Lecture 24: MapReduce Algorithms Wrap-up

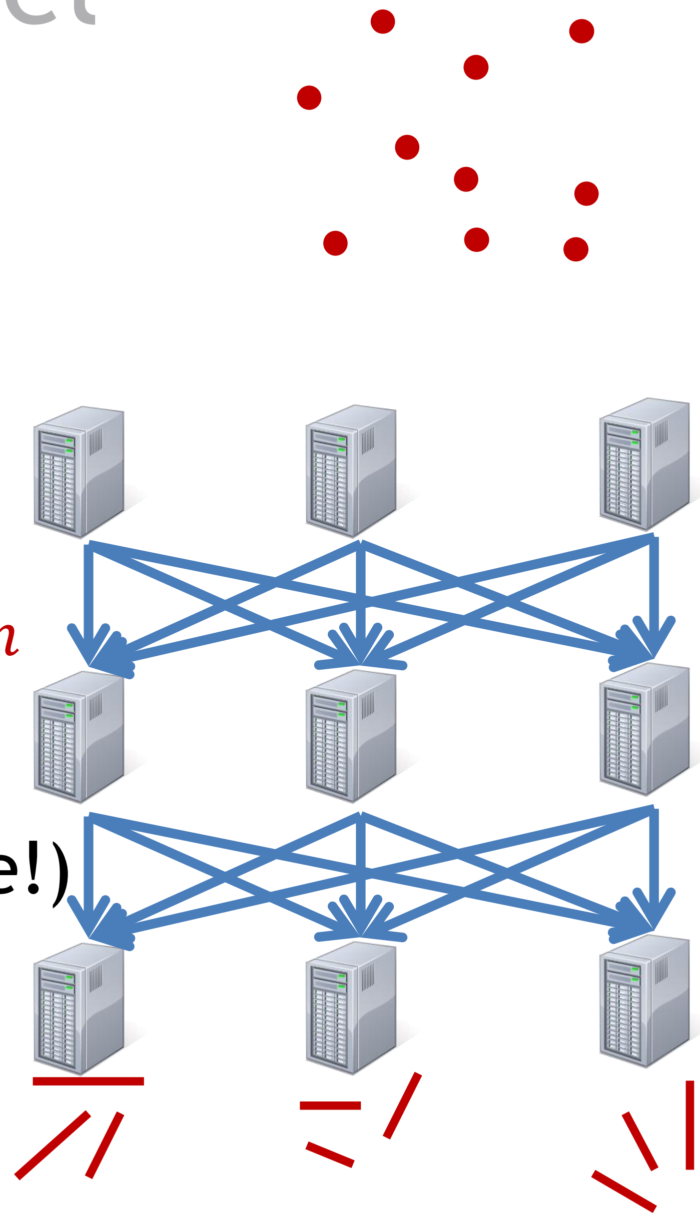
COLUMBIA | ENGINEERING
The Fu Foundation School of Engineering and Applied Science

Admin

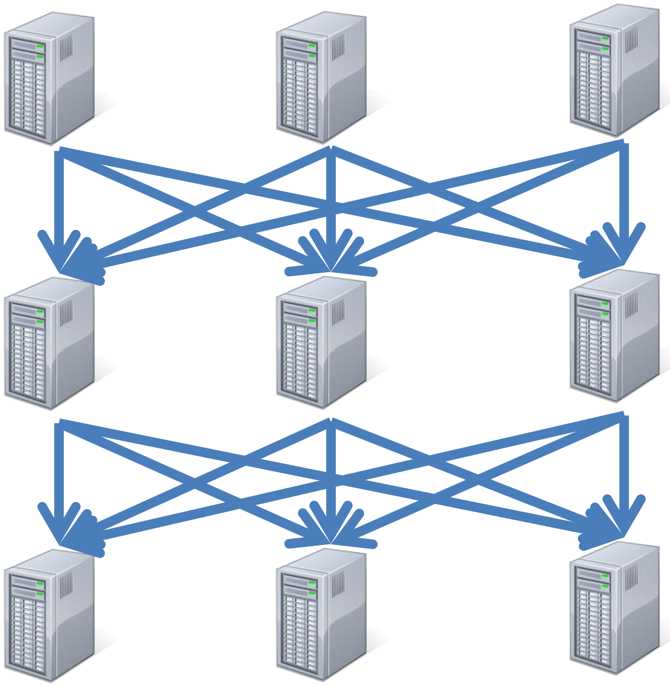
- PS2-4 solutions
- Project presentations next week
  - 20min presentation/team
  - 10 teams => 3 days
  - 3rd time: Fri at 3-4:30pm
  - sign-up sheet online
- Today:
  - MapReduce algorithms
  - Wrap-up

Computational Model

- M machines
- S space per machine
- $M \cdot S \approx$ O(input size)
  – cannot replicate data much
- Input: n elements
- Output: O(input size)=O(n)
  doesn't fit on a machine: $S \ll n$
- Round: shuffle all (expensive!)



