**ADS Assignment 3**

**Describe one approximate algorithm for lossy counting &one for sketch**

Lossy Counting Algorithm:

Lossy Counting is an algorithm designed for counting frequent items in a data stream with limited memory. It provides approximate counts with a specified error bound.

Algorithm Steps:

1. Maintain a sliding window of size `w` to process the stream.

2. Keep track of items and their frequencies using a hash table or any other suitable data structure.

3. Process each element in the stream and update the counts in the hash table.

4. Periodically, prune the infrequent items by decrementing their counts. The frequency threshold for pruning is adjusted dynamically based on the total number of elements processed so far.

5. At the end of the stream, the remaining items in the hash table are approximate frequent items.

Here's a simplified Python code snippet for Lossy Counting:

def lossy\_counting(stream, w, error\_threshold):

counts = {}

bucket\_size = int(w / error\_threshold)

for i, item in enumerate(stream):

counts[item] = counts.get(item, 0) + 1

if i % bucket\_size == 0:

counts = {k: v for k, v in counts.items() if v > 1}

return counts

# Example Usage

data\_stream = [1, 2, 3, 1, 2, 4, 1, 5, 2, 1, 2, 3, 1, 2, 4, 1, 5, 2]

window\_size = 5

error\_threshold = 0.2

result = lossy\_counting(data\_stream, window\_size, error\_threshold)

print(result)

Output

{1: 8, 2: 8}

Sketching Algorithm (Count-Min Sketch):

Count-Min Sketch is a probabilistic data structure for estimating frequencies of items in a stream.

Algorithm Steps:

1. Initialize a 2D array (matrix) `counters` with `k` rows and `w` columns, all initialized to 0.

2. Hash each item in the stream using `k` different hash functions, obtaining `k` hash values for each item.

3. Increment the corresponding counters in the matrix for each hashed value.

4. To estimate the frequency of an item, find the minimum value among the counters corresponding to its hashed values.

Here's a simplified Python code snippet for Count-Min Sketch:

import hashlib

class CountMinSketch:

def \_\_init\_\_(self, width, depth):

self.width = width

self.depth = depth

self.counters = [[0] \* width for \_ in range(depth)]

self.hash\_functions = [hashlib.md5, hashlib.sha1, hashlib.sha256] # Example hash functions

def update(self, item):

for i in range(self.depth):

hash\_value = int(self.hash\_functions[i](str(item).encode()).hexdigest(), 16)

column = hash\_value % self.width

self.counters[i][column] += 1

def estimate\_frequency(self, item):

min\_count = float('inf')

for i in range(self.depth):

hash\_value = int(self.hash\_functions[i](str(item).encode()).hexdigest(), 16)

column = hash\_value % self.width

min\_count = min(min\_count, self.counters[i][column])

return min\_count

# Example Usage

data\_stream = [1, 2, 3, 1, 2, 4, 1, 5, 2, 1, 2, 3, 1, 2, 4, 1, 5, 2]

width = 10

depth = 3

cms = CountMinSketch(width, depth)

for item in data\_stream:

cms.update(item)

estimated\_frequency = cms.estimate\_frequency(1)

print("Estimated Frequency of 1:", estimated\_frequency)

Estimated Frequency of 1: 7