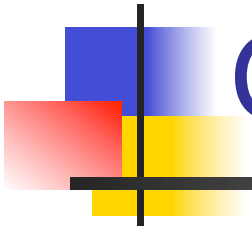
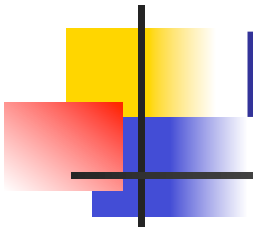


Streaming Algorithms for Geometric Problems

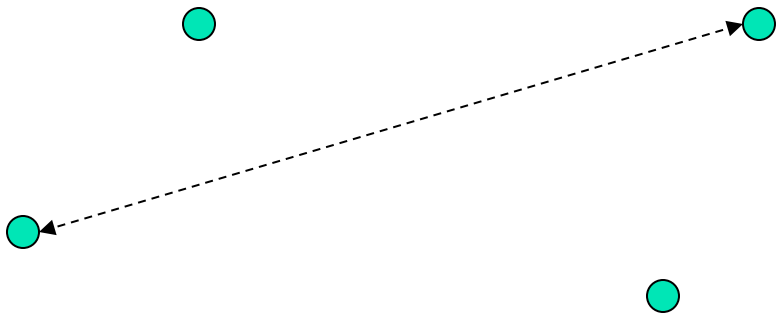
Piotr Indyk
MIT



Recap

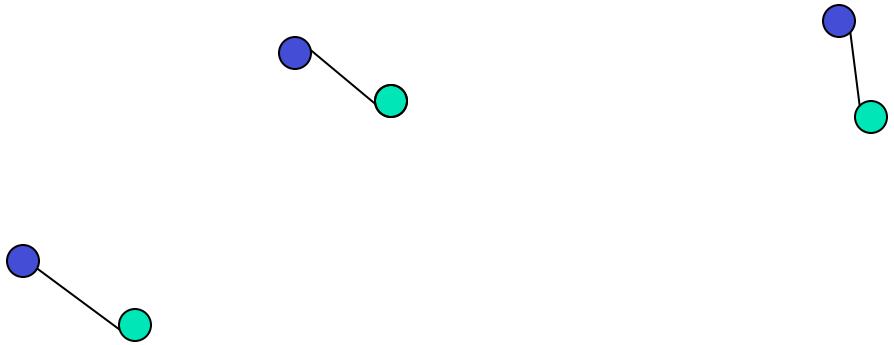
- Have seen:
  - Stream of updates (i,c) to a vector x, interpreted as $x_i = x_i + c$
  - Maintain a linear sketch of x
  - Can compute L_p norm, heavy hitters, sparse approximations
- The next few lectures: diversify
  - Geometric problems (in R^2)
  - Metric problems

Diameter



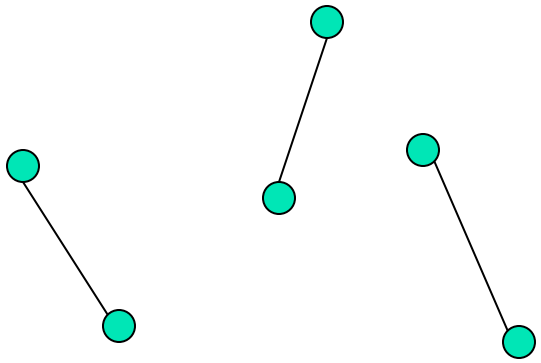
- Estimate the max distance between the points

