Distinct Counting Problem

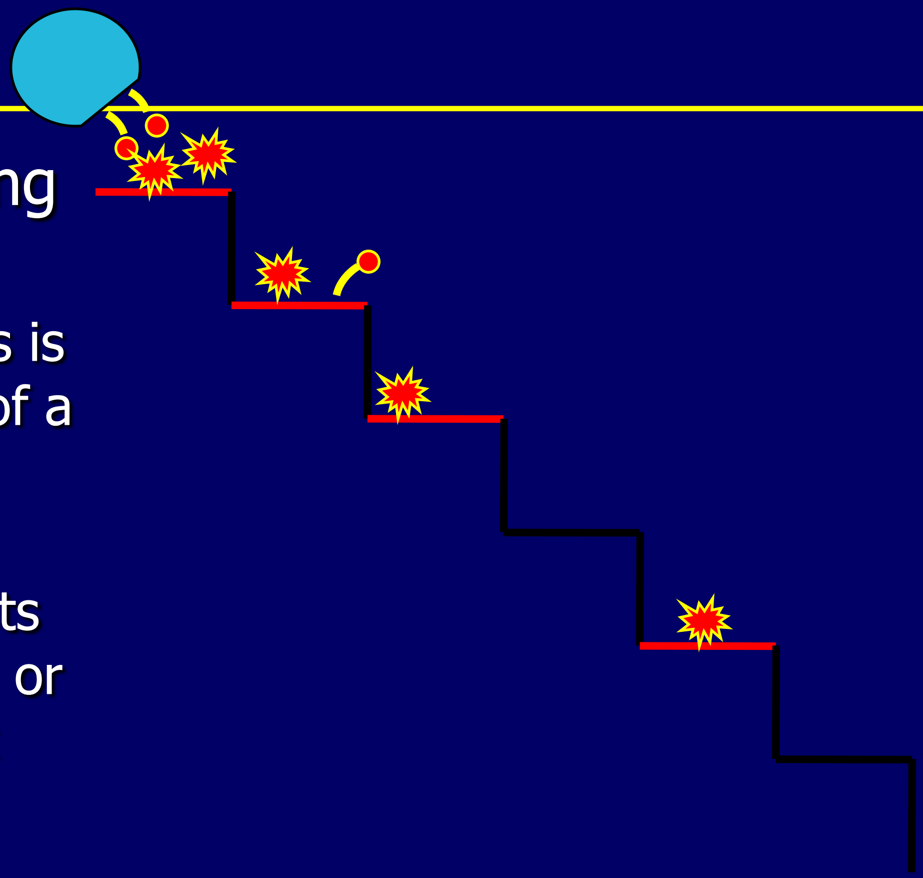
COUNT Sketches

- Problem: Estimate the number of distinct item IDs in a data set with only one pass.
- Constraints:
  - Small space relative to stream size.
  - Small per item processing overhead.
  - Union operator on sketch results.

- Exact COUNT is impossible without linear space.
- First approximate COUNT sketch in [FM'85].
  - O(log N) space, O(1) processing time per item.

