Study Group

Mar 8

Agenda

- Basic Streaming Model (and a few examples)
- Probabilistic Methods
  - Union Bound
  - Local Lemma (Symmetric Case)
- Play VR

Streaming Model (1/3)

Input: A stream (sequence) S of data is given one by one to algorithms.

Requirement: Algorithms are allowed to use $o(|S|)$ space.

Output: Write the solution to a write-only stream.

$O(\text{poly} \log |S|) = O(\log^k |S|)$ for some constant k.

In the literature, streaming algorithms are defined to be those algorithms that use $O(\text{poly} \log |S|)$ space. However, this requirement may be relaxed for some domains.

In this course, we shall concerned with algorithms that compute some function of a massively long input stream σ. In the most basic model (which we shall call the *vanilla streaming model*), this is formalized as a sequence $\sigma = \langle a_1, a_2, \ldots, a_m \rangle$, where the elements of the sequence (called *tokens*) are drawn from the universe $[n] := \{1, 2, \ldots, n\}$. Note the two important size parameters: the stream length, m, and the universe size, n. If you read the literature in the area, you will notice that some authors interchange these two symbols. In this course, we shall consistently use m and n as we have just defined them.

Our central goal will be to process the input stream using a small amount of *space* s, i.e., to use s bits of random-access working memory. Since m and n are to be thought of as "huge," we want to make s much smaller than these; specifically, we want s to be *sublinear* in both m and n. In symbols, we want

$$s = o(\min\{m, n\}) .$$

The holy grail is to achieve

$$s = O(\log m + \log n) ,$$

because this amount of space is what we need to store a constant number of tokens from the stream and a constant number of counters that can count up to the length of the stream. Sometimes we can only come close and achieve a space bound of the form $s = \text{polylog}(\min\{m, n\})$, where $f(n) = \text{polylog}(g(n))$ means that there exists a constant $c > 0$ such that $f(n) = O((\log g(n))^c)$.

Streaming Model (3/3)

Algorithms may scan the input from the beginning to the end multiple times, say p times, which are called p-pass algorithms.

Example 1 - Min

Input: A stream (sequence) S of integers is given one by one to algorithms.

Requirement: Algorithms are allowed to use $o(|S|)$ space.

Output: Write $\min_{x \in S} x$ to a write-only stream.

Yes, $O(1) = o(|S|)$ space suffice.

How to find the k-th smallest integers in S where $k \ll |S|$?

Definition 0.2.1. Let $\mathcal{A}(\sigma)$ denote the output of a randomized streaming algorithm $\mathcal{A}$ on input σ; note that this is a random variable. Let ϕ be the function that $\mathcal{A}$ is supposed to compute. We say that the algorithm (ε, δ)-approximates ϕ if we have

