amazonTechO(n)

Data Stream Algorithms

Florin Manolache
Software Development Manager

amazonTechO(n)

Agenda

- What is a Data Stream?
- Why are Data Streams difficult to process?
- Common problems and generic solutions
- Heavy Hitters problem
- Takeaways

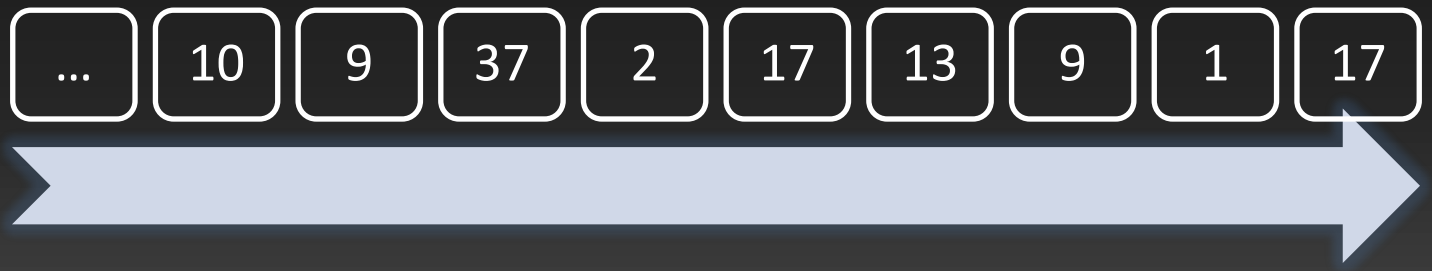
What is a Data Stream?

- Large Data Set which is hard to:
  - Process (by classic algorithms)
  - Transfer
  - Store (in a single location)

- Examples:
  - Sensor data from Curiosity, LHC
  - Traffic on a backbone router
  - Traffic to a popular website (Google, Amazon, Facebook)
  - DDoS traffic to a website

Difficult to process

- Unbounded number of elements

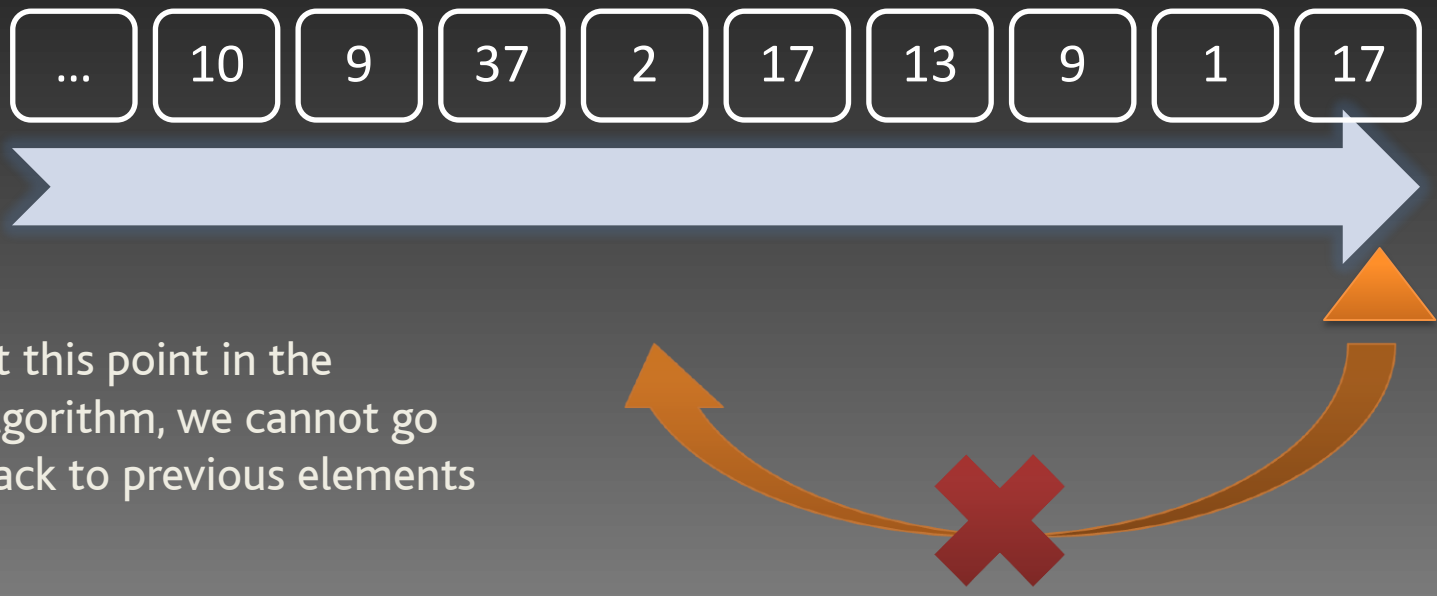


Element	1	2	9	10	13	17	...
Count	1	1	2	1	1	2	...

