EXPERIMENT NO: 7

AIM: implement Bloom Filter using python / R

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 set membership problem, where the set is the list of all registered username. A Bloom filter is a probabilistic data structure that plays a significant role in big data analytics by providing an efficient way to test whether a given element is a member of a set. It is a memory-efficient and scalable tool commonly used in various applications within the realm of distributed and big data systems. 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 given username is already taken but actually it's not.

Interesting Properties of Bloom Filters

- Unlike a standard hash table, a Bloom filter of a fixed size can represent a set with an arbitrarily large number of elements.
- Adding an element never fails. However, the false positive rate increases steadily as elements are added until all bits in the filter are set to 1, at which point all queries yield a positive result.
- Bloom filters never generate **false negative** result, i.e., telling you that a username doesn't exist when it actually exists.
- Deleting elements from filter is not possible because, if we delete a single element by clearing bits at indices generated by k hash functions, it might cause deletion of few other elements. Example – if we delete "geeks" (in given example below) by clearing bit at 1, 4 and 7, we might end up deleting "nerd" also Because bit at index 4 becomes 0 and bloom filter claims that "nerd" is not present.

CODE:

```
import java.util.Scanner;

public class StreamData {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the length of the stream data (m): ");
        int m = scanner.nextInt();
        int[] dataStream = new int[m];

        System.out.print("Enter the number of inputs to train the array (n):
");
        int n = scanner.nextInt();

        // Training the array
        for (int i = 0; i < n; i++) {
            System.out.print("Enter a number to train the array: ");
            int input = scanner.nextInt();
            int hash1 = (input % 5) % m;
            int hash2 = ((2 * input + 3) % 5) % m;

            if (dataStream[hash1] != 1) {
                dataStream[hash1] = 1;
            }
            if (dataStream[hash2] != 1) {
                dataStream[hash2] = 1;
            }
        }

        // Print the training array
        System.out.print("Training Array: ");
        for (int i = 0; i < m; i++) {
            System.out.print(dataStream[i] + " ");
        }
        System.out.println();

        System.out.print("Enter the number of times to test (x): ");
        int x = scanner.nextInt();

        // Testing the array
        for (int i = 0; i < x; i++) {
            System.out.print("Enter a number to test: ");
            int testValue = scanner.nextInt();
            int hash1 = (testValue % 5) % m;
            int hash2 = ((2 * testValue + 3) % 5) % m;
```

```
            if (dataStream[hash1] == 0 && dataStream[hash2] == 0) {
                System.out.println("Number is not in the stream.");
            } else if (dataStream[hash1] == 1 && dataStream[hash2] == 1) {
                System.out.println("Number is in the stream.");
            } else {
                System.out.println("Value not present in the stream.");
            }
        }

        scanner.close();
    }
}
```

OUTPUT:

```
"C:\Program Files\Java\jdk-20\bin\java.exe" "-javaagent:C:\Prog
Enter the length of the stream data (m): 5
Enter the number of inputs to train the array (n): 2
Enter a number to train the array: 9
Enter a number to train the array: 11
Training Array: 1 1 0 0 1
Enter the number of times to test (x): 2
Enter a number to test: 16
Number is in the stream.
Enter a number to test: 15
Value not present in the stream.

Process finished with exit code 0
```