Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Department of Computer Science and Engineering (Data Science)

Subject: Machine Learning – IV Laboratory AY: 2024-25

Name:Jhanvi Parekh
Sap:60009210033
Batch:D11

Experiment 3

Aim: Implement and analyze a Bloom Filter for checking the presence of elements, counting false positives, and true negatives.

Theory:

Introduction:

A Bloom Filter is a space-efficient probabilistic data structure used to test whether a given element is a member of a set or not. It is particularly useful when dealing with large datasets and is designed for fast and efficient lookups. However, a Bloom Filter may produce false positives (indicating an element is present when it's not) but never produces false negatives (if it says an element is not present, it's true). The main idea behind a Bloom Filter is to use multiple hash functions to map elements to multiple positions in a bit array.

Bloom Filter Workflow:

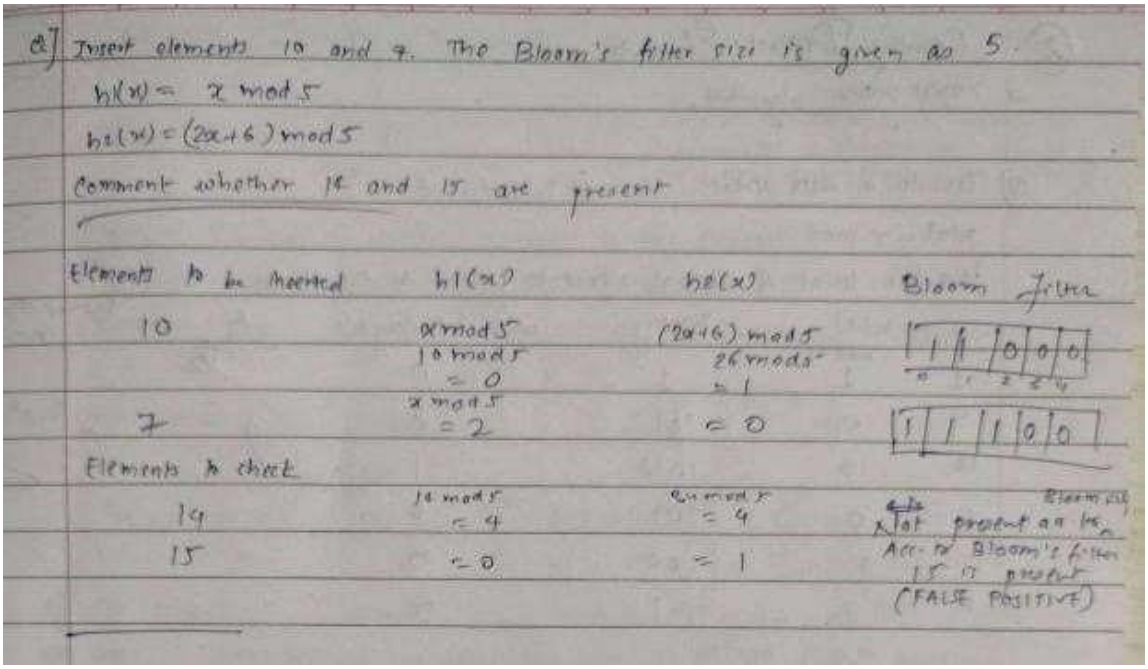
1. Initialization:
    - Create a bit array of 'm' bits, initialized with all zeroes.
    - Choose 'k' independent hash functions, each mapping to one of the 'm' bit positions.
2. Insertion:
    - To add an element to the Bloom Filter, apply each of the 'k' hash functions to the element and set the corresponding bit positions in the bit array to 1.
3. Membership Check (Query):
    - To check if an element is present in the Bloom Filter, apply each of the 'k' hash functions to the element.
    - If all 'k' bit positions are 1, the element is in the set (may have false

positives). Counting False Positives and True Negatives:

- False Positives: These occur when the Bloom Filter incorrectly indicates that an element is in the set when it's not. False positives can happen because of hash collisions or if other elements have set the same bit positions.
- True Negatives: These occur when the Bloom Filter correctly indicates that an element is not in the set. Bloom Filters guarantee that false negatives do not occur.