...but we want finite (pre-determined) space, depending on the problem

Difficult to process

- Short processing time for each element in the stream
- No random data access



Common problems …

- Summary of data
  - Sampling
  - Statistics: average, median, percentile, histogram
  - Distinct elements
  - Most frequent elements, top k elements
- Filtering
- Data stream mining
- Data stream clustering
- Pattern matching

… and generic solutions

- Almost all algorithms are approximate
- We assume worst case input
- Use random inside the algorithm (not in data)

As a result:

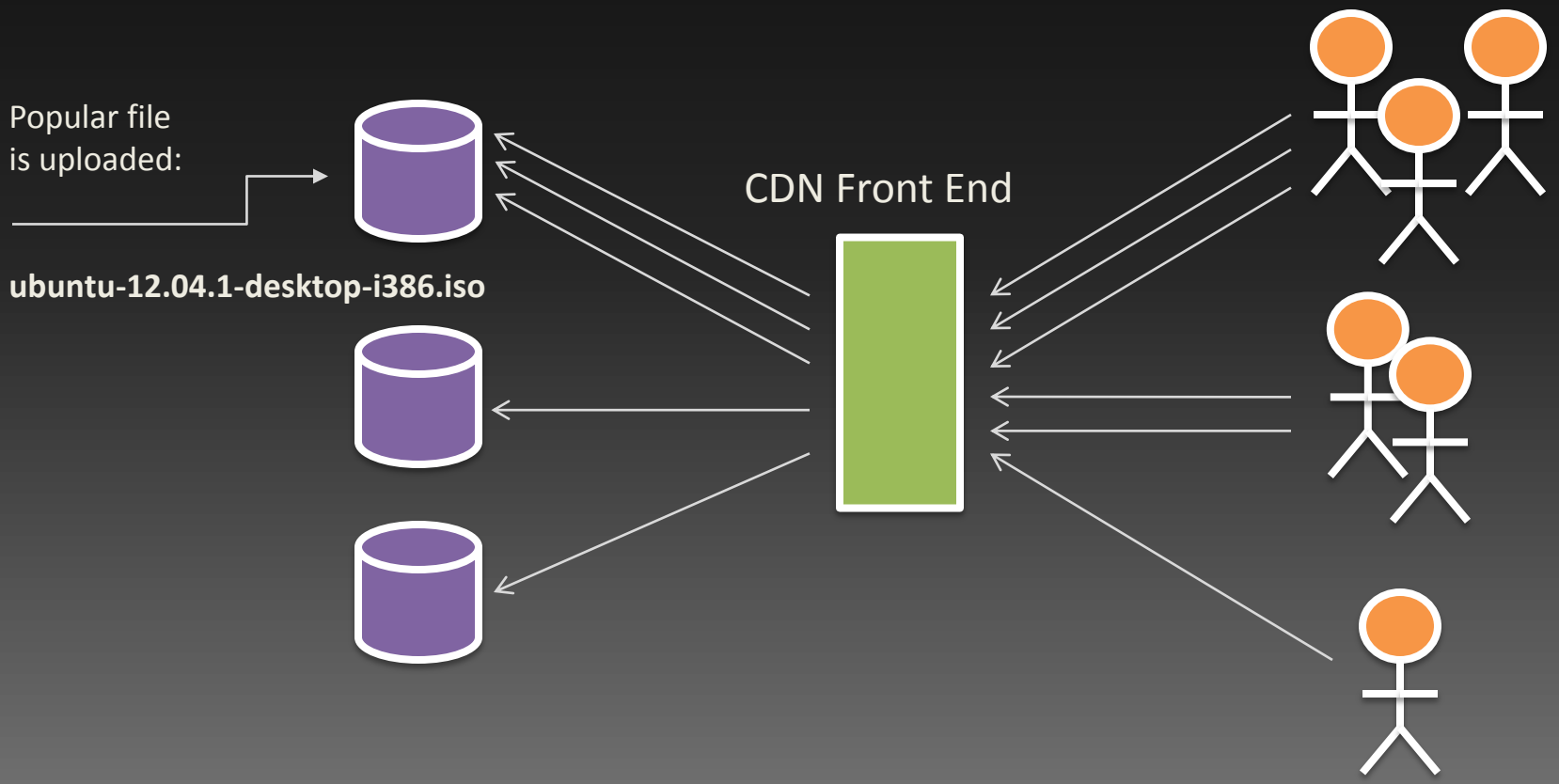
- Non-intuitive (“hard”) algorithms
- Error analysis is hard
- Both space and time complexity are important

… and generic solutions

- "Sketching" a general technique
  - Try to build a small data-structure to represent the data you want to obtain from the stream
  - The smaller the data structure, the less accurate the results
  - Usually use hashing when accessing the data structure

