Querying Data Streams
(By approximation)

Slides in part courtesy of

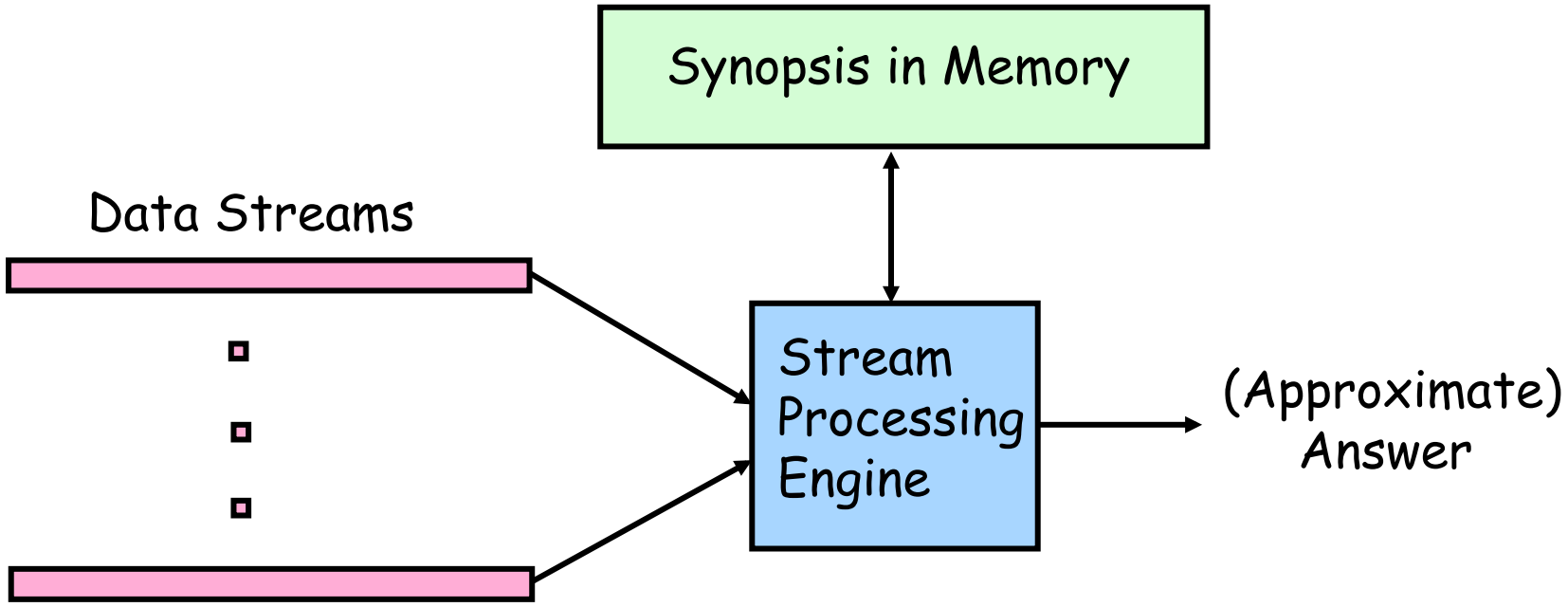
M.Garofalakis, J.Gehrke, and R. Rastogi

Processing Data Streams: Motivation

- A growing number of applications generate streams of data
  - Performance measurements in network monitoring and traffic management
  - Call detail records in telecommunications
  - Transactions in retail chains, ATM operations in banks
  - Log records generated by Web Servers
  - Sensor network data
- Application characteristics
  - Massive volumes of data
  - Records arrive at a rapid rate
- Goal: Mine patterns, process queries and compute statistics on data streams in real-time

