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 INFORMATION TECHNOLOGY

COURSE CODE: DJ19ITL502 **DATE:6/12/22**
COURSE NAME: Advanced Data Structures Laboratory **CLASS/Div: A**
SAP ID:60003200076 **Academic Year 2022-23**

EXPERIMENT NO. 8

CO/LO: LO1

AIM: To Implement Bloom Filter and Count min sketch

DESCRIPTION OF EXPERIMENT:

Bloom Filter :

A bloom filter is a probabilistic data structure that is based on hashing. It is extremely space efficient and is typically used to add elements to a set and test if an element is in a set. Though, the elements themselves are not added to a set. Instead a hash of the elements is added to the set.

When testing if an element is in the bloom filter, false positives are possible. It will either say that an element is definitely not in the set or that it is possible the element is in the set.

A bloom filter is very much like a hash table in that it will use a hash function to map a key to a bucket. However, it will not store that key in that bucket, it will simply mark it as filled. So, many keys might map to same filled bucket, creating false positives.

Count min sketch :

Basically, CM sketch works by aggregating the count of all items in your dataset into several counter arrays. Upon a query, it returns the item's minimum count of all arrays, which promises to minimize the count inflation caused by collisions. Items with a low appearance rate or score ("mouse flows") have a count below the *error rate*, so you lose any data about their real count and they are treated as noise. For items with a high appearance rate or score ("elephant flows"), simply use the count received. Considering CM sketch's size is constant and it can be used for an infinite number of items, you can see the potential for huge savings in storage space.

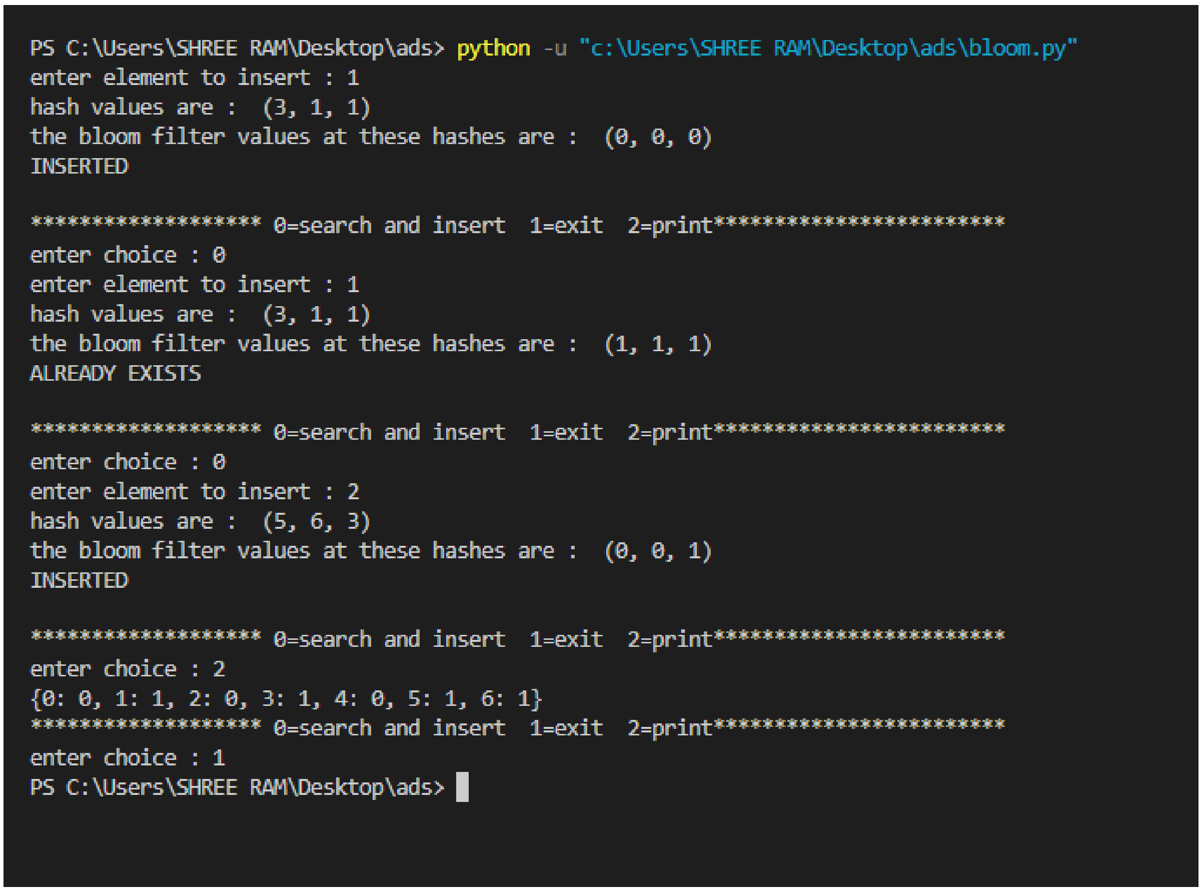
Technology stack used: Python

Bloom filter:

Program:

```
d={0:0,1:0,2:0,3:0,4:0,5:0,6:0}

def h1(e):
    return (2*e+1)%7

def h2(e):
    return (5*e+3)%7

def h3(e):
    return (9*e+6)%7

def search(e):
    a=d[h1(e)]
    b=d[h2(e)]
    c=d[h3(e)]
    print("hash values are : ",(h1(e),h2(e),h3(e)))
    print("the bloom filter values at these hashes are : ",(a,b,c))
    if a==1 and b==1 and c==1:
        print("ALREADY EXISTS\n")
    else:
        d[h1(e)]=1
        d[h2(e)]=1
        d[h3(e)]=1
        print("INSERTED\n")

choice=0
while True:
    if choice==0:
        e=int(input("enter element to insert : "))
        search(e)
        choice=-1
    elif choice==-1:
        print("******************* 0=search and insert\t 1=exit\t
2=print************************")
        choice=int(input("enter choice : "))
    elif choice==1:
        break
    elif choice==2:
        print(d)
        choice=-1
```

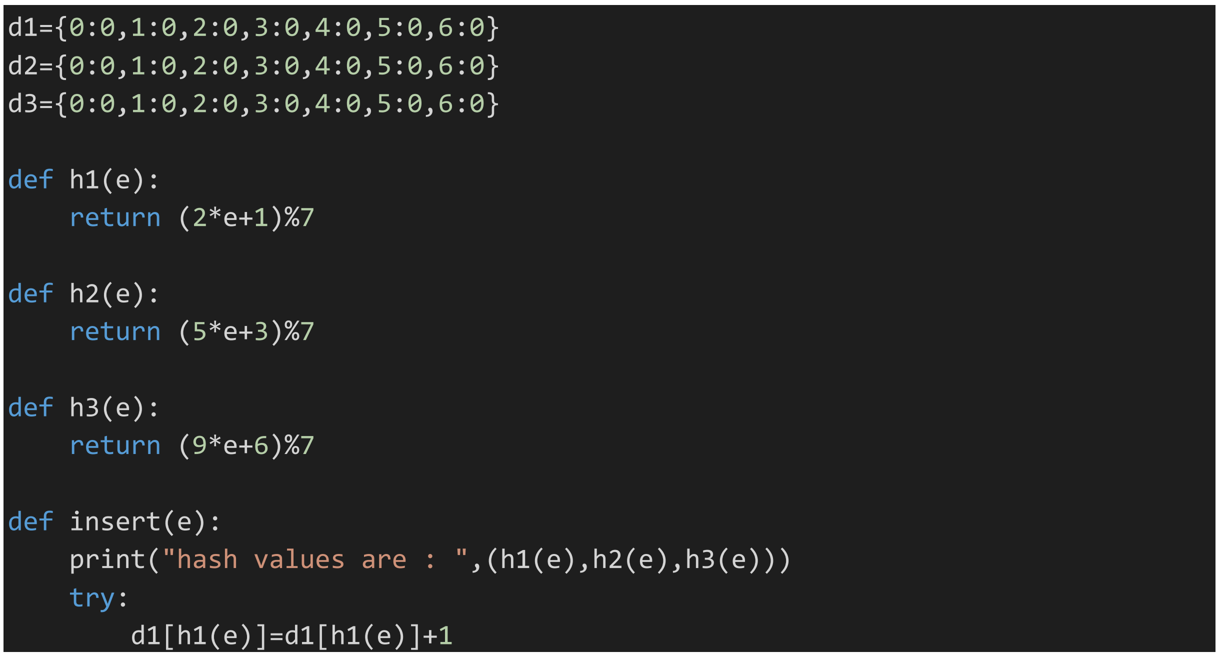
Output:



```
PS C:\Users\SHREE RAM\Desktop\ads> python -u "c:\Users\SHREE RAM\Desktop\ads\bloom.py"
enter element to insert : 1
hash values are :  (3, 1, 1)
the bloom filter values at these hashes are :  (0, 0, 0)
INSERTED

******************* 0=search and insert  1=exit  2=print************************
enter choice : 0
enter element to insert : 1
hash values are :  (3, 1, 1)
the bloom filter values at these hashes are :  (1, 1, 1)
ALREADY EXISTS

******************* 0=search and insert  1=exit  2=print************************
enter choice : 0
enter element to insert : 2
hash values are :  (5, 6, 3)
the bloom filter values at these hashes are :  (0, 0, 1)
INSERTED

******************* 0=search and insert  1=exit  2=print************************
enter choice : 2
{0: 0, 1: 1, 2: 0, 3: 1, 4: 0, 5: 1, 6: 1}
******************* 0=search and insert  1=exit  2=print************************
enter choice : 1
PS C:\Users\SHREE RAM\Desktop\ads>
```

