CS 522

Homework Assignment 2

NAME: DHAYALINI NAGARAJ

A.NO :A20359686

1. Sampling

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) Estimate the average number of courses per university.

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

Explain briefly but clearly how you will create the sample and why.

1) **The average number of courses per university**

According to the given schema Grades, the universities are unique and courseID, studentID are not unique. In order to take sample of data we can combine two components to form a key attribute and then we can do sampling by doing hashing. The average number of courses per university deals with two components. One is University and the other is course. Universities are unique, so this component can be used as key attribute. Each course is linked with university. So the list of courses offered in each university can be obtained using the university. But each student may take the same course which results in courseID repetition. In order to avoid that we should query in such a way that only unique courseID are listed. So courseID can be combined with university as keyattribute. The corresponding course list can be used to calculate the average number of courses per university by doing sampling.

keyattribute (university, courseID)

2) The fraction of student who have a GPA of 3.7 or more

We have to find fraction of student who have a GPA of 3.7. It means we have to find the number of students who have GPA of 3.7 or more. Inorder to calculate the GPA of students, StudentID can be used. Using studentID we can get all the courses that a student has taken and its grade. This can be used to find the GPA of the student. StudentID is not unique and the same student ID may be assigned to different student in different university. The university component is a unique key which can be used to obtain all the studentID linked with the university. So University and StudentID is used as the key attribute.

keyattribute(university, studentID)

The 1/15th sample of data can be created by hashing the key value for each tuple to 15 buckets . That is to take sample of size a/b we hash the keyvalue for each tuple to b buckets. The tuple is accepted to the bucket if it is less than a.

2. Bloom Filter

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.

The bloom filter has

n bits of array, initially all are 0's

Collection of hash functions: h1, h2, h3.....hk

Set S with m key values.

Bloom filter is used to allow the stream elements whose keys are in S and to reject other elements whose keys are not in set S. Sometimes , the element whose key value is not in S may also be passed. So we have to determine the false positive rate. The probability of a false positive iss probability of a 1 bit raised to the k^{th}power $(1-e^{-km/n})^k$ where k is the no.of hash functions, n- n bits array, m- members of set(keys).

False positive rate for 3 hash functions

k= 3, n= 8 bit array (in billion), m=1 (in billion)

False positive rate $= (1-e^{-km/n})^k$

$= (1-e^{-3(1)/8})^3$

$= (1-e^{-0.375})^3$

$= (1-0.687289)^3$

$= (0.31271)^3$

$= 0.030579$

False positive rate for 5 hash functions

k= 5, n= 8 bit array (in billion), m=1 (in billion)

False positive rate $= (1-e^{-km/n})^k$

$= (1-e^{-5(1)/8})^5$

$= (1-e^{-0.625})^5$

$= (1-0.535261)^5$

$= (0.464739)^5$

$= 0.021679$

From the above calculation it is seen when the number of hash functions increase the false positive rate is decreased. Increasing hash functions further may not result in decreasing the false positive rate. This shows that the only specific number of hash functions must be used to reduce the false positive rate.

3. FM Algorithm

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.

Briefly explain each step in your solution.

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

Integers	h(x)=2x +1 mod 32	5- bit binary integer
3	7	00111
1	3	00011
4	9	01001
1	3	00011
5	11	01011
9	19	10011
2	5	00101
6	13	01101
5	11	01011
Tail Length R =0		

When we apply hash function 'h' to a stream element 'a' , then the bit string h(a) will end in some 0's or 1's. If the number ends in 0's call that as tail length of 'a' and 'h'. R is the maximum tail length of 'a' in stream. Then we can use 2^R to find the number of distinct elements seen in a stream.

The integers are hashed using hash function h(x) = 2x +1 mod 32 and tail length is found.
Here **R=0**. Therefore the **number of distinct elements is 2^0 =1**

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

Integers	h(x)=3x +7 mod 32	5- bit binary integer
3	16	10000
1	10	01010
4	19	10011
1	10	01010
5	22	10110
9	2	00010
2	13	01101
6	25	11001
5	22	10110
Tail Length R =4		

The integers are hashed using hash function h(x) = 3x +7 mod 32 and tail length is found.

Though many numbers end with 0's. The first number ends with 4 0's which is the maximum tail length in stream. So **R=4**. Therefore **the number of distinct elements in 2^4 = 16**

4. DGIM Algorithm

1) 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?

We divide the window into buckets consisting of

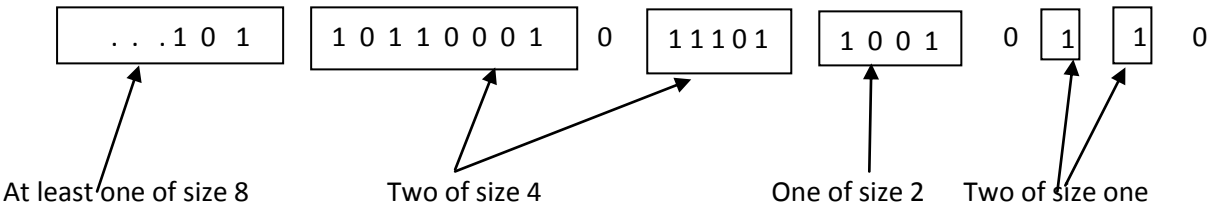
1. The timestamp of its right end.

2. The number of 1's in the bucket. The number should be power of 2.

The six rules that must be followed when we represent stream in buckets are

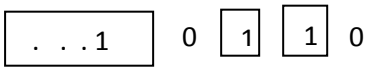
1) The right end of a bucket is always a position with 1.
2) Every position with 1 is in some bucket.
3) No position is in more than one bucket.
4) There are one or two buckets of any given size, up to some maximum size.
5) All sizes must be a power of 2.
6) Buckets cannot decrease in size as we move to the left .

