Big Data Analytics
Experiment No. 08

Title : Implementation of FM Algorithm.

Flajolet Martin (FM) Algorithm:

Flajolet Martin Algorithm, also known as FM algorithm, is used to approximate the number of unique elements in a data stream or database in one pass. The highlight of this algorithm is that it uses less memory space while executing.

The Flajolet-Martin algorithm uses the position of the rightmost set and unset bit to approximate the count-distinct in a given stream. The two seemingly unrelated concepts are intertwined using probability. It uses extra storage of order **O(log m)** where m is the number of unique elements in the stream and provides a practical estimate of the cardinalities.

Pseudo Code-Stepwise Solution:

1. Selecting a hash function h so each element in the set is mapped to a string to at least $\log_2 n$ bits.
2. For each element x, r(x)= length of trailing zeroes in h(x)
3. R= max(r(x))
   => Distinct elements= 2^R

Reasons for using Flajolet Martin algorithm:

Let us compare this algorithm with our conventional algorithm using python code. Assume we have an array*(stream in code)* of data of length 20 with 8 unique elements. Using the brute force approach to find the number of unique elements in the array, each element is taken into consideration. Another array*(st_unique in code)* is formed for unique elements. Initially, the new array is empty*(st_unique length equals zero)*, so naturally, the first element is not present in it. The first element is considered to be unique as it does not exist in the new array and thus a copy of the first element is inserted into the new array*(1 is appended to st_unique)*.

Similarly, all the elements are checked, if they are already present in the new array, they are not considered to be unique, else a copy of the element is inserted into the new array. Running the brute force algorithm for our array, we will get 8 elements in the new array. If each element takes 20 bytes of data, the new array will take 8*20= 160 bytes memory to run the algorithm.

Example:

Data stream={1,4,2,1,2,4,4,4,1,2,4,7}

Hash Function=(3X+1)mod 5

Answer:-

Mod; A%B= A-A/B*B

|**b**|=bit string length= 3

For X=1, ((3*1)+1) mod 5 =4

X	**h(X)**	**Binary Number**	**Trailing Number[r(a)]**
1	4	100	2
4	3	011	0
2	2	010	1
7	2	010	1

R= max[r(a)] = 2

Number of distinct Element (DE) = 2^R

= 2^ 2

= 4.

Python Code:

```
data = [4,2,5,9,1,6,3,7]
print("Hash functions are defined as (a*x+b)\%c, where x is an element of the set.")
inputCount = int(input("Enter the number of hash functions: "))
abcList = []

for i in range(inputCount):
	inputList = input("Enter the space-separated values of a, b and c: ").split(" ")
	abcList.append([int(i) for i in inputList])
finalCountsRecorded = []

for i in abcList:
	binElems = []
	#Evaluates the hash function, then converts it to binary.
	for j in set(data):
		#Appends binary output to a list

		binElems.append(str(bin((i[0]*j+i[1])%i[2])).split("b")[1])
	greatestTrailing = 0
	#Processes every element for that specific hash

	for k in binElems:

		reversedCount = k[::-1]
		count = 0
		for i in reversedCount:

			if(i=='1'):
				if(count>greatestTrailing):
					#The greatest number of trailing zeros are established
					greatestTrailing = count
				break
			else:
				count+=1
	#The formula R = 2^r is applied, where R is number of elems,and r is max
	#trailing zeros recorded
	finalCountsRecorded.append(2**greatestTrailing)

print("Counts recorded for each hash: ",finalCountsRecorded)
```

Drawbacks of the algorithm:

It is important to choose the hash parameters wisely while implementing this algorithm as it has been proven practically that the FM Algorithm is very sensitive to the hash function parameters. The hash function used in the above code is of the form ax+b mod c where x is an element in the stream and a,b,c are 1,6,32 respectively where a,b,c are the hash function parameters. For any other values of a,b,c the algorithm may not give the same result. Thus, it is important to observe the stream and then assign proper values to a,b and c.

Conclusion:

This approach is used to maintain a count of distinct values seen so far, given a large number of values. For example, getting an approximation of the number of distinct URLs surfed by a person on the web. Many companies want to check how many unique users logged in to their website the previous day to check if their advertisement was successful or not. Here, the FM algorithm is an excellent solution for these companies.