Data Streams: Computation Model

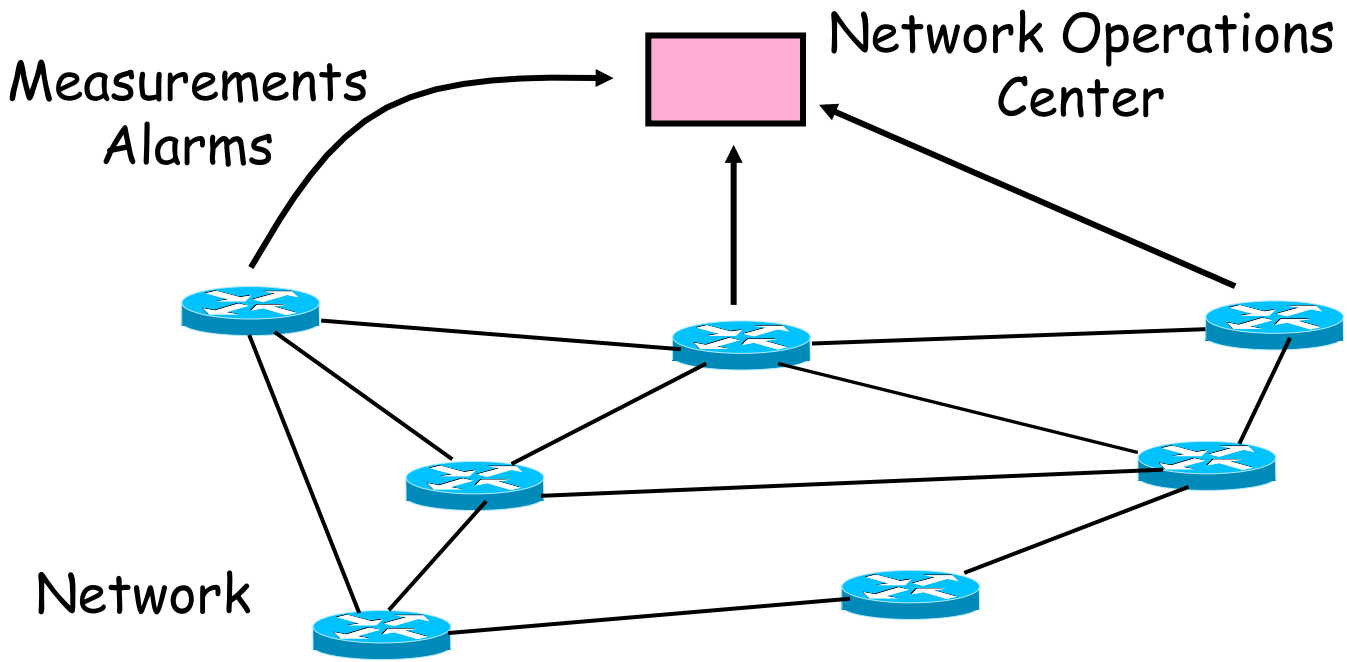
- A data stream is a (massive) sequence of elements: $e_1, \ldots, e_n$



- Stream processing requirements
  - Single pass: Each record is examined at most once
  - Bounded storage: Limited Memory (M) for storing synopsis
  - Real-time: Per record processing time (to maintain synopsis) must be low

Network Management Application

- Network Management involves monitoring and configuring network hardware and software to ensure smooth operation
  - Monitor link bandwidth usage, estimate traffic demands
  - Quickly detect faults, congestion and isolate root cause
  - Load balancing, improve utilization of network resources



IP Network Measurement Data

- IP session data (collected using NetFlow)

Source	Destination	Duration	Bytes	Protocol
10.1.0.2	16.2.3.7	12	20K	http
18.6.7.1	12.4.0.3	16	24K	http
13.9.4.3	11.6.8.2	15	20K	http
15.2.2.9	17.1.2.1	19	40K	http
12.4.3.8	14.8.7.4	26	58K	http
10.5.1.3	13.0.0.1	27	100K	ftp
11.1.0.6	10.3.4.5	32	300K	ftp
19.7.1.2	16.5.5.8	18	80K	ftp

Network Data Processing

- Traffic estimation
  - How many bytes were sent between a pair of IP addresses?
  - What fraction network IP addresses are active?
  - List the top 100 IP addresses in terms of traffic
- Traffic analysis
  - What is the average duration of an IP session?
  - What is the median of the number of bytes in each IP session?
- Fraud detection
  - List all sessions that transmitted more than 1000 bytes
  - Identify all sessions whose duration was more than twice the normal
- Security/Denial of Service
  - List all IP addresses that have witnessed a sudden spike in traffic
  - Identify IP addresses involved in more than 1000 sessions

Data Stream Processing Algorithms

- Generally, algorithms compute approximate answers
  - Difficult to compute answers accurately with limited memory
- Approximate answers - Deterministic bounds
  - Algorithms only compute an approximate answer, but bounds on error
- Approximate answers - Probabilistic bounds
  - Algorithms compute an approximate answer with high probability
    - With probability at least $1-\delta$, the computed answer is within a factor ε of the actual answer
- Single-pass algorithms for processing streams also applicable to massive databases!

