

# CptS 415 Big Data

# Approximate Query Processing

# Srini Badri

## Acknowledgement: Tinghui Wang



# Make Case for Computationally Efficient Queries

---

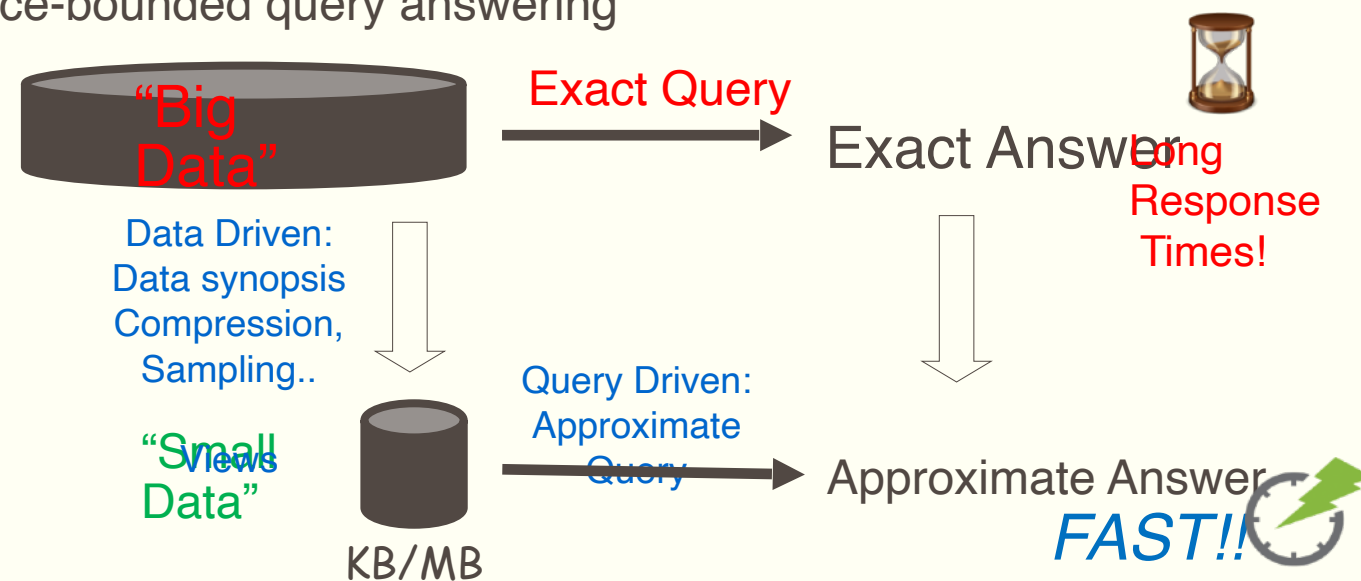
- Approximate Query Answering

- Query-driven approximation

- Rewrite queries to computationally efficient query classes

- Data-drive approximation

- Compact data synopses, materialized views, compression, summaries, sketches, spanners
    - Resource-bounded query answering



# Data-Driven Approximation

---

- Data synopses:
  - Histogram
  - Sampling
  - Wavelet
- Resource Bounded Querying

