Range Reporting

- Range reporting problem
- 1D range reporting
  - Range trees
- 2D range reporting
  - Range trees
  - Fractional cascading
  - kD trees

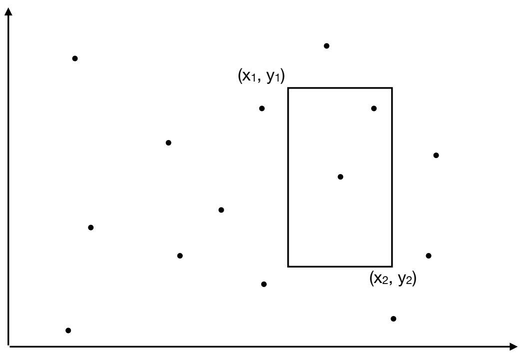
Philip Bille

Range Reporting

- Range reporting problem
- 1D range reporting
  - Range trees
- 2D range reporting
  - Range trees
  - Fractional cascading
  - kD trees

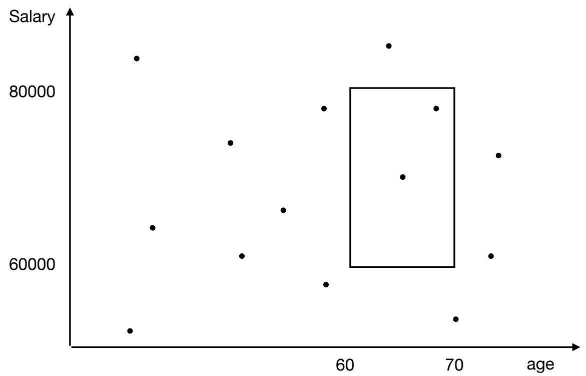
Range Reporting Problem

- 2D range reporting problem. Preprocess at set of points $P \subseteq \Re^2$ to support
  - report(x_1, y_1, x_2, y_2): Return the set of points in $R \cap P$, where R is rectangle given by (x_1, y_1) and (x_2, y_2).



Applications

- Relational databases. SELECT all employees between 60 and 70 years old with a montly salary between 60000 and 80000 DKr



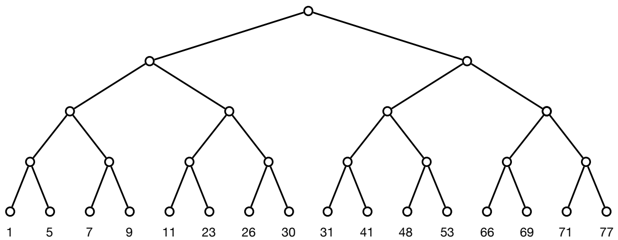
Range Reporting

- Range reporting problem
- 1D range reporting
  - Range trees
- 2D range reporting
  - Range trees
  - Fractional cascading
  - kD trees

1D Range Reporting

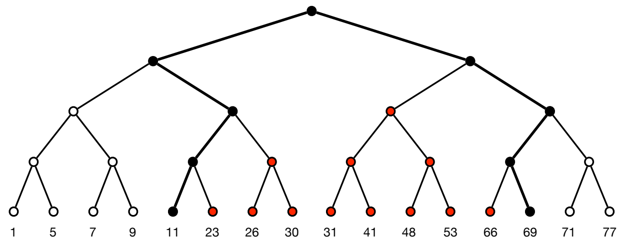
- 1D range reporting. Preprocess a set of n points $P \subseteq \Re$ to support:
  - report(x_1, x_2): Return the set of points in interval [x_1, x_2]
- Simplifying assumption. Only comparison-based techniques (no hashing or bittricks).
- Solutions?

1D Range Reporting



- 1D range tree. Balanced binary tree over P in sorted order.
  - All points stored at leaves.
  - Internal nodes stored range of points below.
- Space. O(n)
- Preprocessing. O(n log n)

1D Range Reporting



- Report(x_1, x_2): Search for predecessor of x_1 + successor of x_2. Traverse all nodes in between.
- Example. Report(20, 68) = {23, 26, 30, 31, 41, 48, 53, 66}.
- Time. O(log n + occ)

1D Range Reporting

- Theorem. We can solve the 1D range reporting problem in
  - O(n) space.
  - O(log n + occ) time for queries.
  - O(n log n) preprocessing time.
- Optimal in comparison-based model.