Sampling: Basics

- Idea: A small random sample S of the data often well-represents all the data
  - For a fast approx answer, apply "modified" query to S
  - Example: select agg from R where R.e is odd

Data stream: 9 3 5 2 7 1 6 5 8 4 9 1 (n=12)

Sample S: 9 5 1 8

  - If agg is avg, return average of odd elements in S — answer: 5
  - If agg is count, return average over all elements e in S of
    - n if e is odd
    - 0 if e is even

answer: 12*3/4 =9

Unbiased: For expressions involving count, sum, avg: the estimator is unbiased, i.e., the expected value of the answer is the actual answer

Probabilistic Guarantees

- Example: Actual answer is 5 ± 1 with prob ≥ 0.9
- Hoeffding's Inequality: Let X1, …, Xm be independent random variables with 0<=Xi <= r. Let $\overline{X} = \frac{1}{m}\sum_i X_i$ and μ be the expectation of $\overline{X}$. Then, for any $\varepsilon > 0$,

$$\Pr(|\overline{X} - \mu| \geq \varepsilon) \leq 2\exp^{\frac{-2m\varepsilon^2}{r^2}}$$

- Application to avg queries:
  - m is size of subset of sample S satisfying predicate (3 in example)
  - r is range of element values in sample (8 in example)
- Application to count queries:
  - m is size of sample S (4 in example)
  - r is number of elements n in stream (12 in example)

Approximating the Frequency Moments

- Let's sample from S=[a1, a2, a3, ... , am] where each aj ∈[n]
- *AMS Sampling:* Return (i,r) with i from [n] chosen w/prob fi/m and r ∈[fi]
  - g is a real-valued fn with g(0)=0, eg. g(x) = x^2 (for 2nd moment)
  - Sample aj for j ∈[m], let i= aj and compute r=|{j'≥j : aj'=aj}|
  - Useful for estimating ∑i g(fi) because E[m*(g(r)-g(r-1))] = ∑i g(fi)
  - ∑i=1..n g(fi) = ∑i=1..n ∑j=1..fi (g(j) – g(j-1)) = ∑i=1..n ∑j=1..fi E[g(r) – g(r-1)]

Computing Stream Sample

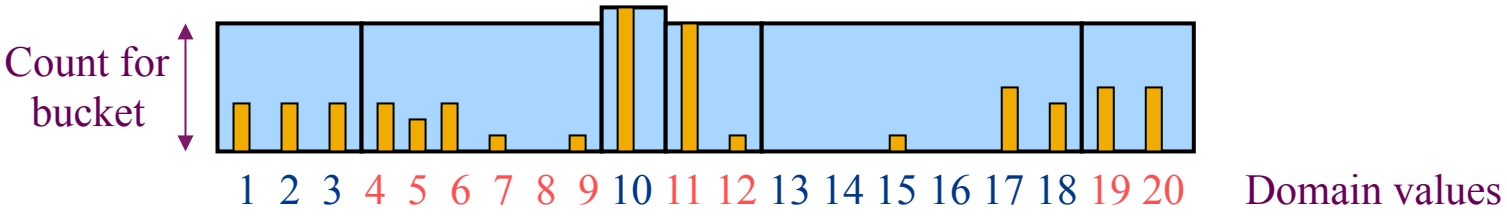
- Reservoir Sampling [Vit85]: Maintains a sample S of a fixed-size M
  - Add each new element to S with probability M/n, where n is the current number of stream elements
  - If add an element, evict a random element from S
  - Instead of flipping a coin for each element, determine the number of elements to skip before the next to be added to S
- Concise sampling [GM98]: Duplicates in sample S stored as <value, count> pairs (thus, potentially boosting actual sample size)
  - Add each new element to S with probability 1/T (simply increment count if element already in S)
  - If sample size exceeds M
    - Select new threshold T' > T
    - Evict each element (decrement count) from S with probability T/T'
  - Add subsequent elements to S with probability 1/T'

Histograms

- Histograms approximate the frequency distribution of element values in a stream
- A histogram (typically) consists of
  - A partitioning of element domain values into buckets
  - A count C_B per bucket B (of the number of elements in B)