Minimum Weight Bi-chromatic Matching



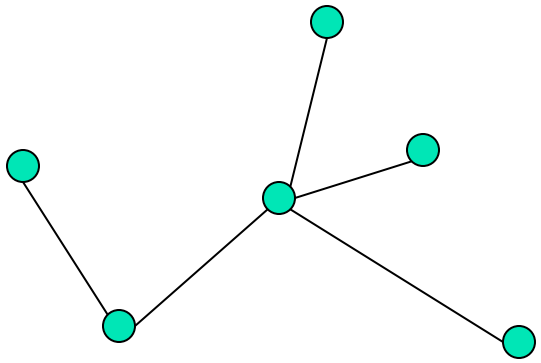
- Estimate the cost of MWBM

Minimum Weight Matching



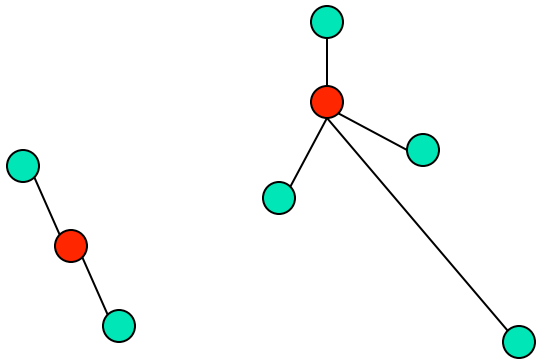
- Estimate the cost of MWM

Minimum Spanning Tree

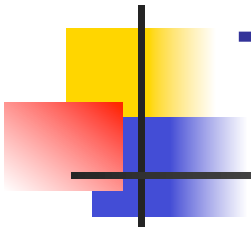


- Estimate the cost of MST

Facility Location



- Goal: choose a set F of facilities to minimize the sum of the distances to nearest facility plus the number of facilities times f
- Again, report the cost

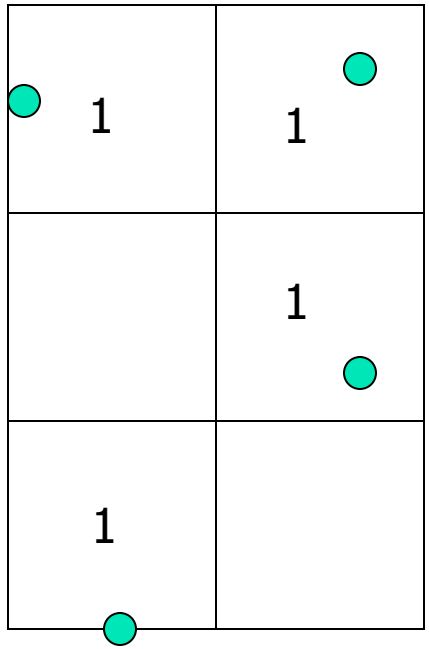


The "great streaming divide"

- Insertions and deletions
  - Maintain a pointset P under insertions and deletions
  - Need to assume coordinates are discrete, i.e., that the points come from $\{1 \ldots \Delta\}^2$
  - Reduction of geometric problems to vector problems
- Insertions only
  - Can assume arbitrary ("real") coordinates
  - Core-set technique

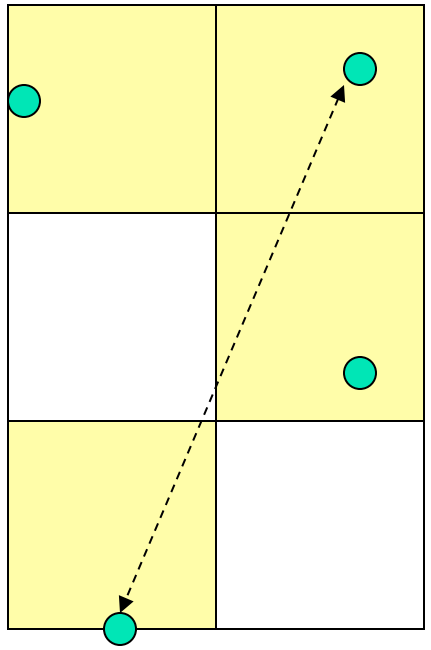
Warmup: diameter

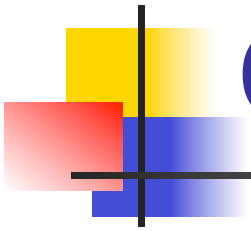
- $(1+O(\varepsilon))$-approximation using $O(1/\varepsilon^{O(1)}\ \mathrm{polylog}(m+\Delta))$ space
- Algorithm:
  - Impose square grids $G_0 \ldots G_k$, with side lengths $(1+\varepsilon)^0, (1+\varepsilon)^1, (1+\varepsilon)^2 \ldots, \Delta$,
  - For each square cell c in G_i, let $n_P^i(c)$ be the number of points from P in c
  - Insertions/deletions of points to P results in updates to vectors n_P^i
  - The algorithms will maintain linear sketches over vectors n_P^i which will allow it to approximately solve:
    - Exact k-sparse recovery problem for $k=O(1/\varepsilon^2)$
    - Norm estimation problem (to check whether a vector is k-sparse)



