Your name
CS 261, Spring 2020
Homework 3
Due Monday, April 20

1. Suppose that you are given the bitmap representation of a set S of non-negative integers. Describe how to find the smallest non-negative integer x that is *not* a member of S, in a constant number of integer operations.

(This is an important operation in combinatorial game theory, where it is called the "minimum excluded value" or "mex". You may assume that the dictionary `set2element` from the lecture notes has already been computed and that looking up a key-value pair in that dictionary is allowed among the constant number of operations that you perform. You may also assume that the resulting value x is within the range of values allowed by your bitmap set representation.)

Initiate x = 0, then compare x with the smallest element in S. While x is bigger than the smallest element, increase x by 1 and delete the smallest element in S. Return x when out of the while loop. The pseudo-code is below:

```
x = 0
while x > set2element[S (S-1)] do
   x += 1
   S = S-1
end while
return x
```

■

2. The lecture notes provide formulas that, for bitmap-based sets, perform the operations of set union, set intersection, and testing whether one set is a subset of another. Which of these formulas work correctly without modification on Bloom filters? (You can assume that the Bloom filter is small enough to fit into a single word of memory, so that the same formulas can be used; the question is about the correctness of these formulas, not about their implementation.) Explain your answers.

Union works: Say there are two bloom filters A and B, when take union, all the ones in both A and B are combined. Thus, if one element belongs to either A or B or both, it must be included in the union.

Intersection does not work: To take intersection, we keep the ones that are shared in two bloom filters. If the two bloom don't have any element in common, it should return empty, thus all zeros. However, there might still be shared ones between the original two bloom set so does not work.

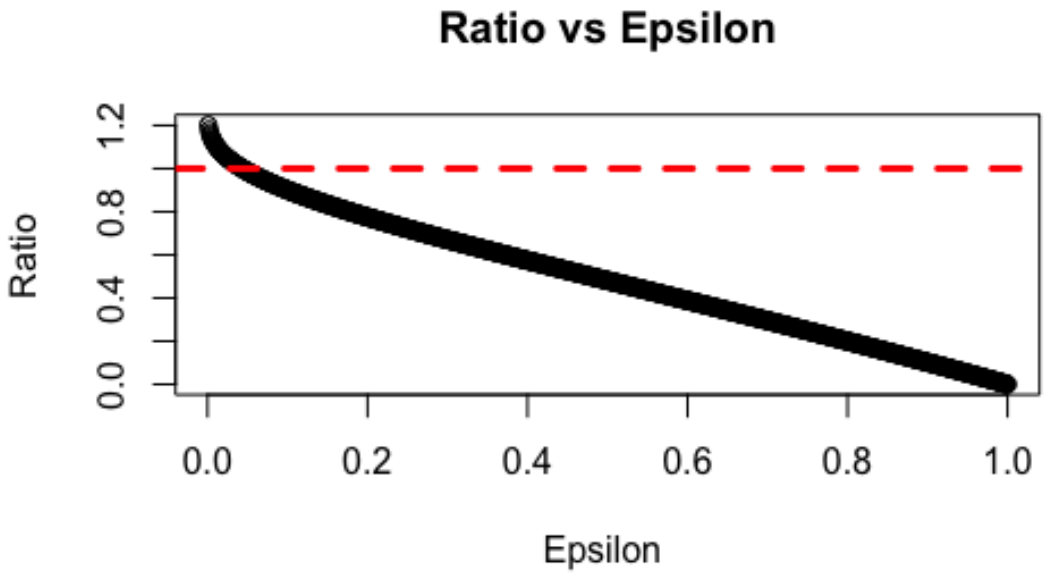
Testing whether one set is a subset of another ■

3. We have seen that, to achieve a false positive rate of ε for a set of n items, Bloom filters need $(n \log_2 1/\varepsilon)/\log 2$ bits of storage. We didn't analyze cuckoo filters as precisely, but many sources give a bound of $n(2 + \log_2 1/\varepsilon)$ on the number of bits of storage needed for them. Assuming that this bound is accurate, for which values of ε do cuckoo filters use fewer bits than Bloom filters, and for which values of ε do cuckoo filters use more bits? (It may be easiest to solve this numerically by writing a short computer program.)

$$ratio = \frac{nlog_2\frac{1}{\epsilon}}{log_e 2}/n(2+log_2\frac{1}{\epsilon}) \quad (1)$$

$$= \frac{log_2\frac{1}{\epsilon}}{log_e 2}/(2+log_2\frac{1}{\epsilon}) \quad (2)$$

When ratio is bigger than 1, cuckoo filters use fewer bits than Bloom filters, and vice versa. ϵ is the false positive rate, which should be within [0,1]. By setting ratio to one and solving the equation, $\epsilon = 0.044$. Thus when $\epsilon < 0.044$, cuckoo filters use fewer bits than Bloom filters. when $\epsilon > 0.044$, cuckoo filters use more bits than Bloom filters. I also wrote a short script and plot the ratio vs ϵ.



■

4. The lectures for this week include a proof that, if the strong exponential time hypothesis (SETH) is true, then for every constant $\varepsilon > 0$ the disjointness data structure problem for N sets of size k cannot be solved with preprocessing time $N^{2-\varepsilon}k^{O(1)}$ and query time $N^{1-\varepsilon}k^{O(1)}$. Prove a stronger bound on the preprocessing, with the same bound on the query time: if SETH is true, then for every two constants $a > 0$ and $\varepsilon > 0$, the disjoint set data structure problem cannot be solved with preprocessing time $N^a k^{O(1)}$ and query time $N^{1-\varepsilon}k^{O(1)}$.

Hint: Follow the same proof strategy as in the lecture notes, but partition the variables of the satisfiability instance into two subsets of unbalanced sizes, depending on a. Which collection of partial truth assignments should be used as the data for the disjointness structure, and which collection of partial truth assignments should be used as the queries?

preprocess: $N^a K^{O(1)}$
query time: $N^{1-\epsilon}K^{O(1)}$
split n clauses into 2 sets, so that $\frac{setsize1}{setsize2} = a$
$(2-\epsilon)^n m^{O(1)} = \frac{1}{2^\delta} \times 2^{\frac{na+n}{a+1}}$
Thus, if SETH is true, then for every two constants a ¿ 0 and $\epsilon > 0$, the disjoint set data structure problem cannot be solved with preprocessing time $N^a K^{O(1)}$ and query time $N^{1-\epsilon}K^{O(1)}$ ■