Streaming Algorithms

By Koko Nanahji

Outline

- Motivation
- Model
- Focus
- Simple Example
- Another Example
- Overview of Exact Deterministic Algorithms
- Approximate Randomized Algorithms
- Approximate Randomized Algorithm Example
- Conclusion

Motivation

- Compute statistics about a given stream
  - For example: number of distinct IP addresses

Outline

- Motivation
- Model
- Focus
- Simple Example
- Another Example
- Overview of Exact Deterministic Algorithms
- Approximate Randomized Algorithms
- Approximate Randomized Algorithm Example
- Conclusion

Model

- Input: sequence of integers $x_1, x_2, \ldots, x_n$
  - $x_i \in U$
- Goal: compute some function f_i on the input stream

Common Models

- Time Series Model:
  - U consists of pairs (k, v) where $k \in \{1, \ldots, m\}$ and $v \in \mathbb{R}$
  - f_i keeps an array A of size m and does the following:
    - For all (k, v) do $A[k] \mathrel{+}= v$
- There are variants of this model which restrict the values of v
  - Turnstile Model: $v \in \mathbb{R}$
  - Strict Turnstile Model: $A[k] \geq 0$ for all k at all times during the execution of the algorithm
  - Cash Register Model: $v > 0$

Common Parameter

- Window:
  - Approximate the objective function for some fixed window of the stream instead of the whole stream
  - Add weights to each value received
    - Weights decay over time

Outline

- Motivation
- Model
- Focus
- Simple Example
- Another Example
- Overview of Exact Deterministic Algorithms
- Approximate Randomized Algorithms
- Approximate Randomized Algorithm Example
- Conclusion

Focus

- Main focus:
  - Reduce the amount of memory used to process the input

Outline

- Motivation
- Model
- Focus
- Simple Example
- Another Example
- Overview of Exact Deterministic Algorithms
- Approximate Randomized Algorithms
- Approximate Randomized Algorithm Example
- Conclusion

Simple Example

- Input: sequence of distinct integers $x_1, x_2, \ldots, x_n$
  - $x_i \in \{1,2, \ldots, n+1\}$
- Goal: Find the missing value

Simple Example

- Naïve solution:
  - Keep an array A of $n + 1$ bits all set to zero
  - If $x_i = j$ then $A[j] = 1$
  - Result: report the index of the array that is set to zero
- Requires: $n + 1$ bits for the array

Simple Example – $O(\log n)$ Solution

- Use variable agg
  - Keep a running aggregate of the values received in agg
  - Report: $\left(\frac{(n+1)(n+2)}{2} - agg\right)$
- agg requires $\log(n+1)$ bits

Outline

- Motivation
- Model
- Focus
- Simple Example
- Another Example
- Overview of Exact Deterministic Algorithms
- Approximate Randomized Algorithms
- Approximate Randomized Algorithm Example
- Conclusion

Another Example

- Input: sequence of integers $x_1, x_2, \ldots, x_n$
  - $x_i \in \{1,2,\ldots,m\}$
- Let f_i be the frequency of element i in the given sequence
- Goal: Given some integer k find the elements that have $f_i > \frac{n}{k}$
- No deterministic algorithm that has one pass over the sequence
- There is a simple two pass algorithm

Misrea – Gries Algorithm (High Level)

- During the first pass:
  - Identify a small set of candidates for k-frequent elements.
- During the second pass:
  - Maintain an explicit count of the number of times each candidate appears

Misrea – Gries Algorithm

- Observation: There can be at most $k-1$ elements that have $f_i > \frac{n}{k}$
- Algorithm:
  - M is map of size $k-1$
  - First Pass when receiving x_i:
    - If x_i is in $M.keys$ then
      - increment $M[x_i]$
    - Else if $|M.keys| < k-1$ then
      - $M[x_i] = 1$
    - Else
      - for all k ∈ $M.keys$ $M[k] -= 1$
      - Remove all entries k from M if $M[k] = 0$
  - Second pass:
    - For each e ∈ $\{1,2,\ldots,m\}$ if $e \in M.keys$ then approx. frequency of e is $M[e]$
      - Else approx. frequency of e is 0

Misrea – Gries Algorithm

- Results:
  - Total space used $O\big(k(\log m + \log n)\big)$
  - For each element the frequency outputted is in $[f_i - \frac{n}{k}, f_i]$

Outline

- Motivation
- Model
- Focus
- Simple Example
- Another Example
- Overview of Exact Deterministic Algorithms
- Approximate Randomized Algorithms
- Approximate Randomized Algorithm Example
- Conclusion

Exact Deterministic Algorithms

- In streaming algorithms:
  - Exact deterministic solutions are extremely rare
  - It can be proven for most problems:
    - Both approximation and randomness are required for non-trivial problems
      - E.g. Frequent items problem: Count Min algorithm

Outline

- Motivation
- Model
- Focus
- Simple Example
- Another Example
- Overview of Exact Deterministic Algorithms
- Approximate Randomized Algorithms
- Approximate Randomized Algorithm Example
- Conclusion

Approximate Randomized Algorithms

- Formal definition:
  - Universe of items U and a family of functions F
  - For a given $f_i : U^n \to \mathbb{R}$ where $f_i \in F$
  - A stream of inputs $x_1, x_2, \ldots, x_n$
  - Goal: Design algorithm that uses little memory and approximates $f_i(x_1, x_2, \ldots, x_n)$
  - A randomized algorithm R_i is said to (ϵ, δ) approximate f_i if for all $x_1, x_2, \ldots, x_n$ we have $\Pr\left(\left|\frac{R_i(x_1,x_2,\ldots,x_n)}{f_i(x_1,x_2,\ldots,x_n)} - 1\right| > \epsilon\right) \leq \delta$
- Here we are interested in the space used and the approximation guarantee.

Outline

- Motivation
- Model
- Focus
- Simple Example
- Another Example
- Overview of Exact Deterministic Algorithms
- Approximate Randomized Algorithms
- **Approximate Randomized Algorithm Example**
- **Conclusion**

Approx. Number of Distinct Elements

- Input: sequence of integers $x_1, x_2, \dots, x_n$
  - $x_i \in \{1,2, \dots, m\}$
- Let f_i be the number of distinct elements in $x_1, x_2, \dots, x_n$
- If we are restricted to either deterministic algorithms, or exact algorithms
  - It is impossible to solve this problem in sublinear space

Approx. Number of Distinct Elements

- For any integer x we define
  - $zeros(x) \coloneqq max\{i: 2^i\ divides\ x\}$
  - i.e. number of zeros x's binary representation ends with

Approx. Number of Distinct Elements

- Algorithm
  - Pick a hash function H from a universal family
  - Have a local variable z initially 0
  - Upon receiving x_i do the following:
    - If $zeros\left(h(x_i)\right) > z$ then z $= zeros\left(h(x_i)\right)$
  - After processing all n elements
    - Output $2^{z+\frac{1}{2}}$
  - Run k copies of this algorithm in parallel, using independent random hash functions, and outputting the median of the k answers
- These would result in “good” approximation to the actual function

Outline

- Motivation
- Model
- Focus
- Simple Example
- Another Example
- Overview of Exact Deterministic Algorithms
- Approximate Randomized Algorithms
- Approximate Randomized Algorithm Example
- **Conclusion**

Conclusion

- Streaming algorithms are very useful in practice
- There isn’t much we can do if we want determinist and exact streaming algorithms
- Randomized approximate algorithms help us find practical algorithms