CSL7630: Sketching & Streaming

Dataset Generation:

```
with open("numbers", 'w') as f:
    no_of_element = rnd.randint(900000000, 1000000000)
    for i in range(no_of_element):
        r = rnd.randint(1000, 10000)
        f.write("{0}\n".format(rnd.randint(1, r)))
```

Executed the above code to generate a dataset of size 2.8 Gb called “numbers” and uploaded the same in Google drive for use to implement the following Sketching & Streaming Algorithms.

Question 1 *(Building Statistics of the Data (10 Marks))*

Find the following exact summary of the data.

- Total number of integers in the file.
- Total number of unique integers in the file.
- Frequency of each unique integers in the file.

Link of Google Colab File:
https://colab.research.google.com/drive/1wLIHpGD-9pmWxuchs3ocl7aiptqfvyGy?usp=sharing

Observations:

1.1. Total number of integers in the file:

Total number of integers in the file (Exact)
275732974

1.2. Total number of unique integers in the file: 10000

First few unique integers present in the file are:

```
Unique numbers present in the stream are:  ['298', '5140', '223', '8167',
'8115', '4619', '433', '302', '3283', '180', '454', '372', '2496', '256',
'4151', '3807',.........]
```

1.3. Frequency of each unique integer in the file:

Frequency of first few unique integers present in the file are {‘num’:frequency}:

```
{'num':count}
{'298': 70393,
 '5140': 20633,
 '223': 70463,
 '8167': 6237,
 '8115': 6455,
 '4619': 23593,
 '433': 70438,
 '302': 70284,
 '3283': 34221,
 '180': 70656,
 '454': 70063,
 '372': 70865,
 '2496': 42262,
 '256': 70450,
 '4151': 26846,
 '3807': 29512,
 '5067': 20961,
 '4690': 22949,
 '423': 70793,
 '7682': 7995,
 '2402': 43706,
 '1266': 63362,
 '5633': 17528,
 '364': 70700,
 '2705': 39695,
 '138': 70602,
 '2297': 45227,
 '3169': 34910,
 '470': 71128,
 '1483': 58776,
 '6638': 12475,
 '762': 70004,
 '2681': 40112,
 '408': 70625,
 '5648': 17645,
 '4415': 25197,
 '707': 70494,
 '2432': 43245,
 '3360': 33508,
 '3294': 34397,
 '1186': 64986,
 '1312': 61819,
 '1616': 55720,
 '2424': 43455,
 '1579': 56479,
 '2346': 44272,
 '4753': 22707,
 '226': 70615,
 '4085': 27476,
 '106': 70768,
 '4042': 27661,
 '744': 70539,
 '2661': 40685,
 '3039': 36473,
 '1447': 59473,
 '14': 70239,
 '2209': 46372,
 '5820': 16599,
 '1078': 68320,
 '2577': 41994,
 '6153': 15073,
 '1608': 56054,
 '1086': 68524,
 '1381': 60895,
 '4734': 22885,
 '1895': 50753,
 '2436': 43112,
 '1214': 64259,
 '4329': 25603,
 '3748': 30227,
 '4520': 24434,
 '7200': 10041,
 '2911': 38036,
 '4172': 26700,
 '3049': 36589,
 '1252': 63801,
 '6854': 11694,
 '1454': 58925,
 '7557': 8529,
 '2999': 37104,
 '598': 70205,
 '898': 71213,
 '4839': 22376,
 '5248': 19900,
 '5933': 15917,
 '167': 70682,
 '1165': 66104,
 '1613': 56070,
 '2491': 42537,
 '2425': 43888,
 '23': 70859,
 '687': 70667,
 '3893': 29001,
 '2736': 39879,
 '3883': 29082,
 '4944': 21562,
 '748': 70262,
 '1774': 53046,
 '3988': 28281,
 '458': 70500,
 '664': 70735,
 '4589': 23846,
 '4934': 21783,
 '485': 70708,
 '3101': 35766,
 '304': 70088,
 '6738': 12232,
 '523': 70342,
 '6988': 10909,
 '4354': 25734,
 '354': 70410,
 '2154': 47248,
 '1322': 61919,
 '8642': 4606,
 '1112': 67636,
 '4866': 22216,
 '6514': 12965,
 '1069': 68707,
 '3370': 33224,
 '128': 70757,
 '3255': 34297,
 '160': 70872,
 '536': 70641,
 '339': 70828,
 '1324': 62084,
 '556': 70684,
 '1752': 53558,
 '2667': 40440,
 '4430': 24763,
 '6779': 12097,
 '294': 70975,
```

Question 2 *(Sketching: Approximate Count (20 Marks))*

Find the approximate count of the data using Morris+ algorithm. Report values of ϵ, δ and any other parameter you have considered. Side-by-side report the actual number of integer (from Question 1) and approximate count (output of this program). What is the value of the counter X?

