Jinyang Ruan

011696096

CPT_S 515 Advanced Algorithms HW#5

1. H is universal, if $\forall x \neq y \in U, Prob[h(x) = h(y)] \leq \frac{1}{M}$, where h is a random variable on H.

   In this case, $M = 2$. H is universal if $\forall x \neq y \in U, Prob[h(x) = h(y)] \leq \frac{1}{2}$.

   We consider two keys $x = 00000000, and\ y = 00000001$, they differ in one digit.

   Thus, we have 7 hash functions mapping them into the same slot.

   $$Prob[h(x) = h(y)] \leq \frac{7}{8}$$

   Therefore, H is not universal.

2. According to the question, we have:

   $h_r(x) = (r \cdot x\ mod\ M), where\ r, x \in [M]^k$, and $r \cdot x = \sum_i r_i \cdot x_i$.

   Decompose keys into $k + 1$ digits, $x =< x_0, x_1, \dots, x_k >, where 0 \leq x_k \leq M - 1$, pick $r =< r_0, \dots, r_k >$, each r_k is chosen randomly from $[M]$. [1]

   All $h_r(x) = (r \cdot x\ mod\ M)$ constitute a family H of hash functions. The number of hash functions in H, called $|H| = M^{k+1}$.

   Let key $x =< x_0, \dots, x_k >$, key $y =< y_0, \dots, y_k >$ be distinct keys. They differ in at least one digit, without loss of generality, position 0.

   When $h_r(x) = h_r(y)$, we have:

   $$\sum_{i=0}^{k} r_i \cdot x_i\ mod\ M = \sum_{i=0}^{k} r_i \cdot y_i\ mod\ M$$

$$\sum_{i=0}^{k}(r_i \cdot x_i - r_i \cdot y_i) \equiv 0 \ (mod\ M)$$

$$r_0(x_0 - y_0) + \sum_{i=1}^{k}(r_i \cdot x_i - r_i \cdot y_i) \equiv 0 \ (mod\ M)$$

$$r_0(x_0 - y_0) \equiv -\sum_{i=1}^{k}(r_i \cdot x_i - r_i \cdot y_i) \ (mod\ M)$$

Number theory: Let m be a prime. For any $Z \in intergers\ mod\ m$, such that $Z \neq 0, \exists\ unique\ Z^{-1} \in intergers\ mod\ m$, such that $Z \cdot Z^{-1} \equiv 1\ (mod\ m)$. Since $x_0 \neq y_0$, $\exists\ (x_0 - y_0)^{-1}$. Thus, we have:

$$r_0 \equiv [-\sum_{i=1}^{k}(r_i \cdot x_i - r_i \cdot y_i)\ ](x_0 - y_0)^{-1}\ (mod\ M)$$

Thus, for any choice of $r_1, r_2, \dots, r_i$, exactly 1 of the M choices for r_0 causes x and y to collide, and no collision for other m-1 choices for r_0. The number of $h_r(x)$'s that cause x, y to collide $= M \cdot M \cdot M \dots \cdot 1 = M^k = \frac{M^{k+1}}{M} = \frac{|H|}{M}$. So, the family of hush functions is universal.

3. Input a graph G

   Step 1: Translate the graph to a Laplacian matrix $L = D - A$, where D is the degree matrix and A is the adjacency matrix of the graph.

   Step 2: Calculate the eigenvalues and spectrum of the Laplacian matrix.

   Step 3: The spectrum of the Laplacian matrix is always real. Decompose the spectrum of the matrix into $m + 1$ digits and hash it into a number.

4. One way: Use the Erdős-Rényi model $G(n, M)$, in which n represents the number of nodes and M represents the total number of edges in G. Assume that there is only one edge between 2 nodes.

   Step 1: In this case, $n = 5$ and $M_{max} = n * (n-1)/2 = 10$.

   Step 2: Use random generator $r(n)$ to generate a random number in $\{0,1, \dots ,10\}$

   Step 3: Randomly choose a pair of nodes without edge, add an edge between them, then $n = n - 1$.