The window shown in Fig 4.2 is



a) k =5

When k= 5 the bits are 10110. To represent the stream in buckets the above mentioned six rules has to be followed.



The 5 bits are represented in buckets. Each position represents the timestamp for each bit as it enters.

The number of 1's in the last k positions can be calculated from the sum of all the 1's present in the stream. So here three 1's are present in the stream. **No. of 1's is 3.**

Let the current timestamp be t. Then the two buckets with one 1, has timestamps t − 1 and t − 2 , the bucket of size 2 has timestamp t − 4, the bucket of size 4 has timestamp t-8.
Here we have the bucket of size 1 and bucket of size2. The two buckets of size 1 are considered as a whole. The bucket of size 2 with timestamp t-4 has only 1 bit. So this bucket is considered partially. So half of its bucket size will be considered.

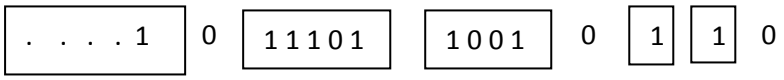
First Bucket of size 1, timestamp t-1	= 1
Second Bucket of size 1, timestamp t-2	= 1
Partially included third bucket of size 2, timestamp t-4 = 2/2	= 1

Total = 3

From the above result we get that the number of 1's in the last k positions is 3 and it is same in both the cases. The estimation is correct and there is no difference in it.

b) k = 15

When k= 15 the bits are 101110110010110. To represent the stream in buckets the above mentioned six rules has to be followed.



The 15 bits are represented in buckets.
To get the number of 1's in the last k positions let us add all the 1's in the stream. There are totally 9 1's in the bit stream. **No. of 1's is 9**.

Now let us calculate the 1's in the bucket based on the algorithm.

Let the current timestamp be t. Then the two buckets with one 1, has timestamps t − 1 and t − 2 , the bucket of size 2 has timestamp t − 4, t-7 and the bucket of size 4 has timestamp t-8, t-10, t-11, t-12 . The two buckets of size 1 are considered as a whole. The bucket of size 2 with timestamp t-4 and t-7 is considered as whole. The bucket of size 4 is also considered fully. The second bucket size of 4 has only 1 bit. So this bucket is considered partially. So half of the bucket size will be considered.

First Bucket of size 1 (timestamp t-1)	= 1
Second Bucket of size 1 (timestamp t-2)	= 1
Third Bucket of size 2 (timestamp t-4 & t-7)	= 2
Fourth Bucket of size 4 (timestamp t-8,t-10, t-11& t-12)	=4
Partially included fifth bucket of size 4 = 4/2	= 2

Total = 10

Thus the number of 1's in the last k positions is 10 here. Though the correct answer is 9, the estimated no. of 1's is 10 is not far off the correct value and it differs only by 1 count.

2) 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 ci 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.

Extensions to the Counting of Ones and Reducing the error.

Let us break the stream into buckets and represent the buckets by the sum of the integers in the buckets rather than the count of 1's. If bucket b is partially in the query range in which half of it is positive integers and other half is negative integers and sum of it is 0. Then bucket b's contribution would be 0. But actually it will contribute anything from 0 to sum of all the positive integers. So estimating the count using this method is meaningless.

So in order to estimate correctly let us consider that our stream has positive integers in the range of **1 to 2^m** for some 'm' where each of the m bits of each integer is treated as a separate stream. Now if we use DGIM method to count the 1's in each bit, then each bit will have a count and it is represented as C_i **.** Now the total sum of the integers is $\sum_{i=0}^{m-1} c_i 2^i$.

Now let us use 'Reducing the error' for this stream. In this we use r-1 or r of each size of buckets for some integer r>2. We relax this condition for buckets of size 1 and buckets of largest size present. If we get r+1 buckets of size 2^j combine leftmost two buckets to form a bucket of size 2^{j+1} and that in turn continues combining buckets of larger size.

Here we treat each m bit as a stream. So each i bit stream will have 2^j buckets where r-1 or r of each size bucket is present. For each bit we have to find the true count and the fractional error.

The largest relative error occurs when only one 1 from the leftmost bucket b is within the query range. Let us take that bucket b is of size 2^j for i^{th} bit. **The true count c_i is $1+(r-1)(2^{j-1}+2^{j-2}+\ldots+1)= 1+(r-1)(2^j-1)$, over estimate is $2^{j-1}-1$. Thus fractional error for each true count c_i is $2^{j-1}-1/1+(r-1)(2^j-1)$**. By picking large r, error is limited to at most e.

The total sum of integers is $\sum_{i=0}^{m-1} c_i 2^i$ then the fractional error of the sum would be total sum of fractional errors of each true count.

So the **total sum of integers true count will be** $\sum_{i=0}^{m-1}(1+(r-1)(2^j-1))2^i$ where 2^j size of the bucket of i bit.

The over estimate is $\sum_{i=0}^{m-1}(2^{j-1}-1)2^i$

Thus the **fractional error of the sum of integers will be**

$$\sum_{i=0}^{m-1}(2^{j-1}-1)2^i \,/\, \sum_{i=0}^{m-1}(1+(r-1)(2^j-1))2^i$$

This further reduces to **$2^{j-1}-1/1+(r-1)(2^j-1)$.** The j value can be any value , this fraction is upper bounded by 1/(r − 1). Thus, by picking sufficiently large r , error can be limited to at most e.