For each $1 \leq i \leq n$, define c_i to be the number of elements before the ith element in the array that are removed before the ith element in the process. Apparently, $0 \leq c_i < i$ for all $1 \leq i \leq n$. Let's call the array $(c_i)_{i=1}^n$ *good* if for each $1 \leq i \leq n$, the condition $0 \leq c_i \leq i-1$ holds. Notice that now the problem can be rephrased as inserting the values of $a_i + i - c_i$ in some order of i, where $1 \leq i \leq n$. We now introduce and prove an interesting claim.

Lemma 0.1 Given a length n, all *good* arrays $(c_i)_{i=1}^n$ are achievable in the process described in the problem.

Proof Note that $c_1 \equiv 0$. Consider c_2: if we want $c_2 = 0$, we could simply delete the second element before deleting the first element. On the other hand, if we want $c_2 = 1$, we can just delete the first element before the second. Thus, it is clear that all *good* arrays of length $n \leq 2$ are achievable.

Now, assume that all *good* arrays of length n are achievable, where $n \geq 2$. We now show that this implies all *good* arrays of length $n+1$ are achievable. Notice that all *good* arrays of length $n+1$ are formed by appending a value x with $0 \leq x \leq n$ at the end of a *good* array of length n. For a specific *good* array of length n, and a particular value of x, we can make this possible by doing the following:

1. delete the first x values in the deletion sequence corresponding to the *good* array of length n;
2. delete the $(n+1)$th element;
3. delete the rest of the element in the deletion sequence corresponding to the *good* array of length n.

Notice how the second step doesn't affect the previous *good* array, and how c_{n+1} is now equal to x. The the *good* array of length $n+1$ formed by the given *good* array of length n appending x is achievable.

Thus, by induction, for all positive integer n, all *good* arrays of length n can be obtained in the process described in the problem. □

We claim that the algorithm:

iterating i from 1 to n, insert the largest integer $a_i + i - c_i$, where $0 \leq c_i \leq i-1$, that is not yet contained in S.

produces the lexicographically largest answer.

Proof If we are now inserting the ith element of the original array, it is obviously the best to choose the largest integer $a_i + i - c_i$, where $0 \leq c_i \leq i-1$, that is not yet contained in S.

So what now remains to be proved is why the order of iteration from 1 to i gives the lexicographically largest answer.

Note that if we assume that during the process we don't face the scenario where when we want to delete element i, there's no available value in the range $[a_i + 1, a_i + i]$, then the resultant answer array is the same for all orders of deletions.

This is proved by considering all values of $a_i + i$, which is the best choice for each i at the beginning. Of course, there might be duplicates, so we have to continue decreasing the values of those duplicated $a_i + i$ until no more duplicates are made. This actually fixes the final answer. For instance, if the values of $a_i + i$ gives one 2 and three 5's, it apparently only corresponds to the answer $2, 3, 4, 5$, where one original 5 is decreased by 1, and the other repeated 5 is decreased by 2.

Moreover, when such a collision scenario occurs, the answer array that is decreased in length apparently doesn't grow lexicographically larger, because collisions are equivalent to removing some answers from the optimal answer array.

Thus, we now only have to show that our algorithm actually gives the longest possible answer sequence.

But this is obvious. Following the order from 1 to n, when we arrive at i, there are i possible choices in the range $[a_i + 1, a_i + i]$ for us to choose, but the given algorithm implies that at most $(i - 1)$ of these i available values are taken, so there is always a possible choice.

Thus, the correctness of the algorithm is proved. □