 by 11,238,513 to get the powerball number, which is 8. Now, find the remainder of *n* when divided by 11,238,513 to get 10,091,896, which represents the rank for computing the values of the white balls. The steps of the process to show how intermediate values are determined are as follows.

Starting point:

```
(?, ?, ?, ?, ?, 8)
n: 10091896, lower bound: 0, size: 5
```

Check whether 0 can be the first number. The number of possible tickets with 0 as the first number is $\binom{68}{4}$. However, since 10,091,896 $>$ $\binom{68}{4}$, we know that the first number cannot be 0. Thus, we reduce *n* to get *n* =10,091,896 – $\binom{68}{4}$ =9,277,511.

We can rule out 1 as the first number because 9,277,511 $>$ $\binom{67}{4}$ = 8,511,031. Thus, again reduce to get *n* = 9,277,511 – $\binom{67}{4}$ = 8,511,031. Repeat this for a total of 24 iterations until finally getting that 75,142 $<$ $\binom{44}{4}$. Thus, our ticket lies in this set of possible tickets, and our first number must be 24.

Simply repeat this process recursively to generate all the other numbers, which is summarized as:

```
(24, ?, ?, ?, ?, 8)
n: 75142, lower bound: 25, size: 4
Since we have used 24, our new lower bound
is 25.
Our new size is 4 since we only have 4 more
to fill now.
62801 = 75142 – Binom(43,3)
...total 7 iterations...
5436 < Binom(36,3)
Thus, add 7 more to the new lower bound.
```

## APPENDIX (CONTINUED)

```
(24, 32, ?, ?, ?, 8)
n: 5436, l: 33, size: 3
4841 = 5436 - Binom(35,2)
... total 13 iterations ...
67 < Binom(22,2)

(24, 32, 46, ?, ?, 8)
n: 67, l: 47, size: 2
46 = 67 - Binom(21,1)
26 = 46 - Binom(20,1)
7 < Binom(19,1)
```

```
(24, 32, 46, 50, ?, 8)
n: 7, low: 51, size: 1
Since size = 1, we can return low + n.

Final result
(24, 32, 46, 50, 58, 8)
```

In the end, increment each value by 1 to have every value start at 1, and the 100,000,000th ticket is `(25, 33, 47, 51, 59, 9)`.