

## Range Reporting

---

- Range reporting problem
- 1D range reporting
- 2D range reporting
  - Range trees
  - Predecessor in nested sets
  - kD trees

Philip Bille

## Range Reporting

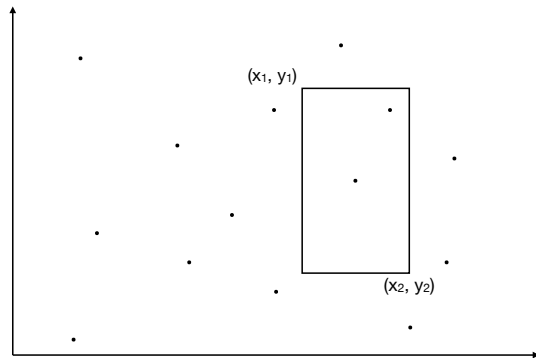
---

- Range reporting problem
- 1D range reporting
- 2D range reporting
  - Range trees
  - Predecessor in nested sets
  - kD trees

## Range Reporting Problem

---

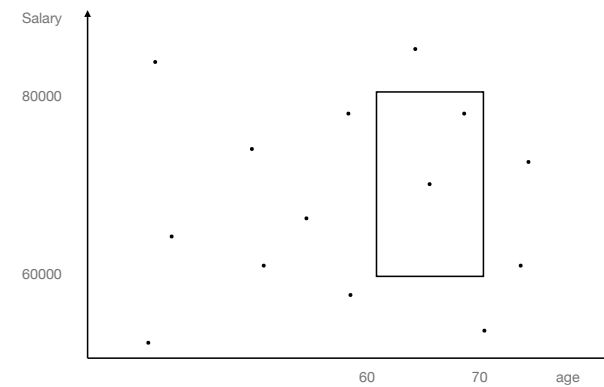
- **2D range reporting problem.** Preprocess at set of points  $P \subseteq \mathbb{R}^2$  to support
  - $\text{report}(x_1, y_1, x_2, y_2)$ : Return the set of points in  $R \cap P$ , where  $R$  is rectangle given by  $(x_1, y_1)$  and  $(x_2, y_2)$ .



## Applications

---

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



## Range Reporting

---

- Range reporting problem
- 1D range reporting
- 2D range reporting
  - Range trees
  - Predecessor in nested sets
  - kD trees

## 1D Range Reporting

---

- **1D range reporting.** Preprocess a set of  $n$  points  $P \subseteq \mathcal{R}$  to support:
  - **report**( $x_1, x_2$ ): Return the set of points in interval  $[x_1, x_2]$
- **Output sensitivity.** Time should depend on the size of the output.
- **Simplifying assumption.** Only **comparison-based** techniques (e.g. no hashing or bittricks).
- Solutions?

## 1D Range Reporting

---

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

- **Sorted array.** Store  $P$  in sorted order.
- **Report**( $x_1, x_2$ ): Binary search for predecessor of  $x_1$ . Traverse array until  $> x_2$ .
- **Time.**  $O(\log n + \text{occ})$
- **Space.**  $O(n)$
- **Preprocessing.**  $O(n \log n)$

## 1D Range Reporting

---

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

## Range Reporting

---

- Range reporting problem
- 1D range reporting
- 2D range reporting
  - Range trees
  - Predecessor in nested sets
  - kD trees

## 2D Range reporting

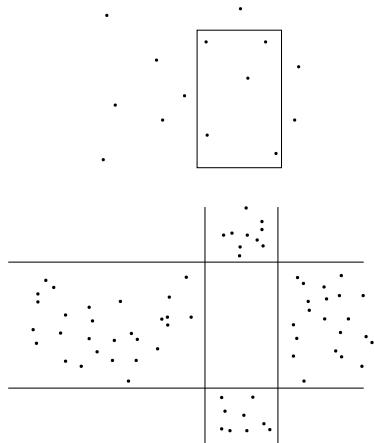
---

- Goal. 2D range reporting with
  - $O(n \log n)$  space and  $O(\log n + \text{occ})$  query time or
  - $O(n)$  space and  $O(\sqrt{n} + \text{occ})$  query time.
- Solution in 4 steps.
  - Generalized 1D range reporting.
  - 2D range trees.
  - 2D range trees with bridges.
  - kD trees.