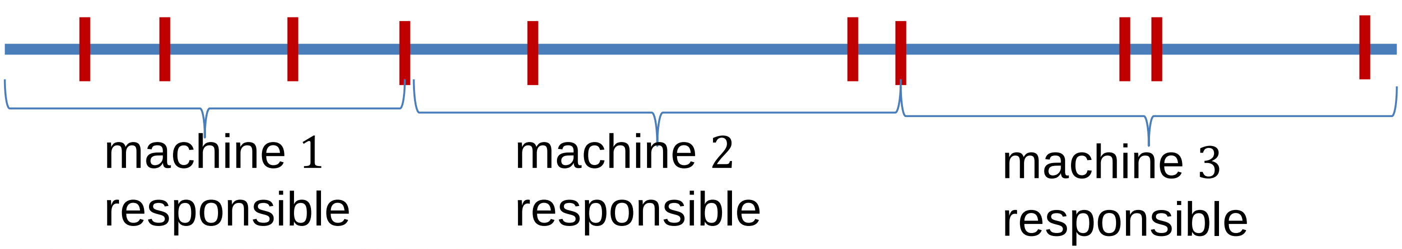
Model Constraints



- Main goal:
  - number of rounds $R = O(1)$
  - for $S \geq n^{\delta}$
    - e.g., $S > \sqrt{n}$ when $S > M$
- Local resources bounded by S
  - $O(S)$ in-communication per round
  - ideally: linear run-time/round
- Model culmination of:
  - Bulk-Synchronous Parallel [Valiant'90]
  - Map Reduce Framework [Feldman-Muthukrishnan-Sidiropoulos-Stein-Svitkina'07, Karloff-Suri-Vassilvitskii'10, Goodrich-Sitchinava-Zhang'11]
  - Massively Parallel Computing (MPC) [Beame-Koutis-Suciu'13]

Problem 1: sorting

- Suppose:
  - $S = O(n^{2/3})$
  - $M = O(n^{1/3})$
- Algorithm:
  - Pick each element with Pr=$\frac{n^{1/2}}{n}$ (locally!)
    - total $\Theta(n^{1/2})$ elements selected
  - Send selected elements to machine #1
  - Choose ~equidistant *pivots* and assign a range to each machine
    - each range will capture about $O(n^{2/3})$ elements
  - Send the pivots to all machines
  - Each machine sends elements in range i to machine $\#i$
  - Sort locally
- 3 rounds!

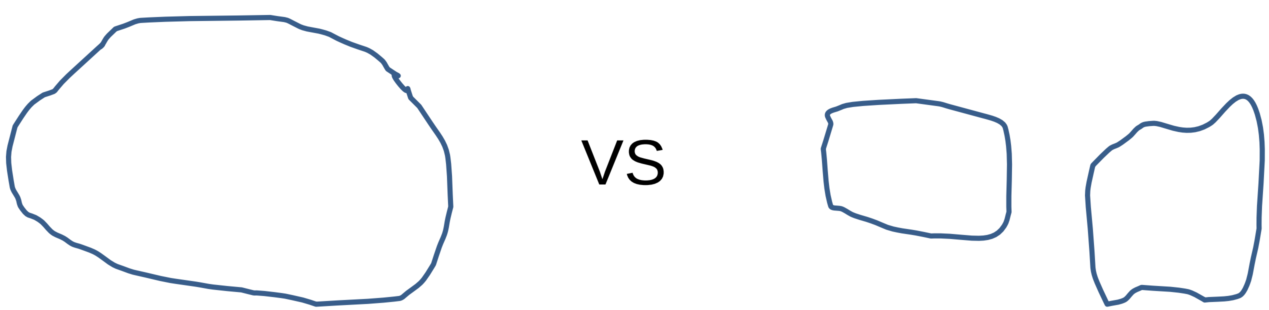


Parallel algorithms from 80-90's

- Can reuse algorithms in Parallel RAM model
  - can simulate PRAM algorithms with
    R=O(parallel time) [KSV'10,GSZ'11]
- Bad news: often ≈ logarithmic…
  - e.g., XOR
    - $\tilde{\Omega}(\log n)$ on CRCW [BH89]
    - Difficulty: information aggregation
    - $O(\log_S n) = const$ on MapReduce/MPC !
- MapReduce as a circuit:
  - $S = n^\delta$ fan-in
  - arbitrary function at a "gate"