# Histograms

---

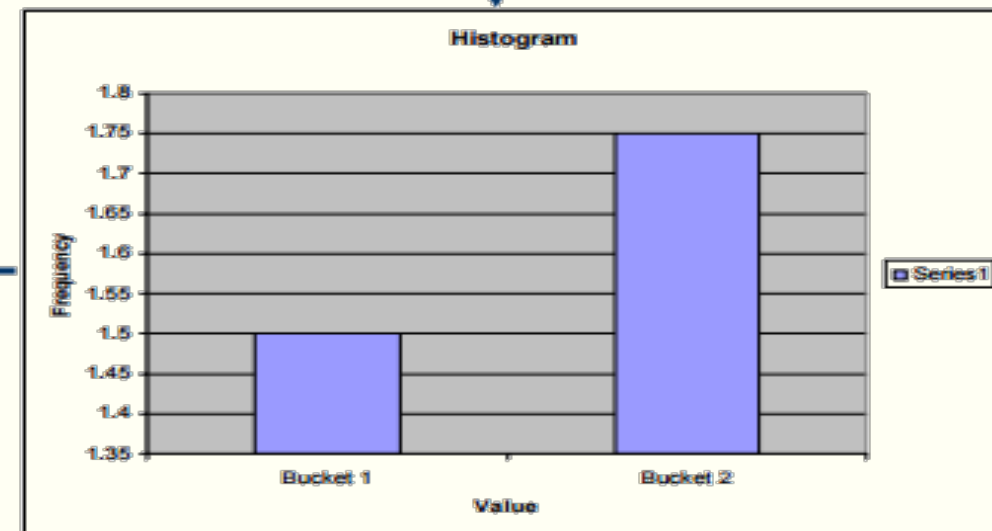
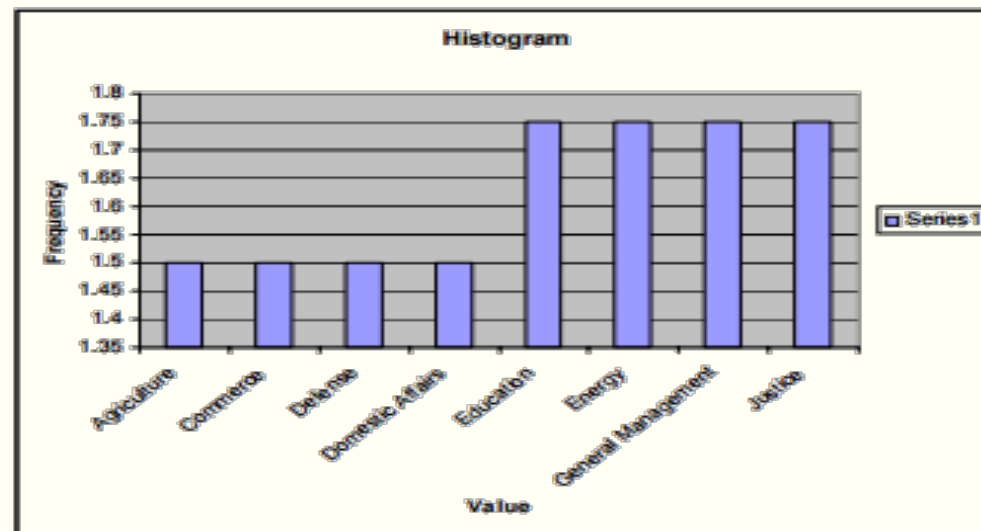
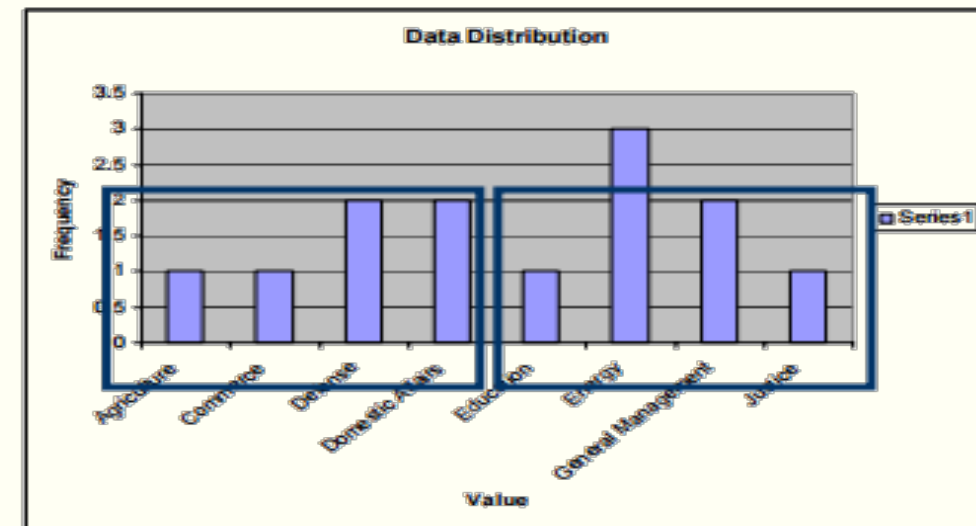
- Partition attribute value(s) domain into a set of buckets
- Estimation of data distribution (mostly for aggregation)
- Approximate the frequencies in each bucket in common fashion
- Equi-width
- Equi-depth
- V-optimal