Link of Google Colab File:
https://colab.research.google.com/drive/1h8NayLMvGeV6vDtEwnR6SIEOF_mKSar7?usp=sharing

Parameters Taken:

Failure Probability (δ)	Error (ϵ)
1/3	0.3

Depending upon following parameters, number of independent copies of Morris Algorithm required for Morris+ are

S = int(1/(2*δ*ϵ^2)) = 16

Observations:

i^{th} Copy of Morris	Counter (X)	Approximate Count
1	28	268435455
2	27	134217727
3	28	268435455
4	27	134217727
5	27	134217727
6	29	536870911
7	27	134217727
8	27	134217727
9	28	268435455
10	27	134217727
11	28	268435455

12	27	134217727
13	28	268435455
14	29	536870911
15	29	536870911
16	28	268435455

Approximate Count of integers in the file using Morris+ Algorithm =

(1/S)*(Summation of Approximate integer count of each independent copy using Morris Algorithm)

Exact Count of integers in the file	Approximate Count of integers in the file using Morris+
275732974	260046847

```
0 th Copy of Morris Algorithm has X count:  28  and approximate count:
268435455
1 th Copy of Morris Algorithm has X count:  27  and approximate count:
134217727
2 th Copy of Morris Algorithm has X count:  28  and approximate count:
268435455
3 th Copy of Morris Algorithm has X count:  27  and approximate count:
134217727
4 th Copy of Morris Algorithm has X count:  27  and approximate count:
134217727
5 th Copy of Morris Algorithm has X count:  29  and approximate count:
536870911
6 th Copy of Morris Algorithm has X count:  27  and approximate count:
134217727
7 th Copy of Morris Algorithm has X count:  27  and approximate count:
134217727
8 th Copy of Morris Algorithm has X count:  28  and approximate count:
268435455
9 th Copy of Morris Algorithm has X count:  27  and approximate count:
134217727
10 th Copy of Morris Algorithm has X count:  28  and approximate count:
268435455
11 th Copy of Morris Algorithm has X count:  27  and approximate count:
134217727
12 th Copy of Morris Algorithm has X count:  28  and approximate count:
268435455
13 th Copy of Morris Algorithm has X count:  29  and approximate count:
536870911
```

```
14 th Copy of Morris Algorithm has X count:  29  and approximate count:
536870911
15 th Copy of Morris Algorithm has X count:  28  and approximate count:
268435455
Approximate Count using Morris+ Algorithm:  260046847
Exact count of number of integers in the file:  275732974
```

Question 3 *(Sketching: Approximate Distinct Count (20 Marks))*

Find the approximate distinct count using any of the practical algorithm we studied in the class. Report the parameters such as ϵ and δ. Side-by-side report the actual number of distinct element and the approximate distinct count. What is the value of the counter(s)?

Link of Google Colab File:
https://colab.research.google.com/drive/1r4J3zKqC9_q_ZmY-nZsZQtK5Av6RIJfz?usp=sharing

Have used FM and FM+ Algorithm to find the approximate distinct count of elements in the dataset.

2.1. Observation using FM Algorithm for Approximate Distinct Count:

Hash Function Used: (7573*x + 9573) mod 10015, where x is the element of the dataset

Exact Distinct element Count of Elements	Approximate Distinct Count using FM
10000	8192

2.2. Observation using FM+ Algorithm for Approximate Distinct Count:

Parameters Taken:

Failure Probability (δ)	Error (ϵ)
1/3	0.3

Depending upon following parameters, number of independent copies of FM Algorithm required for FM+ count are

$$S = \text{int}(1/(\delta * \epsilon^2)) = 33$$

Have created 33 different Hash functions using the formulae: **(a*x+b) mod c,** where a,b and c are randomly created odd numbers.

First 10 Approximate Distinct Count of Elements using FM+ (33 such copies are there in the code file):

i^{th} Copy of FM	Approximate Distinct Count
1	8192
2	11286
3	7096
4	11286
5	8192
6	9261
7	8192
8	6981
9	8192
10	8192

Approximate Distinct Count of elements in the file using FM+ Algorithm =

(1/S)*(Summation of Approximate distinct count of elements of each independent copy using FM Algorithm)

Exact Distinct element Count of Elements	Approximate Distinct Count using FM
10000	8896

Question 4 *(Sketching: Frequency Query (20 Marks))*

Create a CountMin Sketch data structure for calculating the frequency of any number. Report side-by-side the actual frequency of the number and estimated frequency of number by CountMin and Count Sketch for 15 random numbers of the file.