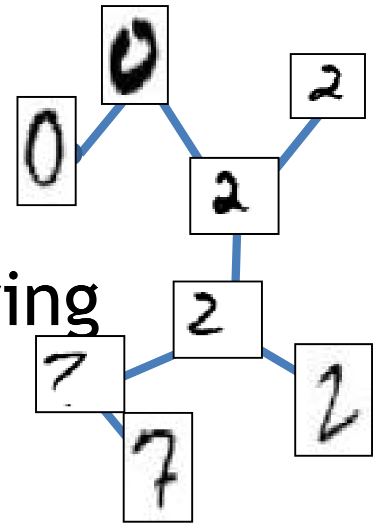
Graph problems: connectivity

- Dense: if $S \gg$ solution size
  - "Filtering": filter input until fits on a machine
  - $S = n^{1+\delta}$ can do in $O\left(\frac{1}{\delta}\right)$ rounds [KSV'10, EIM'11…]
- Sparse: if $S \ll$ solution size
  - $S = \sqrt{n}$
  - Hard: big open question to do s-t connectivity in $\ll \log n$ rounds
  - Lower bounds for restricted algorithms [BKS13]



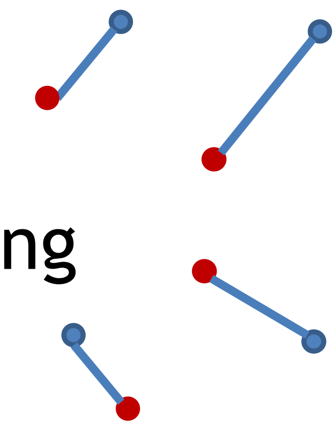
Geometric Graphs

- Implicit graph on n points in $\mathbb{R}^d$
  - distance = Euclidean distance
- Minimum Spanning Tree
  - Agglomerative hierarchical clustering
    [Zahn'71, Kleinberg-Tardos]
- Earth-Mover Distance
- etc



Geometric Graphs

- Implicit graph on n points in $\mathbb{R}^d$
  - distance = Euclidean distance
- Minimum Spanning Tree
  - Agglomerative hierarchical clustering
    [Zahn'71, Kleinberg-Tardos]



- Earth-Mover Distance
- etc

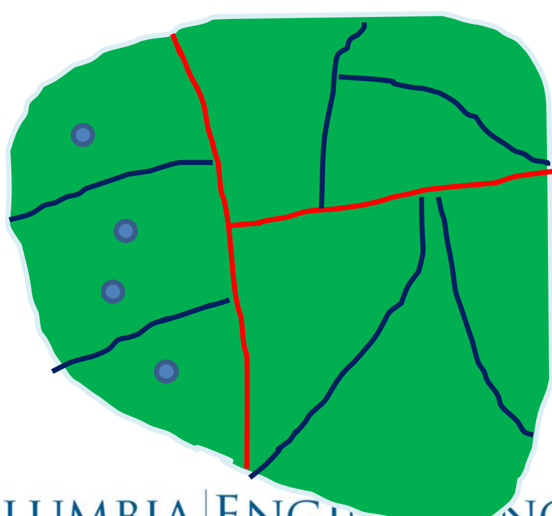
Results: MST & EMD algorithms

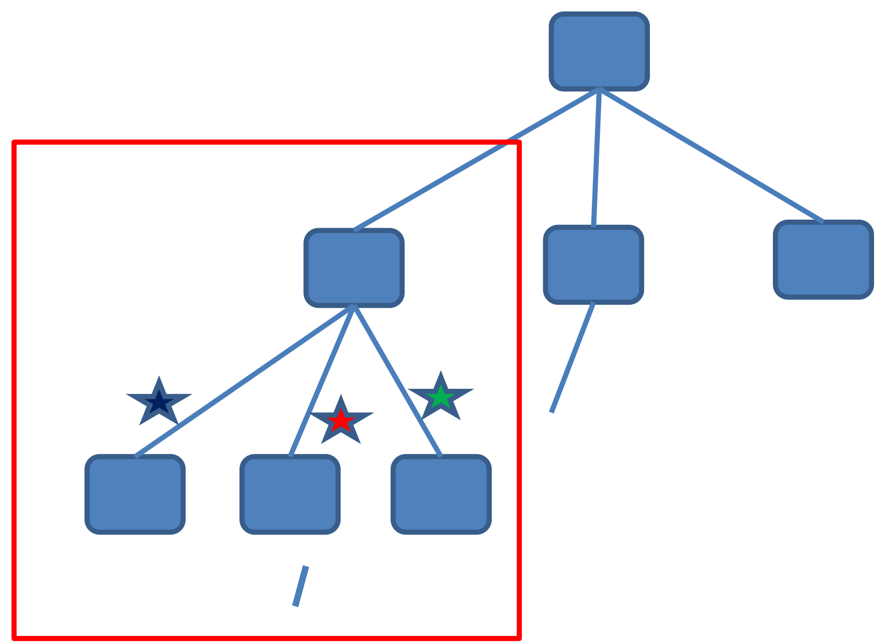
[A-Nikolov-Onak-Yaroslavtsev'14]