Counting Paintballs

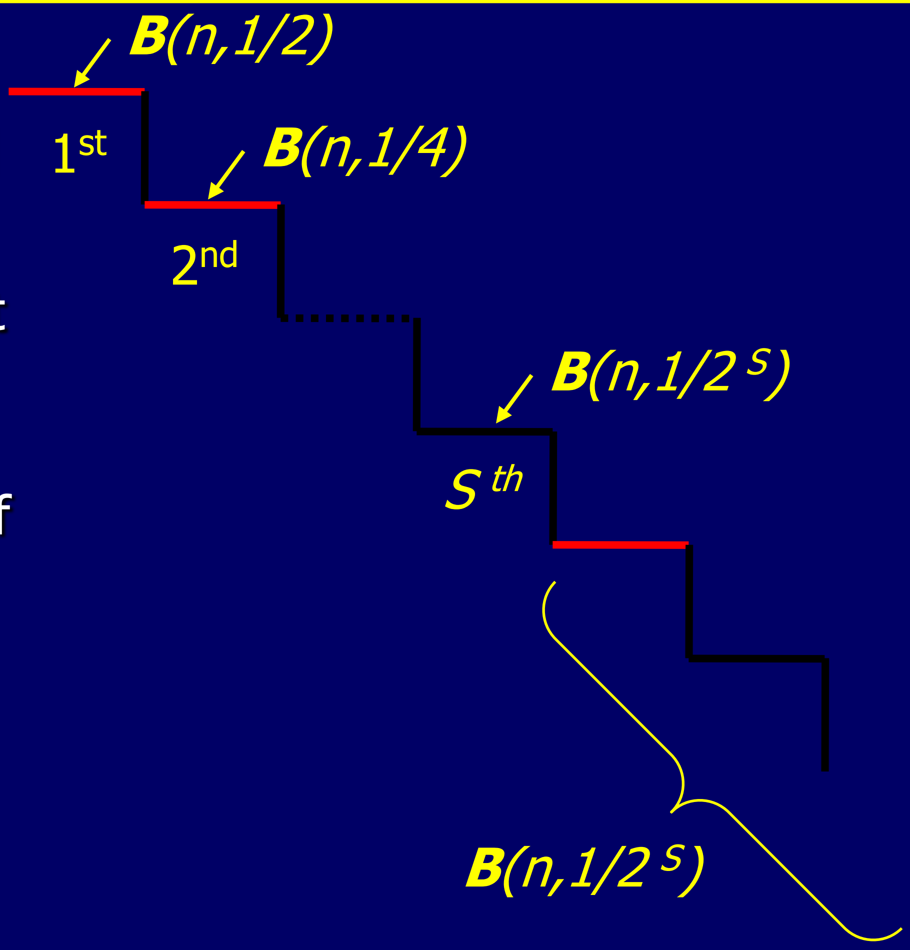
- Imagine the following scenario:
  - A bag of *n* paintballs is emptied at the top of a long stair-case.
  - At each step, each paintball either bursts and marks the step, or bounces to the next step. 50/50 chance either way.



Looking only at the pattern of marked steps, what was *n*?

Counting Paintballs (cont)

- What does the distribution of paintball bursts look like?
  - The number of bursts at each step follows a binomial distribution.
  - The expected number of bursts drops geometrically.
  - Few bursts after $log_2\, n$ steps



Counting Paintballs (cont)

- Many different estimator ideas [FM'85,AMS'96,GGR'03,DF'03,...]
- Example: Let *pos* denote the position of the highest unmarked stair,

$$E(pos) \approx \log_2(0.775351\, n)$$

$$\sigma^2(pos) \approx 1.12127$$

- Standard variance reduction methods apply
- Either O(log n) or O(log log n) space

Back to COUNT Sketches

- The COUNT sketches of [FM'85] are equivalent to the paintball process.
  - Start with a bit-vector of all zeros.
  - For each item,
    - Use its ID and a hash function for coin flips.
    - Pick a bit-vector entry.
    - Set that bit to one.
- These sketches are duplicate-insensitive:

$\{x\} \rightarrow$	1	0	0	0	0
$\{y\} \rightarrow$	0	0	1	0	0
$\{x,y\} \rightarrow$	1	0	1	0	0

$$\forall A,B\ \ (Sketch(A) \vee Sketch(B)) = Sketch(A \cup B)$$

SUM Sketches

- Problem: Estimate the sum of values of distinct <*key, value*> pairs in a data stream with repetitions. (*value* ≥ 0, integral).

- Obvious start: Emulate *value* insertions into a COUNT sketch and use the same estimators.

  – For <*k,v*>, imagine inserting

  <k, v, 1>, <k, v, 2>, …, <k, v, v>

SUM Sketches (cont)

- But what if the value is 1,000,000?

- Main Idea (details on next slide):
  - Recall that all of the low-order bits will be set to 1 w.h.p. inserting such a value.
  - Just set these bits to one immediately.
  - Then set the high-order bits carefully.

