Homework 2 Solution

1. *Sampling*

(20 points) Exercise 4.2.1 : *Suppose we have a stream of tuples with the schema*

Grades(university, courseID, studentID, grade)

Assume universities are unique, but a courseID is unique only within a university (i.e., different universities may have different courses with the same ID, e.g., "CS101") and likewise, studentID's are unique only within a university (different universities may assign the same ID to different students). Suppose we want to answer certain queries approximately from a 1/15th sample of the data. For each of the queries below, indicate how you would construct the sample. That is, tell what the key attributes should be.

(a) For each university, estimate the average number of courses per university.

(b) Estimate the fraction of students who have a GPA of 3.7 or more.

Solution:

As it is required in the problem that we need to use 1/15th sample of the original steam data to answer certain query questions, this is a sampling fixed proportion problem. We will construct a representative sample from the data. The basic process to deal with sampling fixed proportion problem is: ***use a hash function to generate random integers in [0...14] for each incoming tuple under certain tuple key attribute; store the tuple if the integer is 0, otherwise discard***. Therefore, the choice of key attribute is the most important step for different queries. For each key attribute we will then process each occurrence of it in the stream

(a) The query wants to generate average number of courses per university. For each tuple, the "university" field is unique, then we chose "university" as the key. To take a sample of 1/15th, we hash the key for each tuple to a random integer in 15 buckets from 0 to 14, and accept the tuple for the sample if the hash value is 0. Thus, we store only 1/15th of the tuples as the sample, and discard others. For each university in the sample, we can easily count the average number of courses.

(b) The query wants to estimate the fraction of students who have a GPA of 3.7 or more. Since the "studentID" is unique only within a university, it cannot be alone used to identify one tuple in the stream. In this case, we need to build a composite key, [university, studentID], to identify each tuple in the global domain. Therefore, for this case, we hash the composite key to a random integer in 15 buckets from 0 to 14, and accept the tuple for the sample if the hash value is 0. Thus, we store only 1/15th of the tuples as the sample and discard others. Then for each student in the sample, we can easily conduct query to calculate the fraction of students who have a GPA of 3.7 or more.

2. Bloom Filter

(15 points) Exercise 4.3.1 : *For the situation of our running example (8 billion bits, 1 billion members of the set S), calculate the false-positive rate if we use 3 and 5 hash functions? Briefly explain each step in your solution*

Solution:

As discussed in the text book, the false positive rate for k hash functions with sample size m and data size n is:

$$\left(1 - e^{-\frac{km}{n}}\right)^k$$

Therefore, for this example, if we use 3 hash functions, we have false positive rate of

$$\left(1 - e^{-\frac{3*1}{8}}\right)^3 = 0.03057935$$

If we use 5 hash functions, the false positive rate would be:

$$\left(1 - e^{-\frac{5*1}{8}}\right)^5 = 0.02167922$$

From the example from the book we know that the probability of a false positive with k=1 is 0.1175, and with k=2 is 0.0493. So we observe that we achieve the biggest decrease in the false positive rate with the first few additional hash functions. If we have a target rate of false positive that is acceptable for our application, increasing k is one way to achieve that. The second way to increase the size of the array n.

3. FM Algorithm

(10 points) Exercise 4.4.1:

Suppose our stream consists of the integers 3, 1, 4, 1, 5, 9, 2, 6, 5. Our hash functions will all be of the form h(x) = ax+b mod 32 for some a and b. You should treat the result as a 5-bit binary integer. Determine the tail length for each stream element and the resulting estimate of the number of distinct elements if the hash function is:

(a) h(x) = 2x + 1 mod 32.

(b) h(x) = 3x + 7 mod 32.

Solution:

For each element x, let r(x) be the number of tailing 0s in h(x), and let R = max(r(x)) over all the items x seen so far. Then the estimate number of distinct elements is equal to 2^R. Thus, we have the following result for the three different hash functions:

Hash Function		3	1	4	1	5	9	2	6	5	R	Estimate
h(x) = 2x+1 mod 32	h(x)	7	3	9	3	11	19	5	13	11	0	1
	r(x)	0	0	1	0	0	0	0	0	0		
h(x) = 3x+7 mod 32	h(x)	16	10	19	10	22	2	13	25	22	4	16
	r(x)	4	1	0	1	1	1	0	0	1		

Therefore, the estimated number of distinct elements for (a) is 1, for (b) is 16. The correct answer is 7. We conclude that both estimates are off.