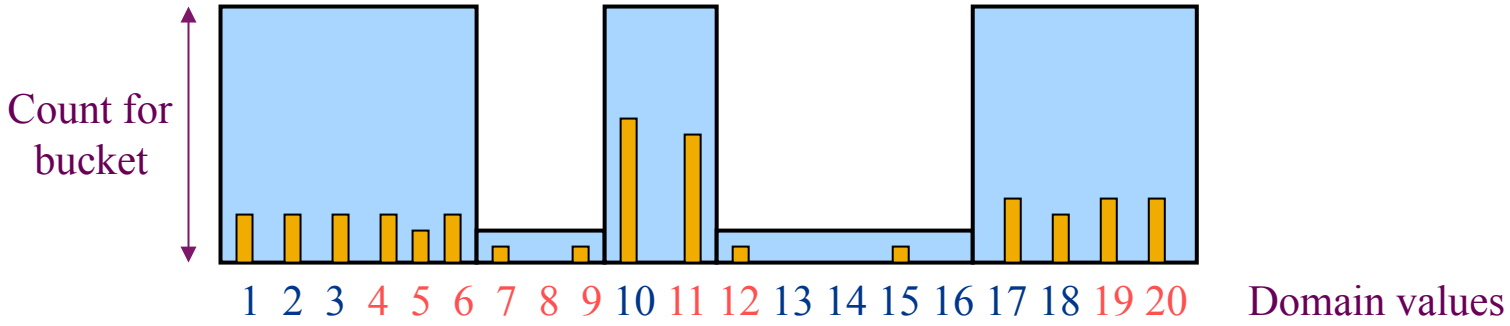
Types of Histograms

- Equi-Depth Histograms
  - Idea: Select buckets such that counts per bucket are equal



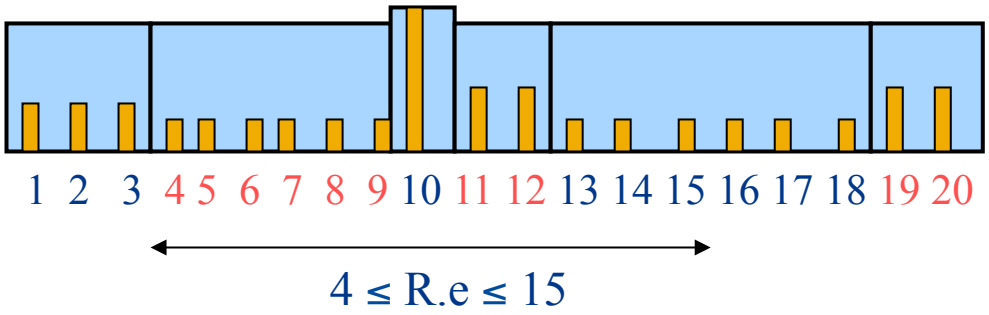
- V-Optimal Histograms
  - Idea: Select buckets to minimize frequency variance within buckets

$$\text{minimize} \sum_B \sum_{v \in B} (f_v - \frac{C_B}{V_B})^2$$



Answering Queries using Histograms

- (Implicitly) map the histogram back to an approximate relation, & apply the query to the approximate relation
- Example: select count(*) from R where 4 <= R.e <= 15