Range Reporting

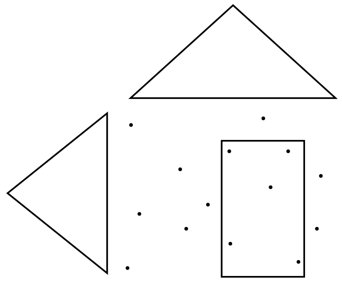
- Range reporting problem
- 1D range reporting
  - Range trees
- 2D range reporting
  - Range trees
  - Fractional cascading
  - kD trees

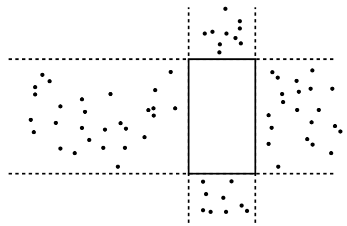
2D Range reporting

- Goal. 2D range reporting with
  - O(n log n) space and O(log n + occ) query time or
  - O(n) space and $O(n^{1/2} + occ)$ query time.
- Solution in 4 steps.
  - Generalized 1D range reporting.
  - 2D range trees.
  - 2D range trees with fractional cascading.
  - kD trees.

Generalized 1D Range Reporting

- Data structure.
  - 1D range tree T_x over x-coordinate
  - 1D range tree T_y over y-coordinate
- Report(x_1, y_1, x_2, y_2):
  - Compute all points R_x in x-range.
  - Compute all points R_y in y-range.
  - Return $R_x \cap R_y$
- Time?



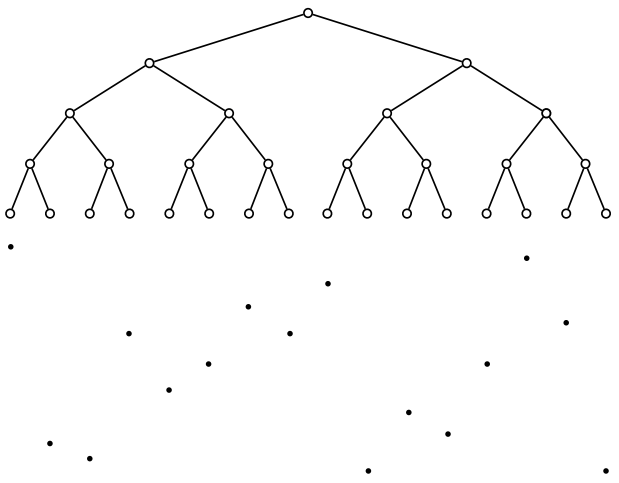


2D Range Trees

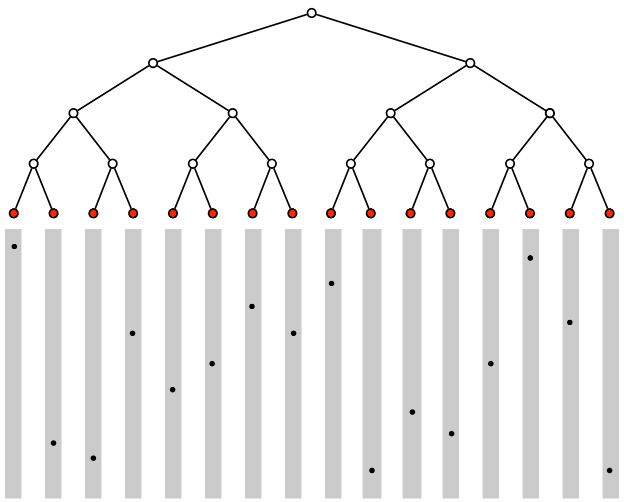
- Data structure.
  - A 1D range tree T_x over x-coordinate.
  - For each node v in T_x: Store a 1D range over y-coordinate for the subset of P below v.