Name	Salary	Department
Zeus	100K	General Management
Poseidon	80K	Defense
Pluto	80K	Justice
Aris	50K	Defense
Ernis	60K	Commerce
Apollo	60K	Energy
Hefestus	50K	Energy
Hera	90K	General Management
Athena	70K	Education
Aphrodite	60K	Domestic Affairs
Demeter	60K	Agriculture
Hestia	50K	Domestic Affairs
Artemis	60K	Energy

Department	Frequency
General Management	2
Defense	2
Education	1
Domestic Affairs	2
Agriculture	1
Commerce	1
Justice	1
Energy	3

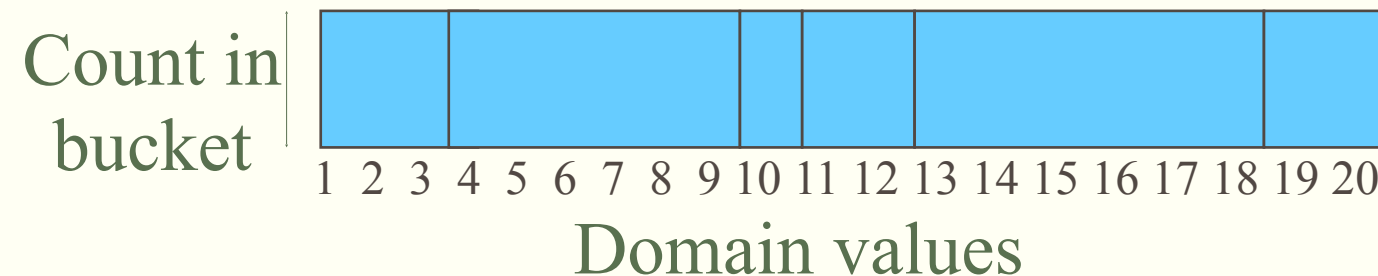
# Equi-Width

Department	Histogram H1	
	Frequency in Bucket	Approximate Frequency
Agriculture	1	1.5
Commerce	1	1.5
Defense	2	1.5
Domestic Affairs	2	1.5
Education	①	1.75
Energy	③	1.75
General Management	②	1.75
Justice	①	1.75



# Equi-Depth

---



- Goal: Equal number of rows per bucket (B buckets in all)
- Can construct by first sorting then take B-1 equally-spaced splits

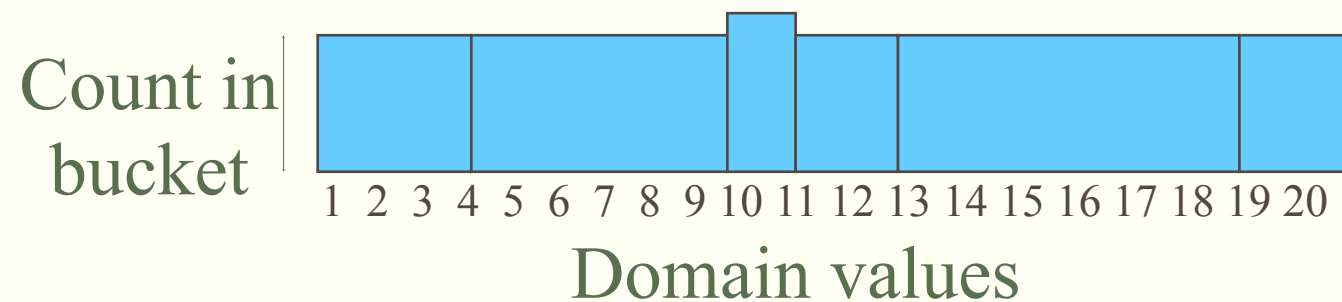
1 2 2 3 4 7 8 9 10 10 10 10 11 11 12 12 14 16 16 18 19 20 20 20