Link of Google Colab File:
https://colab.research.google.com/drive/1hxxph44h4n8LTlMuWnipQy0-Mru-dRcl?usp=sharing

4.1 Count Min Data Structure to find the frequency of Number:

Used the following three different hash functions to find the hash value of any given number:

- hash1 = ((7173*x+3013)%10993)%10000
- hash2 = ((783*x+639)%10993)%10000
- hash3 = ((8124*x+8276)%10993)%10000

Where x is the number of the dataset

Created a hash table for the elements of the dataset using following hash values:

```
#Hash table
print(hash_table)
```

```
[[57144. 18711. 45924. ...  6165. 70175. 30128.]
 [33176. 19913.  8728. ... 68246. 42338. 28367.]
 [23029. 33982. 48311. ...  2410.  6330. 11066.]]
```

Observations:

Element	Exact Frequency	Frequency using Count min
298	70393	70393
5140	20633	20633
223	70463	70463
8167	6237	6237
8115	6455	6455
4619	23593	23593
433	70438	70438
302	70284	70284
3283	34221	34221
180	70656	70656
2329	44366	49365

454	70063	70063
3509	31784	32814
372	70865	70865
2496	42262	42262

4.2 Count Data Structure to find the frequency of Number:

Used the following three different hash functions to find the hash value of any given number:

- hash1 = ((7173*x+3013)%10993)%10000
- hash2 = ((783*x+639)%10993)%10000
- hash3 = ((8124*x+8276)%10993)%10000
-

Used the following three sing functions to get the sign value of any given number:

- sign_hash1 = (x)%2
- sign_hash2 = (x)%4
- sign_hash3 = (x)%5

Where x is the number of the dataset

Created a hash table for the elements of the dataset using the above hash and sign values:

```
count_hash_table
```

```
array([[-57144., -18711.,  37636., ...,   6165., -70175., -30128.],
       [-33176., -19913.,  -8728., ..., -68246., -42338., -28367.],
       [ 23029., -33982., -48311., ...,  -2410.,  -6330., -11066.]])
```

Observations:

Element	Exact Frequency	Frequency using Count
5140	20633	20633
8167	6237	6237
433	70438	70438
302	70284	70284
3283	34221	34221
180	70656	70656

298	70393	70392
454	70063	70063
3807	29512	29512
4690	22949	22949
4085	27476	24476
1165	66104	107451
4787	22527	51861
423	70793	70793
7682	7995	7995

If we increase the number of hash functions in each Count min and Count sketch, the accuracy of the algorithm will increase. I have used only 3 different hash functions.

Question 5 *(Building the Streaming version of the Sketches (30 Marks))*

Make all the above problem into a streaming one. That is a person can query at anytime during the execution and the system should show the estimated values at that moment.

Link of Google Colab File:
https://colab.research.google.com/drive/1h5mbhkkadBQN9Sv37aNhR-xB2zRcBVQw?usp=sharing

We have taken queries at 9 intervals when the count of the elements in the stream are:
[100,1000,10000,100000,1000000,10000000,100000000,200000000, and at terminating the stream]

Our Query at each interval includes (Question 1 to 4):

- Finding Exact Element Count in the stream
- Finding Approximate Element Count in the stream using Morris
- Finding Exact Distinct Element Count in the stream
- Finding Approximate Distinct Element Count in the stream using FM
- Finding Exact Frequency of each element in the stream
- Finding Frequency of element in the stream using Count min and count sketch

Observations:

5.1. Query1:

Exact Element Count	Approx. element Count using Morris	Exact distinct element count	Approx. distinct element count using FM
100	63	98	32

Element	Exact Frequency	Frequency using Count min	Frequency using Count
298	1	1	1
5140	1	1	1
223	1	1	1
8167	1	1	1

5.2. Query2:

Exact Element Count	Approx. element Count using Morris	Exact distinct element count	Approx. distinct element count using FM
1000	1023	912	512

Element	Exact Frequency	Frequency using Count min	Frequency using Count
298	1	1	1
5140	1	1	1
223	1	1	1
8167	1	1	1

5.3. Query3:

Exact Element Count	Approx. element Count using Morris	Exact distinct element count	Approx. distinct element count using FM
10000	8191	5263	4096

Element	Exact Frequency	Frequency using Count min	Frequency using Count
298	4	4	4
5140	2	2	2
223	5	5	8
8167	1	1	1

5.4. Query4:

Exact Element Count	Approx. element Count using Morris	Exact distinct element count	Approx. distinct element count using FM
100000	131071	9147	8192

Element	Exact Frequency	Frequency using Count min	Frequency using Count
298	23	23	23
5140	5	5	5
223	30	30	50
8167	3	3	3

5.5. Query5:

Exact Element Count	Approx. element Count using Morris	Exact distinct element count	Approx. distinct element count using FM
1000000	1048575	9911	8192