- Theorem: can get
  - $1 + \epsilon$ approximation in low dimensional space ($\mathbb{R}^d$)
  - Constant number of rounds: $R = (\log_S n)^{O(1)}$
- For:
  - Minimum Spanning Tree (MST):
    - as long as $S \geq \epsilon^{-O(d)}$
  - Earth-Mover Distance (EMD):
    - as long as $S \geq n^{o(1)}$ for constant ϵ, d

Framework: Solve-And-Sketch

- Partition the space hierarchically in a “nice way”
- In each part
  - Compute a pseudo-solution for the local view
  - Sketch the pseudo-solution using small space
  - Send the sketch to be used in the next level/round





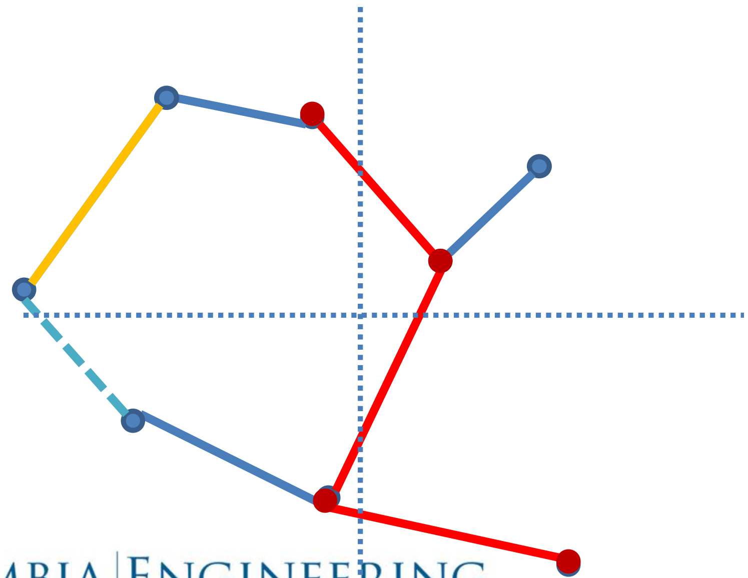
MST algorithm: attempt 1

quad trees!

- Partition the space hierarchically in a "nice way"
- In each part
  - Compute a pseudo-solution for the local view
  - Sketch the pseudo-solution using small space
  - Send the sketch to be used in the next level/round