- Faster construction: Sample & take equally-spaced splits in sample
  - Nearly equal buckets
  - Faster algorithm: one-pass quantile

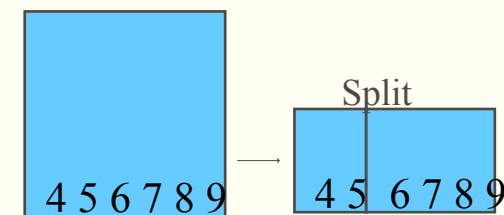
“Space-Efficient Online Computation of Quantile Summaries”, Michael Greenwald, et al., SIGMOD 2001

# Equi-Depth

---

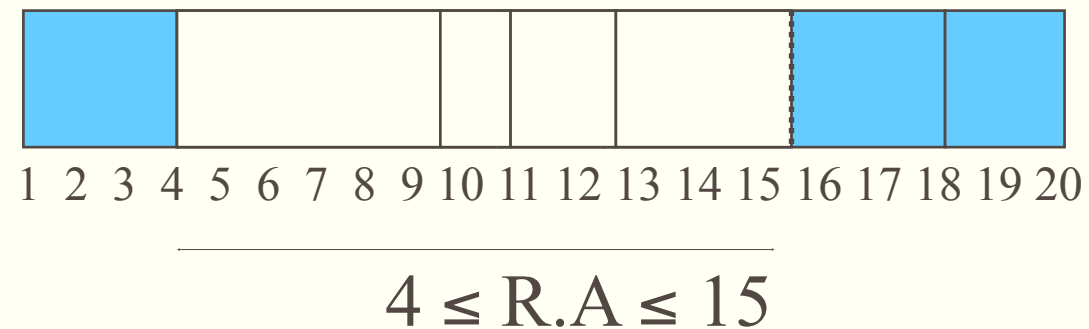


- Can maintain using one-pass algorithms (insertions only), or
- Maintain a backing sample on disk in support of histogram maintenance
  - Keep histogram bucket counts up to date by incrementing on row insertion, decrementing on row deletion
  - Merge adjacent buckets with small counts
  - Split any bucket with a large count, using the sample to select a split value, i.e., take median of the sample points in bucket range.



# Equi-Depth: Query Processing

---



- Answering queries:

- Select  $count(*)$  from  $R$  where  $4 \leq R.A \leq 15$
- Approximate answer:  $F * |R| / B$ 
  - $F$ : Number of buckets, including fractions, that overlap the range
- Answer:  $3.6 * 24 / 6 = 14$ , actual count: 13
- Error Bound:  $24/6 = 4$



# Sampling

---

- Intuition: A small random sample  $S$  of the data often well-represents all the data
  - For a fast approximation, apply the query to  $S$  and “scale” the result.
  - E.g.  $R.a$  is  $\{0, 1\}$ ,  $S$  is a 20% sample
  - `Select count(*) from R where R.a = 0`
  - `Select 5 * count(*) from S where S.a = 0`
- Unbiased sampling:
  - For expression involving count, sum, average: the estimator is unbiased, i.e. the expected value of the answer is the actual answer, even for (most) queries with predicates

# Sampling: Basics

---

$$AVERAGE_{estimated} = AVERAGE_{sample\_set}$$

$$COUNT_{estimated} = COUNT_{sample\_set} \times \frac{N}{n}$$

$$SUM_{estimated} = SUM_{sample\_set} \times \frac{N}{n} = AVERAGE_{sample\_set} \times N$$

$$MAX_{estimated} = MAX_{sample\_set} \qquad MIN_{estimated} = MIN_{sample\_set}$$

Normal query:

```
SELECT count(*), sum(sales),  
average(sales) , ...  
FROM SalesFact, Time, ...  
WHERE joins and restrictions  
GROUP BY group conditions
```