Element	Exact Frequency	Frequency using Count min	Frequency using Count
298	263	263	263
5140	69	69	69
223	279	279	507
8167	28	28	28

5.6. Query6:

Exact Element Count	Approx. element Count using Morris	Exact distinct element count	Approx. distinct element count using FM
10000000	8388607	9991	8192

Element	Exact Frequency	Frequency using Count min	Frequency using Count
298	2615	2615	2615
5140	724	724	724
223	2584	2584	4925
8167	240	240	240

5.7. Query7:

Exact Element Count	Approx. element Count using Morris	Exact distinct element count	Approx. distinct element count using FM
100000000	67108863	9999	8192

Element	Exact Frequency	Frequency using Count min	Frequency using Count
298	25530	25530	25530
5140	7429	7429	7429
223	25618	25618	49192
8167	2311	2311	2311

5.8. Query8:

Exact Element Count	Approx. element Count using Morris	Exact distinct element count	Approx. distinct element count using FM
200000000	167908261	10000	8192

Element	Exact Frequency	Frequency using Count min	Frequency using Count
298	50894	50894	50894
5140	14934	14934	14934
223	51050	51050	98193
8167	4534	4534	4534

5.9. Query9:

Exact Element Count	Approx. element Count using Morris	Exact distinct element count	Approx. distinct element count using FM
275732974	268435455	10000	8192

Element	Exact Frequency	Frequency using Count min	Frequency using Count
298	70393	70393	70393
5140	20633	20633	20633
223	70463	70463	135376
8167	6237	6237	6237

Snippet of output:

```
************************************************************Query 6 ************************************************************
  Exact Element Count    Approximate Element Count using Morris    Exact Distinct Element Count    Approximate Distinct elements count using FM
---------------------  ----------------------------------------  ------------------------------  ---------------------------------------------
             10000000                                   8388607                            9991                                          8192

Frequency of each distinct Element in the stream {'num':count}
{'298': 2615, '5140': 724, '223': 2584, '8167': 240, '8115': 246, '4619': 867, '433': 2560, '302': 2492, '3283': 1242, '180': 2522, '454': 2514,

Frequency of each distinct Element in the stream using Count min {'num':count}:
{'298': 2615, '5140': 724, '223': 2584, '8167': 240, '8115': 246, '4619': 867, '433': 2560, '302': 2492, '3283': 1242, '180': 2522, '454': 2514,
Frequency of each distinct Element in the stream using Count {'num':count}:
{'298': 2615, '5140': 724, '223': 4925, '8167': 240, '8115': 614, '4619': 1984, '433': 2560, '302': 2492, '3283': 1242, '180': 2522, '454': 2514,
************************************************************Query 7 ************************************************************
  Exact Element Count    Approximate Element Count using Morris    Exact Distinct Element Count    Approximate Distinct elements count using FM
---------------------  ----------------------------------------  ------------------------------  ---------------------------------------------
            100000000                                  67108863                            9999                                          8192

Frequency of each distinct Element in the stream {'num':count}
{'298': 25530, '5140': 7429, '223': 25618, '8167': 2311, '8115': 2375, '4619': 8534, '433': 25545, '302': 25284, '3283': 12446, '180': 25511, '454

Frequency of each distinct Element in the stream using Count min {'num':count}:
{'298': 25530, '5140': 7429, '223': 25618, '8167': 2311, '8115': 2375, '4619': 8534, '433': 25545, '302': 25284, '3283': 12446, '180': 25511, '454
Frequency of each distinct Element in the stream using Count {'num':count}:
{'298': 25530, '5140': 7429, '223': 49192, '8167': 2311, '8115': 6123, '4619': 19828, '433': 25545, '302': 25284, '3283': 12446, '180': 25511, '45
```

Exact Element Count	Approximate Element Count using Morris	Exact Distinct Element Count	Approximate Distinct elements count using FM
275732974	268435455	10000	8192

Frequency of each distinct Element in the stream {'num':count}
{'298': 70393, '5140': 20633, '223': 70463, '8167': 6237, '8115': 6455, '4619': 23593, '433': 70438, '302': 70284, '3283': 34221, '180': 70656, '

Frequency of each distinct Element in the stream using Count min {'num':count}:
{'298': 70393, '5140': 20633, '223': 70463, '8167': 6237, '8115': 6455, '4619': 23593, '433': 70438, '302': 70284, '3283': 34221, '180': 70656, '
Frequency of each distinct Element in the stream using Count {'num':count}:
{'298': 70393, '5140': 20633, '223': 135376, '8167': 6237, '8115': 16617, '4619': 54508, '433': 70438, '302': 70284, '3283': 34221, '180': 70656,

---