Counting 1's

Approximating Counts
Exponentially Growing Blocks
DGIM Algorithm

Counting Bits

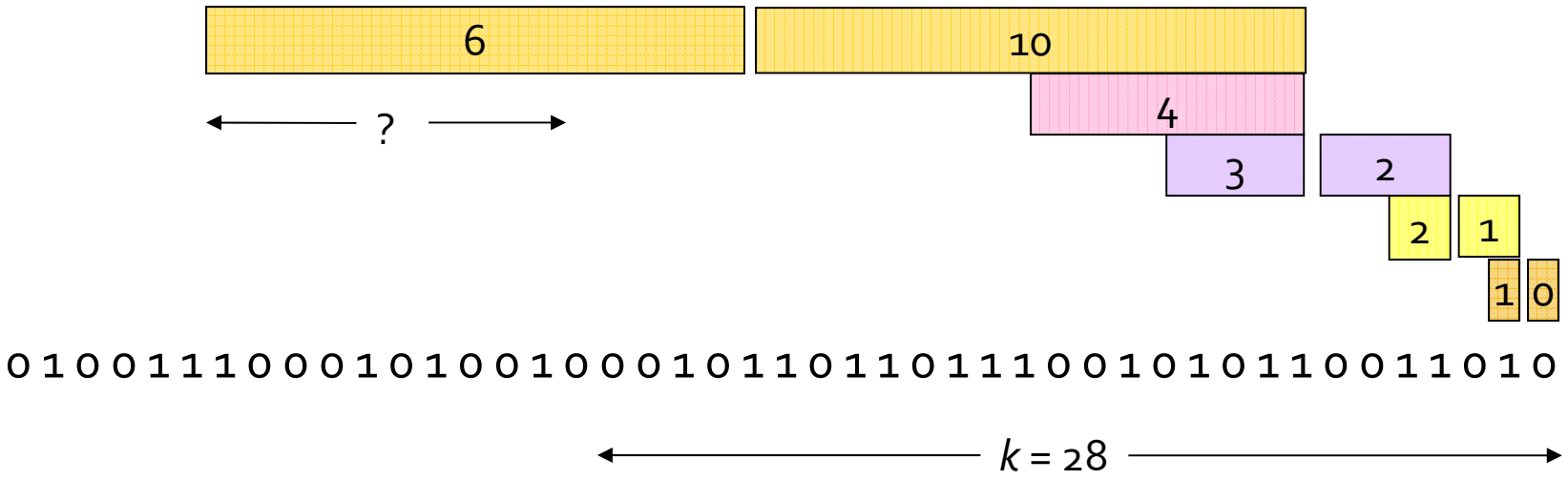
- Problem: given a stream of 0's and 1's, be prepared to answer queries of the form "how many 1's in the last *k* bits?" where $k \leq N$.
- Obvious solution: store the most recent *N* bits.
- But answering the query will take O(k) time.
  - Very possibly too much time.
- And the space requirements can be too great.
  - Especially if there are many streams to be managed in main memory at once, or N is huge.

Something That Doesn't (Quite) Work

- Summarize exponentially increasing blocks of the stream, looking backward.
- Drop small blocks if they begin at the same point as a larger region or a larger region begins to their right.
  - Thus, never more than two blocks of any size.

Example

We can construct the count of the last *k* bits, except we're not sure how many of the last 6 are included.



What's Good?

- Stores only $O(\log^2 N)$ bits.
  - $O(\log N)$ counts of $\log_2 N$ bits each.
- Easy update as more bits enter.
- Error in count no greater than the number of 1's in the "unknown" area.

What's Not So Good?

- As long as the 1's are fairly evenly distributed, the error due to the unknown region is small – no more than 50%.
- But it could be that all the 1's are in the unknown area at the end.
- In that case, the error is unbounded.

Fixup

- Instead of summarizing fixed-length blocks, summarize blocks with specific numbers of 1's.
  - Let the block *sizes* (number of 1's) increase exponentially.
- When there are few 1's in the window, block sizes stay small, so errors are small.

