Cpt S 515 Homework #5

No late homework!

1. Consider a family $\mathcal{H}$ of hash functions:

$$\mathcal{H} = \{h_i : 1 \leq i \leq 8\}.$$

Each h_i is to map an array of eight bits into its i-th component: $h_i(a_1 \cdots a_8) = a_i$. Is $\mathcal{H}$ universal? why or why not?

2. Here is a classic example of universal family of hash functions. Let M be a prime number and, as usual, $[M] = \{0, 1, \cdots, M-1\}$. Consider the following family of hash functions:

$$h_{\mathbf{r}}(\mathbf{x}) = \quad (\mathbf{r} \cdot \mathbf{x} \bmod M),$$

where $\mathbf{r}, \mathbf{x} \in [M]^k$ (where k is a given constant like 10), and $\mathbf{r} \cdot \mathbf{x} = \sum_i r_i x_i$. Show that the family of hash functions (for the given k) is universal.

3. So far, what we have learned about hasing is to hash an array of numbers into one number (e.g., locality sensitive hashing). Can you suggest a way to hash a graph into a number (which could be a real number)?

4. Randomized quicksort is a Las Vegas algorithm where the first step is to create a random permutation of the input arrayof numbers before the second step of running quicksort. Now, we assume that we have a high quality psuedo random generator $r(n)$ that will generate a random number in $1..n$. Please show how to generate a "random" graph with 5 nodes.

5. Mr. X drives on I-90 all the way from Pullman to New York (Let's assume that Pullman is Spoakne). On his car, there is a device that can suggest all the interesting places nearby that Mr. X might visit (and spend some money at these places of course). These places are stored in a set S and will be updated automatically while Mr. X is driving. Please suggest a way to implement the S so that Mr. X can query (e.g., "Is there a restrant nearby?", etc.). You shall use Bloom filter to store S. Feel free to look up papers on the Internet.

6. We know many ways to hash an array of integrs into a number. However, hash itself is loosy — that is, the function many not be one-to-one. Can you suggest a way to hash an array of 10 bits into a number such that a. the hash is one-to-one, and, b. the hash is locality sensitive (i.e., when the Hamming distance between two such arrays of 10 bits is small, then so is the distance between their hash values). (I have a terrific way to do this — but I wont tell you. You shall figure out your own ways to do this. This problem concerns a lot of fundamental applicational problems in computer science.)

Assignment 5

Sheryl Mathew (11627236)

November 17, 2018

1. Question 1

A family H of hash functions mapping U to [M] is called universal if for any two keys $x \neq y \in U$, we have $\Pr[h(x) = h(y)] \leq \frac{1}{M}$.

Here the family of hash functions, $H = \{h_i : 1 \leq i \leq 8\}$ where each h_i maps to an array to eight bits. Here M is 2 since bits consists of only 0 and 1. Therefore H is universal if $\Pr[h(x) = h(y)] \leq \frac{1}{2}$

Example 1: Consider 2 keys: 00000000 and 00000001. The 2 keys differ by 1 bit. Therefore, we have 7 hash functions which map them to the same bit.

$$\Pr[h(x) = h(y)] = \frac{7}{8}.$$

Example 2: Consider 2 keys: 00001000 and 00000001. The 2 keys differ by 2 bits. Therefore, we have 6 hash functions which map them to the same bit.

$$\Pr[h(x) = h(y)] = \frac{6}{8}.$$

In both examples 1 and 2, $\Pr[h(x) = h(y)] \geq \frac{1}{2}$. Therefore H is not universal.

2. Question 2

A family H of hash functions mapping U to [M] is called universal if for any two keys $x \neq y \in U$, we have $\Pr[h(x) = h(y)] \leq \frac{1}{M}$.

Here the family of hash functions, $h_r(x) = (r.x \ mod \ M)$, where $r, x \in [M]^k$. M is a prime number [M] = {0,1,… M-1}. k is a given constant.

Total number of hash functions in the family = M^{k+1}. This is because the total number of x is k+1 and each x has M possible values.

Let x and y be 2 distinct keys such that $x \neq y$.

$$x = < x_0, x_1, \dots x_k >$$

$$y = < y_0, y_1, \dots y_k >$$

If x and y collide then $h(x) = h(y)$

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

$$\sum_{i=0}^{k} r_i x_i \equiv \sum_{i=0}^{k} r_i y_i \bmod M$$

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

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

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

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

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

The number of possible $r_0's$ that can cause x and y to collide is M^k as each of the r's can take M possible values.

$$\Pr[h(x) = h(y)] = \frac{M^k}{M^{k+1}} = \frac{1}{M}$$

Therefore the family of functions is universal.

