CS168 P1 Part2

Count-Min Sketch

Goal: The goal of this part is to understand the count-min sketch (from Lecture #2) via an implementa- tion, and to explore the benefits of a “conservative updates” optimization. \ \ **Description**: You’ll use a count-min sketch with 4 independent hash tables, each with 256 counters. You will run 10 independent trials. This lets you measure not only the accuracy of the sketch, but the distribution of the accuracy over multiple datasets with the same frequency distribution. Your sketch should take a “trial” as input, and the hash value of an element x during trial i (i = 1,2,...,10) for table j (j = 1,2,3,4) is calculated as follows: \ \ • Consider the input x as a string, and append i – 1 as a string to the end of the string. \ • Calculate the MD5 score of the resulting string. modern programming languages have packages that calculate MD5 scores for you. For example, in Python 3, you can use the hashlib library and hashlib.md5(foo.encode(’utf-8’)).hexdigest() to compute the MD5 score of the string foo (returning a hexidecimal string). \ • The hash value is the j-th byte of the score.

(a) *(5 points)* Implement the count-min sketch, as above.

In [1]:

```
"""
    Implementation in count_min_sketch.py
"""
from count_min_sketch import *
```

We will be feeding data streams (i.e., sequences of elements) into count-min sketches. Every element of each stream is an integer between 1 and 9050 (inclusive). The frequencies are given by: \ \ • Integers $1000·(i−1)+1 $ to $1000·i$, for $1≤i≤9$,appear i times in the stream. Thatis, the integers 1 to 1000 appear once in the stream; 1001 to 2000 appear twice; and so on. \ \ • An integer $9000 + i$, for $1 ≤ i ≤ 50$, appears i^2 times in the stream. For example, the integer 9050 appears 2500 times. \ (Each time an integer appears in the stream, it has a count of 1 associated with it.)

(b) *(2 points)* Call an integer a heavy hitter if the number of times it appears is at least 1% of the total number of stream elements. How many heavy hitters are there in a stream with the above frequencies?

In [2]:

```
def get_frequency(integer):
    if ((integer <= 9050) and (integer >= 1)):
        if ((integer >= 9001)):
            return (integer - 9000)**2
        else:
            return ((integer - 1) // 1000) + 1
```

In [3]:

```
N = sum([get_frequency(i) for i in range(1, 9051)])
print(N)
```

```
87925
```

We have $N = 87925$ numbers at all \ Thats why we consider element the heavy hitter if it appears more that $\nu = \frac{N}{100} = \frac{87925}{100} = 879.25$, \ $[\nu] = 880$

In [4]:

```
heavy_hitter_min_count = 880
```

Next, we will consider 3 different data streams, each corresponding to the elements above in a different order.

1. Forward: the elements appear in non-decreasing order.
2. Reverse: the elements appear in non-increasing order.
3. Random: the elements appear in a random order.

In [5]:

```
def forward_stream():
    for integer in range(1, 9051):
        freq = get_frequency(integer)
        for j in range(freq):
            yield integer
```

In [6]:

```
def backward_stream():
    for integer in range(9050, 0, -1):
        freq = get_frequency(integer)
        for j in range(freq):
            yield integer
```

In [7]:

```
def random_stream():
    integers = np.random.choice(9050, 9050, replace = False) + 1
    for integer in integers:
        freq = get_frequency(integer)
        for j in range(freq):
            yield integer
```

(c) *(6 points)* For each of the three data streams, feed it into a count-min sketch (i.e., successively insert its elements), and compute the values of the following quantities, averaged over the 10 trials, for each order of the stream: \ – The sketch's estimate for the frequency of element 9050.\ – The sketch's estimate for the number of heavy hitters (elements with estimated frequency at least 1% of the stream length). Record the mean estimate for each of the three orders. Does the order of the stream affect the estimated counts? Explain your answer.