Rewritten query:

```
SELECT sum(1/SP), sum(sales/SP),  
sum(sales/SP)/sum(1/SP), ...  
FROM SW_SalesFact, SW_Time, ...  
WHERE joins and restrictions  
GROUP BY group conditions
```

- Leverage extensive literature on confidence intervals for sampling
  - Actual answer is within the interval [a,b] with a given probability
    - E.g., 54,000 ± 600 with prob ≥ 90%

# Sampling: Confidence Intervals

---

Method	90% Confidence Interval	Guarantees?
Central Limit Theorem	$1.65 * \sigma(S) / \sqrt{ S }$	As $ S  \rightarrow \infty$
Hoeffding	$1.22 * (\text{Max-Min}) / \sqrt{ S }$	Always
Chebychev (known $\sigma(R)$ )	$3.16 * \sigma(R) / \sqrt{ S }$	Always
Chebychev (estimated $\sigma(R)$ )	$3.16 * \sigma(S) / \sqrt{ S }$	As $\sigma(S) \rightarrow \sigma(R)$

- If predicates, S above is subset of sample that satisfies the predicate
- Quality of the estimate depends only on the variance in R and |S| after the predicate
  - E.g. 10K sample may suffice for 10Billion row relation.
  - Advantage for large samples: can handle more selective predicates

# One-Pass Uniform Sampling

---

- Best choice for incremental maintenance
  - Low overheads, no random data access
- Reservoir sampling [Vit85]: Maintains a sample  $S$  of a fixed-size  $k$ 
  - Add each new item to  $S$  with probability  $k/N$ , where  $N$  is the current number of data item
  - If an item is added, evict a random item from  $S$
  - Instead of flipping a coin for each item, determine the number of items to skip before the next to be added to  $S$

```
ReservoirSample(S[1..n], R[1..k])
  // fill the reservoir array
  for i = 1 to k
    R[i] := S[i]

  // replace elements with gradually decreasing probability
  for i = k+1 to n
    j := random(1, i)  // important: inclusive range
    if j <= k
      R[j] := S[i]
```

# Wavelets

---

- In signal processing community, wavelets are used to break the complicated signal into components in both time and scale
- Similarly, in approximate query processing, wavelets are used to break the dataset into simple components
- Haar wavelet: Simple wavelet, easy to understand and compute

# Haar Wavelet Calculation

---

- Wavelets: Mathematical tool for hierarchical decomposition of functions/ signals
- Haar wavelets: simplest wavelet basis, easy to understand and implement

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]

$[2.75, -1.25, 0.5, 0, 0, -1, -1, 0]$


Haar wavelet decomposition

# Haar Wavelet Coefficients

---

- Using wavelet coefficients one can pull the raw data
- Keep only the large wavelet coefficients and pretend other coefficients to be 0