3. Question 3

Let Graph G of n vertices be the input.

Step 1: Create an adjacency matrix A of size nxn. 1 will represent the presence of an edge from one node to another and 0 will represent the absence of an edge from one node to another.

Step 2: Find the eigenspectrum of A. Scan the absolute values of the eigenspectrum to find the largest eigenvalue. The largest eigenvalue is the Perron number PF according to Perron-Frobenius theorem.

Step 3: Perform LSH on each row of A. Then perform LSH again on the generated n-dimensional vector. The output will be a numerical value.

Step 4: Both Steps 2 and 3 will convert a graph into a number

4. Question 4

The pseudo random generator r(n) will generate a random number between 1 and n.

Total number of node pairs for n nodes = $\frac{n(n-1)}{2}$

Total number of node pairs for 5 nodes = $\frac{5x4}{2} = 10$

Let n =10, the pseudo random generator r(10) will generate a random number between 1 and 10. Let the number returned by r(10) be x.

Step 1: Construct 5 nodes.

Step 2: For each potential edge between a pair of nodes, we find the value x

Step 3: If x>5 then we add an edge between the 2 nodes else we do not add an edge between the 2 nodes.

Step 4: Repeat Steps 2 and 3 for all possible pair of nodes.

Step 5: Random graph of 5 nodes is generated.

5. Question 5

Bloom filters is a probabilistic data structure that is used to test whether an element is a member of a set or not. False positive means that the query will return either possibly in set or definitely not in set. Elements can be added to the set but cannot be removed.

In our problem, the interesting places to visit our stored in set S and is updated automatically when Mr X is driving through I-90. If we keep adding all the places that he passes to set S, the size of set S will become very large. To overcome this we need to remove places from set S after a particular time period. Since removing elements is not possible from Bloom filter we use Forgetful Bloom filters (FBF). FBF automatically expires older items with the time period being adjustable. FBF uses 3 bloom filters: future, present and past to maintain a moving window of recent operations.

Step 1: For every place passed by Mr X, check if it is present in the past, present or future bloom filters.

Step 2: If it is present, ignore the place. If it is not present then add it to the present and future bloom filters.

Step 3: After a time period t, perform the below operations:

- Drop the past bloom filter
- Current present bloom filter becomes new past bloom filter
- Current future bloom filter becomes new present bloom filter
- New empty future bloom filter is added to FBF.

Step 4: A moving window FBF is created to store the places Mr X passes while driving.

Reference: Idempotent Distributed Counters Using a Forgetful Bloom Filter [Rajath Subramanyam, Indranil Gupta, Luke M Leslie, Wenting Wang]

6. Question 6

Hamming distance is used to measure the similarity between two arrays. Here each array is 10 bits.

The general family for LSH is $H: h: U \rightarrow S$ is (r_1, r_2, p_1, p_2) such that for all $p, p' \subset U$. If the distance between p, p' is at most r_1, then the probability of $h(p)$ and $h(p')$ being equal is greater than or equal to p_1. If the distance between p, p' is greater than r_2, then the probability of $h(p)$ and $h(p')$ being equal is less than or equal to p_2.

S is the set of 10-bit arrays $\{a_1, a_2 .. a_n\}$. Hamming distance be between s_i and s_k then $H = [h_1 | h_i(a_1, a_2 .. a_n) = a_i for\ c = 1,2,..n]$ where h_i is a function in the hash family and a_i is a bit array.

Step 1: Find all pairs with a small distance between p, p' based on the hash function.

Step 2: If they have a small distance, then the collision is allowed by insertion of each bit array as an element of a linked list. This allows similar bit arrays with same hash value to be kept together. This is called chaining. This generates a hash table M

Step 3: Create a large hash table M' of slot size n x m and making the load factor $1 - \frac{1}{m}$

Step 4: Scan the current table M for each slot with and without linked list values

Step 5: If the chained items in a slot have a consecutive distance of 1, them items are inserted into M' using the new function

Step 6: After placing each item into M', we get one-to-one mapping of all bit arrays with mapping to hash values done based on how small their hamming distance is.