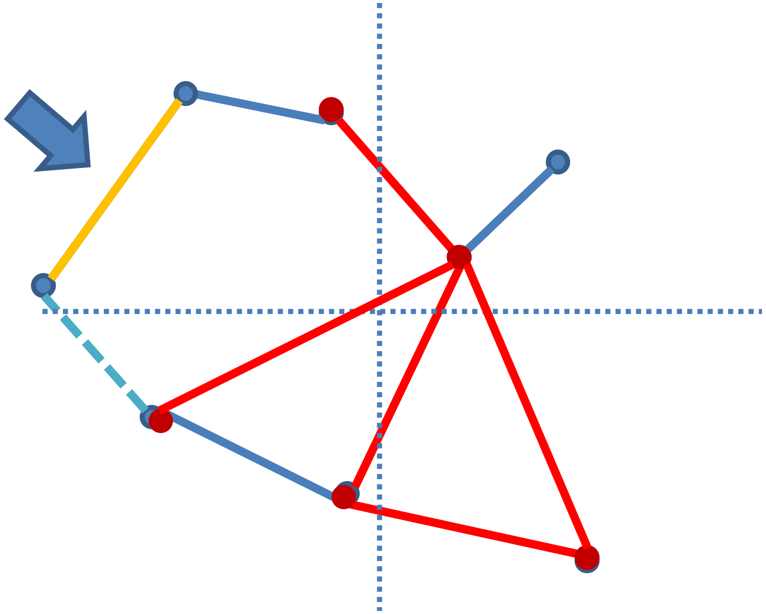
local MST

send any point as a representative



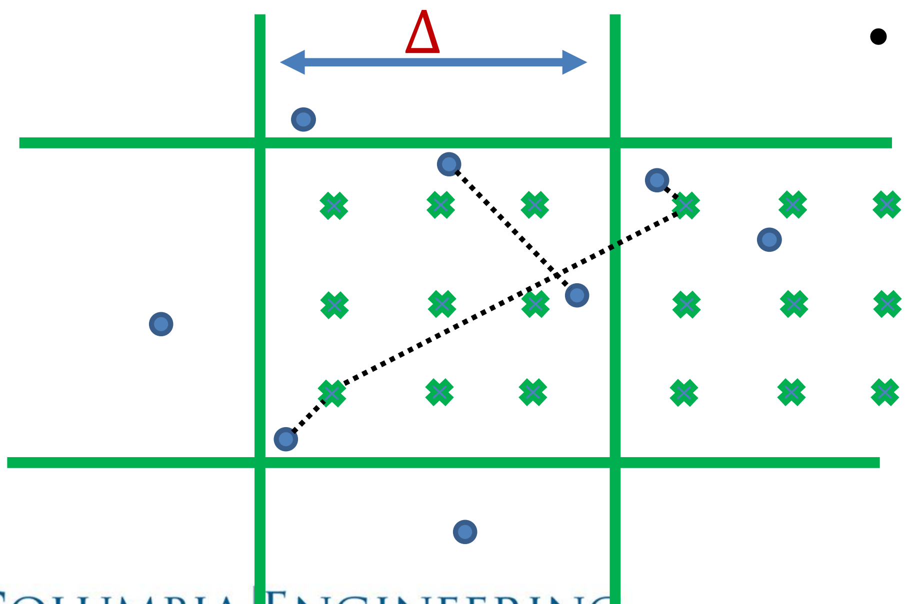
Difficulties

- Quad tree can cut MST edges
  - forcing irrevocable decisions
- Choose a wrong representative



New Partition: Grid Distance

- Randomly shifted grid [Arora'98, …]
- Take an $\epsilon\Delta$-net N
- Net points are entry/exit portals for the cell
- $d'(p,q) =$
  - Old distance if in the same cell
  - Snap each point to closest net-point + net-point to net-point distance
- Claim: all distances preserved up to $1+8\epsilon$ in expectation



- Proof:

  fix pair p, q

  $$\delta = \Pr[p, q\ cut] \leq \frac{2||p-q||}{\Delta}$$

  Hence:

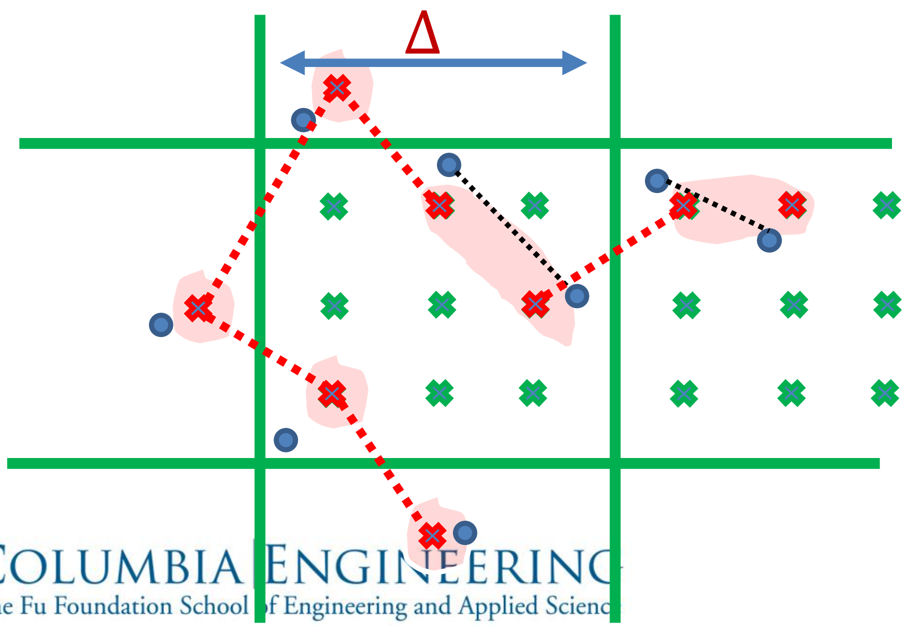
  $$E[d'(p,q)] \leq ||p-q|| + 4\delta \cdot \epsilon\Delta$$
  $$\leq ||p-q|| \cdot (1+8\epsilon)$$

