OI-OPPA. European Social Fund
Prague & EU: We invest in your future.

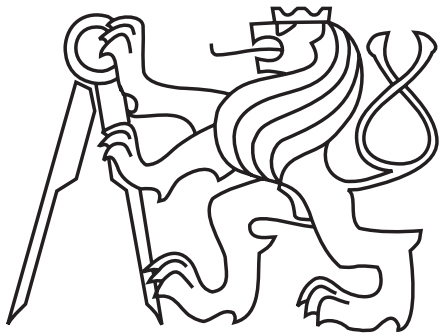
Range-Searching Using kd-Trees and Range Trees

Lecture for Computational Geometry (A4M39VG) Practice

Oskar Hollmann

Open Informatics
Faculty of Electrical Engineering
Czech Technical University in Prague

October 6, 2012



Geometric Searching Problems

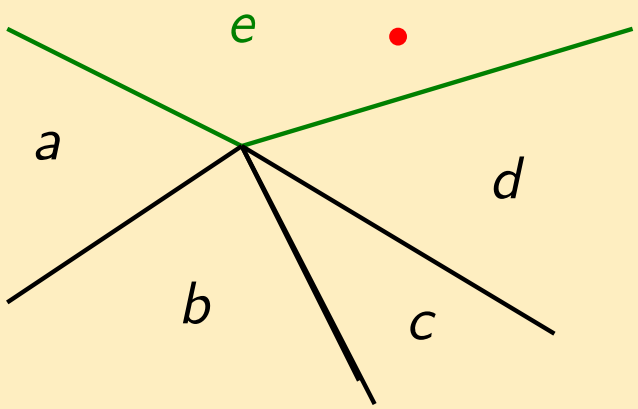
- can be solved using usual searching algorithms
- but we can use their geometric nature to make better algorithms
- 2 major categories in geometric searching:

Geometric Searching Problems

- can be solved using usual searching algorithms
- but we can use their geometric nature to make better algorithms
- 2 major categories in geometric searching:

Location Problems

Which region holds the point?



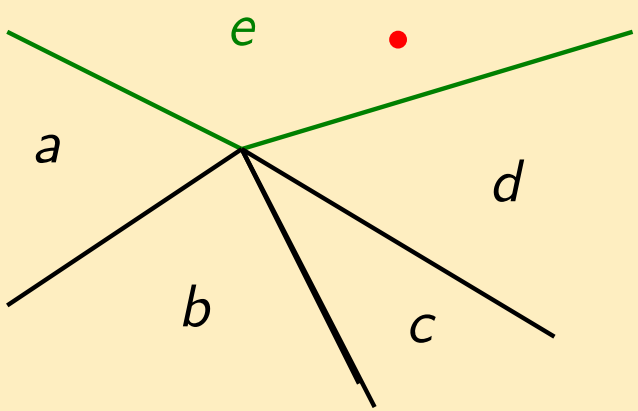
Answer: Region *e*.

Geometric Searching Problems

- can be solved using usual searching algorithms
- but we can use their geometric nature to make better algorithms
- 2 major categories in geometric searching:

Location Problems

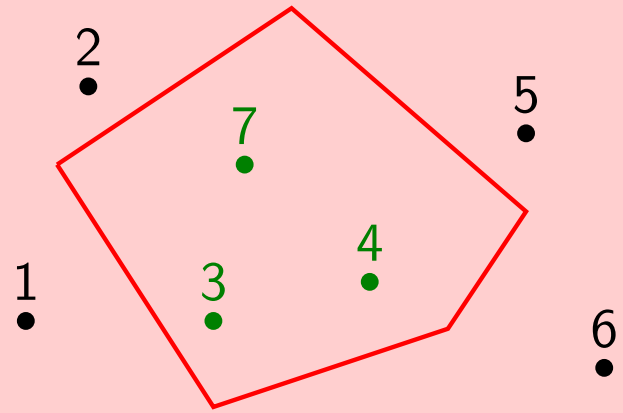
Which region holds the point?



Answer: Region e.

Range-Search Problems

Which points are in the region?