Estimation

- Let D be the diameter of P, and $(1+\varepsilon)^i \le \varepsilon D \le (1+\varepsilon)^{i+1}$
- Then n_P^i has only $k=O(1/\varepsilon^2)$ non-zero entries
- Estimation:
  - Let i^* be the smallest value such that $n_P^{i^*}$ has <=k non-zero entries
  - Recover the set S of non-zero cells in $n_P^{i^*}$
  - Return $(1+\varepsilon)^{i^*}$ times the diameter of S





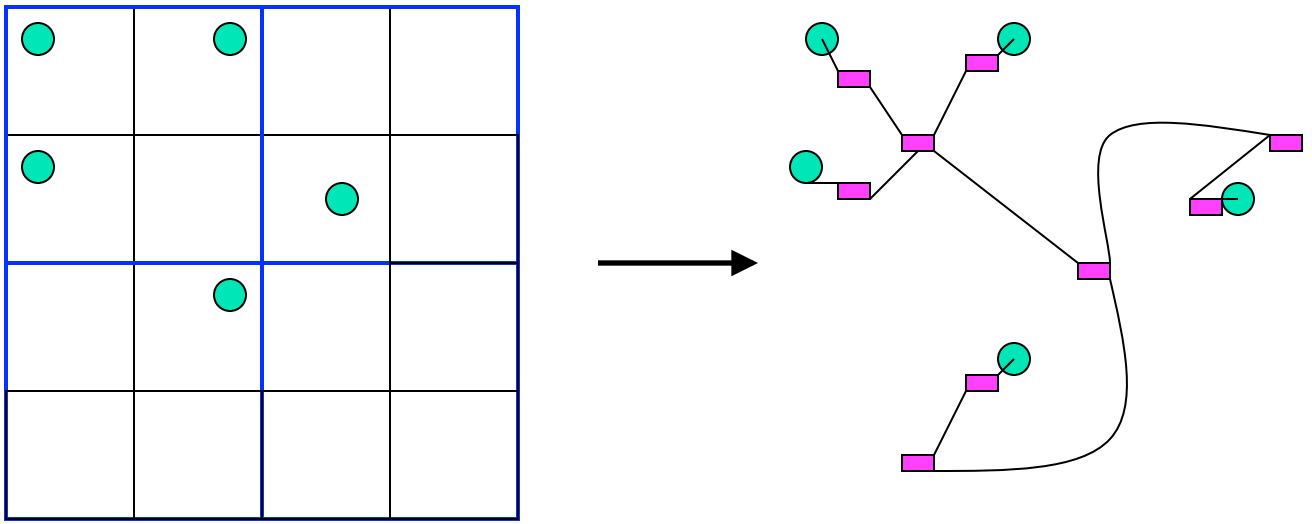
Other Problems

- MST, Matching, etc
- Approximation factor: $O(\log \Delta)$
  - Can be improved to $1+\varepsilon$ for MST
- Main idea: embedding R^2 into a (quad)-tree metric

Probabilistic embedding [Bartal'96,…]

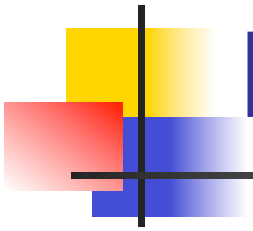
Grids $G_0, G_1, ..$ with cell lengths $2^0, 2^1, 2^2 \ldots$

Tree T (edge length=cell side length)



Fact: If the grids are shifted by random vector $v \in \{1 \ldots \Delta\}^2$ then:

1. $||p\text{-}q||_1 \leq D_{tree}(p,q)$
2. $E[\, D_{tree}(p,q)\,] \leq ||p\text{-}q||_1 * O(\log \Delta)$



Proof:

- Property 1: by design (tree edge lengths are long enough to avoid contraction)
- Property 2:
  - Observe that for a randomly shifted grid with cell side length 2^i , the probability that points p and q belong to different cells is at most $||p-q||_1/2^i$
  - If the grid separates those points, it contributes $O(2^i)$ to $D_{tree}(p,q)$
  - Thus, the i-th level edges contribute $O(||p-q||)$ to $D_{tree}(p,q)$ in the expectation
  - Summing up over all levels gives $||p-q||_1 * O(\log \Delta)$ expected bound on $D_{tree}(p,q)$

