Algorithm 1: Method that creates the index dictionary and the probability mass dictionary.

```
Input: M, Length of the range of n values considered
Output: index_dict, a dictionary of indices by size of subset
        prob_dict, a dictionary of probability masses with the same structure as index_dict
possible_indices ← {1, ..., M};
for i ← M ... 1 do
    starting_index ← 0;
    indices_for_this_size ← empty list;
    while TRUE do
        if starting_index + i > M then
            Break out of the while loop;
        end
        index_subset ← possible_indices[(starting_index + 1) : (starting_index + i)];
        Add index_subset to indices_for_this_size;
        starting_index += 1;
    end
    Add indices_for_this_size as the i^th element of index_dict;
end
Set prob_dict equal to a copy index_dict with NA values filled in;
```

Algorithm 2: Method that calculates the probability mass of a given subset of n values.

Input: index_subset, the subset of indices of interest
range_of_n_values, the actual values of n. Indices in former set refer to these actual values
index_dict, a dictionary of indices by size of subset
prob_dict, a dictionary of probability masses with the same structure as index_dict
my_data, the observed $Bin(n, P)$ values
Output: index_dict, prob_dict, updated with the calculated probability mass of index_subset

```
possible_indices ← {1, ..., M};
min_size_considered ← length of index_subset;
max_size_considered ← largest set size in index_subset;
prob_mass ← 0;
for i ← {min_size_considered, ..., max_size_considered do
    for A ∈ index_dict such that |A| == i do
        if A == index_subset then
            min_n ← range_of_n_values[minimum index of A];
            max_n ← range_of_n_values[maximum index of A];
            prob_mass ← ∏_{x∈my_data} (F_X(x, max_n, P) − F_X(x − 1, min_n, P)) for X ~
              binomial;
        else
            prob_mass ← prob_mass − {probability mass of A from prob_dict};
    end
end
```

Algorithm 3: Main method to find all probability masses for every possible subset of n

Input: my_data, the observed $Bin(n, P)$ values
ϵ, precision value associated with the lower cutoff for feasible probability mass
Output: index_dict, a dictionary of indices by size of subset
prob_dict, a dictionary of probability masses with the same structure as index_dict

```
n_values_considered ← output from Algorithm 4;
index_dict, prob_dict ← [Algorithm 1][length(n_values_considered)];
for i ← { length(n_values_considered) ... 1 } do
    for A ∈ index_dict such that |A| == i do
        index_dict, prob_dict ← [Algorithm 2][A, n_values_considered,
          index_dict, prob_dict, my_data];
        Zero out probability value of A if its ratio to the maximum probability mass is
          below ε threshold;
    end
end
Normalize all probability values;
```

Algorithm 4: Function to find, and return, reasonable values of n

Input: my_data, the observed $Bin(n, P)$ values $\{X_i\}_{i=1}^{n}$

ϵ, precision value associated with the lower cutoff for feasible probability mass

Output: feasible_n, range of n values that are feasible based off of precision cutoff

n_currently_considered $\leftarrow \max\{$my_data$\}$;

max_prob_mass $\leftarrow \prod_{i=1}^{n}(F(X_i,$ n_currently_considered$) - F(X_i - 1,$ n_currently_considered$))$;

Add n_currently_considered to feasible_n;

while *TRUE* **do**

  n_currently_considered $+= 1$;

  current_prob_mass $\leftarrow$
  $\prod_{i=1}^{n}(F(X_i,$ n_currently_considered$) - F(X_i - 1,$ n_currently_considered$))$;

  if $\log($current_prob_mass$) - log($max_prob_mass$) \geqslant 0$ **then**

    Add n_currently_considered to feasible_n;

    max_prob_mass $\leftarrow$ current_prob_mass;

  else if $\log($current_prob_mass$) - log($max_prob_mass$) > \epsilon$ **then**

    Add n_currently_considered to feasible_n;

  else

    Break from for loop;

  end

end