PAREENITA A.SHIRSATH B.E.A.I.&.D.S. ROLL.NO : 57

BDA EXPERIMENT NO :

```
import math
from collections import deque

class DGIM:
    def __init__(self, N):
        self.N = N              # Window size
        self.buckets = deque()  # Each bucket: (timestamp, size)
        self.current_time = 0

    def _expire_buckets(self):
        # Remove buckets outside the window
        while self.buckets and self.buckets[0][0] <= self.current_time - self.N:
            self.buckets.popleft()

    def _merge_buckets(self):
        # Merge buckets with same size if more than 2 exist
        # Start from the right (most recent)
        i = len(self.buckets) - 1
        while i >= 2:
            # Ensure there are at least 3 buckets and they have the same size
            if (len(self.buckets) > i and
                self.buckets[i][1] == self.buckets[i-1][1] == self.buckets[i-2][1]):

                # Merge the two oldest buckets of the three
                # Create a new bucket with double the size of the merged buckets
                merged_size = self.buckets[i-1][1] * 2
                merged_timestamp = self.buckets[i-1][0]

                # Remove the two oldest buckets (i-2 and i-1)
                del self.buckets[i-2]
                del self.buckets[i-2] # After deleting i-2, the original i-1 is now at i-2

                # Insert the new merged bucket
                self.buckets.insert(i-2, (merged_timestamp, merged_size))

                # Re-check for merges from the right after insertion
                i = len(self.buckets) - 1
            else:
                i -= 1


def add_bit(self, bit):
    self.current_time += 1
    self._expire_buckets()

    if bit == 1:
        # Add new bucket of size 1 with current timestamp
        self.buckets.append((self.current_time, 1))
        self._merge_buckets()

def query(self, k):
    # Approximate number of 1s in the last k bits (k <= N)
    count = 0
    threshold_time = self.current_time - k
    # Iterate from the newest buckets (right)
    for i in range(len(self.buckets) - 1, -1, -1):
        timestamp, size = self.buckets[i]
        if timestamp > threshold_time:
            count += size
        else:
            # This bucket partially overlaps with the k window
            # Add half the size of this bucket as estimation
            count += size // 2
            break # Stop once we find the first bucket that is too old

    return count


# Example usage:
dgim = DGIM(32)  # Window size 32

stream = [1,0,1,1,0,1,0,0,1,1,1,0,1,0,1,0,1,1,0,0,1,1,1,0,0,1,1,0,1,0,0,1]

for bit in stream:
    dgim.add_bit(bit)

print("Approximate count of 1s in last 16 bits:", dgim.query(16))
print("Approximate count of 1s in last 32 bits:", dgim.query(32))
```

```
Approximate count of 1s in last 16 bits: 14
Approximate count of 1s in last 32 bits: 18
```