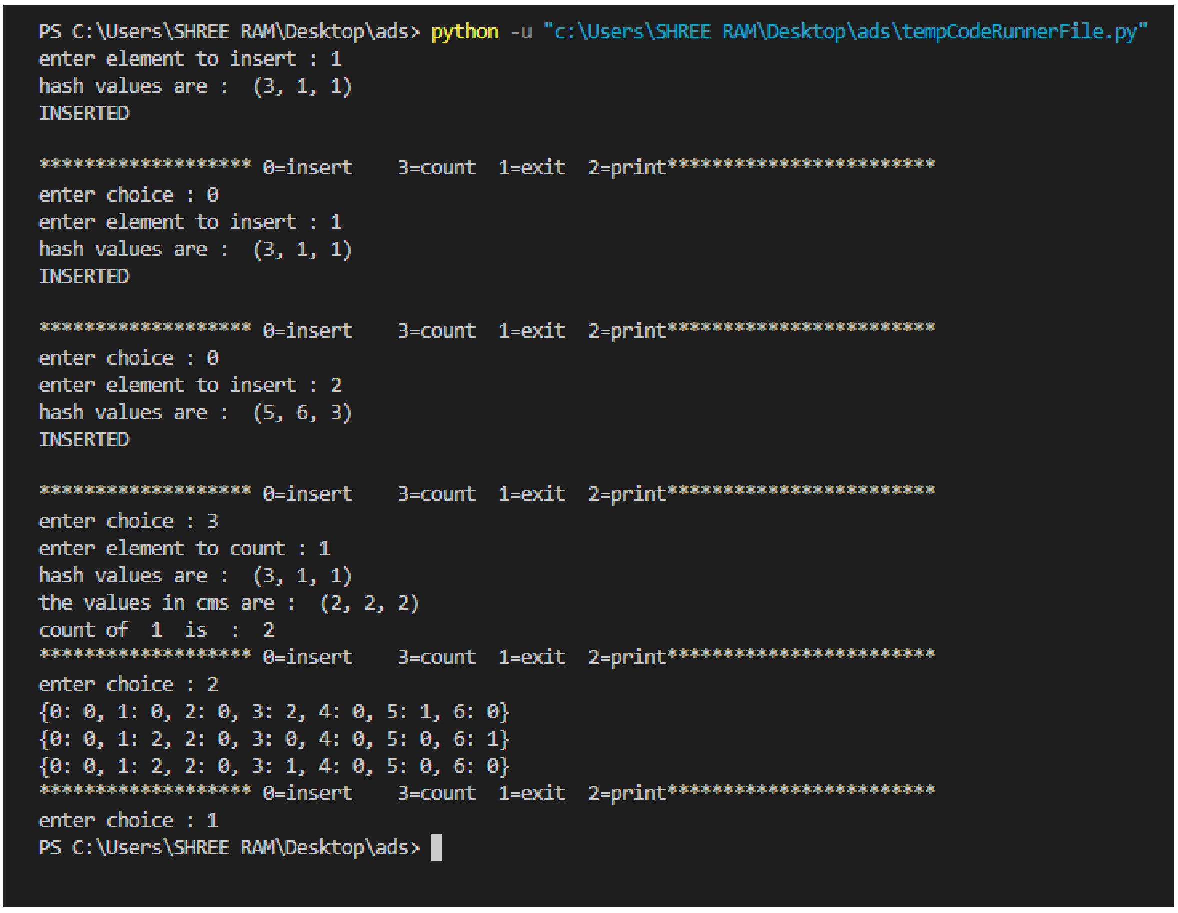
Count min sketch:

Program:



```
d1={0:0,1:0,2:0,3:0,4:0,5:0,6:0}
d2={0:0,1:0,2:0,3:0,4:0,5:0,6:0}
d3={0:0,1:0,2:0,3:0,4:0,5:0,6:0}

def h1(e):
    return (2*e+1)%7

def h2(e):
    return (5*e+3)%7

def h3(e):
    return (9*e+6)%7

def insert(e):
    print("hash values are : ",(h1(e),h2(e),h3(e)))
    try:
        d1[h1(e)]=d1[h1(e)]+1
    except:
        d1[h1(e)]=1
    try:
        d2[h2(e)]=d2[h2(e)]+1
    except:
        d2[h2(e)]=1
    try:
        d3[h3(e)]=d3[h3(e)]+1
    except:
        d3[h3(e)]=1
    print("INSERTED\n")

def search(e):
    a=d1[h1(e)]
    b=d2[h2(e)]
    c=d3[h3(e)]
    print("hash values are : ",(h1(e),h2(e),h3(e)))
    print("the values in cms are : ",(a,b,c))
    print("count of ",e," is  : ", min([a,b,c]))

choice=0
while True:
    if choice==0:
        e=int(input("enter element to insert : "))
        insert(e)
        choice=-1
    elif choice==-1:
        print("******************* 0=insert\t3=count\t 1=exit\t
2=print************************")
        choice=int(input("enter choice : "))
    elif choice==3:
        e=int(input("enter element to count : "))
        search(e)
        choice=-1
    elif choice==1:
        break
    elif choice==2:
        print(d1)
        print(d2)
        print(d3)
        choice=-1
```

Output:



```
PS C:\Users\SHREE RAM\Desktop\ads> python -u "c:\Users\SHREE RAM\Desktop\ads\tempCodeRunnerFile.py"
enter element to insert : 1
hash values are :  (3, 1, 1)
INSERTED

****************** 0=insert    3=count  1=exit  2=print************************
enter choice : 0
enter element to insert : 1
hash values are :  (3, 1, 1)
INSERTED

****************** 0=insert    3=count  1=exit  2=print************************
enter choice : 0
enter element to insert : 2
hash values are :  (5, 6, 3)
INSERTED

****************** 0=insert    3=count  1=exit  2=print************************
enter choice : 3
enter element to count : 1
hash values are :  (3, 1, 1)
the values in cms are :  (2, 2, 2)
count of  1  is  :  2
****************** 0=insert    3=count  1=exit  2=print************************
enter choice : 2
{0: 0, 1: 0, 2: 0, 3: 2, 4: 0, 5: 1, 6: 0}
{0: 0, 1: 2, 2: 0, 3: 0, 4: 0, 5: 0, 6: 1}
{0: 0, 1: 2, 2: 0, 3: 1, 4: 0, 5: 0, 6: 0}
****************** 0=insert    3=count  1=exit  2=print************************
enter choice : 1
PS C:\Users\SHREE RAM\Desktop\ads> 
```

APPLICATIONS:

The applications of Bloom Filter and count min sketch are:

- Weak password detection
- Internet Cache Protocol
- Safe browsing in Google Chrome
- Wallet synchronization in Bitcoin
- Hash based IP Traceback
- Cyber security like virus scanning

Conclusion:

Thus we have implemented Bloom Filter and Count min sketch