In [8]:

```
"""
    Implementation of Approximate Heavy Hitter Problem Solver
"""
from heavy_hitter_solver import Heavy_Hitters_Solver
```

In [9]:

```
element_to_check = 9050
```

An integer $9000 + i$, for $1 ≤ i ≤ 50$, appears i^2 times in the stream \ True Heavy Hitters amount is amount of elements with estimated frequency at least 1% of the stream length \ i.e. at least 880 times \ which is equal to $ 50 - [\sqrt {880}] = 50 - 29 = 21$ times

In [10]:

```
true_element_frequency = get_frequency(9050)
true_heavy_hitters_amount = 21
```

In [11]:

```
def perform_trial(trial_i, stream, conservative_updates = False):
    sketch = CountMinSketch(conservative_updates = conservative_updates)
    hh_solver = Heavy_Hitters_Solver(sketch, heavy_hitter_min_count)
    for element in stream:
        trial_element = str(element) + str(trial_i - 1)
        hh_solver.Inc(trial_element)

    heavy_hitters = hh_solver.heavy_hitters()

    # ---------------------------------------------------------------------
    trial_element_to_check = str(element_to_check) + str(trial_i - 1)
    element_estimated_frequency = sketch.Count(trial_element_to_check)
    heavy_hitters_amount = len(heavy_hitters)
    # ---------------------------------------------------------------------
```

```
    return element_estimated_frequency, heavy_hitters_amount
```

Conducting research

In [31]:

```
def percent_mistake(a,b):
    return str( 100 * (a - b) / b) + "%"

def conduct_research(stream_func, verbose = True, conservative_updates = False):
    a = []
    b = []
    print("Conservative Updates =", conservative_updates)
    for trial_i in range(1, 11):
        element_estimated_frequency, heavy_hitters_amount = perform_trial(trial_i, stream_func(), c
onservative_updates)
        a.append(element_estimated_frequency)
        b.append(heavy_hitters_amount)
        if (verbose == True):
            print("trial -", trial_i)
            print("estimated 9050 frequency -", element_estimated_frequency, ", true 9050 frequency
-", true_element_frequency,
                  ", mistake -", percent_mistake(element_estimated_frequency,
true_element_frequency))
            print("estimated heavy hitters amount -", heavy_hitters_amount, ", true heavy hitters a
mount -", true_heavy_hitters_amount,
                  ", mistake -", percent_mistake(heavy_hitters_amount, true_heavy_hitters_amount))
            print()

    mean_etimated_frequency = sum(a) / len(a)
    mean_hh_amount = sum(b) / len(b)
    print()
    print(stream_func)
    print("mean element estimated frequency of 9050 -", mean_etimated_frequency, ", true -", true_e
lement_frequency,
          ", mistake -", percent_mistake(mean_etimated_frequency, true_element_frequency))
    print("mean heavy hitters amount -", mean_hh_amount, ", true -", true_heavy_hitters_amount,
          ", mistake -", percent_mistake(mean_hh_amount, true_heavy_hitters_amount))
    print()
    return mean_etimated_frequency, mean_hh_amount
```

Forwad Stream

In [23]:

```
print("performing forward stream")
print()
conduct_research(forward_stream)
```

```
performing forward stream

Conservative Updates = False
trial - 1
estimated 9050 frequency - 2626 , true 9050 frequency - 2500 , mistake - 5.04%
estimated heavy hitters amount - 24 , true heavy hitters amount - 21 , mistake -
14.285714285714286%

trial - 2
estimated 9050 frequency - 2633 , true 9050 frequency - 2500 , mistake - 5.32%
estimated heavy hitters amount - 24 , true heavy hitters amount - 21 , mistake -
14.285714285714286%

trial - 3
estimated 9050 frequency - 2654 , true 9050 frequency - 2500 , mistake - 6.16%
estimated heavy hitters amount - 23 , true heavy hitters amount - 21 , mistake -
9.523809523809524%

trial - 4
estimated 9050 frequency - 2654 , true 9050 frequency - 2500 , mistake - 6.16%
estimated heavy hitters amount - 24 , true heavy hitters amount - 21 , mistake -
14.285714285714286%
```

```
trial - 5
estimated 9050 frequency - 2623 , true 9050 frequency - 2500 , mistake - 4.92%
estimated heavy hitters amount - 23 , true heavy hitters amount - 21 , mistake -
9.523809523809524%

trial - 6
estimated 9050 frequency - 2655 , true 9050 frequency - 2500 , mistake - 6.2%
estimated heavy hitters amount - 23 , true heavy hitters amount - 21 , mistake -
9.523809523809524%

trial - 7
estimated 9050 frequency - 2645 , true 9050 frequency - 2500 , mistake - 5.8%
estimated heavy hitters amount - 23 , true heavy hitters amount - 21 , mistake -
9.523809523809524%

trial - 8
estimated 9050 frequency - 2653 , true 9050 frequency - 2500 , mistake - 6.12%
estimated heavy hitters amount - 23 , true heavy hitters amount - 21 , mistake -
9.523809523809524%

trial - 9
estimated 9050 frequency - 2641 , true 9050 frequency - 2500 , mistake - 5.64%
estimated heavy hitters amount - 24 , true heavy hitters amount - 21 , mistake -
14.285714285714286%

trial - 10
estimated 9050 frequency - 2673 , true 9050 frequency - 2500 , mistake - 6.92%
estimated heavy hitters amount - 23 , true heavy hitters amount - 21 , mistake -
9.523809523809524%

mean element estimated frequency of 9050 - 2645.7 , true - 2500 , mistake - 5.827999999999992%
mean heavy hitters amount - 23.4 , true - 21 , mistake - 11.428571428571422%
```