## Generalized 1D Range Reporting

---

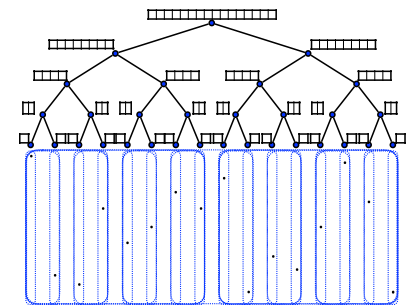
- Data structure.
  - Sorted array over x-coordinate
  - Sorted array over y-coordinate
- Report( $x_1, y_1, x_2, y_2$ ):
  - Compute all points  $R_x$  in x-range.
  - Compute all points  $R_y$  in y-range.
  - Return  $R_x \cap R_y$
- Time?



## 2D Range Trees

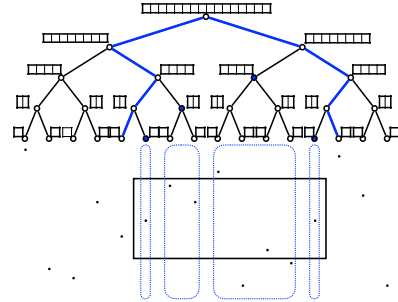
---

- Data structure.
  - Perfectly balanced binary tree over x-coordinate.
  - Each node  $v$  stores array of points below  $v$  sorted by y coordinate.
- Space?
  - $O(n \log n)$ .
- Preprocessing time.  $O(n \log n)$



## 2D Range Trees

- **Report**( $x_1, y_1, x_2, y_2$ ): Find paths to predecessor of  $x_1$  and successor of  $x_2$ .
  - At each **off-path node** do 1D query on  $y$ -range.
  - Return union of results.
- **Time?**
  - Predecessor + successor:  $O(\log n)$
  - $< 2 \log n$  1D queries:  $O(\log n + \text{occ in subrange})$  time per query.
  - $\Rightarrow$  total  $O(\log^2 n + \text{occ})$  time.



## 2D Range Reporting

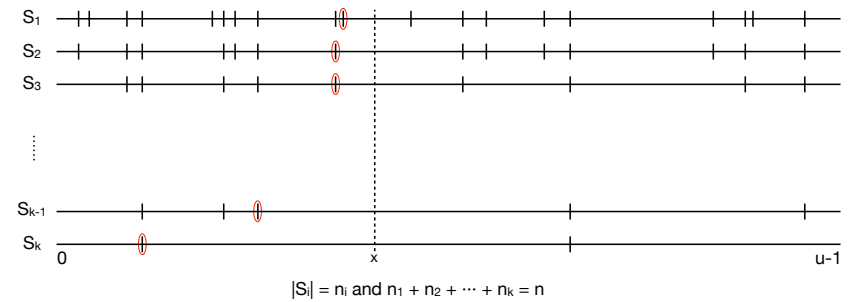
- **Theorem.** We can solve the 2D range reporting problem in
  - $O(n \log n)$  space.
  - $O(\log^2 n + \text{occ})$  time for queries.
  - $O(n \log n)$  preprocessing time.
- **Challenge.** Do we really need the  $\log^2 n$  term for queries? Can we get (optimal)  $O(\log n + \text{occ})$  instead?

## Range Reporting

- Range reporting problem
- 1D range reporting
- 2D range reporting
  - Range trees
  - Predecessor in nested sets
  - kD trees

## Predecessor in Nested Sets

- Predecessor problem in nested sets. Let  $S = \{S_1, S_2, \dots, S_k\}$  be a family of sets from universe  $U$  such that  $U \supseteq S_1 \supseteq S_2 \supseteq \dots \supseteq S_k$ .
  - predecessor( $x$ ): return the predecessor of  $x$  in each of  $S_1, S_2, \dots, S_k$ .

