Homework 7, additional problems, solution set

1. Given a membership query on x, we report $x \in T$ if the Bloom filter test on T reports $x \in T$ and the Bloom filter test on S reports $x \notin S$. Likewise, we report $x \in S$ if the Bloom filter test reports $x \in S$.

Clearly, if $x \in S$ then we always report $x \in S$. If $x \in T$ we will ensure that the test on the Bloom filter for S always reports $x \notin S$, and therefore we always report $x \in T$. For $x \in U \setminus (S \cup T)$, a report that it is in T will occur with probability at most $1/2^{m_t}$ by assumption.

As the Bloom filter on S has a probability of an incorrect answer of at most $(s/2t)^{m_s}$, the expected number of wrong answers on all of T is at most $t \cdot (s/2t)^{m_s}$, and as $s^2 \le t$, this is upper bounded by $t/(2\sqrt{t})^{m_s}$. If we choose $m_s = 2$, this is at most $1/4$.

We build a Bloom filter for S using hash functions drawn uniformly at random. We then check in $O(t)$ time whether it gives an incorrect answer on any item in T. This happens with probability at most $1/4$. (For if the probability is greater than $1/4$, then the expected number of wrong answers is more than $1/4 \cdot 1$, a contradiction.) If there is an incorrect answer, we draw new hash functions uniformly at random and iterate until hash functions producing no collisions are obtained. This takes expected time $O(t)(1 + 1/4 + 1/4^2 + \ldots)) = O(t)$.

2. This analysis applies both to hashing with chaining (with a small modification in the constants is the table sizes, it would also apply to the open addressing setting).

We analyze the cost of rebuilding the tables. Note that if a table H of size 2^i is replaced by a table H' of size 2^{i+1} there must have been at least 2^{i-1} insertions since H was created. To build H', we need to initialize every entry in H' to **nil**, then scan each of the 2^i locations in H and rehash into H' the 2^i items stored at these locations. This takes at most $c \cdot 2^i$ time, where $c > 0$ is a constant, which is linear in the number of insertions since H was created. Similarly, if H is replaced by a table H' of size 2^{i-2} there must have been at least 2^{i-2} insertions since H was created. To build H', we need to scan each of the 2^i locations in H and rehash into H' the 2^{i-2} items stored at these locations. This takes at most time $c' \cdot 2^i$), where $c' > 0$ is a constant, which is linear in the number of deletions since H was created. Summing over all table replacements, we see that the replacement cost is linear in the total number of insertions and deletions.