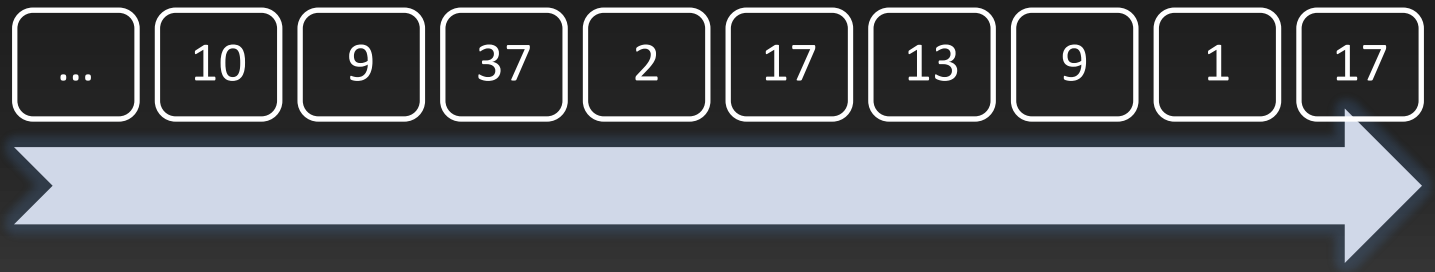
amazonTechO(n)

Content Delivery Network Example



Heavy Hitters

Given a stream of integers, find the ones which appear most frequently



In our example we want to find 9 and 17 as the most frequent elements

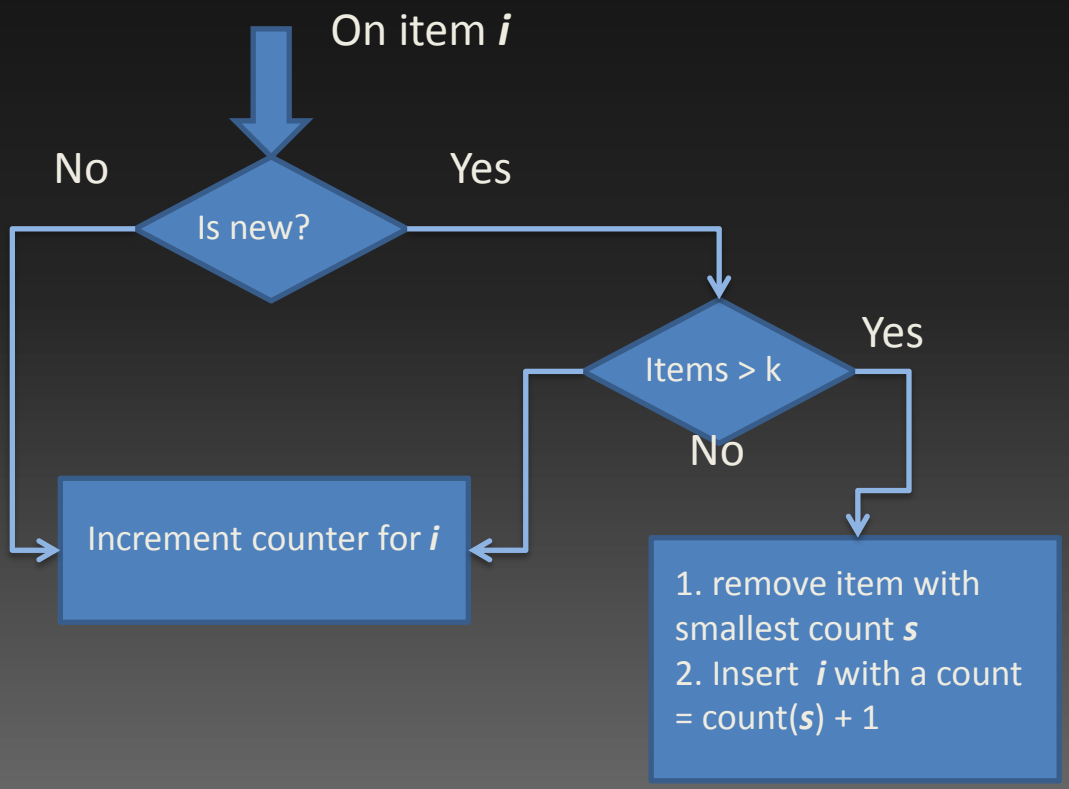
Space Saving algorithm

- Keep k items and counts initially zero
- Count first k distinct items exactly
- On seeing a new item:
  - If it is already in our set, increment counter
  - If not, replace item with least count, increment count

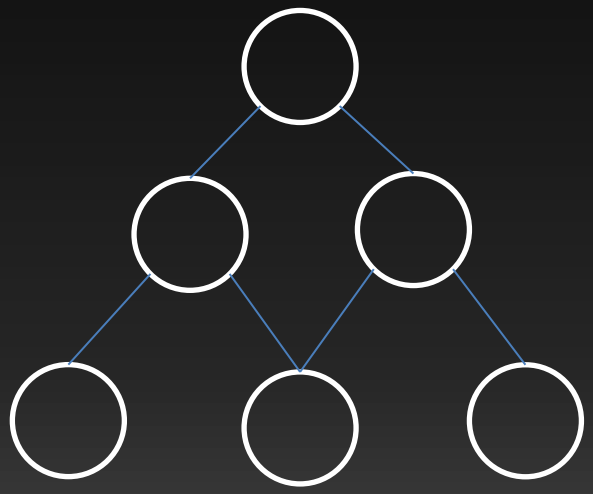
The algorithm overestimates, when replacing an item we can keep the old counter in our data structure as the "overestimate" value

