Experiment No 7 :Implementing DGIM Algorithm

AIM: To implement DGIM algorithm using any programming language.

Code:

```
import math
from collections import deque

class Dgim(object):
    def __init__(self, N, error_rate=0.5):
        """
        Constructor
        param N: sliding window width.
        param error_rate: the maximum error made by the algorithm.
        """
        self.N = N
        if not (0 < error_rate <= 1):
            error_msg = ("Invalid value for error_rate: {}. "
                         "Error rate should be in ]0, 1].".format(error_rate))
            raise ValueError(error_msg)
        self.error_rate = error_rate

        self._r = math.ceil(1/error_rate)
        self._r = max(self._r, 2)

        self._queues = []
        if N == 0:
            max_index = -1
        else:
            max_index = int(math.ceil(math.log(N)/math.log(2)))

        self._queues = [deque() for _ in range(max_index + 1)]

        self._timestamp = 0
        self._oldest_bucket_timestamp = -1

    def update(self, elt):
        """
        Update the stream with one element.
        """
        if self.N == 0:
            return
        self._timestamp = (self._timestamp + 1) % (2 * self.N)
        if (self._oldest_bucket_timestamp >= 0 and
                self._is_bucket_too_old(self._oldest_bucket_timestamp)):
            self._drop_oldest_bucket()
        if elt is not True:
            return
```

```
    carry_over = self._timestamp
    if self._oldest_bucket_timestamp == -1:
        self._oldest_bucket_timestamp = self._timestamp
    for queue in self._queues:
        queue.appendleft(carry_over)
        if len(queue) <= self._r:
            break
        last = queue.pop()
        second_last = queue.pop()
        carry_over = second_last
        if last == self._oldest_bucket_timestamp:
            self._oldest_bucket_timestamp = second_last

def get_count(self):
    """
    Returns an estimate of the number of "True"
    in the last N elements of the stream.
    """
    result = 0
    max_value = 0
    power_of_two = 1
    for queue in self._queues:
        queue_length = len(queue)
        if queue_length > 0:
            max_value = power_of_two
            result += queue_length * power_of_two
        power_of_two = power_of_two << 1
    result -= math.floor(max_value/2)
    return int(result)

def _is_bucket_too_old(self, bucket_timestamp):
    return (self._timestamp - bucket_timestamp) % (2 * self.N) >= self.N

@property
def nb_buckets(self):
    """
    Returns the number of buckets.
    """
    result = 0
    for queue in self._queues:
        result += len(queue)
    return result

def _drop_oldest_bucket(self):
    """Drop oldest bucket timestamp."""
    for queue in reversed(self._queues):
        if len(queue) > 0:
            queue.pop()
            break
```

```
        self._oldest_bucket_timestamp = -1
        for queue in reversed(self._queues):
            if len(queue) > 0:
                self._oldest_bucket_timestamp = queue[-1]
                break

print("This algorithm uses O(log²N) bits to represent a window of N bit,  ")
print("allows to estimate the number of 1's in the window with and error of no more than 50%.")
dgim_question="101011000101110110010110"
print("The string we are going to use for this is ")
print(dgim_question)

dgim = Dgim(N=32, error_rate=0.6)
for i in dgim_question:
    if i == '1':
        dgim.update(True)
    if i == '0':
        dgim.update(False)

dgim_result = dgim.get_count()
dgim_buc=dgim.nb_buckets

print("The estimate no of true statements or 1 in the stream are : ",dgim_result)
print("The estimate no of Buckets are : ",dgim_buc)
```

Result : You have successfully executed the dgim algorithm using python .

Screenshot: