



Experiment No.7
Data Stream Algorithms: Implement Flajolet Martin algorithm using any programming Language
Date of Performance:
Date of Submission:



**Aim:** Data Stream Algorithms:

Implement Flajolet Martin algorithm using any programming language

**Theory:**

Flajolet-Martin algorithm approximates the number of unique objects in a stream or a database in one pass. If the stream contains  $n$  elements with  $m$  of them unique, this algorithm runs in  $O(n)$  time and needs  $O(\log(m))$  memory.

Algorithm:

1. Create a bit vector (bit array) of sufficient length  $L$ , such that  $2^L > n$ , the number of elements in the stream. Usually a 64-bit vector is sufficient since 264 is quite large for most purposes.
2. The  $i$ -th bit in this vector/array represents whether we have seen a hash function value whose binary representation ends in  $0^i$ . So initialize each bit to 0.
3. The  $i$ -th bit in this vector/array represents whether we have seen a hash function value whose binary representation ends in  $0^i$ . So initialize each bit to 0.
4. The  $i$ -th bit in this vector/array represents whether we have seen a hash function value whose binary representation ends in  $0^i$ . So initialize each bit to 0.
5. Once input is exhausted, get the index of the first 0 in the bit array (call this  $R$ ). By the way, this is just the number of consecutive 1s (i.e. we have seen  $0, 00, \dots, 0^{R-1}$  as the output of the hash function) plus one.
6. Calculate the number of unique words as  $2^R / \phi$ , where  $\phi$  is 0.77351. A proof for this can be found in the original paper listed in the reference section.
7. The standard deviation of  $R$  is a constant:  $\sigma(R) = 1.12$ . (In other words,  $R$  can be off by about 1 for  $1 - 0.68 = 32\%$  of the observations, off by 2 for about  $1 - 0.95 = 5\%$  of the observations, off by 3 for  $1 - 0.997 = 0.3\%$  of the observations using the Empirical rule of statistics). This implies that our count can be off by a factor of 2 for 32% of the observations, off by a factor of 4 for 5% of the observations, off by a factor of 8 for 0.3% of the observations and so on.



**CODE:**

```
stream=[1,2,3,4,5,6,4,2,5,9,1,6,3,7,1,2,2,4,2,1]
print('Using Flajolet Martin Algorithm:')maxnum=0
for i in range(0,len(stream)):
    val= bin((1*stream[i] + 6) % 32)[2:]sum=0
    for j in range(len(val)-1,0,-1):if val[j]=='0':
        sum+=1 else:
            break
    if sum>maxnum:maxnum=sum
print('distinct elements', 2**maxnum)
```

**Output:**

Using Flajolet Martin Algorithm:distinct elements 8

**CONCLUSION:**

The Flajolet-Martin algorithm is a powerful and efficient data stream algorithm used for estimating the cardinality of large sets or counting distinct elements in massive data streams. It relies on probabilistic methods and bitwise operations to provide an approximate count, making it well-suited for scenarios where memory and computational resources are limited. By processing data in a single pass, it offers scalability and speed. Implementing the Flajolet-Martin algorithm in a programming language, such as Python or Java, can enable the rapid estimation of unique elements in real-time data streams, making it a valuable tool for big data analytics and applications like network traffic monitoring and recommendation systems.