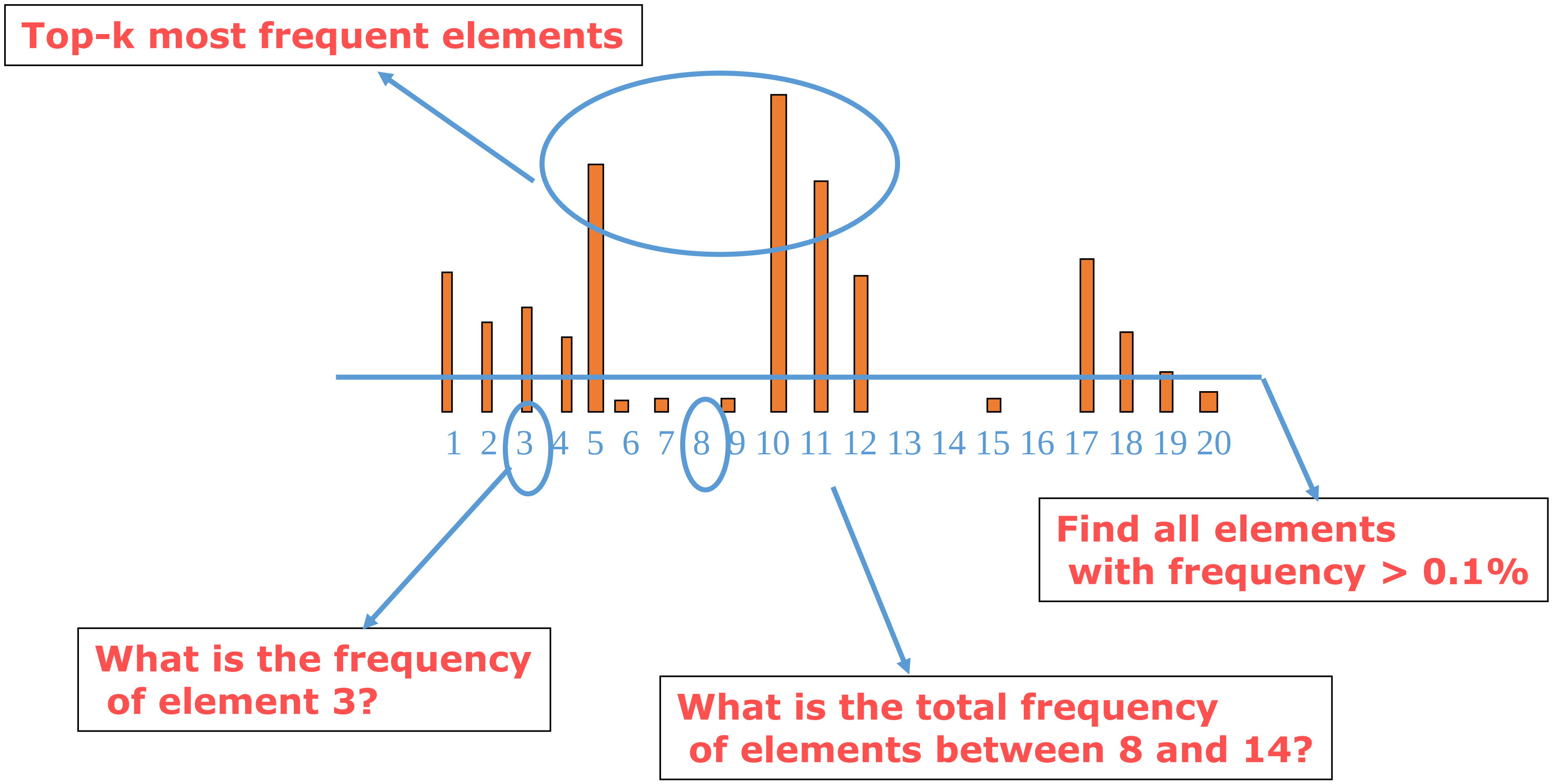
$$\Pr\left[\left|\frac{\mathcal{A}(\sigma)}{\phi(\sigma)} - 1\right| > \varepsilon\right] \le \delta .$$

Notice that the above definition insists on a multiplicative approximation. This is sometimes too strong a condition when the value of $\phi(\sigma)$ can be close to, or equal to, zero. Therefore, for some problems, we might instead seek an additive approximation, as defined below.

Definition 0.2.2. In the above setup, the algorithm $\mathcal{A}$ is said to (ε, δ)-additively-approximate ϕ if we have

$$\Pr\left[|\mathcal{A}(\sigma) - \phi(\sigma)| > \varepsilon\right] \le \delta .$$

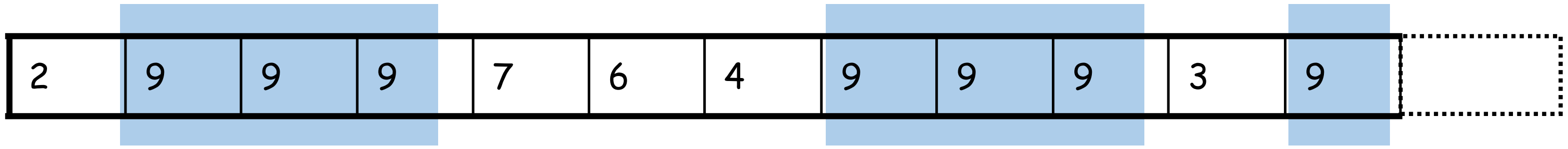
Frequency Related Problems



How many elements have non-zero frequency?

An Old Chestnut: Majority

- A sequence of N items.
- ~~You have constant memory.~~
- In one pass, decide if some item is in majority (occurs > N/2 times)?



N = 12; item 9 is majority

Misra-Gries Algorithm (‘82)

- A counter and an ID.
  - If new item is same as stored ID, increment counter.
  - Otherwise, decrement the counter.
  - If counter <0, store new item with count = 1.

- If counter > 0, then its item is the only candidate for majority.
- Space O(logN) (for the counter) + O(logM) (for the ID)

	2	9	9	9	7	6	4	9	9	9	3	9
ID	2	2	9	9	9	9	4	4	9	9	9	9
count	1	0	1	2	1	0	1	0	1	2	1	2

A generalization: Frequent Items (Karp 03)

Find k items, each occurring at least N/(k+1) times.

ID	ID_1	ID_2	.	.	.	.	ID_k
count	.	.					

- Algorithm:
  - Maintain k items, and their counters.
  - If next item x is one of the k, increment its counter.
  - Else if a zero counter, put x there with count = 1
  - Else (all counters non-zero) decrement all k counters

Problem of False Positives

- False positives in Misra-Gries(MG) algorithm
  - It identifies all true heavy hitters, but not all reported items are necessarily heavy hitters.
  - How can we tell if the non-zero counters correspond to true heavy hitters or not?
- A second pass is needed to verify.
- False positives are problematic if heavy hitters are used for billing or punishment.
- What guarantees can we achieve in one pass?

Approximation Guarantees

- Find heavy hitters with a guaranteed approximation error [MM02]

- Manku-Motwani (*Lossy Counting*)
  - Suppose you want φ-heavy hitters --- items with freq > φN
  - An approximation parameter ε, where $\varepsilon << \varphi$.
    (E.g., $\varphi = .01$ and $\varepsilon = .0001$; $\varphi = 1\%$ and $\varepsilon = .01\%$)

  - **Identify all items with frequency $> \varphi N$**
  - **No reported item has frequency $< (\varphi - \varepsilon)N$**

- The algorithm uses $O(1/\varepsilon \log (\varepsilon N))$ memory

The Union Bound

The union bound, a.k.a. Boole's inequality, is stated as follows. For any countable set of probablistic events A_1, A_2, ..., we have

$$\Pr\left[\bigcup_i A_i\right] \leq \sum_i \Pr[A_i]$$

Note that A_i's may be dependent or independent.

Coloring Hypergraphs

We say a hypergraph is 2-colorable if there exists a coloring on nodes so that every edge is not monochromatic.

Recall that a hypergraph $H = (V, E)$ is defined as ordinary graphs except that each edge $e \in E$ is a subset of V. We say a hypergraph k-uniform if all edges in E has cardinality k.

Coloring Hypergraphs

We say a hypergraph is 2-colorable if there exists a coloring on nodes so that every edge is not monochromatic.

Recall that a hypergraph $H = (V, E)$ is defined as ordinary graphs except that each edge $e \in E$ is a subset of V. We say a hypergraph k-uniform if all edges in E has cardinality k.

Theorem 2. Let H be a k-uniform hypergraph.

If $|E| \leq 2^{k-1}$, then H is 2-colorable.

Proof Strategy. Assign a random 2-coloring on H.

Coloring Hypergraphs

Let A_e for each $e \in H$ be the event that e is monochromatic.

Thus, $Pr[A_e] = 1/2^{k-1}$. (Why?)

By the Union Bound $$\Pr\left[\bigcup_i A_i\right] \leq \sum_i \Pr[A_i]$$

Pr [Some e is monochromatic] $\leq 2^{k-1} \times 1/2^{k-1} = 1$

Pr [No e is monochromatic] ≥ 0

Objectives

- Introduce Lovasz Local Lemma (LLL)
  - one of the most elegant and useful tools in the probabilistic method

- Two versions:
  - symmetric case
  - general case

Lovasz Local Lemma

- Let $E_1, E_2, \ldots, E_n$ be a set of BAD events
- Suppose each occurs with prob < 1

Fact: If they are mutually independent, it is easy to see that

$$\Pr(\text{no BAD events}) > 0$$ … [why?]

- However, in many natural scenario, the BAD events are not mutually independent

Problem: Can we still easily show that

$$\Pr(\text{no BAD events}) > 0 \ ?$$

Lovasz Local Lemma (2)

- In general, probably not...
- But, if there are not many dependency among the BAD events, then the set of events are 'roughly' mutually independent

  we may still be able to show

  Pr(no BAD events) > 0 ...

- Lovasz Local Lemma gives sufficient conditions when we can do so ...
  - It relies on a concept of dependency graph defined as follows (next slide)

Dependency Graph

Let E be an event

Definition: E is mutually independent of a set of events $\{E_1, E_2, \ldots, E_n\}$ if

for any $I \subseteq [1,n]$, $\Pr(E \mid \cap_{j \in I} E_j) = \Pr(E)$

Definition: A dependency graph for a set of events $\{E_1, E_2, \ldots, E_n\}$ is a graph $G=(V,E)$, $V = \{1,2,\ldots,n\}$ such that

for any j, E_j is mutually independent of the events $\{ E_k \mid (j,k) \notin E \}$

Dependency Graph (2)

Test your understanding:

1. Let S be a set of pair-wise independent events. Is a graph with no edges always a dependency graph of S ?

2. Let S be a set of events.
   Is the dependency graph of S unique?

The answers are NO for both questions…

Dependency Graph (3)

Consider flipping a fair coin twice.

Let E_1 = the first flip is head

E_2 = the second flip is tail

E_3 = the two flips are the same

the events are pairwise independent

We see that if a graph has less than 2 edges, it must not be a dependency graph

On the other hand, any graph with 2 or more edges is a dependency graph !!!

Lovasz Local Lemma

(Symmetric Case)

Theorem: Let G be a dependency graph of a set of BAD events $\{ E_1, E_2, \ldots, E_n \}$. If

(i) $\Pr(E_j) \leq p < 1$ for each E_j,

(ii) $1 \leq \mathrm{maxdeg}(G) \leq d$, and

(iii) $ep(d+1) \leq 1$

then $\Pr(\text{no BAD events}) > 0$

Remark: If $\mathrm{maxdeg}(G) = 0$, then $\Pr(\text{no BAD events}) > 0$ since all events are mutually independent

Coloring Hypergraphs (Revisited)

We say a hypergraph is 2-colorable if there exists a coloring on nodes so that every edge is not monochromatic.

Recall that a hypergraph H = (V, E) is defined as ordinary graphs except that each edge $e \in E$ is a subset of V. We say a hypergraph k-uniform if all edges in E has cardinality k.

Theorem 2. Let H be a k-uniform hypergraph and each edge in H has an non-empty intersection with at most d other edges.

If $(d+1) \leq 2^{k-1}/e$, then H is 2-colorable.

Comparison

Union Bound:

If $|E| \leq 2^{k-1}$, then H is 2-colorable.

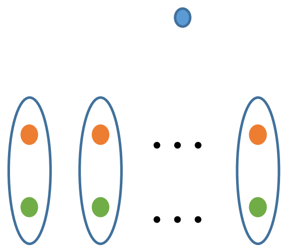
LLL:

If $(d+1) \leq 2^{k-1}/e$, then H is 2-colorable.

Q: Which bound is tighter?

Comparison

Generally, LLL is more powerful than Union Bound.
But consider this example:



Union Bound:

If $|E| \leq 2^{k-1}$, then H is 2-colorable.

LLL:

If $(d+1) \leq 2^{k-1}/e$, then H is 2-colorable.

1 blue node, (k-1) orange nodes, (k-1) green nodes
Each edge contains the blue node and one node from each circle

$|E| = 2^{k-1}$, so the Union Bound works
But (d+1) also equals to 2^{k-1}, so the (Symmetric) LLL fails

Why?