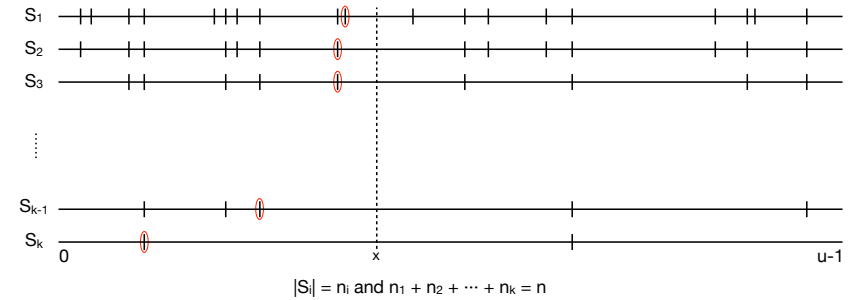


## Predecessor in Nested Sets

- **Goal.** Predecessor in nested sets with  $O(n)$  space and  $O(\log n + k)$  query time.
- **Solution in 3 steps.**
  - **Sorted arrays.** Slow and linear space.
  - **Tabulation.** Fast but too much space.
  - **Sorted arrays with bridges.** Fast and little space.

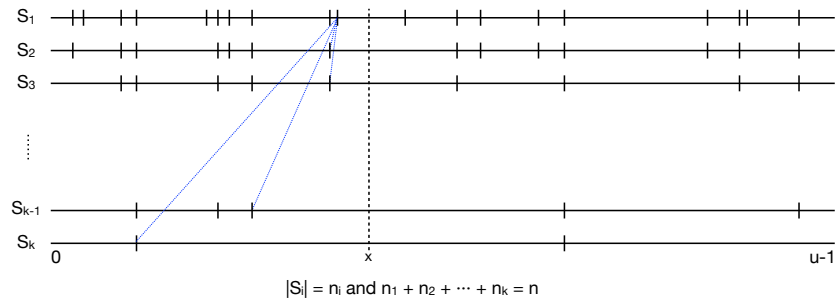
## Solution 1: Sorted Arrays

- **Data structure.** Sorted arrays for each set.
- **Predecessor(x):** Binary search in each array.
- **Time.**  $O(\log n_1 + \log n_2 + \dots + \log n_k) = O(k \log n)$
- **Space.**  $O(n)$



## Solution 2: Tabulation

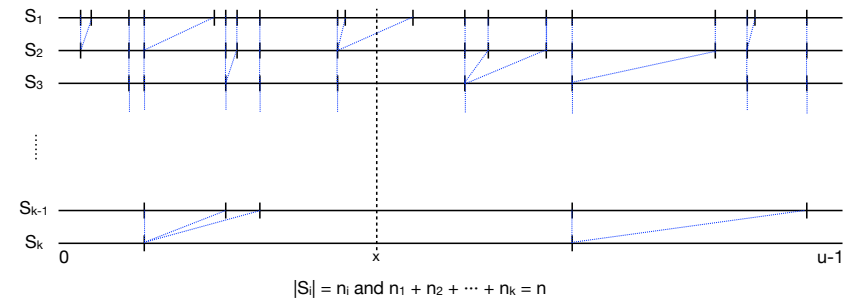
- **Data structure.** Sorted array on  $S_1$  + each entry stores  $k-1$  predecessors in  $S_2, \dots, S_k$ .
- **Predecessor(x):** Binary search in  $S_1$  array + report predecessors.
- **Time.**  $O(\log n_1 + k) = O(\log n + k)$
- **Space.**  $O(nk)$



- **Challenge.** Can we get the best of both worlds?

## Solution 3: Sorted Arrays with Bridges

- **Data structure.** Sorted arrays for each set + **bridges**.
- **Predecessor(x):** Binary search in  $S_1$  array + traverse bridges and report elements.
- **Time.**  $O(\log n_1 + k) = O(\log n + k)$
- **Space.**  $O(n)$

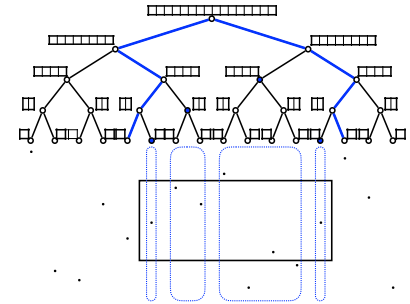


## Predecessor in Nested Sets

- **Theorem.** We can solve the predecessor in nested sets problem in
  - $O(n)$  space.
  - $O(\log n + k)$  query time.
  - $O(n \log n)$  preprocessing time.
- **Extensions.**
  - Predecessor  $\Rightarrow$  1D range reporting.
  - More tricks  $\Rightarrow$  works for non-nested sets. Called **fractional cascading**.
- **Challenge.** How can we use predecessor in nested sets for 2D range reporting?