Out[23]:

```
(2645.7, 23.4)
```

Backward Stream

In [24]:

```
print("performing backward stream")
print()
conduct_research(backward_stream)
```

```
performing backward stream

Conservative Updates = False
trial - 1
estimated 9050 frequency - 2626 , true 9050 frequency - 2500 , mistake - 5.04%
estimated heavy hitters amount - 21 , true heavy hitters amount - 21 , mistake - 0.0%

trial - 2
estimated 9050 frequency - 2633 , true 9050 frequency - 2500 , mistake - 5.32%
estimated heavy hitters amount - 21 , true heavy hitters amount - 21 , mistake - 0.0%

trial - 3
estimated 9050 frequency - 2654 , true 9050 frequency - 2500 , mistake - 6.16%
estimated heavy hitters amount - 22 , true heavy hitters amount - 21 , mistake -
4.761904761904762%

trial - 4
estimated 9050 frequency - 2654 , true 9050 frequency - 2500 , mistake - 6.16%
estimated heavy hitters amount - 22 , true heavy hitters amount - 21 , mistake -
4.761904761904762%

trial - 5
estimated 9050 frequency - 2623 , true 9050 frequency - 2500 , mistake - 4.92%
estimated heavy hitters amount - 21 , true heavy hitters amount - 21 , mistake - 0.0%

trial - 6
estimated 9050 frequency - 2655 , true 9050 frequency - 2500 , mistake - 6.2%
estimated heavy hitters amount - 22 , true heavy hitters amount - 21 , mistake -
4.761904761904762%
```

```
trial - 7
estimated 9050 frequency - 2645 , true 9050 frequency - 2500 , mistake - 5.8%
estimated heavy hitters amount - 22 , true heavy hitters amount - 21 , mistake -
4.761904761904762%

trial - 8
estimated 9050 frequency - 2653 , true 9050 frequency - 2500 , mistake - 6.12%
estimated heavy hitters amount - 21 , true heavy hitters amount - 21 , mistake - 0.0%

trial - 9
estimated 9050 frequency - 2641 , true 9050 frequency - 2500 , mistake - 5.64%
estimated heavy hitters amount - 21 , true heavy hitters amount - 21 , mistake - 0.0%

trial - 10
estimated 9050 frequency - 2673 , true 9050 frequency - 2500 , mistake - 6.92%
estimated heavy hitters amount - 21 , true heavy hitters amount - 21 , mistake - 0.0%

mean element estimated frequency of 9050 - 2645.7 , true - 2500 , mistake - 5.827999999999992%
mean heavy hitters amount - 21.4 , true - 21 , mistake - 1.904761904761898%
```

