Big Data Analytics

Experiment No: 6

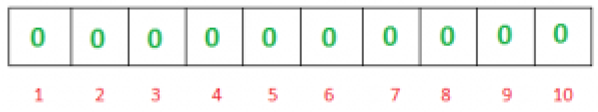
Title:- Implement Bloom Filter using any programming language.

Theory:-

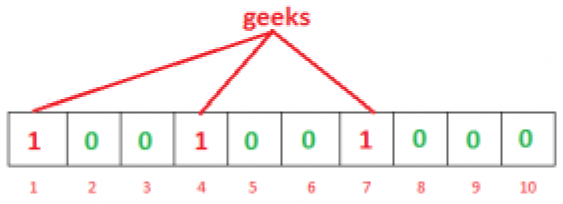
A Bloom filter is a space-efficient probabilistic data structure that is used to test whether an element is a member of a set. For example, checking availability of username is a set membership problem, where the set is the list of all registered usernames. The price we pay for efficiency is that it is probabilistic in nature that means, there might be some False Positive results. False positive means, it might tell that a given username is already taken but actually it's not.

Working of Bloom Filter:-

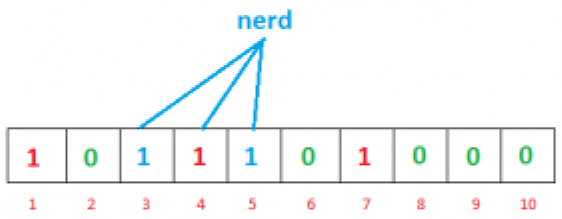
An empty bloom filter is a bit array of m bits, all set to zero, like this –



We need k number of hash functions to calculate the hashes for a given input. When we want to add an item in the filter, the bits at k indices h1(x), h2(x), … hk(x) are set, where indices are calculated using hash functions.
Example – Suppose we want to enter "geeks" in the filter, we are using 3 hash functions and a bit array of length 10, all set to 0 initially. First we'll calculate the hashes as following:
h1("geeks") % 10 = 1
h2("geeks") % 10 = 4
h3("geeks") % 10 = 7
Note: These outputs are random for explanation only.
Now we will set the bits at indices 1, 4 and 7 to 1



Again, we want to enter "nerd", similarly we'll calculate hashes
h1("nerd") % 10 = 3
h2("nerd") % 10 = 5
h3("nerd") % 10 = 4
Set the bits at indices 3, 5 and 4 to 1



Now we want to check if "geeks" is present in the filter or not. We'll do the same process but this time in reverse order. We calculate respective hashes using h1, h2 and h3 and check if all these indices are set to 1 in the bit array. If all the bits are set then we can say that "geeks" is probably present. If any of the bits at these indices are 0 then "geeks" is definitely not present.

False Positive in Bloom Filters

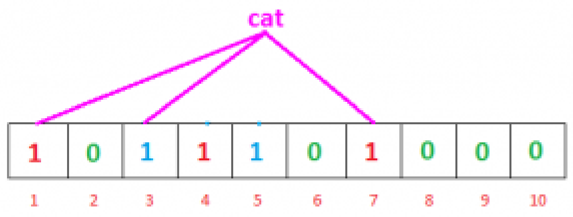
The question is why we said "probably present", why this uncertainty. Let's understand this with an example. Suppose we want to check whether "cat" is present or not. We'll calculate hashes using h1, h2 and h3

h1("cat") % 10 = 1

h2("cat") % 10 = 3

h3("cat") % 10 = 7

If we check the bit array, bits at these indices are set to 1 but we know that "cat" was never added to the filter. Bit at index 1 and 7 was set when we added "geeks" and bit 3 was set when we added "nerd".



So, because bits at calculated indices are already set by some other item, bloom filters erroneously claim that "cat" is present and generate a false positive result. Depending on the application, it could be a huge downside or relatively okay.