2D Range Trees

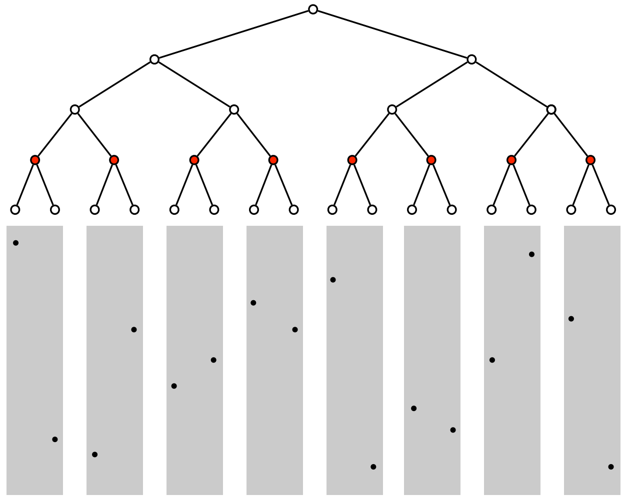
2D Range Trees



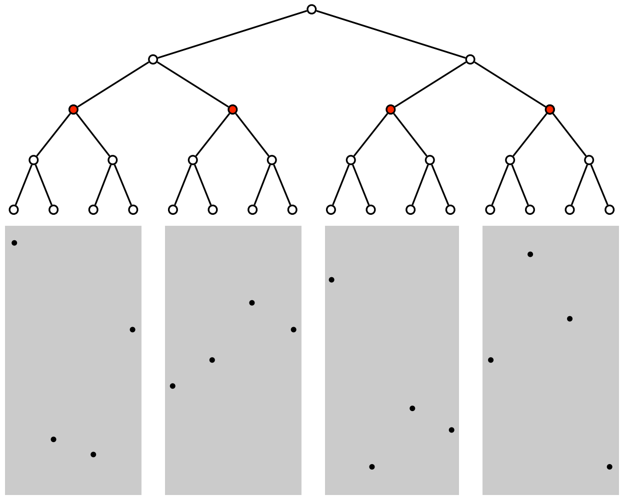
2D Range Trees



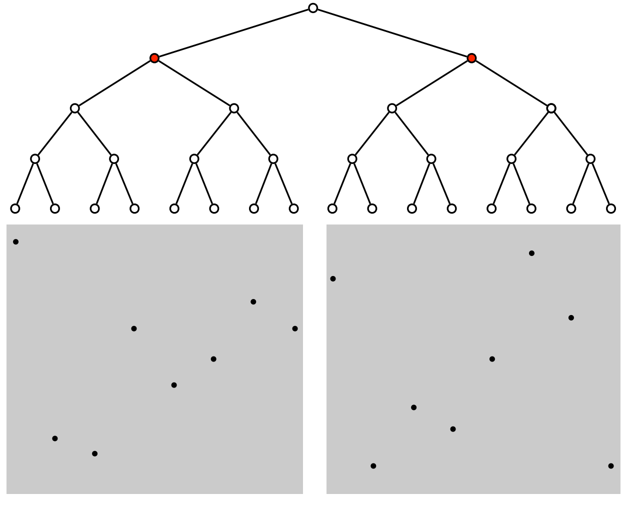
2D Range Trees



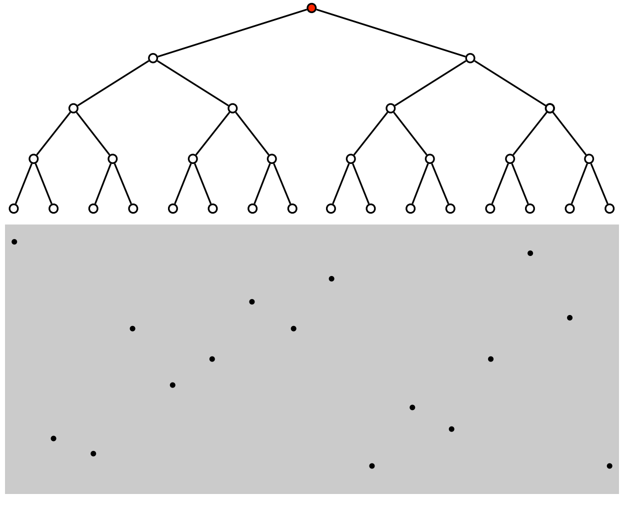
2D Range Trees



2D Range Trees



2D Range Trees



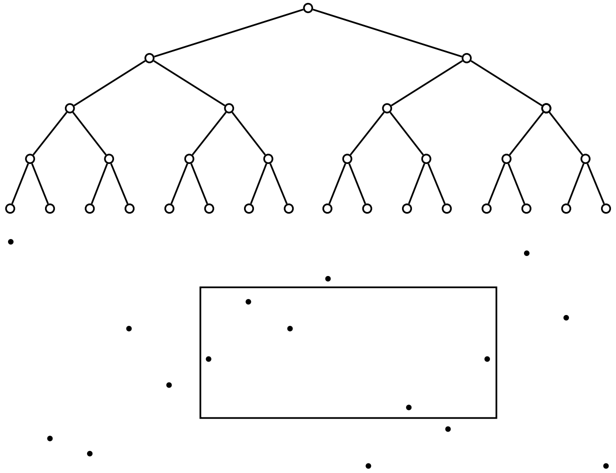
2D Range Trees

- Space.
    - Each point stored in ~log n range trees ⟹ O(n log n) space.
- Preprocessing time. O(n log n)