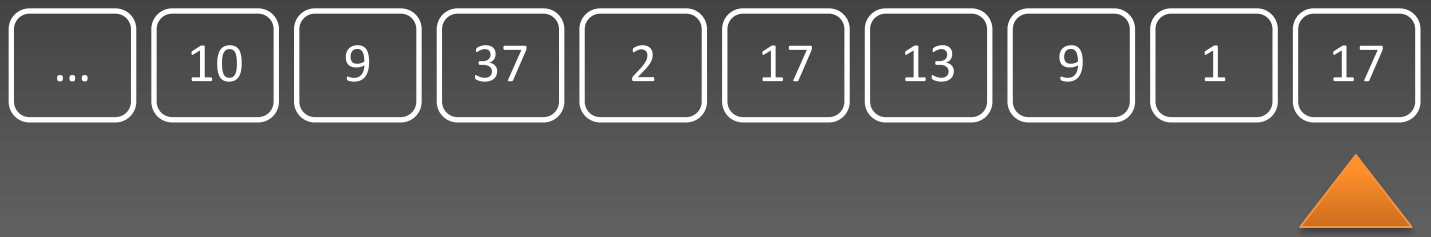
amazonTechO(n)

Space Saving algorithm

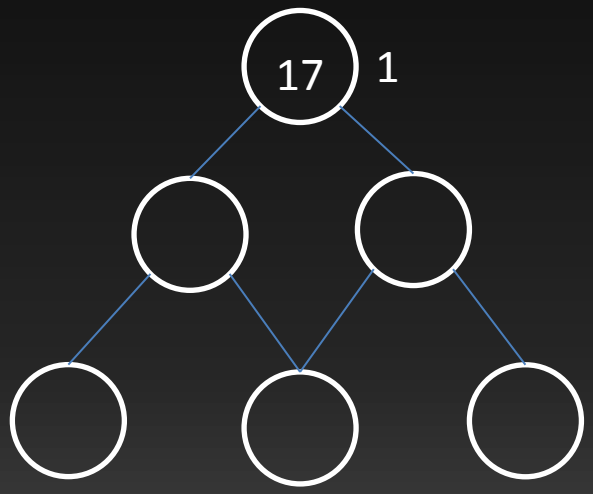


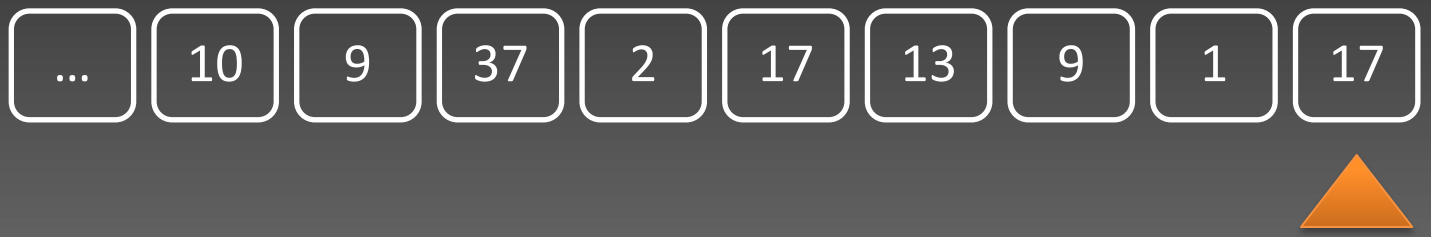
Space Saving algorithm



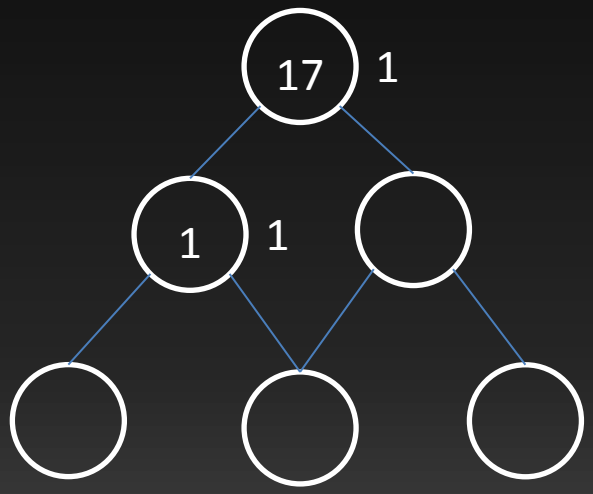


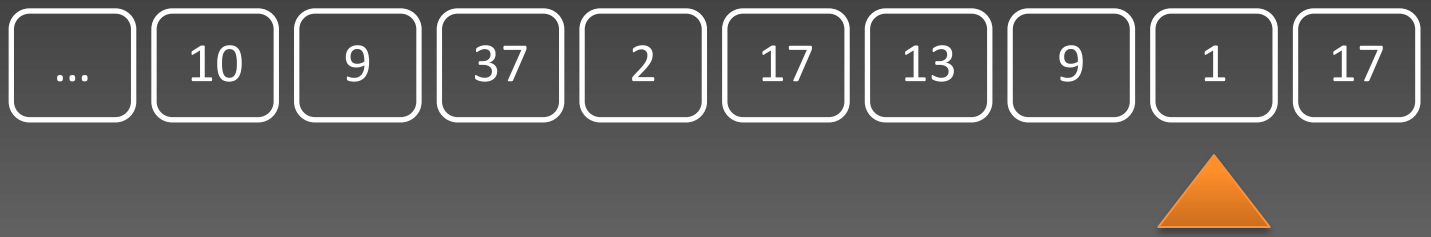
Space Saving algorithm



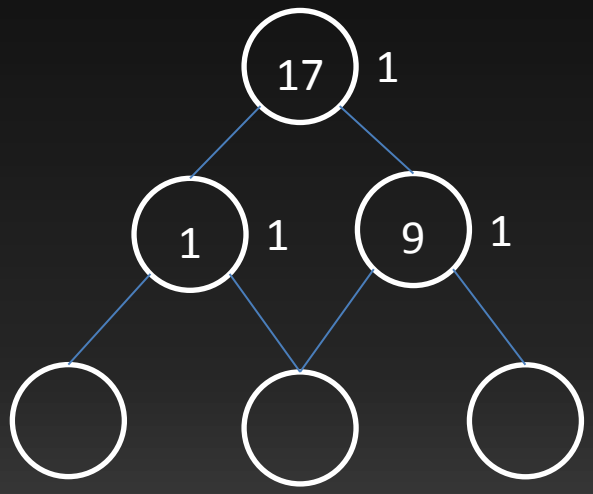


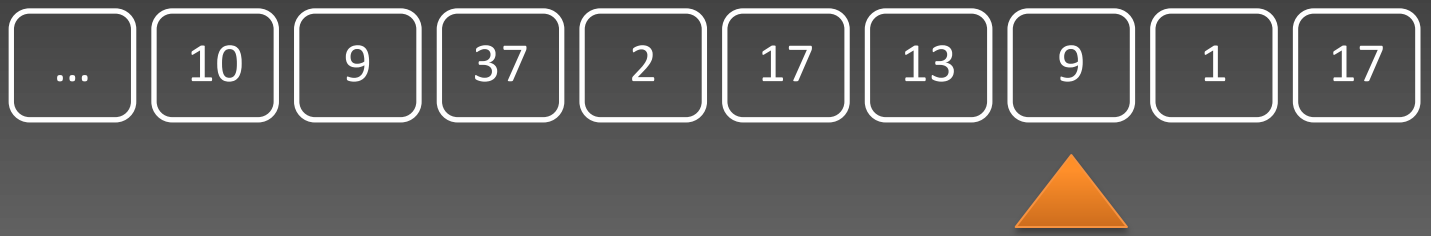
Space Saving algorithm



