HW7

1. *a. For each item $x \notin S$ show that the probability that Bi[hi(x)] = 1 is at most 1/2.*

   > As mentioned in class, the probability that $h_j(x) \neq h_i(x)$ is $1 - \frac{1}{n}$. We take one of B_i into consideration, we only set s bits in each B_i. Therefore, the probability that there are no collisions with $h_i(x)$ is $(1 - \frac{1}{n})^s$. The probability that there is a collision is $1 - (1 - \frac{1}{n})^s$.
   >
   > Let $f(x) = (1 - \frac{1}{2x})^x$, $f'(x) = \left(\ln\left(1 - \frac{1}{2x}\right) + \frac{1}{2x-1}\right)\left(1 - \frac{1}{2x}\right)^x \geq 0$ when $x \geq 1$. Hence $f(x) >= f(1) = \frac{1}{2}$
   >
   > $$\begin{aligned} 1 - (1 - \frac{1}{n})^s &= 1 - (1 - \frac{1}{2s})^s \\ &\leq 1 - \frac{1}{2} \\ &= \frac{1}{2} \end{aligned} \tag{1}$$
   >
   > Hence, the probability that $B_i[hi(x)] = 1$ is at most $\frac{1}{2}$.

   b. Deduce that the probability that x ∈/ S yet x is reported as being in S is at most 1/2m.

   > The probability that $Bi[hi(x)] = 1$ is at most $\frac{1}{2}$. As there are m arrays for storing bits, the probability that there is a collision with every $h_i(x)$ is $(1 - (1 - \frac{1}{n})^s)^m$.
   >
   > $$\begin{aligned} (1 - (1 - \frac{1}{n})^s)^m &= (1 - (1 - \frac{1}{2s})^s)^m \\ &\leq (\frac{1}{2})^m \\ &= \frac{1}{2^m} \end{aligned} \tag{2}$$
   >
   > Hence, the probability that $x \notin S$ yet x is reported as being in S is at most $1/2^m$.

2. *Show how to implement the two level hash table so that each query takes expected O(1) time, and the m insertions and deletions require expected O(m) time in total. In addition, at all times the overall table size must be O(|S|).*

   > We use the method of rebuilding hash tables described in the additional problems to make sure the rebuilding after insertions and deletions will cost expected $O(m)$ in a single hash table.
   >
   > That is, we have families $H_i = \{h_{i1}, h_{i2}, \ldots\}$ of hash functions for hash tables of sizes 2^i for $i = 1, 2, 3, \ldots$ When size 2^i is full or is $1/4$ full, we rebuilt the hash tables to maintain the property.
   >
   > To begin with, we initialize the first-level hash table with $i = 1$, i.e., with size 2 and for each position in the first level, we initialize a second-level hash table with size 2, which will cost $O(1)$. (To make sure the size is what we need, we can use hash with chaining to make sure the expected $O(1)$ time for insertion and query by a family of hash functions, then the pointer will be stored in the doubly linked list to avoid collisions. So does the second level hash table)

In this case, when inserting an item, we first calculate $h_{i1}(s)$ to get the position in the doubly linked list in the first hash table, which cost $O(1)$. Then go to the second hash table by pointer and use $h_{i1}^{'}(v)$ to find the position in the second hash table, which cost $O(1)$ time. The total cost will be $O(1)$.

If querying the item, it will be similar to the insertion. Find the position by $h_{i1}(s)$ and go to the second hash table, then calculate $h_{i1}^{'}(v)$ to get the item. The total cost will be $O(1)$.

If inserting m items, we focus on the first level first. It is like inserting at most m different sizes into the hash table. The hash table will cost in expected $O(m)$ time by building and inserting, because each step is like querying 2^i items and insert them into a bigger table like 2^{i+1}. If $m = 2^i + c$, the size of the hash table will be 2^i. The procedure will be like inserting 2^1 first, then rebuild to 2^2, query 2^1, inserting 2^2, rebuild to 2^3 . . .So the total expected cost will be
$2^1 + 1 + 2^1 + 2^2 + 1 + 2^2 + 2^3 + \cdots = i + 2(1 + 2^1 + 2^2 + \cdots + 2^i) + c = i + 2^{i+2} + c = O(m)$

For the second level, there are at most m different value inserted into the second level. For each second hash table, there will be m_i items inserted into it, each of which will cost $O(m_i)$. Hence, the total cost will be $\sum_{i=0}^{m} cm_i = cm$, which is $O(m)$. Deletion is the same, the total expected cost will be less than $i + 2^{i+2} + c$, maybe $i + 2^i + c$, but still $O(m)$ time.