Out[24]:

```
(2645.7, 21.4)
```

Random Stream

In [25]:

```
print("performing random stream")
print()
conduct_research(random_stream)
```

```
performing random stream

Conservative Updates = False
trial - 1
estimated 9050 frequency - 2626 , true 9050 frequency - 2500 , mistake - 5.04%
estimated heavy hitters amount - 21 , true heavy hitters amount - 21 , mistake - 0.0%

trial - 2
estimated 9050 frequency - 2633 , true 9050 frequency - 2500 , mistake - 5.32%
estimated heavy hitters amount - 21 , true heavy hitters amount - 21 , mistake - 0.0%

trial - 3
estimated 9050 frequency - 2654 , true 9050 frequency - 2500 , mistake - 6.16%
estimated heavy hitters amount - 22 , true heavy hitters amount - 21 , mistake -
4.761904761904762%

trial - 4
estimated 9050 frequency - 2654 , true 9050 frequency - 2500 , mistake - 6.16%
estimated heavy hitters amount - 22 , true heavy hitters amount - 21 , mistake -
4.761904761904762%

trial - 5
estimated 9050 frequency - 2623 , true 9050 frequency - 2500 , mistake - 4.92%
estimated heavy hitters amount - 23 , true heavy hitters amount - 21 , mistake -
9.523809523809524%

trial - 6
estimated 9050 frequency - 2655 , true 9050 frequency - 2500 , mistake - 6.2%
estimated heavy hitters amount - 23 , true heavy hitters amount - 21 , mistake -
9.523809523809524%

trial - 7
estimated 9050 frequency - 2645 , true 9050 frequency - 2500 , mistake - 5.8%
estimated heavy hitters amount - 21 , true heavy hitters amount - 21 , mistake - 0.0%

trial - 8
estimated 9050 frequency - 2653 , true 9050 frequency - 2500 , mistake - 6.12%
estimated heavy hitters amount - 22 , true heavy hitters amount - 21 , mistake -
4.761904761904762%

trial - 9
estimated 9050 frequency - 2641 , true 9050 frequency - 2500 , mistake - 5.64%
estimated heavy hitters amount - 22 , true heavy hitters amount - 21 , mistake
```

```
estimated heavy hitters amount - 22 , true heavy hitters amount - 21 , mistake -
4.761904761904762%

trial - 10
estimated 9050 frequency - 2673 , true 9050 frequency - 2500 , mistake - 6.92%
estimated heavy hitters amount - 23 , true heavy hitters amount - 21 , mistake -
9.523809523809524%

mean element estimated frequency of 9050 - 2645.7 , true - 2500 , mistake - 5.827999999999992%
mean heavy hitters amount - 22.0 , true - 21 , mistake - 4.761904761904762%
```

Out[25]:

```
(2645.7, 22.0)
```

In [32]:

```
conduct_research(forward_stream, verbose = False)
conduct_research(backward_stream, verbose = False)
conduct_research(random_stream, verbose = False)
```

```
Conservative Updates = False

<function forward_stream at 0x1063d6ef0>
mean element estimated frequency of 9050 - 2645.7 , true - 2500 , mistake - 5.827999999999992%
mean heavy hitters amount - 23.4 , true - 21 , mistake - 11.428571428571422%

Conservative Updates = False

<function backward_stream at 0x10799ce60>
mean element estimated frequency of 9050 - 2645.7 , true - 2500 , mistake - 5.827999999999992%
mean heavy hitters amount - 21.4 , true - 21 , mistake - 1.9047619047619898%

Conservative Updates = False

<function random_stream at 0x1079a8440>
mean element estimated frequency of 9050 - 2645.7 , true - 2500 , mistake - 5.827999999999992%
mean heavy hitters amount - 21.9 , true - 21 , mistake - 4.285714285714279%
```

Out[32]:

```
(2645.7, 21.9)
```

Summary (without conservative updates)

Forward Stream \ mean element estimated frequency of 9050 - 2645.7 , true - 2500, mistake - 5.8% \ mean heavy hitters amount - 23.4 , true - 21, mistake - 11.4% \ \ **Backward Stream** \ mean element estimated frequency of 9050 - 2645.7 , true - 2500, mistake - 5.8% \ mean heavy hitters amount - 21.4 , true - 21, mistake - 1.9% \ \ **Random Stream** \ mean element estimated frequency of 9050 - 2645.7 , true - 2500, , mistake - 5.8% \ mean heavy hitters amount - 21.8 , true - 21, mistake - 3.8%

Adding Conservative Updates

(d) *(3 points)* Implement the conservative updates optimization, as follows. When updating the counters during an insert, instead of incrementing all 4 counters, we only increment the subset of these 4 counters that have the lowest current count (if two or more of them are tied for the minimum current count, then we increment each of these). \ \ *Conservative updates implemented in count_min_sketch.py*

(e) *(3 points)* Explain why, even with conservative updates, the count-min sketch never underestimates the count of a value. \ \ **Becuase we never substract\ And once element is accounted its frequency never decreases**

(f) *(6 points)* Repeat part **(c)** with conservative updates.