$[2.75, -1.25, 0.5, 0, 0, -1, -1, 0]$



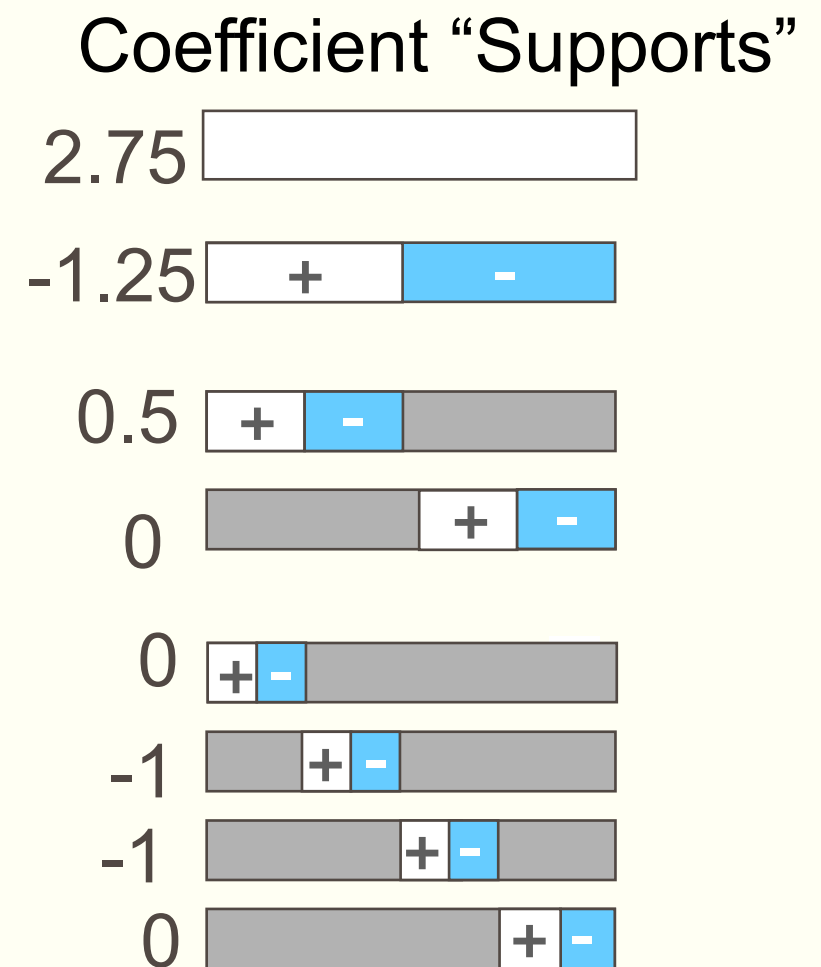
$[2.75, -1.25, 0.5, 0, 0, 0, 0, 0]$ -synopsis of the data

- The elimination of small coefficients introduces only small error when reconstructing the original data

---

- 
- ```

graph TD
    A((2.75)) -->|+| B((-1.25))
    A -->|-| C((0))
    B -->|+| D((0.5))
    B -->|-| E((0))
    D -->|+| F((0))
    D -->|-| G((-1))
    E -->|+| H((-1))
    E -->|-| I((0))
    F -->|+| J(2)
    F -->|-| K(2)
    G -->|+| L(0)
    G -->|-| M(2)
    H -->|+| N(3)
    H -->|-| O(5)
    I -->|+| P(4)
    I -->|-| Q(4)
  
```
- Original data





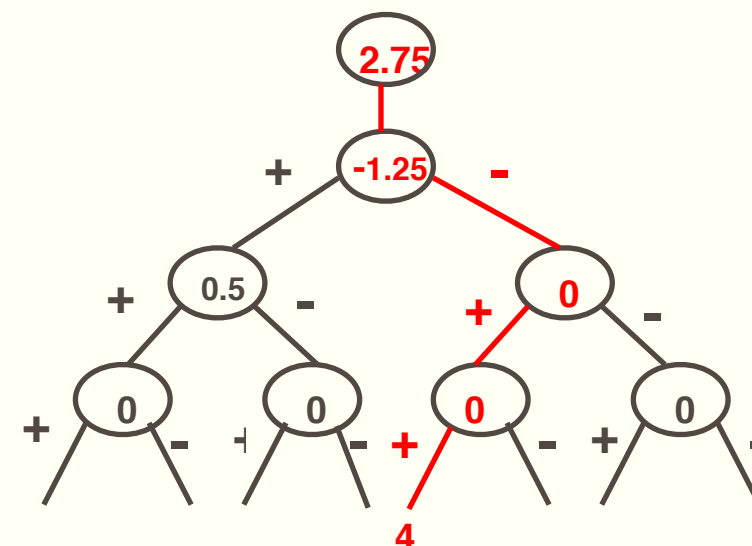
# Example: Selection/Projection Query

---

- Query:
  - SELECT salary FROM employee WHERE empid=5
- Result:
  - By using the synopsis [2.75,- 1.25, 0.5, 0, 0, 0, 0, 0] and constructing the tree on fly, salary=4 will be returned, whereas the correct result is salary=3.
- This error is due to truncation of wavelength

Employee

| Empid | Salary |
|-------|--------|
| 1     | 2      |
| 2     | 2      |
| 3     | 0      |
| 4     | 2      |
| 5     | 3      |
| 6     | 5      |
| 7     | 4      |
| 8     | 4      |



