Homework 7, Solution set

1. a. As h_i was drawn uniformly at random from $\mathcal{H}$, for $x \notin S$ and $y \in S$, $\Pr\left[h_i(x) = h_i(y)\right] = 1/n$ by assumption. Let $S = \{y_1, y_2, \dots y_s\}$. By the union bound, $\Pr\left[h_i(x) = h_i(y_2) \text{ or } h_i(x) = h_i(y_2) \text{ or } \dots \text{ or } h_i(x) = h_i(y_s)\right] \le s/n \le 1/2$ as $n = 2s$. But this is the probability that $B_i[h_i(x)] = 1$.

b. Since the choices of the h_i are all independent, the probability that $B[h_i(x)] = 1$ for every $1 \le i \le m$ is at most $1/2^m$; this is the probability that x is reported as being in S when $x \notin S$.

2. Corresponding to the initial empty S, we begin by requesting a table of size $4 = 2^2$ for the first hash table and initializing its entries to **nil**. We will use hashing by chaining. The first level table will hash on size. For each size, we will keep a pointer to a second level hash table hashing on value. For each level 2 hash table we keep a counter of the number of items in the table, and a separate variable indicating its size (note this size has nothing to do with the size attribute). Likewise, for the level 1 table we keep a count of how many level 2 tables it points to, and a separate variable indicating its size.

As in additional problem 2, when a hash table of size 2^i is full, i.e. it holds 2^i items, we put the items in a hash table of size 2^{i+1} by drawing a hash function uniformly at random for the new table, and rehashing all 2^i items at hand. Likewise, whenever a table of size 2^i is 1/4 full, i.e. it holds $2^i/4$ items, we move all the items, 2^{i-2} of them, into a table of size 2^{i-1}. In both cases, when a new table is created it will be exactly half full, with the exception that a base case table (size 4), when replacing no table, will be 1/4 full.

A search is performed by hashing on size and then on value. Each hash takes expected $O(1)$ time for a total of expected $O(1)$ time. To perform an insertion, we start by inserting the item into the top level hash table, and then into the level 2 hash table. Similarly, for a deletion, we locate the item in the second level table, remove it from there, and if this table is now empty remove this level 2 table from the top level table. We also update counts for these two tables. These operations also take expected $O(1)$ time as they both involve just two hashes.

It remains to analyze the cost of rebuilding the tables. As in Problem 2 in the additional problems, the expected cost of rebuilding the level 2 tables is $O(m)$, , where m is the total number of insertions and deletions. We also have to bound the cost of creating new size 4 tables to replace previously not having a table at the relevant location. Each such creation follows a distinct insertion, so the cost of creating all these size 4 tables is also $O(m)$.

When we rebuild a level 1 table, the level 2 tables are kept unchanged. Again, all we have to do is to rehash the sizes present in the level 1 table. This will take time linear in the following quantity: the number of new sizes inserted in the level 1 table plus the number of sizes deleted from the level 1 table; this quantity is upper bounded by the number of items inserted plus the number of items deleted.

Thus the cost of creating all the tables is $O(m)$.

Finally, we note that the total size of the tables is $O(s)$ at all times. For each level 2 table of size 2^i is storing at least 2^{i-2} items, and the different level 2 tables store disjoint sets of items, Similarly, if the level 1 table has size 2^j, it will be storing pointers to at least 2^{j-2} level 2 hash tables each of which stores at least one item. So the total space used by the two levels of hash tables is $O(s)$.

3.a. Suppose for a contradiction that for some f and g with $1 \leq f < g < 2^{i+j}$, $f(x-y) = g(x-y) \bmod 2^{i+j}$. Then $(f-g)\cdot(x-y) = \lambda 2^{i+j}$ for some integer λ. As $f \neq g$ and $x \neq y$, $\lambda \neq 0$. As $x-y$ is odd, 2^{i+j} must divide $f-g$ exactly. But this is not possible as $1 \leq f < g < 2^{i+j}$. Therefore the values $x-y, 2(x-y), 3(x-y), \ldots, (2^{i+j})-1)\cdot(x-y) \bmod 2^{i+j}$ are all distinct. Furthermore none of these values is 0 (because otherwise if $f(x-y) = 0 \bmod 2^{i+j}$ for some $1 \leq f < 2^{i+j}$ then as in the previous argument, 2^{i+j} would divide f exactly, which is not possible). Therefore there is a single value α such that $\alpha(x-y) = 1$, and by definition $\alpha = (x-y)^{-1}$.

The condition $a(x-y) = c' - d' \bmod 2^{i+j}$ is the same as $a = (x-y)^{-1}(c'-d') \bmod 2^{i+j}$. Thus there is exactly one value of a satisfying this condition.

b. Given the choice of a from (a) above, to obtain $ax + b = c' \bmod 2^{i+j}$ we need $b = c' - ax \bmod 2^{i+j}$. Again, this choice is unique. So there is one pair (a, b). (Note that if $a(x-y) = c' - d' \bmod 2^{i+j}$ and $ax + b = c' \bmod 2^{i+j}$, then $ax + b = d' \bmod 2^{i+j}$.)

c. To obtain $h_{a,b}(x) = c$, we need $\lfloor c'/2^j \rfloor = c$. There are exactly 2^j such c'. Similarly to obtain $h_{a,b}(y) = d$, we need $\lfloor d'/2^j \rfloor = d$. There are exactly 2^j such d'. This yields 2^{2j} pairs $(c'd')$ satisfying both these conditions.

d. For each pair (c', d') satisfying $\lfloor c'/2^j \rfloor = c$ and $\lfloor d'/2^j \rfloor = d$, by parts (a) and (b), there is one pair (a, b) that yields $ax + b = c' \bmod 2^{i+j}$ and $ay + b = d' \bmod 2^{i+j}$; this occurs with probability $1/2^{2(i+j)}$. As there are 2^{2j} pairs (c', d') for which $\lfloor c'/2^j \rfloor = c$ and $\lfloor d'/2^j \rfloor = d$, the probability that $h_{a,b}(x) = c$ and $h_{a,b}(y) = d$ is $2^{2j}/2^{2(i+j)} = 1/2^{2i}$.