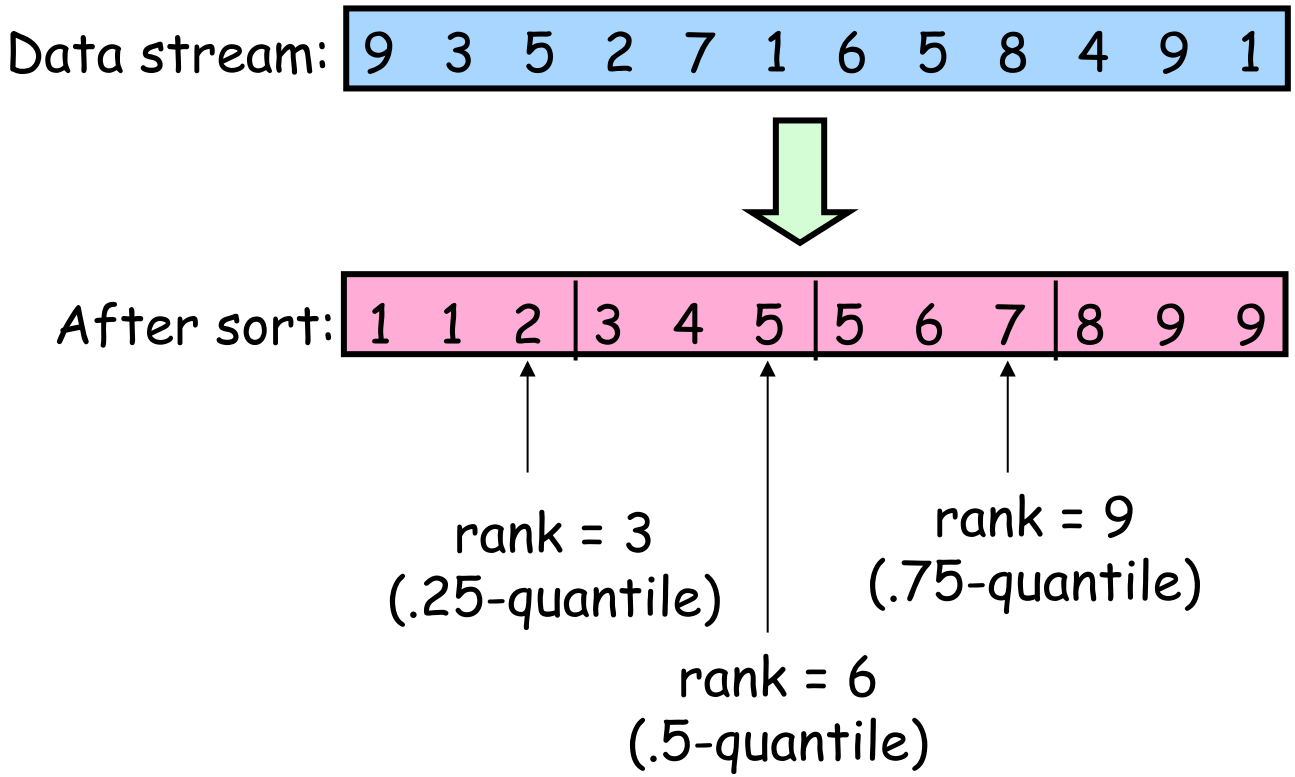
Count spread evenly among bucket values

answer: 3.5 * C_B

- For equi-depth histograms, maximum error: $\pm 2 * C_B$

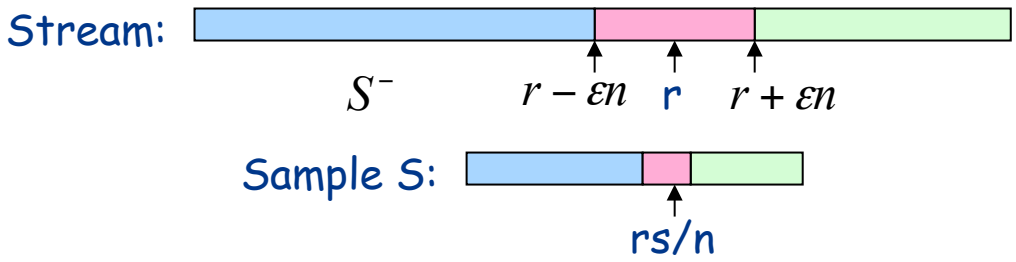
Equi-Depth Histogram Construction

- For histogram with b buckets, compute elements with rank n/b, 2n/b, ..., (b-1)n/b
- Example: (n=12, b=4)



Computing Approximate Quantiles Using Samples

- Problem: Compute element with rank r in stream
- Simple sampling-based algorithm
  - Sort sample S of stream and return element in position rs/n in sample (s is sample size)
  - With sample of size $O(\frac{1}{\varepsilon^2}\log(\frac{1}{\delta}))$, possible to show that rank of returned element is in $[r-\varepsilon n, r+\varepsilon n]$ with probability at least $1-\delta$
    - Hoeffding's Inequality: probability that S contains greater than rs/n elements from S^- is no more than $\exp^{-2s\varepsilon^2}$