Simulating a set of insertions

- Set all the low-order bits in the "safe" region.
  - First $S = \log v - 2 \log \log v$ bits are set to 1 w.h.p.
- Statistically estimate number of trials going beyond "safe" region
  - Probability of a trial doing so is simply 2^{-S}
  - Number of trials $\sim \boldsymbol{B}(v, 2^{-S})$. [Mean = $O(\log^2 v)$]
- For trials and bits outside "safe" region, set those bits manually.
  - Running time is O(1) for each outlying trial.

Expected running time:

$O(\log v)$ + time to draw from $\boldsymbol{B}(v, 2^{-S})$ + $O(\log^2 v)$

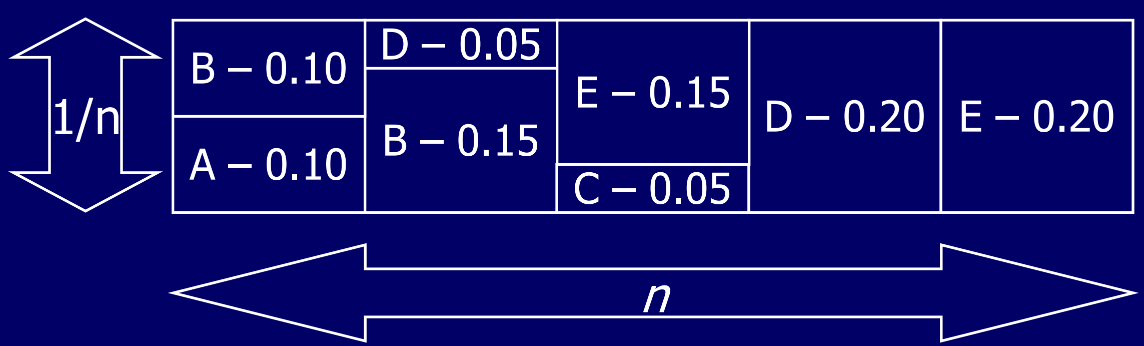
Sampling for Sensor Networks

- We need to generate samples from ***B*** (*n, p*).
  - With a slow CPU, very little RAM, no floating point hardware
- General problem: sampling from a discrete pdf.
- Assume can draw uniformly at random from [0,1].
- With an event space of size *N*:
  - O(log *N*) lookups are immediate.
    - Represent the cdf in an array of size *N*.
    - Draw from [0, 1] and do binary search.
  - Cleverer methods for O(log log *N*), O(log* *N*) time

Amazingly, this can be done in constant time!

Walker’s Alias Method

- Theorem [Walker ’77]: For any discrete pdf D over a sample space of size n, a table of size *O(n)* can be constructed in *O(n)* time that enables random variables to be drawn from D using at most two table lookups.



Binomial Sampling for Sensors

- Recall we want to sample from $\boldsymbol{B}(v,2^{-S})$ for various values of v and S.
  - First, reduce to sampling from $\boldsymbol{G}(1-2^{-S})$.
  - Truncate distribution to make range finite (recursion to handle large values).
  - Construct tables of size 2^S for each S of interest.
  - Can sample $\boldsymbol{B}(v,2^{-S})$ in $O(v \cdot 2^{-S})$ expected time.

Fact

- Suppose we have a method to repeatedly draw at random from G(1-p). Let d be the random variable that records the number of draws from G(1-p) until the sum of the draws exceeds n. The value d-1 is then equivalent to a random draw from B(n, p).

The Bottom Line

- SUM inserts in
  - $O(\log^2(v))$ time with $O(v / \log^2(v))$ space
  - $O(\log(v))$ time with $O(v / \log(v))$ space
  - $O(v)$ time with naïve method
- Using $O(\log^2(v))$ method, 16 bit values ($S \leq 8$) and 64 bit probabilities
  - Resulting lookup tables are ~ 4.5KB
  - Recursive nature of $\boldsymbol{G}(1 - 2^{-S})$ lets us tune size further
- Can achieve $O(\log v)$ time at the cost of bigger tables