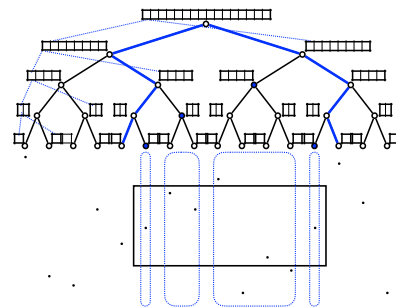
## 2D Range Reporting

- **Goal.** 2D range reporting in  $O(n \log n)$  space and  $O(\log n)$  time
- **Idea.** Consider node  $v$  with children  $v_l$  and  $v_r$ .
  - Arrays at  $v_l$  and  $v_r$  are **subsets** of array at  $v$ .
  - All searches in arrays during a query are on the **same** y-range.



## 2D Range Reporting

- **Data structure.** 2D range tree with bridges.
  - Each point in array at  $v$  stores bridges to arrays in  $v_l$  and  $v_r$ .
- **Report** $(x_1, y_1, x_2, y_2)$ : As 2D range tree query
  - Binary search in root array + traverse bridges for remaining 1D queries.
- **Time.**  $O(\log n + occ)$
- **Space.**  $O(n \log n)$
- **Preprocessing.**  $O(n \log n)$



## 2D Range Reporting

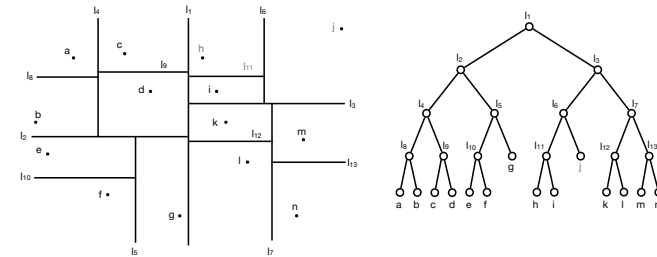
- **Theorem.** We can solve the 2D range reporting problem in
  - $O(n \log n)$  space
  - $O(\log n + occ)$  time for queries.
  - $O(n \log n)$  preprocessing time.
- What can we do with only linear space?

## Range Reporting

- Range reporting problem
- 1D range reporting
- 2D range reporting
  - Range trees
  - Predecessor in nested sets
  - kD trees

## kD Trees

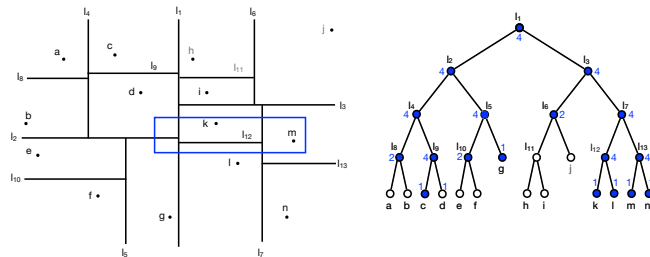
- The 2D tree ( $k = 2$ ).
  - A balanced binary tree over point set  $P$ .
  - Recursively partition  $P$  into rectangular regions containing (roughly) same number of points. Partition by alternating horizontal and vertical lines.
  - Each node in tree stores region and line.



- 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.  $O(\sqrt{n})$

## kD trees

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

## 2D Range Reporting

---

- **Theorem.** We can solve 2D range reporting in either
  - $O(n \log n)$  space and  $O(\log n + \text{occ})$  query time
  - $O(n)$  space and  $O(\sqrt{n} + \text{occ})$  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
- 2D range reporting
  - Range trees
  - Predecessor in nested sets
  - kD trees