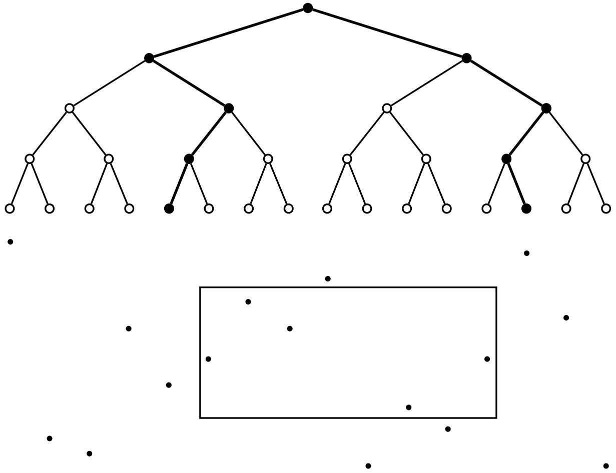
2D Range Trees

- Report(x_1, y_1, x_2, y_2):
    - Search in T_x for x-range.
        - For each node v hanging of search path within x-range:
        - Do a 1D report query with y-range.
    - Return the union of the results.

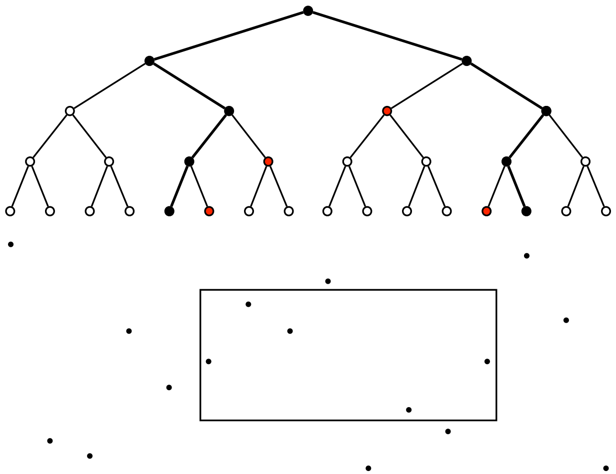
2D Range Trees



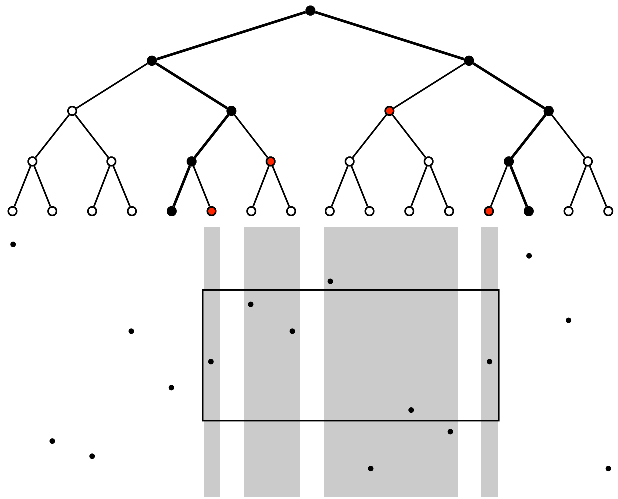
2D Range Trees



2D Range Trees



2D Range Trees



2D Range Trees

- Time.
  - 1D range query on x-range: O(log n) time
  - < 2log n 1D range queries: Each uses O(log n + occ in subrange) time.
  - ⟹ in total O($\log^2$ n + occ) time.