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)
Department of Computer Science and Engineering (Data Science)

Sample problem:



Lab Exercise:

Step 1: Initialize a Bloom Filter with a specified size 'm' and the number of hash functions 'k'.

Step 2: Insert a set of elements into the Bloom Filter using the 'k' hash functions.

Step 3: Test the presence of various elements in the Bloom Filter and observe if there are any false positives.

Step 4: Count and record the number of false positives and true negatives.

[Carry out application of a Bloom's filter, like checking if a word is present or not in a document provided time permits]

Code with Output:

```
[8]  1  !pip install mmh3
     2  !pip install bitarray
```

```
Requirement already satisfied: mmh3 in /usr/local/lib/python3.10/dist-packages (4.1.0)
Requirement already satisfied: bitarray in /usr/local/lib/python3.10/dist-packages (2.9.2)
```

```
 1
 2  import mmh3
 3  from bitarray import bitarray
 4
 5  class BloomFilter:
 6      def __init__(self, size, hash_count):
 7          self.size = size
 8          self.hash_count = hash_count
 9          self.bit_array = bitarray(size)
10          self.bit_array.setall(0)
11
12      def add(self, item):
13          for i in range(self.hash_count):
14              # Generate hash value for each hash function
15              digest = mmh3.hash(item, i) % self.size
16              self.bit_array[digest] = 1
17
18      def check(self, item):
19          for i in range(self.hash_count):
20              digest = mmh3.hash(item, i) % self.size
21              if self.bit_array[digest] == 0:
22                  # If any bit is not set, item is definitely not in filter
23                  return False
24          return True
25
```

```python
# Parameters
bloom_filter_size = 10000  # Size of the bit array
hash_count = 10 # Number of hash functions

# Initialize Bloom Filter
bloom = BloomFilter(size=bloom_filter_size, hash_count=hash_count)

# Insert a set of elements
elements_to_add = ["apple", "banana", "cherry", "date", "elderberry"]

for element in elements_to_add:
    bloom.add(element)

print("Elements added to Bloom Filter.")

```

```
Elements added to Bloom Filter.
```

```python
# Test elements
test_elements = ["apple", "grape", "watermelon", "banana", "fig"]

for element in test_elements:
    result = bloom.check(element)
    print(f"'{element}' is in Bloom Filter: {result}")

```

```
'apple' is in Bloom Filter: True
'grape' is in Bloom Filter: False
'watermelon' is in Bloom Filter: False
'banana' is in Bloom Filter: True
'fig' is in Bloom Filter: False
```

```python
# Example document
document = """A Bloom Filter is a space-efficient probabilistic data structure used to test whether a given element is a
member of a set or not. It is particularly useful when dealing with large datasets and is designed for fast
and efficient lookups. However, a Bloom Filter may produce false positives (indicating an element is
present when it's not) but never produces false negatives (if it says an element is not present, it's true).
The main idea behind a Bloom Filter is to use multiple hash functions to map elements to multiple
positions in a bit array. """

```

```python
# Add words to Bloom Filter
# Parameters
bloom_filter_size = 10000  # Size of the bit array
hash_count = 10 # Number of hash functions
bloom = BloomFilter(size=bloom_filter_size, hash_count=hash_count)
words = document.lower().split()
for word in words:
    bloom.add(word)

# Check for a word in the document
word_to_check = "bloom"
if bloom.check(word_to_check):
    print(f"The word '{word_to_check}' might be in the document.")
else:
    print(f"The word '{word_to_check}' is definitely not in the document.")

```

```
The word 'bloom' might be in the document.
```

```python

word_to_check = "cat"
if bloom.check(word_to_check):
    print(f"The word '{word_to_check}' might be in the document.")
else:
    print(f"The word '{word_to_check}' is definitely not in the document.")
```

```
The word 'cat' is definitely not in the document.
```

Conclusion:Thus, gained hands-on experience estimating cardinality, counting false positives, and analysing true negatives using Bloom’s Filter.