DGIM Method

- Name refers to the inventors:
  - Datar, Gionis, Indyk, and Motwani.
- Store $O(\log^2 N)$ bits per stream.
- Gives approximate answer, never off by more than 50%.
  - Error factor can be reduced to any fraction > 0, with more complicated algorithm and proportionally more stored bits.

Timestamps

- Each bit in the stream has a *timestamp*, starting 0, 1, …
- Record timestamps modulo *N* (the window size), so we can represent any relevant timestamp in $O(\log_2 N)$ bits.

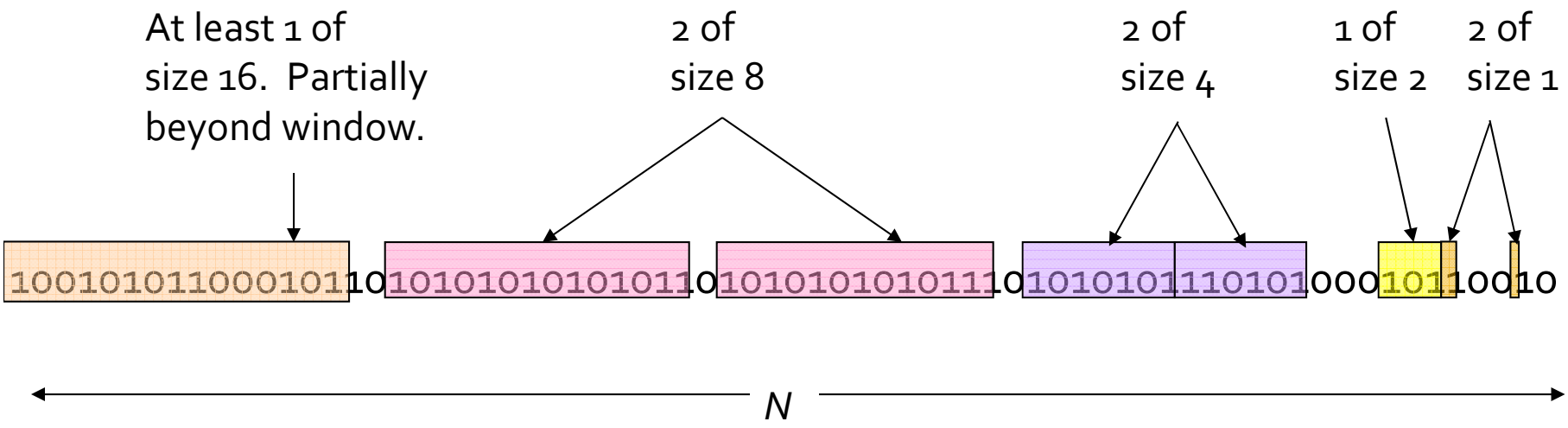
Buckets

- A *bucket* is a segment of the window; it is represented by a record consisting of:
  1. The timestamp of its end [O(log *N*) bits].
  2. The number of 1's between its beginning and end [O(log log *N*) bits].
     - Number of 1's = *size* of the bucket.
- Constraint on buckets: number of 1's must be a power of 2.
  - That explains the log log *N* in (2).

Representing a Stream by Buckets

- Either one or two buckets with the same power-of-2 number of 1's.
- Buckets do not overlap.
- Buckets are sorted by size.
  - Earlier buckets are not smaller than later buckets.
- Buckets disappear when their end-time is > *N* time units in the past.