2D Range Reporting

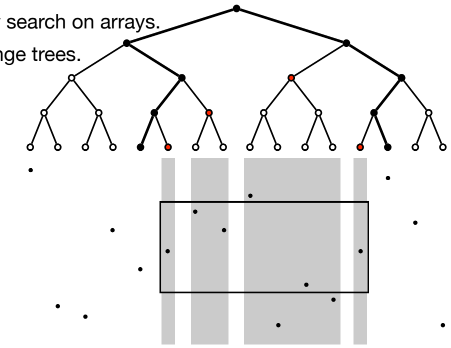
- Theorem. We can solve the 2D range reporting problem in
  - O(n log n) space.
  - O($\log^2$ n + occ) time for queries.
  - O(n log n) preprocessing time.
- Do we really need the $\log^2$ n term for queries? Can we get (optimal) O(log n) time?

Range Reporting

- Range reporting problem
- 1D range reporting
  - Range trees
- 2D range reporting
  - Range trees
  - Fractional cascading
  - kD trees

Fractional Cascading

- Goal. 2D range reporting in O(n log n) space and O(log n) time
- Idea. Exploit properties of the O(log n) searches on y-range.
  - All searches on the same y-range.
  - Points at node v is a subset of points at parent of v.
- Solution in 2 steps.
  - Fractional cascading for binary search on arrays.
  - Fractional cascading on 2D range trees.



Fractional Cascading

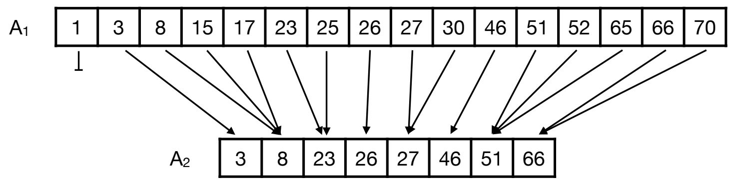
- Binary search on two arrays. Let A_1 and A_2 be two sorted arrays such that $A_2 \subseteq A_1$.
- Goal. Implement binary search for key k on both A_1 and A_2.
- Solution 1. Do a binary search for k on A_1. Do a binary search for k on A_2
- Challenge. Can we add some data structure so we can do it with one binary search?

A_1	1	3	8	15	17	23	25	26	27	30	46	51	52	65	66	70

A_2	3	8	23	26	27	46	51	66

Fractional Cascading

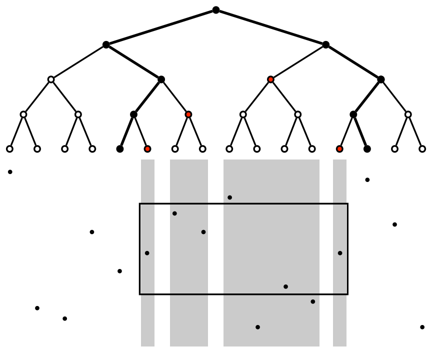
- Solution 2.
  - For each i store pointer to predecessor of $A_1[i]$ in A_2
  - Binary search for k in A_1. Follow pointer and locate predecessor in A_2.



- Space. $O(|A_1| + |A_2|)$
- Time. $O(\log |A_1|)$
- Binary search ⟹ 1D range reporting.
- Generalizes to 2+ arrays.

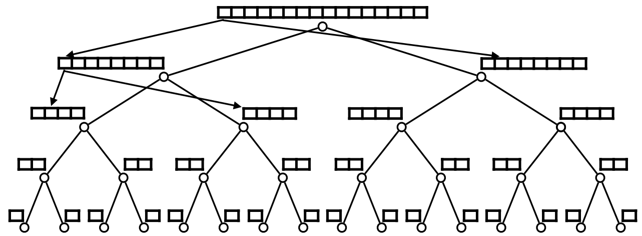
Fractional Cascading

- 2D range trees need O(log n) searches on y-range.
  - All searches on the same y-range.
  - Points at node v is a subset of points at parent of v.
- ⟹ we can implement searches using fractional cascading.



Fractional Cascading

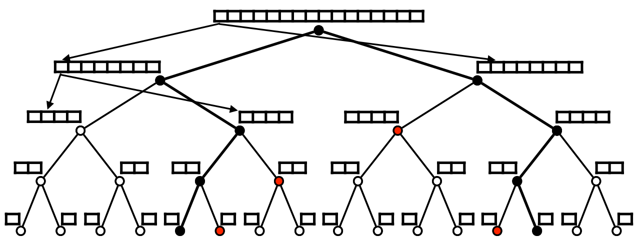
- Data structure.
  - Store a 1D range tree T_x over x-coordinate.
  - For each node v in T_x store a sorted array over y-coordinate for the subset of P below v.
  - Add predecessor pointers from the array for v to the arrays for children of v.



