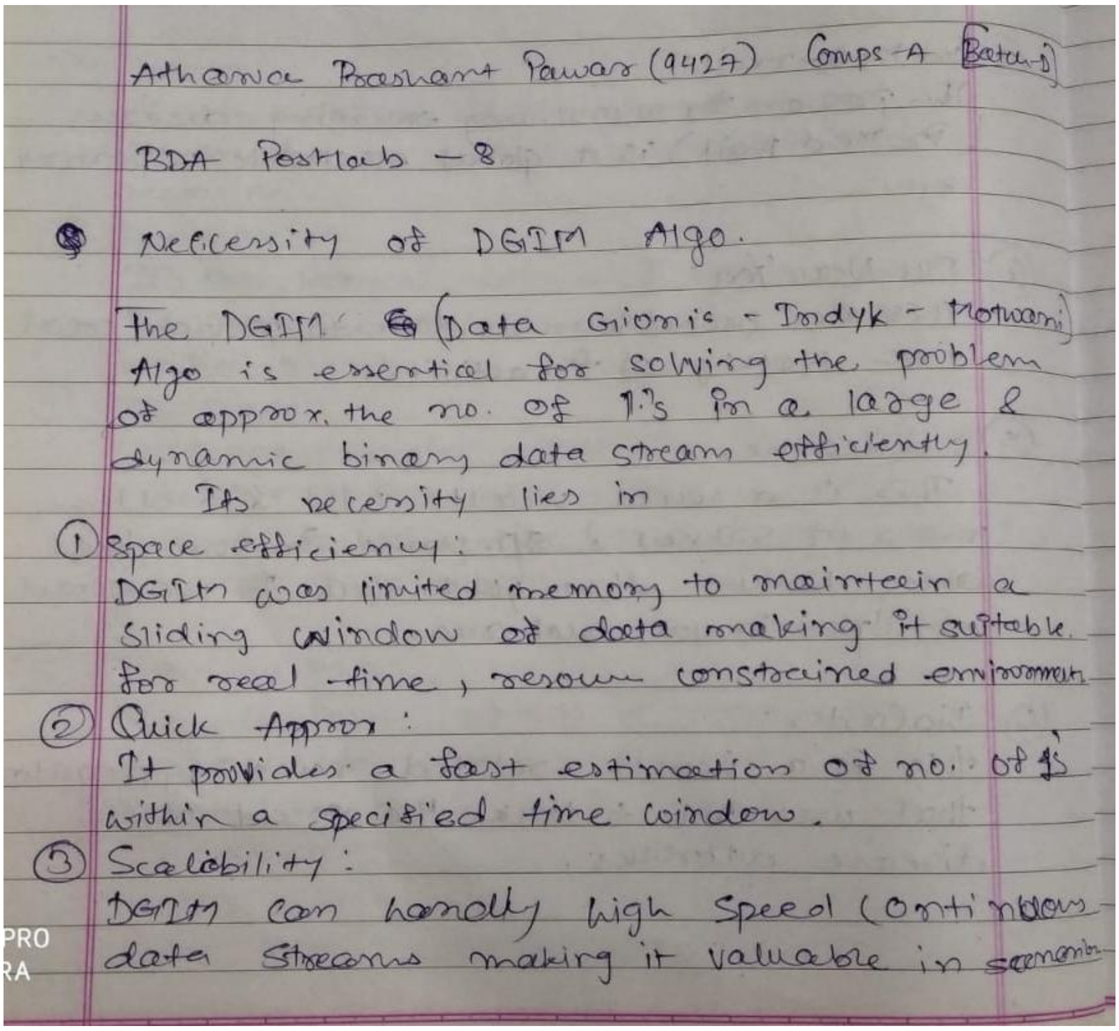
USING DGMI ALOGIRHTM

```
import IPython
import sys
import itertools
import time
import math

def checkAndMergeBucket(bucketList, t):
    bucketListLength = len(bucketList)
    for i in range (bucketListLength):
        if len(bucketList[i]) > 2:
            bucketList[i].pop(0)
            if i + 1 >= bucketListLength:
                bucketList[i].pop(0)
            else:
                bucketList[i+1].append(bucketList[i].pop(0))

K = 1000
N = 1000
k = int(math.floor(math.log(N, 2)))
t = 0
onesCount = 0
bucketList = []
for i in range(k+1):
    bucketList.append(list())

with open('dgmi.txt') as f:
    while True:
        c = f.read(1)
        if not c:
            for i in range(k+1):
                for j in range(len(bucketList[i])):
                    print ("Size of bucket: %d | Power of 2: %d | timestamp: %d" % (pow(2,i), i, bucketList[i][j]))
                    earliestTimestamp = bucketList[i][j]
            for i in range(k+1):
                for j in range(len(bucketList[i])):
                    if bucketList[i][j] != earliestTimestamp:
                        onesCount = onesCount + pow(2,i)
                    else:
                        onesCount = onesCount + 0.5 * pow(2,i)
            print ("Number of ones in last %d bits: %d" % (K, onesCount))
            break
        t = (t + 1) % N
        for i in range(k+1):
            for bucketTimestamp in bucketList[i]:
                if bucketTimestamp == t:
                    bucketList[i].remove(bucketTimestamp)
        if c == '1':
            bucketList[0].append(t)
            checkAndMergeBucket(bucketList, t)
        elif c == '0':
            continue
```

```
Size of bucket: 1 |  Power of 2: 0 | timestamp: 261
Size of bucket: 1 |  Power of 2: 0 | timestamp: 267
Size of bucket: 2 |  Power of 2: 1 | timestamp: 259
Size of bucket: 4 |  Power of 2: 2 | timestamp: 245
Size of bucket: 8 |  Power of 2: 3 | timestamp: 227
Size of bucket: 16 |  Power of 2: 4 | timestamp: 191
Size of bucket: 32 | Power of 2: 5 | timestamp: 123
Number of ones in last 1000 bits: 48
```

dgmi.txt ×

```
1 0 1 1 0 1 0 0 1 0 1 0 1 0 0 0 1 1 0 1 0 1 0 0 0 1 1 1 1 0 1 0 0 0 1 0 1 1 0 0 1 1 0 1 0 0 0 1 0 0 1 0 1 0 0 1 0 1 0 1 0 1 1 0 0 1 1 0 0
```



Atharva Prashant Pawar (9427) Comps-A $\boxed{\text{Batch-D}}$

BDA PostLab – 8

Necessity of DGIM Algo.

The DGIM (Data Gionis - Indyk - Motwani) Algo is essential for solving the problem of approx. the no. of 1's in a large & dynamic binary data stream efficiently.

Its necessity lies in

① Space efficiency:
DGIM was limited memory to maintain a sliding window of data making it suitable for real-time, resoure constrained environments.

② Quick Approx:
It provides a fast estimation of no. of 1's within a specified time window.

③ Scalability:
DGIM can handly high speed continuous data streams making it valuable in scenarios

PRO
RA