# Example: Range Query

- Query:
  - SELECT sum(salary) FROM employee WHERE 3 <= empid <= 7
- Find the Haar wavelet transformation and construct the tree

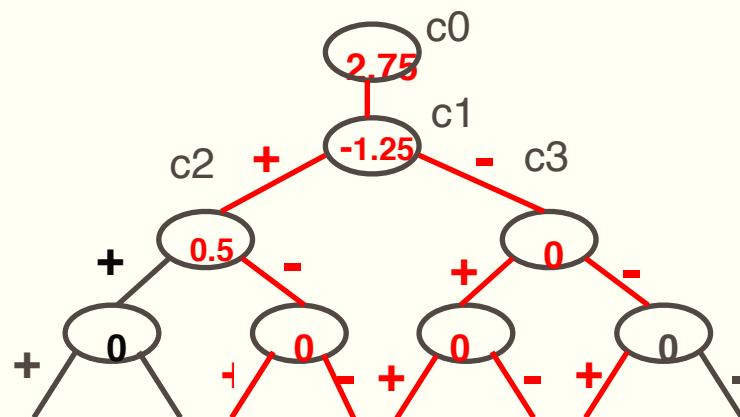
$$A(l : h) = \sum_{c_j \in \text{path}(A[l]) \cup \text{path}(A[h])} x_j,$$

$$x_j = \begin{cases} (h - l + 1) \cdot c_j, & \text{if } j = 0 \\ (|\text{leftleaves}(c_j, l : h)| - |\text{rightleaves}(c_j, l : h)|) \cdot c_j, & \text{otherwise.} \end{cases}$$

$$A(2 : 6) = 5c_0 + (2 - 3)c_1 - 2c_2 =$$

- Result:

Synopses for Massive Data: Samples, Histograms, Wavelets, Sketches, Graham, et.al <http://db.ucsd.edu/static/Synopses.pdf>



| Empid | Salary |
|-------|--------|
| 1     | 2      |
| 2     | 2      |
| 3     | 0      |
| 4     | 2      |
| 5     | 3      |
| 6     | 5      |
| 7     | 4      |
| 8     | 4      |

# Comparison with Sampling

---

- For Haar wavelet transformation, all the data must be numerical.
  - In the above example, every “empid” must be numeric and must be sorted
- Sampling gives the probabilistic error measure whereas Haar wavelet does not provide any
- Haar wavelet is more robust than sampling. The final average gives the average of all data values. Hence, all the tuples are involved

# Summary: Approximate query answering

---

- Challenges: to get real-time answers
  - Big data and costly queries
  - Limited resources
- Query-driven approximation
  - Cheaper queries
  - Retain sensible answers
- Data-driven approximation
  - Different type of data synopses construction methods
    - histogram, sample, wavelet, sketch, spanner, sparsifier
  - Dynamic data reduction
  - Query-guided search

# Papers to Review

---

- G. Gou and R. Chirkova. Efficient algorithms for exact ranked twig-pattern matching over graphs. In SIGMOD, 2008. <http://dl.acm.org/citation.cfm?id=1376676>
- H. Shang, Y. Zhang, X. Lin, and J. X. Yu. Taming verification hardness: an efficient algorithm for testing subgraph isomorphism. PVLDB, 2008. <http://www.vldb.org/pvldb/1/1453899.pdf>
- R. T. Stern, R. Puzis, and A. Felner. Potential search: A bounded-cost search algorithm. In ICAPS, 2011. (search Google Scholar)
- S. Zilberstein, F. Charpillet, P. Chassaing, et al. Real-time problem solving with contract algorithms. In IJCAI, 1999. (search Google Scholar)
- W. Fan, X. Wang, and Y. Wu. Diversified Top-k Graph Pattern Matching, VLDB 2014. (query-driven approximation)
- W. Fan, X. Wang, and Y. Wu. Querying big graphs with bounded resources, SIGMOD 2014. (data-driven approximation)