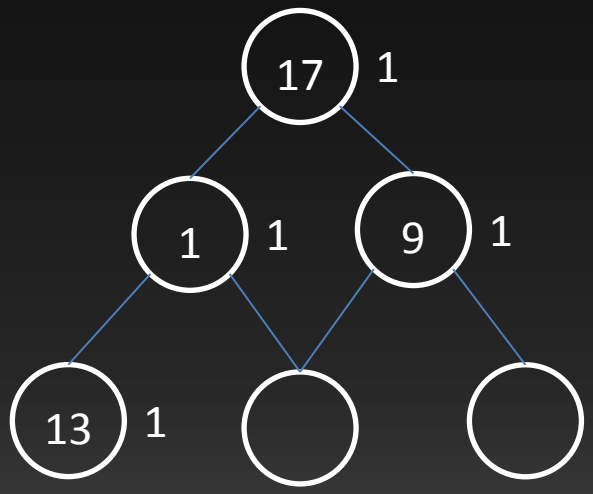


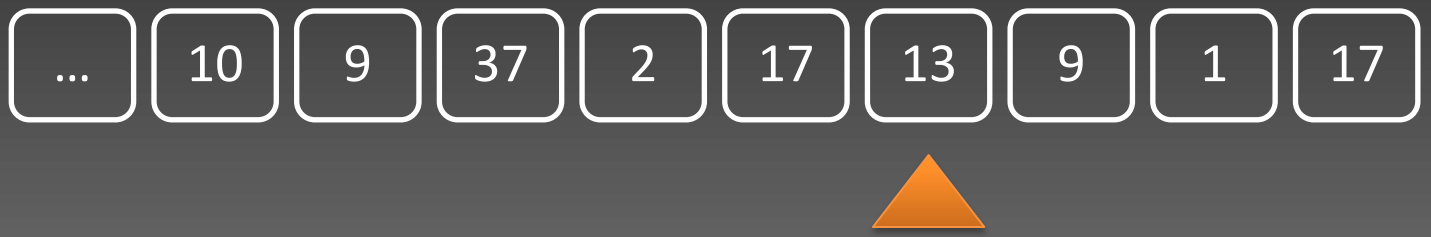
Space Saving algorithm



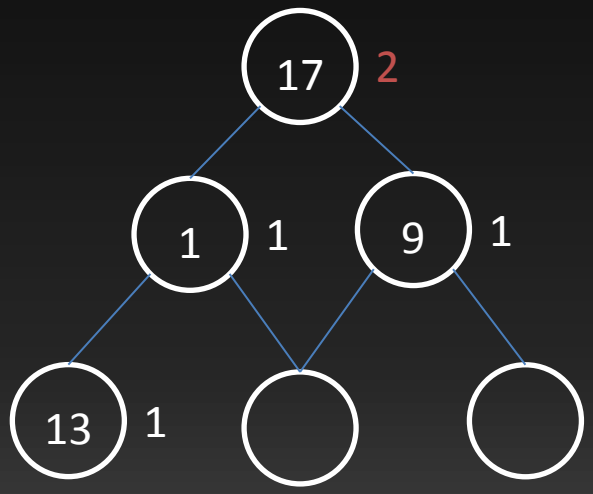


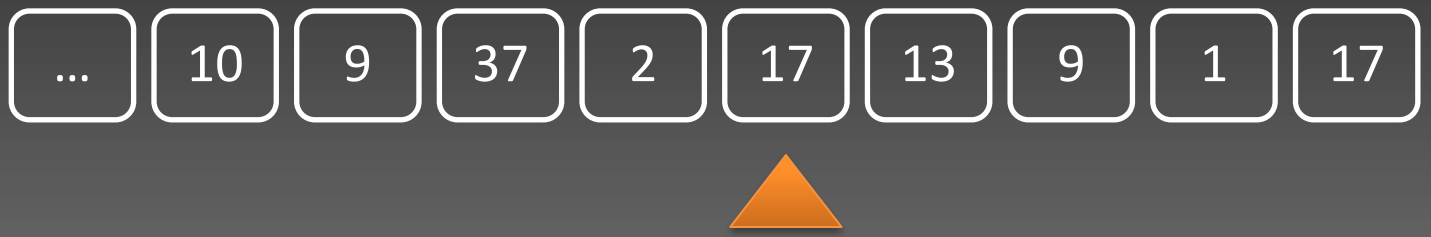
Space Saving algorithm



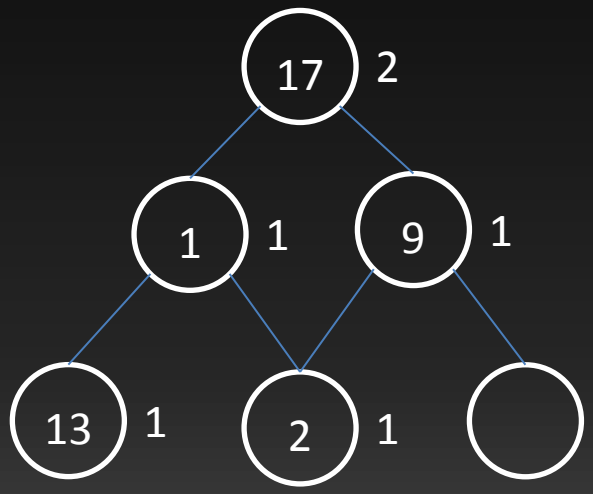


Space Saving algorithm

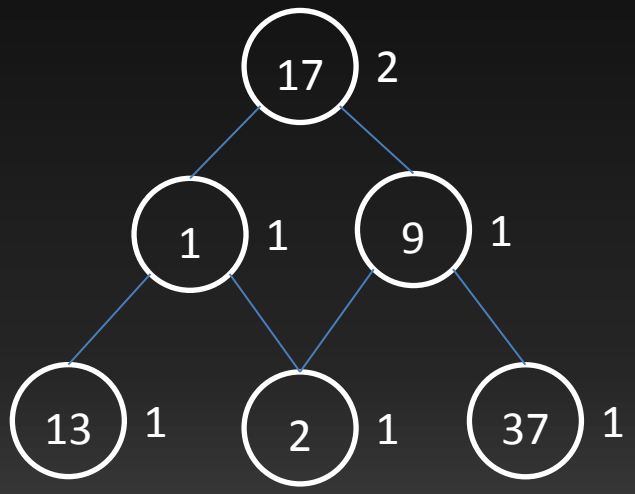


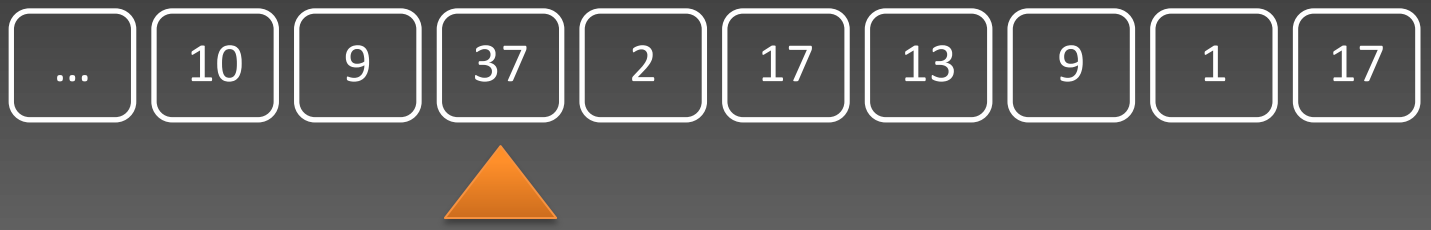


Space Saving algorithm