- Space. O(n log n)
- Preprocessing. O(n log n)

Fractional Cascading

- Report(x_1, y_1, x_2, y_2):
  - Search in T_x for x-range.
  - Search in root array for y-range.
  - For each node v hanging off the search paths within the x-range:
    - Do a 1D report query with y-range using predecessor pointers.
  - Return the union of the results.



Fractional Cascading

- Time.
  - 1D range query on x-range: O(log n) time
  - 1D range query on y-range on root array: O(log n) time
  - Pointer walking: O(log n) time.
  - Report points in subrange: O(occ in subrange) time.
  - ⟹ in total O(log n + occ) time.

Fractional Cascading

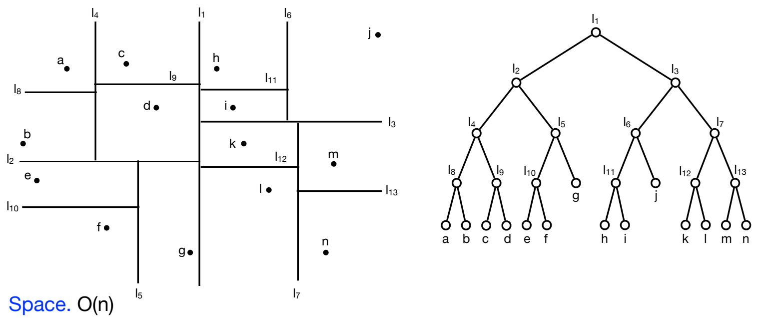
- Theorem. We can solve the 2D range reporting problem in
  - O(n log n) space
  - O(log n + occ) time for queries.
  - O(n log n) preprocessing time.
- What can we do with only linear space?

Range Reporting

- Range reporting problem
- 1D range reporting
  - Range trees
- 2D range reporting
  - Range trees
  - Fractional cascading
  - kD trees

kD Trees

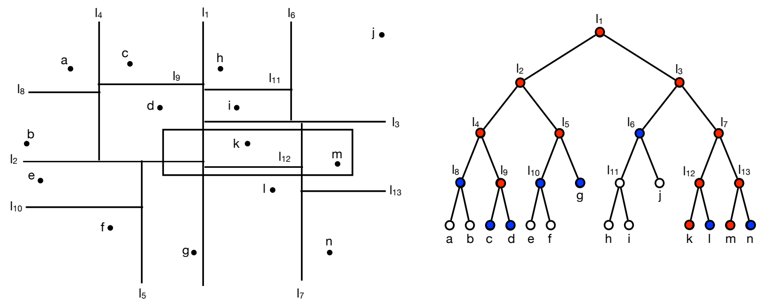
- The 2D tree (k = 2).
  - A balanced binary tree over point set P.
  - Recursively partition P into rectangular regions containing (roughly) same number of points. Partition by alternating horizontal and vertical lines.
  - Each node in tree stores region and line.



- Space. O(n)
- Preprocessing. O(n log n)

kD Trees

- Report(x_1, y_1, x_2, y_2): Traverse 2D tree starting at the root. At node v:
  - Case 1. v is a leaf: report the unique point in region(v) if contained in range.
  - Case 2. region(v) is disjoint from range: stop.
  - Case 3. region(v) is contained in range: report all points in region(v).
  - Case 4. region(v) intersects range, and v is not a leaf. Recurse left and right.



  - Time. $O(n^{1/2})$

kD trees

- Theorem. We can solve the 2D range reporting problem in
  - O(n) space
  - $O(n^{1/2} + occ)$ time
  - O(n log n) preprocessing

2D Range Reporting

- Theorem. We can solve 2D range reporting in either
  - O(n log n) space and O(log n + occ) query time
  - O(n) space and $O(n^{1/2} + occ)$ query time.
- Extensions.
  - More dimensions.
  - Inserting and deleting points.
  - Using word RAM techniques.
  - Other shapes (circles, triangles, etc.)

Range Reporting

- Range reporting problem
- 1D range reporting
  - Range trees
- 2D range reporting
  - Range trees
  - Fractional cascading
  - kD trees