One-Dimensional Haar Wavelets

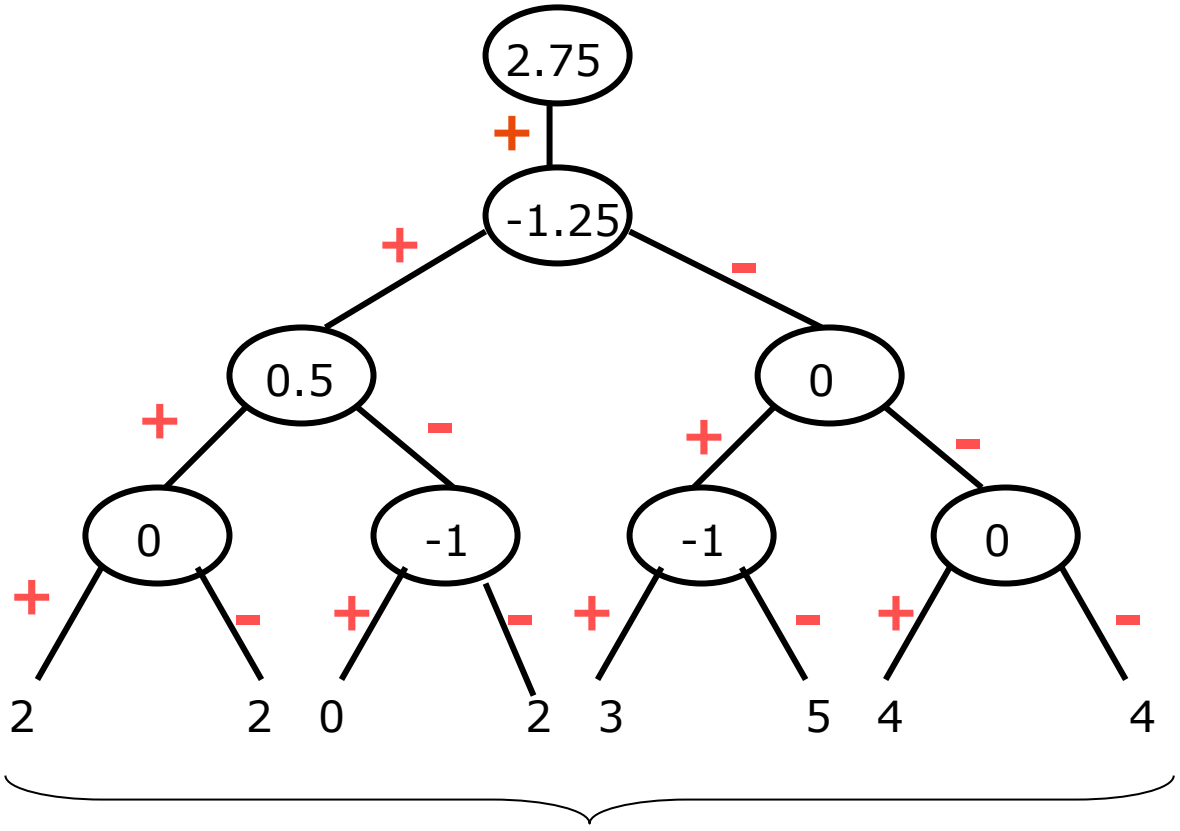
- Wavelets: Mathematical tool for hierarchical decomposition of functions/signals
- Haar wavelets: Simplest wavelet basis, easy to understand and implement
  - *Recursive pairwise averaging and differencing* at different resolutions

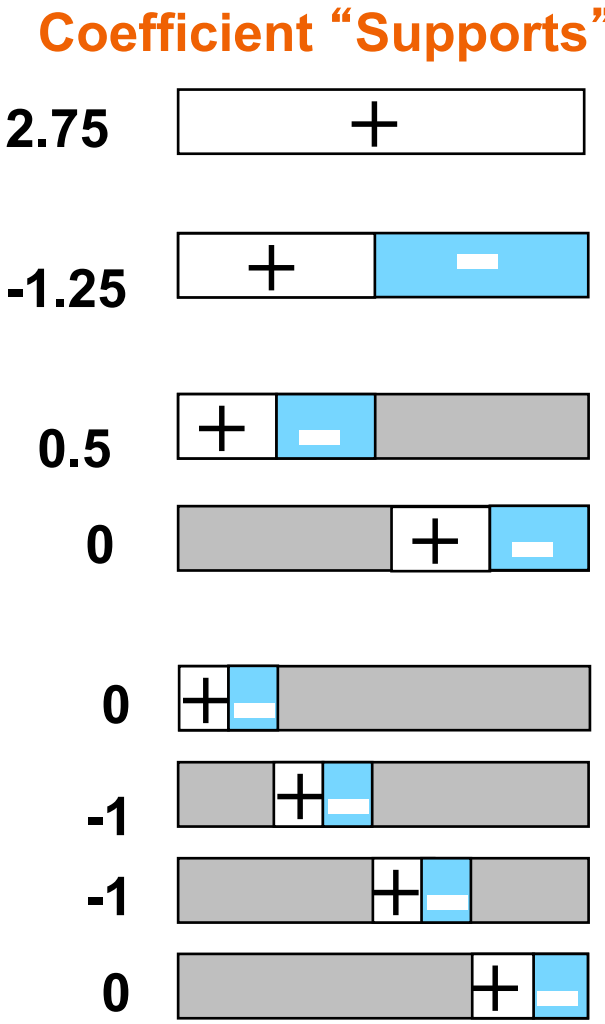
Resolution	Averages	Detail Coefficients
3	[2, 2, 0, 2, 3, 5, 4, 4]	----
2	[2, 1, 4, 4]	[0, -1, -1, 0]
1	[1.5, 4]	[0.5, 0]
0	[2.75]	[-1.25]