In [33]:

```
conduct_research(forward_stream, verbose = False, conservative_updates = True)
conduct_research(backward_stream, verbose = False, conservative_updates = True)
conduct_research(random_stream, verbose = False, conservative_updates = True)
```

```
Conservative Updates = True

<function forward_stream at 0x1063d6ef0>
mean element estimated frequency of 9050 - 2577.2 , true - 2500 , mistake - 3.0879999999999925%
mean heavy hitters amount - 22.0 , true - 21 , mistake - 4.761904761904762%

Conservative Updates = True

<function backward_stream at 0x10799ce60>
mean element estimated frequency of 9050 - 2500.0 , true - 2500 , mistake - 0.0%
mean heavy hitters amount - 21.2 , true - 21 , mistake - 0.952380952380949%

Conservative Updates = True

<function random_stream at 0x1079a8440>
mean element estimated frequency of 9050 - 2549.7 , true - 2500 , mistake - 1.9879999999999927%
mean heavy hitters amount - 21.5 , true - 21 , mistake - 2.380952380952381%
```

Out[33]:

```
(2549.7, 21.5)
```

In [34]:

```
conduct_research(backward_stream, verbose = True, conservative_updates = True)
```

```
Conservative Updates = True
trial - 1
estimated 9050 frequency - 2500 , true 9050 frequency - 2500 , mistake - 0.0%
estimated heavy hitters amount - 21 , true heavy hitters amount - 21 , mistake - 0.0%

trial - 2
estimated 9050 frequency - 2500 , true 9050 frequency - 2500 , mistake - 0.0%
estimated heavy hitters amount - 21 , true heavy hitters amount - 21 , mistake - 0.0%

trial - 3
estimated 9050 frequency - 2500 , true 9050 frequency - 2500 , mistake - 0.0%
estimated heavy hitters amount - 22 , true heavy hitters amount - 21 , mistake -
4.761904761904762%

trial - 4
estimated 9050 frequency - 2500 , true 9050 frequency - 2500 , mistake - 0.0%
estimated heavy hitters amount - 22 , true heavy hitters amount - 21 , mistake -
4.761904761904762%

trial - 5
estimated 9050 frequency - 2500 , true 9050 frequency - 2500 , mistake - 0.0%
estimated heavy hitters amount - 21 , true heavy hitters amount - 21 , mistake - 0.0%

trial - 6
estimated 9050 frequency - 2500 , true 9050 frequency - 2500 , mistake - 0.0%
estimated heavy hitters amount - 21 , true heavy hitters amount - 21 , mistake - 0.0%

trial - 7
estimated 9050 frequency - 2500 , true 9050 frequency - 2500 , mistake - 0.0%
estimated heavy hitters amount - 21 , true heavy hitters amount - 21 , mistake - 0.0%

trial - 8
estimated 9050 frequency - 2500 , true 9050 frequency - 2500 , mistake - 0.0%
estimated heavy hitters amount - 21 , true heavy hitters amount - 21 , mistake - 0.0%

trial - 9
estimated 9050 frequency - 2500 , true 9050 frequency - 2500 , mistake - 0.0%
estimated heavy hitters amount - 21 , true heavy hitters amount - 21 , mistake - 0.0%

trial - 10
estimated 9050 frequency - 2500 , true 9050 frequency - 2500 , mistake - 0.0%
estimated heavy hitters amount - 21 , true heavy hitters amount - 21 , mistake - 0.0%


<function backward_stream at 0x10799ce60>
mean element estimated frequency of 9050 - 2500.0 , true - 2500 , mistake - 0.0%
mean heavy hitters amount - 21.2 , true - 21 , mistake - 0.952380952380949%
```

```
mean heavy hitters amount - 21.2 , true - 21 , mistake - 0.9523809523809524%
```

Out[34]:

```
(2500.0, 21.2)
```

In backward stream estimated frequency error is 0% \ It seems interesting