3. *The task is to show this family is strongly universal.*

   - Assume $x - y$ is odd, $(x - y)^{-1}$ exists when mod 2^{i+j}, i.e.,$(x - y)(x - y)^{-1} \equiv 1$ (mod 2^{i+j}):

     Let $exgcd(a, n)$ be the function of the extended Euclidean algorithm, then we can get $(x - y)z_1 + 2^{i+j}z_2 = g$, where g is the greatest common factor of $x - y$ and 2^{i+j}. As $x - y$ is odd, the greatest common factor of $x - y$ and 2^{i+j} is 1. Hence, $z_1 + k2^{i+j}(k \in Z)$ are all the inverse of $x - y$, so for values $x - y, 2(x - y), 3(x - y), \ldots, (2^{i+j} - 1) \cdot (x - y)$ mod 2^{i+j}, there is only one value that fits that solution when $k = 0$. As these values are increasing, $x \in [0 : 2^j - 1]$, then $x - y \in [0 : 2^j - 1]$. If there are two values are not distinct, i.e., there exists $a(x - y) = b(x - y)$ mod $2^{i+j}, 0 < b < 2^{i+j}$, then $(a - b)(x - y) = 0$ mod 2^{i+j}, so $(a - b)(x - y) = \lambda 2^{i+j}, \lambda \geq 1$. Both factors are less than 2^{i+j}. So it contradicts the assmption that they are not distinct. Hence, the values are distinct and there is exactly one of these term equals 1.

     Then, $a = (x - y)^{-1}(c' - d')$ mod 2^{i+j}. $(x - y)^{-1}$ is fixed, c' can choose from 0 to 2^{i+j}, so can d'. The difference of $(c' - d')$ is between 0 to 2^{i+j} as $a \geq 0$. Hence, there are 2^{i+j} choices of a.

     The probability $Pr[a = (x - y)^{-1}(c' - d')] = \frac{1}{2^{i+j}}, Pr[b = c' - ax] = \frac{1}{2^{i+j}}$.

   - Then, c', d' can be in any postion of the 2^{i+j}, so there are $2^{2(i+j)}$ pairs for either $ax + b = c'$ mod 2^{i+j} and $ax + b = d'$ mod 2^{i+j}

   - As for the desired conditions $h_{a,b(x)} = c$ and $h_{a,b}(y) = d$, c', d' will be divided by 2^j, which will shrink the range from 2^{i+j} to $2^{i+j}/2^j = 2^i$, so the pairs will be 2^i . Each location c in hash will store $\leq \lceil 2^{i+j}/2^i \rceil = \lceil 2^j \rceil$ values.

   - Hence, the probability that $h_{a,b}(x) = c$ and $h_{a,b}(y) = d$ when a and b are chosen uniformly at random is $\frac{1}{2^{2i}}$.

- When $x - y$ is even, let's say $x - y = 2^h t$, so for values $x - y, 2(x - y), 3(x - y), \ldots, (2^{i+j} - 1) \cdot (x - y)$ mod 2^{i+j} can be seen as $t, 2t, 3t, \ldots, (2^{i+j} - 1) \cdot t$ mod 2^{i+j-h}. In this case, the inverse of $(x - y)$ mod 2^{i+j} can be converted to the inverse of t mod 2^{i+j-h}. For $a(x - y) = c' - d'$ mod 2^{i+j}, $a = (x - y)^{-1}(c' - d')$ mod 2^{i+j} . Thus, the distinct values occurs 2^h times during the reverse , $(x - y)^{-1}$ will have multiply results. Hence, the one in the range $[0 : 2^{i+j-h}]$ will be satisfied.