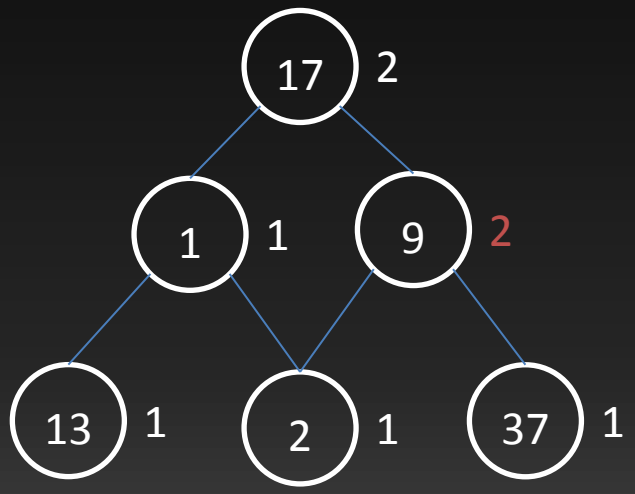


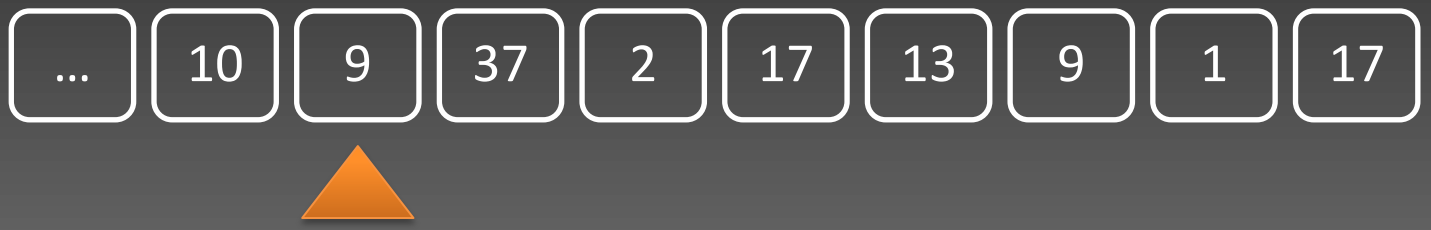
Space Saving algorithm



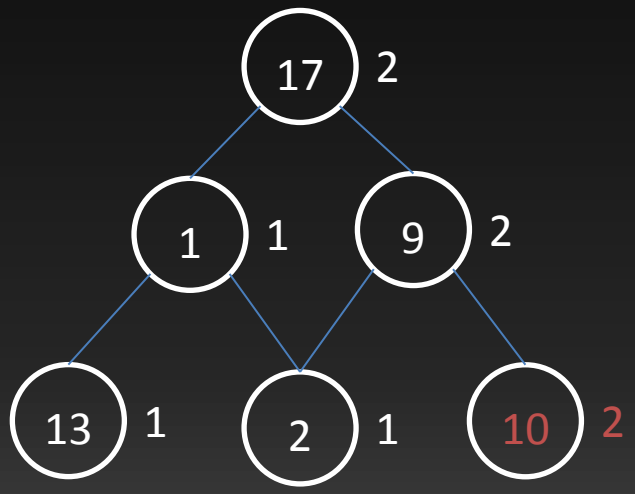


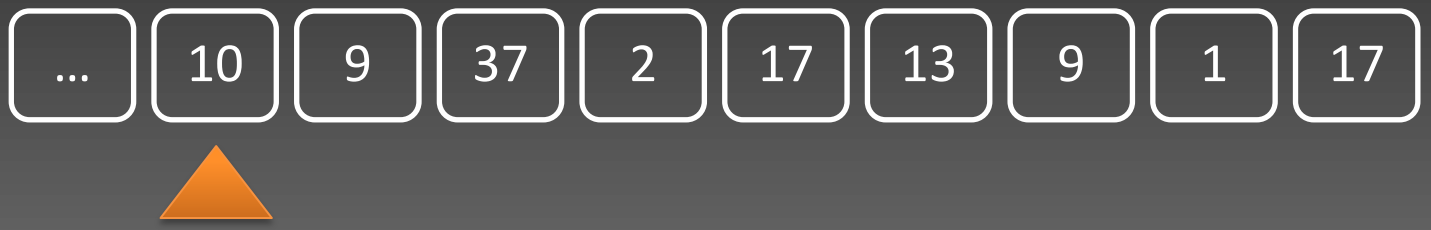
Space Saving algorithm





Space Saving algorithm





Space Saving algorithm analysis

- Smallest counter value, *min,* is at most *n/k*
- True count of an uncounted item is between *0* and *min*
- Any item x whose true count > *n/k* is stored

- So the algorithm finds all items with counts > *n/k* and the error in counts <= *n/k*

Experimental notes

- Multiple algorithms because this was one of the most studied problems in data streams, but Space Saving is the best performer

- Even for Space Saving we have many implementation details: hashtables to check if an item is in our data set?, heap to find the current minimum?, etc