Haar wavelet decomposition: [2.75, -1.25, 0.5, 0, 0, -1, -1, 0]

Haar Wavelet Coefficients

- Hierarchical decomposition structure (a.k.a. "error tree")



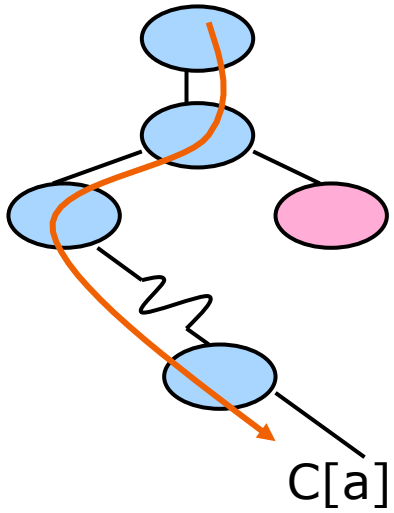


Wavelet-based Histograms [MVW98]

- Problem: Range-query selectivity estimation
- Key idea: Use a compact subset of Haar/linear wavelet coefficients for approximating frequency distribution
- Steps
  - Compute cumulative frequency distribution *C*
  - Compute Haar (or linear) wavelet transform of *C*
  - Coefficient *thresholding* : only m<<n coefficients can be kept
    - Take largest coefficients in *absolute normalized value*
      - Haar basis: divide coefficients at resolution j by $\sqrt{2^j}$
      - *Optimal* in terms of the overall Mean Squared (L2) Error
    - Greedy heuristic methods
      - Retain coefficients leading to large error reduction
      - Throw away coefficients that give small increase in error

Using Wavelet-based Histograms

- Selectivity estimation: count(a<= R.e<= b) = C' [b] - C' [a-1]
  - C' is the (approximate) "reconstructed" cumulative distribution
  - Time: O(min{m, logN}), where m = size of wavelet synopsis (number of coefficients), N= size of domain

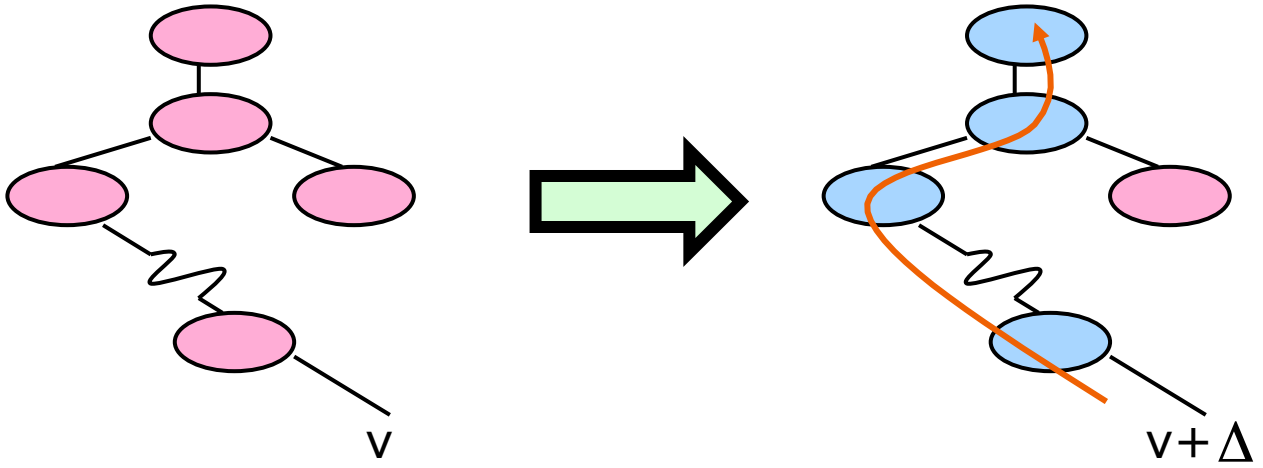


- At most logN+1 coefficients are needed to reconstruct any C value

- Empirical results over synthetic data
  - Improvements over random sampling and histograms