It is true that this is a small sample so we cannot expect our estimation to be very precise. But what we see right away here is that the choice of the hash function is important. The first hash function maps x to only add integers so for each the number of trailing 0 will be 0 and our estimate will always be 1. The second hash function maps to odd and even integers, so we will get trailing 0 for some stream tuples.

If we try the third hash function from the exercise, we will see that it maps only to even integers and each tuple will get some trailing 0. That function will give the most accurate estimate of 8.

Hash Function		3	1	4	1	5	9	2	6	5	R	Estimate
h(x) = 4x mod 32	h(x)	6	2	8	2	10	18	4	12	10	3	8
	r(x)	1	1	3	1	1	2	2	2	1		

4. *DGIM Algorithm*

1) (15 points) Exercise 4.6.1

Suppose the window is as shown in Fig. 4.2. Estimate the number of 1's the last k positions, for k = (a) 5 (b) 15. In each case, how far off the correct value is your estimate?

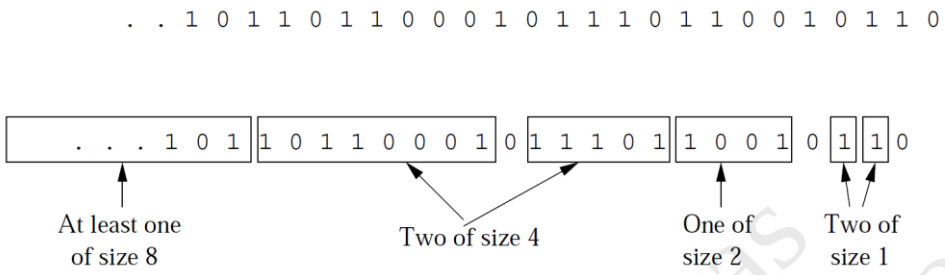

Figure 4.2: A bit-stream divided into buckets following the DGIM rules

Solution:

(a) If k = 5, the earliest bucket that at overlaps with k is the third bucket from the right. Thus, the estimate number of 1's would be sum of size of more recent bucket plus half size of the third bucket, which is:

1+1+0.5*2 = 3

The actual number of 1's for last 5 positions is 3, which means our estimate is equal to the correct value.

(b) If k = 15, the earliest bucket that at overlaps with k is the fifth bucket. Thus, the estimate number of 1's would be:

1+1+2+4+0.5*4 = 10

The actual number of 1's for last 15 positions is 9, which means our estimate is 11% far off the correct value.

2) **(20 points)** Study the example in section 4.6.7 **Extensions to the Counting of Ones**. Use the technique of Section 4.6.6 to estimate the total error. Show that if each c_i has fractional error at most *e*, then the estimate of the true sum has error at most *e*. Briefly explain each step in your solution.

Solution:

The worst case occurs when all the c_i's are overestimated or all are underestimated by the same fraction e.

We know from the error analysis for the original DGIM algorithm in sections 4.6.4 and 4.6.6 that the error e in estimating the sum will be e<=50% or e<=1/(1-r) of the

actual sum. In our case we apply the same counting algorithm for m streams. For each stream the error is bounded by e.

We know that each stream s_1, .., s_m counts one bit from the binary representation for the integers up to 2^m. We know the position of the bit in the binary representation that that stream s_i counts.

We know that the contribution of each stream count c_i to the sum of integers is c_i*$2^i$where i is the stream index, with the lower-end bits being counted in the stream with i=0.

If each bit overestimates the count, it's contribution will be (c_i+c_i*e)* 2^i. And so the total sum will be

$$S = \sum_{i=0}^{m-1}(c_i + c_i * e)2^i = \sum_{i=0}^{m-1}(c_i)2^i + \sum_{i=0}^{m-1}(c_i * e)2^i.$$

If each stream has its own over- or underestimation error e_i, then

$$\sum_{i=0}^{m-1}(c_i * e\_i)2^i \leq \sum_{i=0}^{m-1}(c_i * e)2^i = e\sum_{i=0}^{m-1} c_i * 2^i$$

Thus the worst case overestimation happens when all streams overestimate with e. And we also see that the total sum's overestimation is e:

$$\sum_{i=0}^{m-1}(c_i * e)2^i = e\sum_{i=0}^{m-1} c_i * 2^i = e*S.$$

Thus, the sum is overestimated with the same fraction e.

Similar calculations show this for underestimation.