1.a

To calculate the probability that deleting one item from B does not cause false negative results for any of the remaining n items in B, we need to consider the following factors:

Now, the probability that deleting one item from B does not cause false negative results for any of the remaining n items in B is the same as the probability that none of the k bits for each of the n items are set to 0 after deleting one item. This is equal to

. The probability that a given bit in B is set to 1 after inserting n+1 items is

$$1-(1-\frac{1}{b})^{k(n+1)}$$

. The probabilty that a given bit in B is shared by the deleted item and another item is

$$\frac{k^2}{b}$$

. The probabilty that a given bit in B is not shared by the deleted item and another item is

$$1-\frac{K^{-}}{b}$$

. The probability that a given bit in B is set to 0 after delting one item is

$$(1-\frac{1}{b})^{\text{[illegible]}}+\frac{K}{b}(1-\frac{1}{b})^{\text{[illegible]}}$$

. The probabilty that a given bit in B is still 1 after deleting one item is

$$1-(1-\frac{1}{b})^{k(n+1)}-\frac{k^2}{b}(1-\frac{1}{b})^{k(n)}$$

Now, the probability that deleting one item from B does not cause false negative results for any of the remaining n items in B is the same as the probability that none of the k bits for each of the n items are set to 0 after deleting one item. This is equal to

$$(1-(1-\frac{1}{b})^{k(n+1)}-\frac{k^2}{b}(1-\frac{1}{b})^{k(n)})^{kn}$$

1.b

The expected number of bits which are set due to insert of the first n items and which are unset by the delete of item (n + 1) is equal to the expected number of bits which are shared by the deleted item and any of the previous n items. This is because those are the bits that will be set to zero by the deletion operation, and they are also the bits that will cause collisions when inserting the (n + 1)-st item.

To calculate the expected number of shared bits, we can use the linearity of expectation and sum over all the b bits in the Bloom filter. The probability that a given bit is shared by the deleted item and another item is :

$$\frac{K}{b}$$

as we have seen in part a). Therefore, the expected number of shared bits is:

$$b \cdot \frac{k^2}{b} = k^2$$

Hence, the expected number of bits which are set due to insert of the first n items and which are unset by the delete of item (n + 1) is

$$k^2$$

2.

One possible solution using Bloom filters to accelerate the counting is to use a Counting Bloom Filter (CBF) with a threshold of 4. A CBF is a variant of the standard Bloom Filter (SBF) that allows deletion of elements by storing the frequency of occurrence of each element in an array of counters instead of bits. A CBF can also be used to estimate the frequency of an element by taking the minimum of the counters corresponding to its hash values. However, a CBF may overestimate the frequency due to collisions, so it is not very accurate for elements with low frequency. Therefore, we can use a threshold of 4 to filter out the elements that appear less than four times, and only store the elements that exceed the threshold in a separate data structure, such as a hash table or a heap.

To determine the optimal parameters k and b for the CBF, we need to consider the trade-off between the space efficiency and the false positive rate. The space efficiency is the ratio of the number of bits required by the CBF to the number of bits required by an exact representation of the set. The false positive rate is the probability that the CBF returns a frequency greater than or equal to the threshold for an element that is not in the set. We can use the following formulas to calculate these metrics :

- Space efficiency:

$$\frac{b}{n\log_2\left(\frac{n}{n_4}\right)}$$

-False positive rate:

$$(1 - e^{-kn/n_4})^k$$

where n is the total number of elements, n_4 is the number of elements that appear four or more times, and b is the number of bits in the CBF. We can assume that n_4 is approximately 3.47% of n, which is 34.7 million.

To find the optimal k and b, we can use a heuristic method that minimizes the false positive rate for a given space efficiency, or vice versa. One such method is to use the formula

$$k = \frac{b}{n_4} \ln 2$$

which minimizes the false positive rate for a given b .

Then, we can choose a suitable b based on the desired space efficiency or the acceptable false positive rate. For example, if we want a space efficiency of 10%, we can choose b = 0.1 * n * log2(n / n_4) = 2.6 billion bits. Then, using the formula for k, we get k = 2.6. Since k must be an integer, we can round it up to 3. The false positive rate for this choice of k and b is 0.0004, which means that only 0.04% of the elements that are not in the set will be falsely reported as having a frequency of 4 or more.

Alternatively, if we want a false positive rate of 0.001, we can use the formula for the false positive rate to solve for b, and get b = -n_4 * ln(0.001) / (k * ln(2)) = 2.9 billion bits. Then, using the formula for k, we get k = 2.9. Again, we can round it up to 3. The space efficiency for this choice of k and b is 8.9%, which means that the CBF uses only 8.9% of the bits required by an exact representation of the set.

Therefore, a possible solution using CBF is to use k = 3 and b = 2.9 billion bits, with a threshold of 4. This will give us a false positive rate of 0.001 and a space efficiency of 8.9%. We can then use the CBF to filter out the elements that appear less than four times, and store the remaining elements and their exact frequencies in another data structure. This will accelerate the counting process and reduce the memory usage.