   Step 4: Repeat step 3 until $n = 0$. Now we get a random graph with 5 nodes.

   Another way: Use the Erdős-Rényi model $G(n, p)$, in which n represents the number of nodes in G and p represents the independent occurrence probability of each edge. Assume that there is only one edge between 2 nodes.

   Step 1: Use $r(n)$ to generate a random number n in [1,100], $p = n/100$.

   Step 2: Label all pairs of nodes. Choose a pair of nodes we have never chosen before, Use $r(n)$ to generate a random number n in [1,100], $r = n/100$, if $r > p$, add an edge between these two nodes, otherwise, no edge added.

   Step 3: Repeat step 2 until all pairs of nodes have been chosen. Now we get a random graph with 5 nodes.

5. Use a new data structure called Forgetful Bloom Filter (FBF) to store S. [2]

   An FBF contains Bloom filters:

   i) a future Bloom filter,

   ii) a present Bloom filter,

   iii) a past Bloom filter.

   All these Bloom filters are equal in size and identical in their use of hash functions.

When an element needs to be inserted, it is first checked for membership in the FBF. If it is not present, it is inserted only into the future and present Bloom filters, but not in the past Bloom filter.

A membership check can be performed by checking if the tested element is present in at least one of the three constituent Bloom filters – if so, the check returns true. If the element is absent in all the three constituent filters, it is considered to be not present in the FBF. In this case, when Mr. X queries a place, check this place whether presents in one of the three constituent filters.

Algorithm:

Step 1: Insert elements into the FBF. Check every place nearby Mr. X if in the FBF, if not, store this place in present Bloom filter and future Bloom filter.

Step 2: In order to forget older elements, periodically (every t time units), an FBF undergoes a refresh operation. In a basic FBF, at a refresh point, the following operations are performed atomically:

The past Bloom filter is dropped,

The current present Bloom filter is turned into the new past Bloom filter,

The current future Bloom filter is turned into the new present Bloom filter,

A new, empty future Bloom filter is added to the FBF.

6. Given two arrays a and b of 10 bits.

One way: Step 1: Calculate the Hamming distance between two arrays $D(a, b)$. we perform their XOR operation, $(a \oplus b)$, and then count the total number of 1's in the resultant string. $D(a, b) \in \{0,1, \dots ,10\}$

Step 2: Consider the hush function for Hamming distance that maps a d-dimensional bit vector to its value on a random coordinate. We have:

$$Prob\big(h(a) = h(b)\big) = 1 - \frac{D(a,b)}{d}$$

The hash functions are more likely to be equal if the distance between two strings is smaller.

Step 3: Define the hash family H. For a parameter $c > 1$, probability values $p_1 > p_2$, and Hamming distance $D(a,b) \in \{0,1,\dots,10\}$, a hash family H is said to be (r, cr, p_1, p_2)-Locality Sensitive. $\forall q, a, b$

$$If\ D(a,b) < r, then\ Prob\big(h(a) = h(b)\big) \geq p_1, and$$

$$if\ D(a,b) > cr, then\ Prob\big(h(a) = h(b)\big) \leq p_2,$$

where the probability is over $h \in H$ chosen at random.

Step 4: Build such hash family H. Super-bit Locality-Sensitive hashing. (I am still thinking a way to build such hash family.)

Step 5: Randomly select $h \in H$, until

$$\sum_{j=0}^{N-1} n_j^2 \leq 4N$$

For each $j - th$ slot with n_j keys ($n_j > 1$), select h_j such that the n_j keys are hashed into n_j^2 slots without collision.

References

[1] C. Leiserson & E. Demaine. *MIT*. Retrieved from: http://open.163.com/newview/movie/free?pid=M6UTT5U0I&mid=M6V2TGI3A

[2] R. Subramanyam, I. Gupta, L. M. Leslie and W. Wang, "Idempotent Distributed Counters Using a Forgetful Bloom Filter," 2015 International Conference on Cloud and Autonomic Computing, Boston, MA, 2015, pp. 113-124, doi: 10.1109/ICCAC.2015.30.