The Datar-Gionis-Indyk-Motwani (DGIM) Algorithm

The DGIM algorithm is a method for estimating the number of 1s in a sliding window of bits using **$O(\log^2 N)$** bits of memory, where N is the window length. It provides an approximate count with an error margin of no more than 50%.

Core Principles

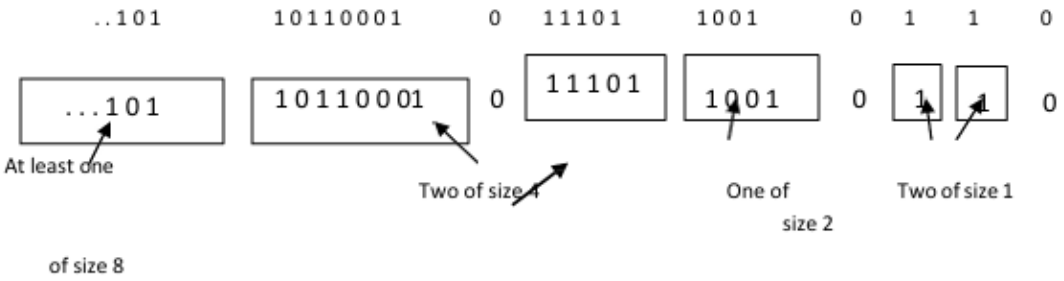
1. **Timestamping**:
    a. Each bit in the stream is assigned a timestamp based on its arrival order. The first bit has a timestamp of 1, the second has a timestamp of 2, and so forth.
    b. To identify positions within a window of length N, timestamps are represented as **timestamps modulo N**. This reduces timestamp storage to **$\log_2 N$** bits.
2. **Bucketization**:
    a. The window is divided into **buckets**, where each bucket represents a collection of consecutive bits.
    b. Each bucket contains:
        i. **Timestamp of its right end**: The most recent bit in the bucket.
        ii. **Number of 1s in the bucket**: The count of 1s, which must be a power of 2 and is referred to as the **bucket size**.
    c. To represent a bucket:
        i. **Timestamp** requires **$\log_2 N$** bits (since timestamps are modulo N).
        ii. **Bucket size** requires **$\log_2(\log_2 N)$** bits because sizes are powers of 2. If a bucket has `2^j` 1s, we can represent the size by storing `j` in binary, which requires **$\log_2(\log_2 N)$** bits.
    d. This results in **$O(\log N)$** bits required per bucket.

Bucket Representation Rules

To ensure accurate representation, DGIM enforces six key rules:

1. The right end of each bucket always coincides with a 1.
2. Every 1 in the stream is included in a bucket.
3. Each bit is part of only one bucket.
4. There are one or two buckets of any given size, up to a certain maximum.
5. All bucket sizes are powers of 2.

6. Bucket sizes do not decrease as we move leftward (backward in time) in the stream.



Advantages of DGIM

- **Efficient Storage**: The algorithm uses only **$O(\log^2 N)$** bits, divided across **$O(\log N)$** counts, each requiring **$\log_2 N$** bits.
- **Simple Updates**: As more bits arrive, buckets can be updated easily, and the error in the count is limited to the number of 1s in the "unknown" area.

Drawbacks of DGIM

- **Limited Error Control**:
  - When the 1s are fairly evenly distributed, the error remains small (no more than 50%).
  - However, if most 1s are concentrated in the unknown area at the end of the window, the error may become unbounded.

Let us take an example to understand the algorithm. Estimating the number of 1's and counting the buckets in the given data stream.

DGIM rules with example →

① Every bucket should contain at least a single 1 in it.

[10100011] ✓ [00000000] ×

② Right side of bucket should strictly start from 1.

[10100011] ✓ [11101000] ×

③ Length of the bucket is equal to the no. of 1s in it.

[10100011]

Length of the bucket is 4.

④ Every bucket length should be in powers of 2.

$2^0 = 1$, $2^1 = 2$, $2^2 = 4$, $2^3 = 8$, $2^4 = 16$ ...

[10100011] ✓ [11000001] ×

⑤ As we move to left, bucket size should not ↓

[10 1000 1011 0 1000 11] ×

[10 1000 11 0 11 0 1000 1 1] ✓

⑥ No more than 2 buckets can have the same size.

[10 1000 11 0 11 0 1000 1 1] ×

[1010001 1 0 11 0 10001 1] ✓