Experimental notes

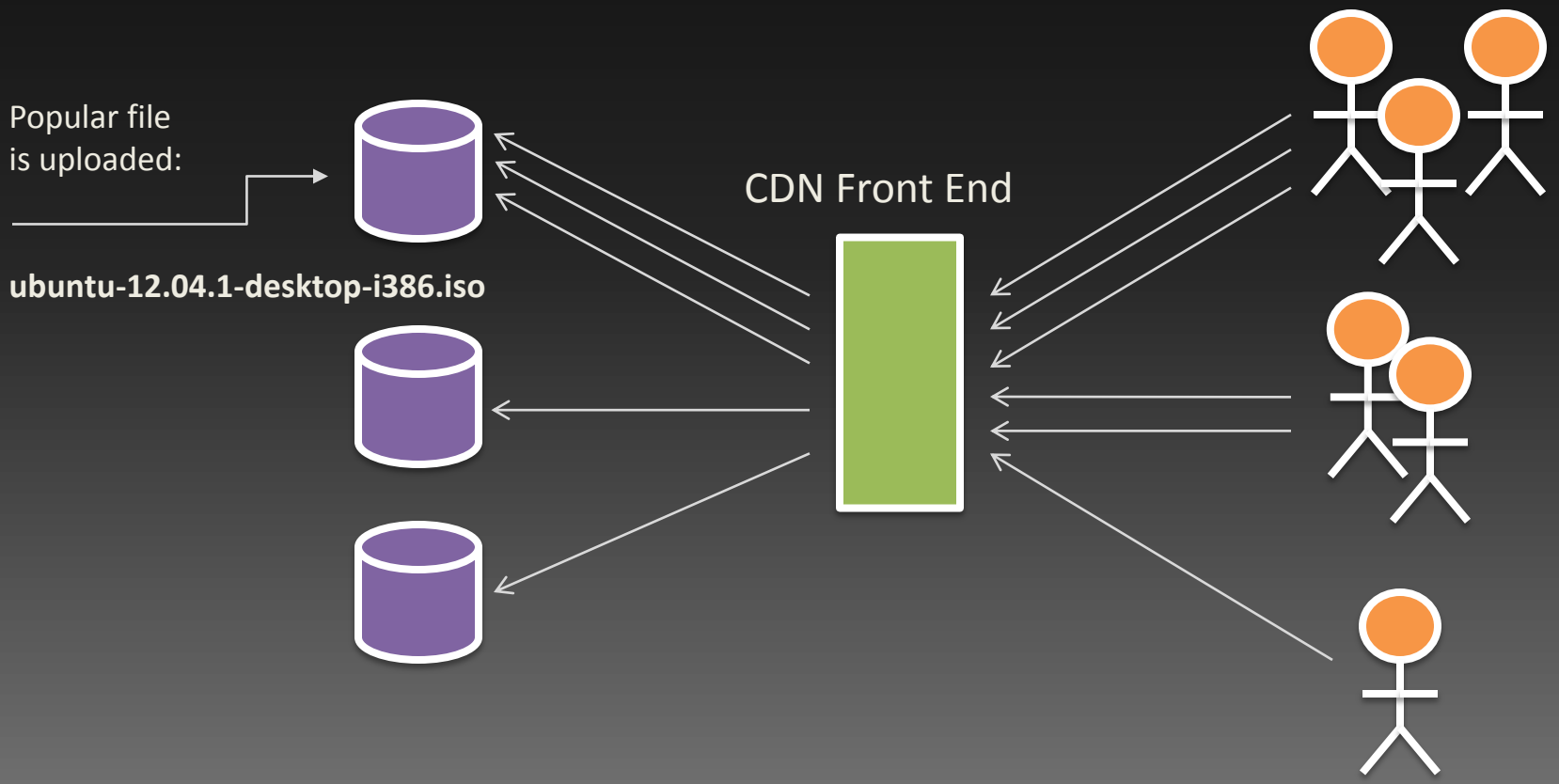
- Very fast in practice: ten millions of updates per second
- Very good practical accuracy and memory usage
  - easy to obtain 100% accuracy for real-life data with only a small additional space (usually 1.1 * k)

Extensions

- Weighted items (find items whose total weight is high)
- Support deletion of items (negative weight)
- Heavy Changers (which items have largest change)
- Distinct heavy hitters (which sources contact the most distinct addresses)
- Time Decay (weight of items decay with age)

amazonTechO(n)

Content Delivery Network Example



Takeaways

- Recent area of research (last 20 years), but increasingly important
- Time and space complexity matter a lot
- Different algorithms make different trade-offs (space vs time vs accuracy)

Learn more…

- "Probabilistic Counting Algorithms for Database Applications" - *Flajolet and Martin (1985)*
- "The space complexity of approximating the frequency moments" - *Alon, Noga; Matias, Yossi; Szegedy, Mario (1999)*
- Data Streams: Models and Algorithms - *Charu C. Aggarwal*
- Implementations of heavy hitters algorithms (and more!) at: *http://www.research.att.com/~marioh/frequent-items*

amazonTechO(n)

Q&A