EXAMPLE

m=5

h1(x) = x mod 5
h2(x) = (2x+3) mod 5

x	h1(x)	h2(x)	B
Insert - 9	4	1	0 1 0 0 1 0 1 2 3 4
Insert - 11	1	0	1 1 0 0 1 0 1 2 3 4

1	1	0	0	1
0	1	2	3	4

1. **Query:- 15**

   h1(15) = 0
   h2(15) = 3

   15 not present

2. **Query:- 16**

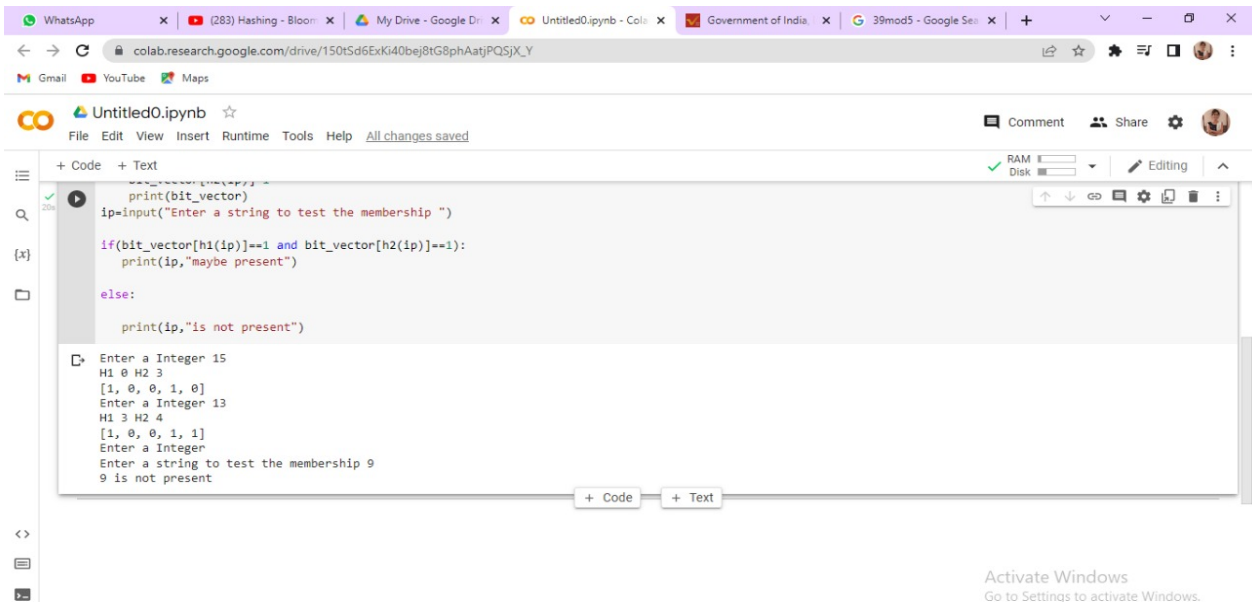
   h1(16) = 1
   h2(16) = 0

   16 maybe present

Program:-

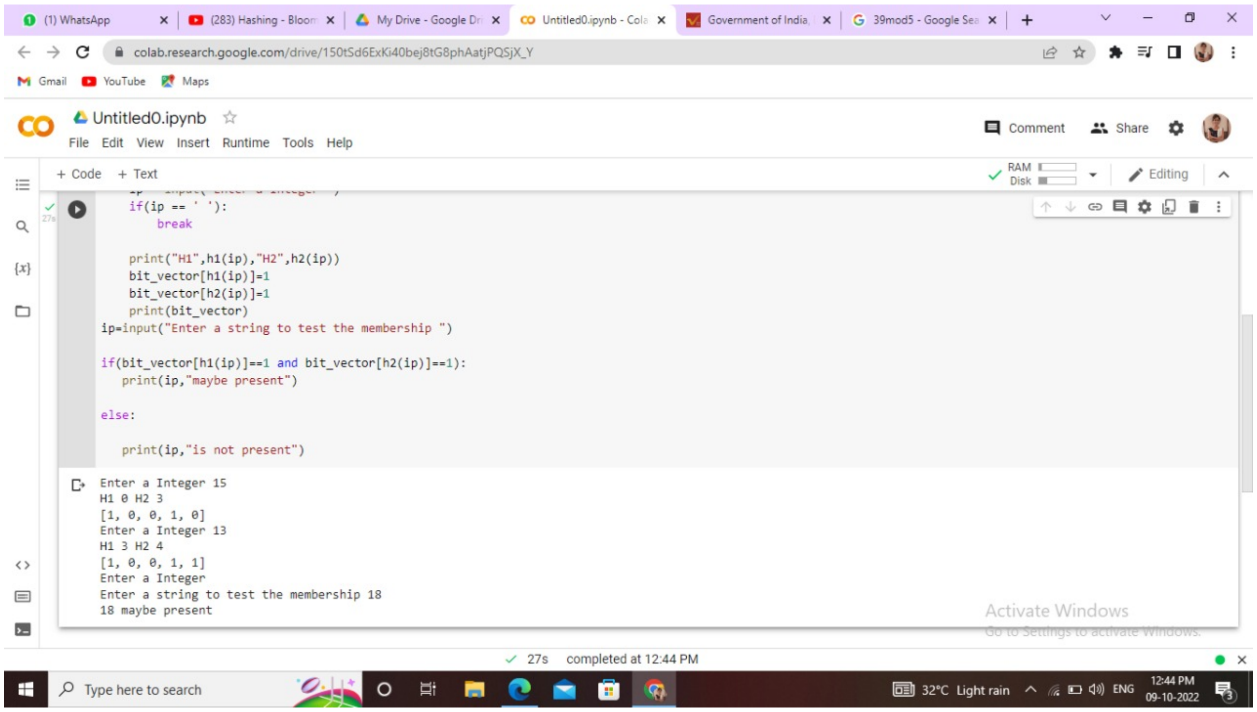
```
def sum(ip):
    s=0
    for i in ip:
      s = int(i)
    return s

def h1(ip):
    s = sum(ip)
    return (s)%5

def h2(ip):
    s = sum(ip)
    return (2*s+3)%5
bit_vector = [0, 0, 0, 0, 0]

while True:
    ip = input("Enter a Integer ")
    if(ip == ' '):
        break

print("H1",h1(ip),"H2",h2(ip))
    bit_vector[h1(ip)]=1
    bit_vector[h2(ip)]=1
    print(bit_vector)
ip=input("Enter a string to test the membership ")

if(bit_vector[h1(ip)]==1 and bit_vector[h2(ip)]==1):
   print(ip,"maybe present")

else:

   print(ip,"is not present")
```

OUTPUT:-



```
    print(bit_vector)
ip=input("Enter a string to test the membership ")

if(bit_vector[h1(ip)]==1 and bit_vector[h2(ip)]==1):
   print(ip,"maybe present")

else:

   print(ip,"is not present")
```

```
Enter a Integer 15
H1 0 H2 3
[1, 0, 0, 1, 0]
Enter a Integer 13
H1 3 H2 4
[1, 0, 0, 1, 1]
Enter a Integer
Enter a string to test the membership 9
9 is not present
```



```
    if(ip == ' '):
        break

    print("H1",h1(ip),"H2",h2(ip))
    bit_vector[h1(ip)]=1
    bit_vector[h2(ip)]=1
    print(bit_vector)
ip=input("Enter a string to test the membership ")

if(bit_vector[h1(ip)]==1 and bit_vector[h2(ip)]==1):
   print(ip,"maybe present")

else:

   print(ip,"is not present")
```

```
Enter a Integer 15
H1 0 H2 3
[1, 0, 0, 1, 0]
Enter a Integer 13
H1 3 H2 4
[1, 0, 0, 1, 1]
Enter a Integer
Enter a string to test the membership 18
18 maybe present
```

Conclusion: We studied and implemented Bloom Filter in Python.