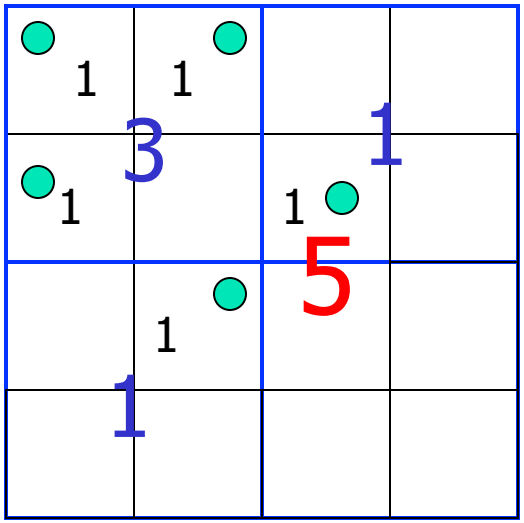
Estimators

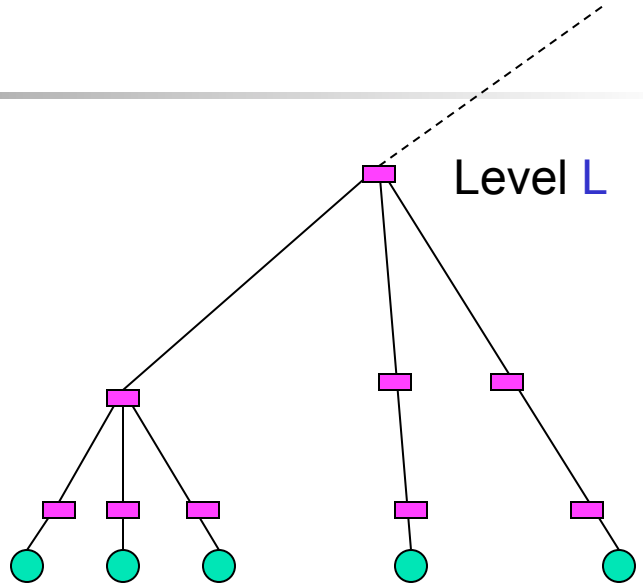
- MST: $\sum_i^{L-1} 2^i \sum_{c \in G^i} [n_P^i(c)>0]$
  - L is the smallest level with exactly one non-zero entry in the count vector (see also later)
- MWM: $\sum_i 2^i \sum_{c \in G^i} [n_P^i(c) \text{ is odd}]$
- MWBM: $\sum_i 2^i \sum_{c \in G^i} |n_G^i(c)-n_B^i(c)|$
- Fac. Loc.: $\sum_i 2^i \sum_{c \in G^i} \min[n_P^i(c), T_i]$

Maintain #non-zero entries

Maintain L_1 difference

MST

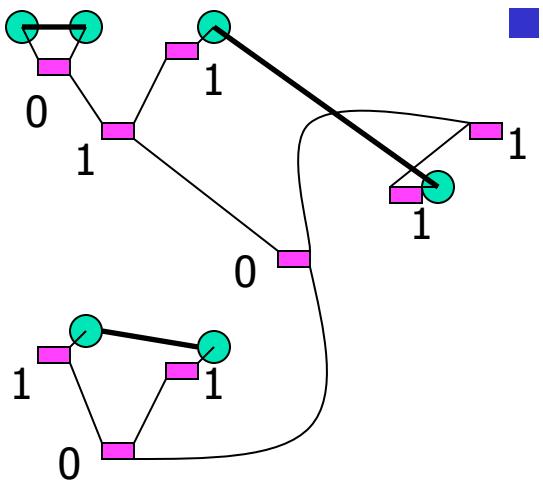




- Let T′ be the MST of P, T″ be the image of T′ in T (union of all images of edges of T′ , removing duplicates)
- E[Cost(T″)] = O(log Δ) Cost(T′), and Cost(T″) =Ω(Cost(T′))
- Cost(T″) = ?

  = Cost(T up to level L)
- Cost(T up to level L) =$\sum_i^{L-1} 2^i \sum_{c \in Gi} [n_P^i(c)>0]$

Matching



- Algorithm for the tree:
  - Match what you can at the current level
  - Odd leftovers wait for the next level
  - Repeat
- Cost=$\sum_i 2^i \sum_{c \in Gi} [n_P(c)$ is odd$]$