Dynamic Maintenance of Wavelet-based Histograms

- Build Haar-wavelet synopses on the original frequency distribution
  - Similar accuracy with CDF, makes maintenance simpler
- Key issues with dynamic wavelet maintenance
  - Change in single distribution value can affect the values of many coefficients (path to the root of the decomposition tree)

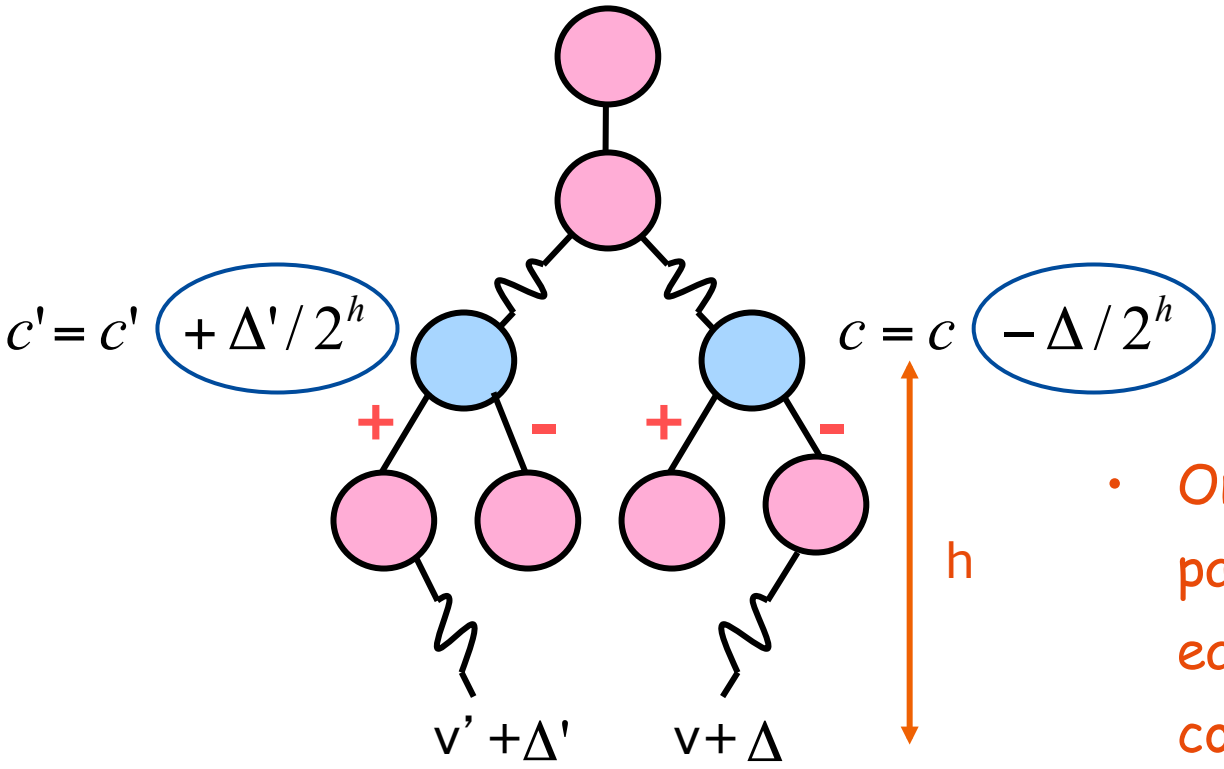


Change propagates up to the root coefficient

  - As distribution changes, "most significant" (e.g., largest) coefficients can also change!
    - Important coefficients can become unimportant, and vice-versa

Effect of Distribution Updates

- Key observation: for each coefficient c in the Haar decomposition tree
  - c = (AVG(leftChildSubtree(c)) - AVG(rightChildSubtree(c))) / 2



- Only coefficients on path(v) are affected and each can be updated in constant time

Maintenance Algorithm [MWV00] - Simplified Version

- Histogram H: Top m wavelet coefficients
- For each new stream element (with value v)
  - For each coefficient c on path(v) and with "height" h
    - If c is in H, update c (by adding or substracting $1/2^h$)
  - For each coefficient c on path(v) and not in H
    - Insert c into H with probability proportional to $1/(\min(H)*2^h)$ (*Probabilistic Counting*)
      - Initial value of c: min(H), the minimum coefficient in H
    - If H contains more than m coefficients
      - Delete minimum coefficient in H