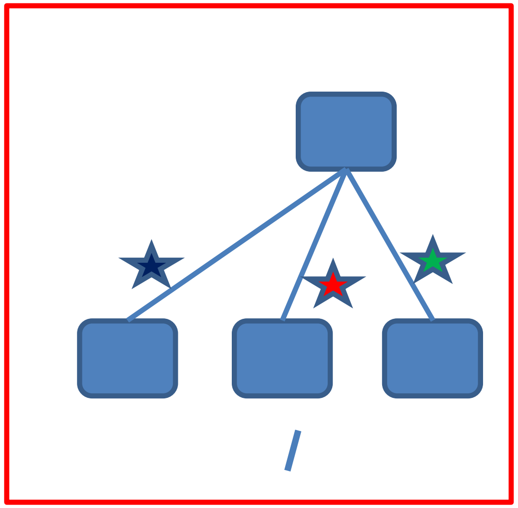
MST Algorithm: Final

- Assume entire pointset in a cube of size $n^{2/3} \times n^{2/3}$

  also $S \gg n^{2/3}$
- Partition:
  - Randomly-shifted grid with $\Delta = n^{1/3}$
  - 2 levels of partition: local size $\Delta \times \Delta < S$
- Pseudo-solution:
  - Run Kruskal's algorithm locally, for edges up to length $\epsilon\Delta$
- Sketch of a pseudo-solution:
  - Snap points to $\epsilon^2\Delta$-net N_2, and store their connectivity => size $O\left(\frac{1}{\epsilon^4}\right)$

Kruskal's MST algorithm: connect the points with the shortest edge that does not introduce a cycle

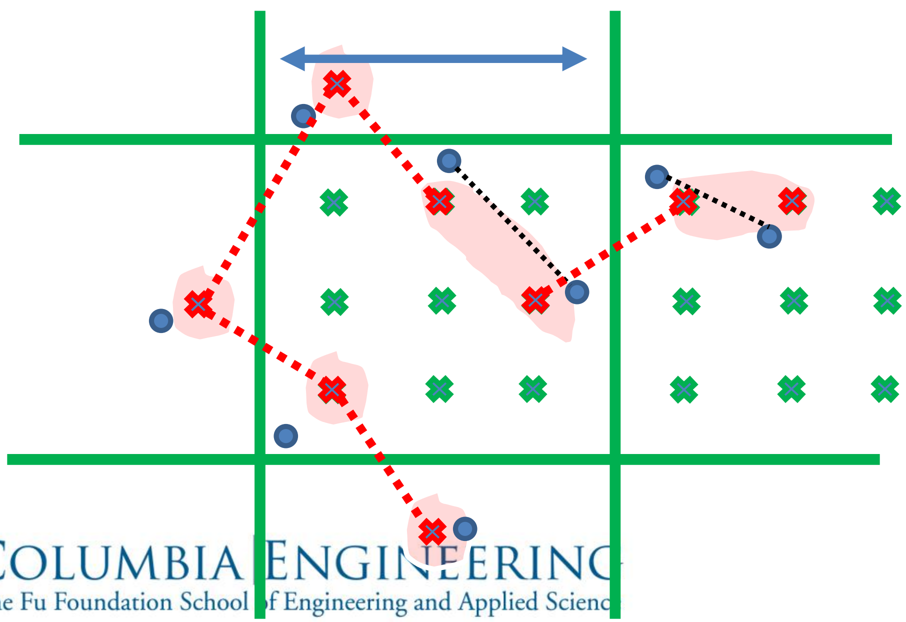




MST Analysis

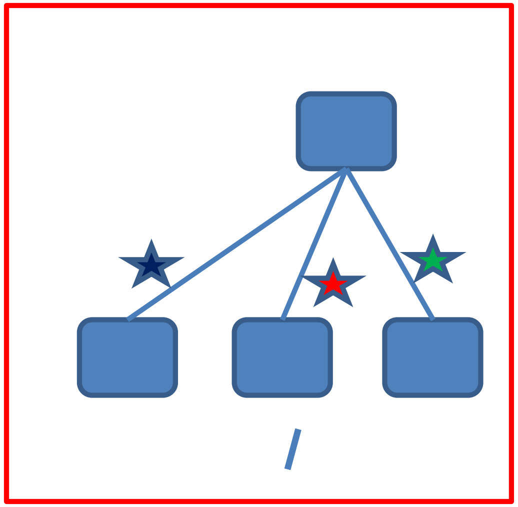
Kruskal's MST algorithm: connect the points with the shortest edge that does not introduce a cycle