Example: Bucketized Stream



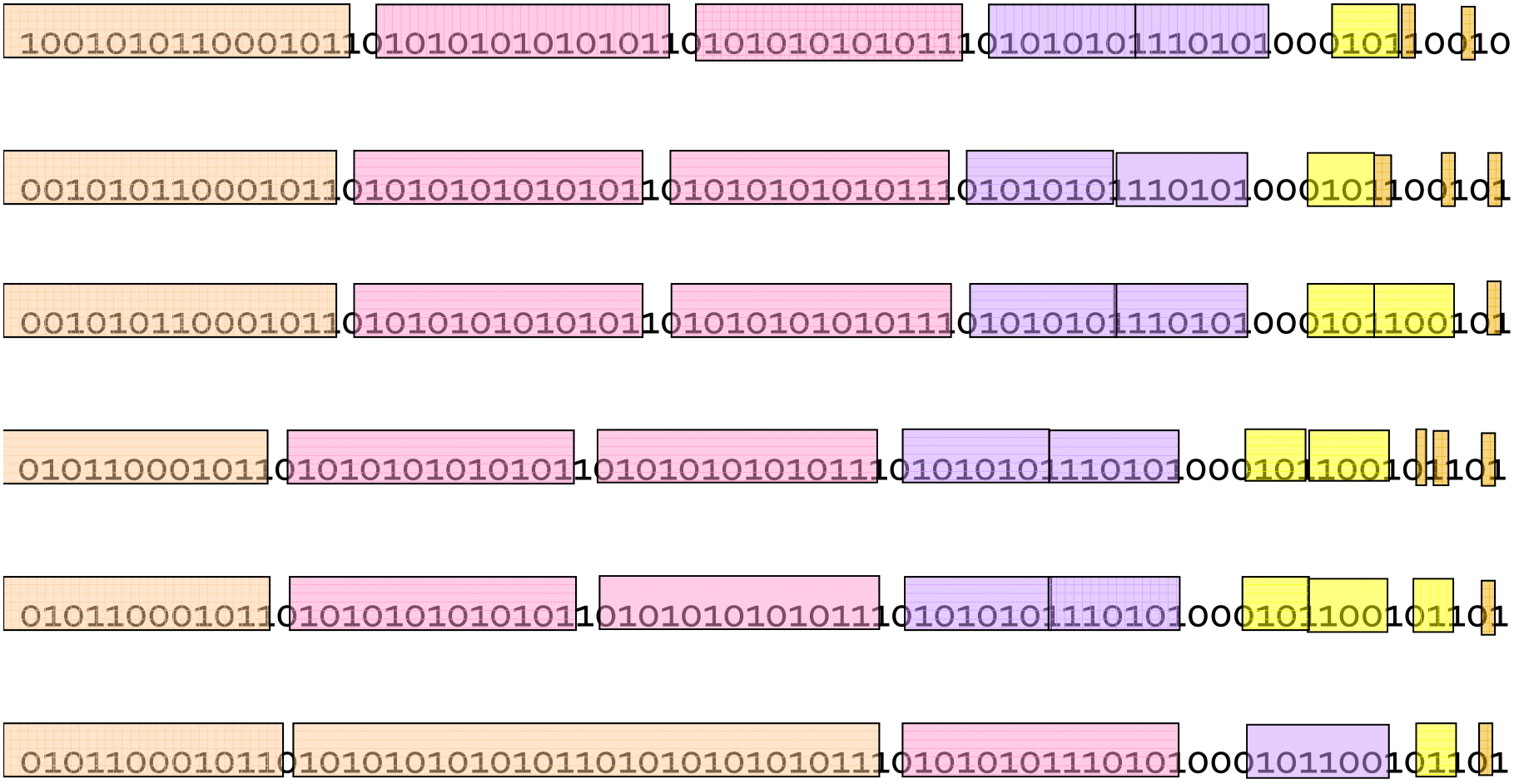
Updating Buckets

- When a new bit comes in, drop the last (oldest) bucket if its end-time is prior to N time units before the current time.
- If the current bit is 0, no other changes are needed.

Updating Buckets: Input = 1

- If the current bit is 1:
  1. Create a new bucket of size 1, for just this bit.
     - End timestamp = current time.
  2. If there are now three buckets of size 1, combine the oldest two into a bucket of size 2.
  3. If there are now three buckets of size 2, combine the oldest two into a bucket of size 4.
  4. And so on …

Example



Querying

- To estimate the number of 1's in the most recent $k \leq N$ bits:
  1. Restrict your attention to only those buckets whose end time stamp is at most k bits in the past.
  2. Sum the sizes of all these buckets but the oldest.
  3. Add half the size of the oldest bucket.
- Remember: we don't know how many 1's of the last bucket are still within the window.

Error Bound

- Suppose the oldest bucket within range has size 2^i.
- Then by assuming 2^{i-1} of its 1's are still within the window, we make an error of at most 2^{i-1}.
- Since there is at least one bucket of each of the sizes less than 2^i, and at least 1 from the oldest bucket, the true sum is no less than 2^i.
- Thus, error at most 50%.