- Claim: our algorithm is equivalent to running Kruskal on the distance d', up to $1 + O(\epsilon)$ approximation
  - Any distance across cells is $\geq \epsilon\Delta$
  - Safe to run Kruskal **locally inside each cell** up to this threshold!
  - Snapping to $\epsilon^2\Delta$-net points: introduces $1 + 2\epsilon$ factor error only since all distances are now at least $\epsilon\Delta$



MST Wrap-up

- Conclusion:
  - We find an MST with cost at most $1 + 2\epsilon$ time the MST under the distance d'
  - Hence: $E[cost\ of\ MST] \leq \big(1 + O(\epsilon)\big) \cdot MST_{opt}$
- Local run-time?
  - Linear: using approximate Kruskal
- How is the solution represented?
  - Each machine has a number of edges from the MST
  - The top machine has the remaining edges

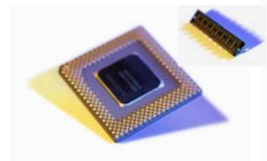


Wrap-up

1) Streaming algorithms

- Streaming algorithms

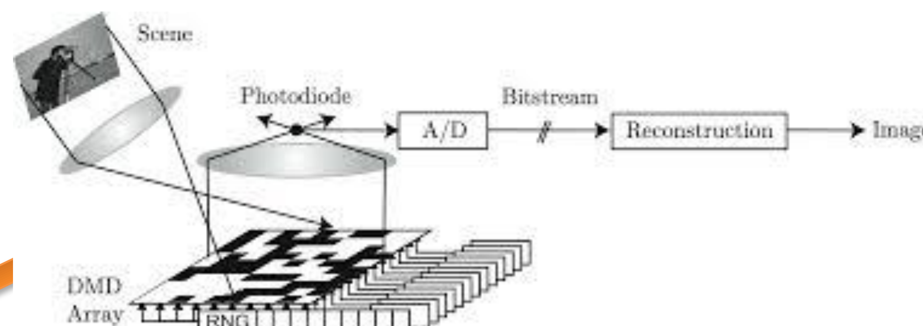
IP	Frequency
160.39.142.2	3
18.9.22.69	2
80.97.56.20	2



- Frequency moments, heavy hitters
- Graph algorithms
- *Algorithms for lists: Median selection, longest increasing sequence*
- *Algorithms for geometric objects: clustering, MST, various approximation algorithms*

Sketching &dimension reduction

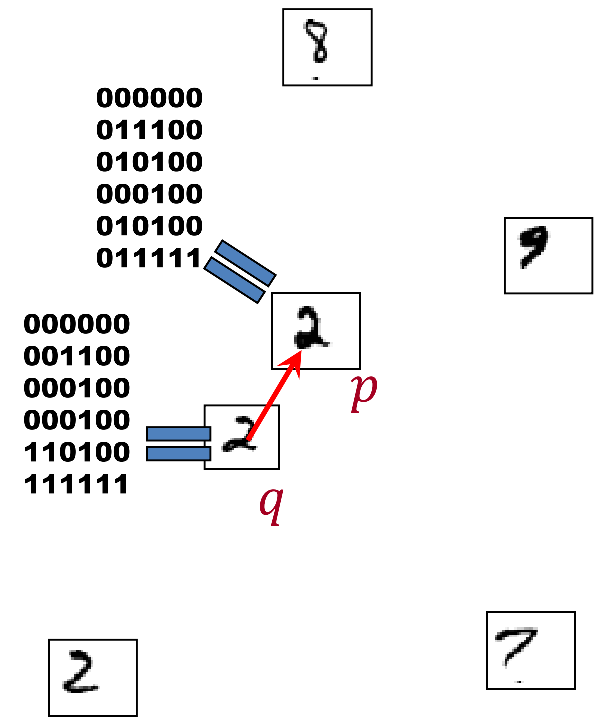
- Power of linear sketches: $S(a+b) = S(a) + S(b)$
- For frequency vectors, dynamic graphs
- Ultra-efficient for ℓ_1, ℓ_2: $1+\epsilon$ approximation in constant space!
- Dimension reduction: Johnson-Lindenstrauss
- Fast JL, using Fast Fourier Transform
- Can speed-up numerical linear algebra!
- Compressed sensing: *many algorithms/models*



source: http://dsp.rice.edu/sites/dsp.rice.edu/files/cs/cscam-SPIEJan06.pdf

Nearest Neighbor Search

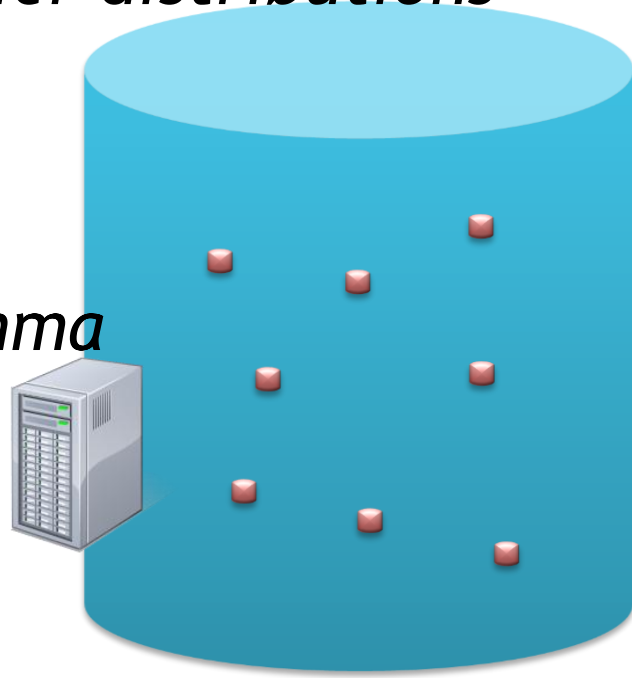
- Can use sketching for NNS
- Even better via Locality Sensitive Hashing
- Data-dependent LSH
- Embeddings: reduce harder distances to easier ones



- *NNS for general metrics*
- *Complexity dependent on "intrinsic dimension"*

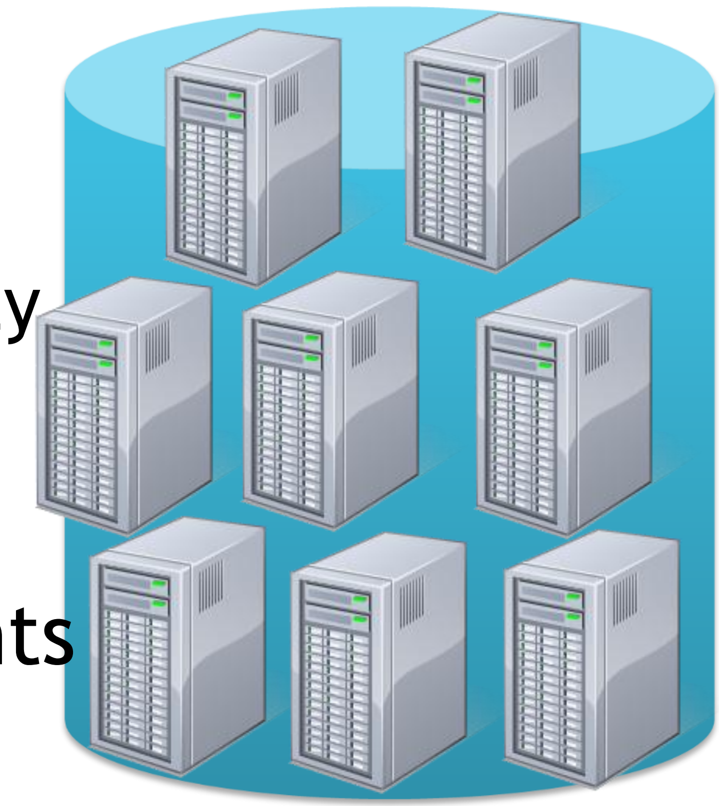
Sampling, property testing

- Distribution testing:
  - Get samples from a distribution, deduce its properties
  - Uniformity, identity
  - *Many others in the literature!*
  - *Instance optimal: better for easier distributions*
- Property testing:
  - Is this graph connected or far from connected?
  - *For dense graphs: regularity lemma*
- Sublinear time approximation:
  - Estimate the MST cost, matching size, *etc*



Parallel algorithms: MapReduce

- Model: limited space/machine
- Filtering: throw away part of the input locally, send only important stuff
- Dense graph algorithms
- Solve-And-Sketch:
  - find a partial solution locally
  - sketch the solution
  - work with sketches up
- Good for problems on points



Algorithms for massive data

- Computer resources << data
- Access data in a limited way
  - Limited **space** (main memory << hard drive)
  - Limited **time** (time << time to read entire data)

power of randomization and approximation

Introduction to *Sublinear* Algorithms