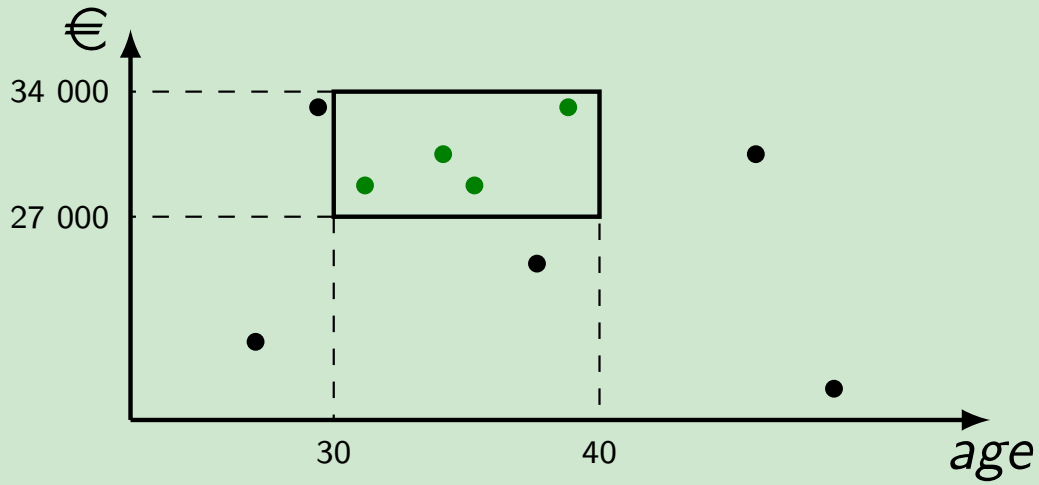
Answer: Points $3, 4, 7$.

Range-Search Problems

- set of points $P = (p_1, \ldots, p_n)$, domain (or range) D
- interesting are the algorithms for repeated range-searching
  - the set of points is fixed, different domains are queried
- cases with hyperrectangular domain are well-known
- applications in GIS, statistics or databases

Example: Database Query (Preparata and Shamos, 1985, p. 70)

"How many employees whose age is between 30 and 40 earn salary between €27 000 and €34 000?"

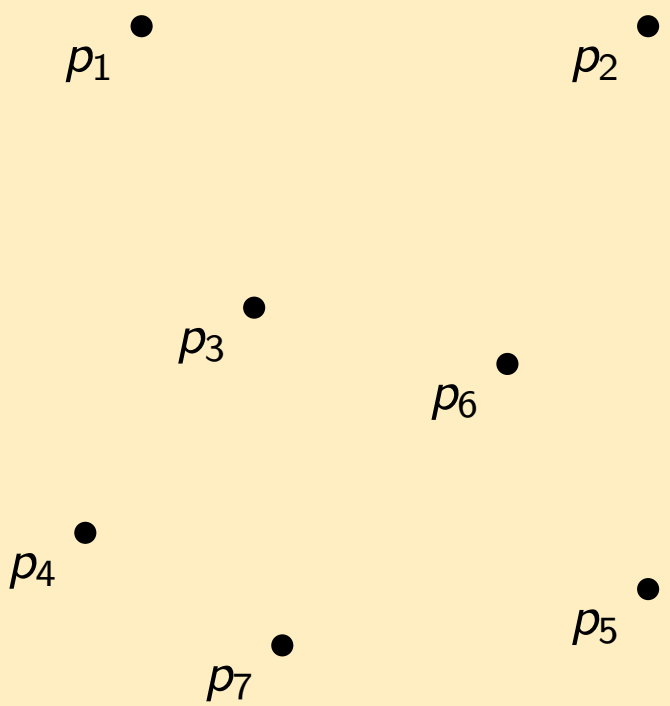


Using kd-Trees to Solve Range-Search Problems

- one-time searches $\rightarrow$ not interesting
- repeated searches $\rightarrow$ room for more sophisticated algorithms
- for the price of some preprocessing time, queries can be speeded up
- one of the methods are Kd-Trees, one needs to
  1. build the tree
  2. run searches
- demonstrated on 2d-Tree
  - the tree is build recursively using bisection, until each segment contains one point
  - nodes are the separating lines
  - leaves are the points
- we consider only rectangle ranges (simple case)

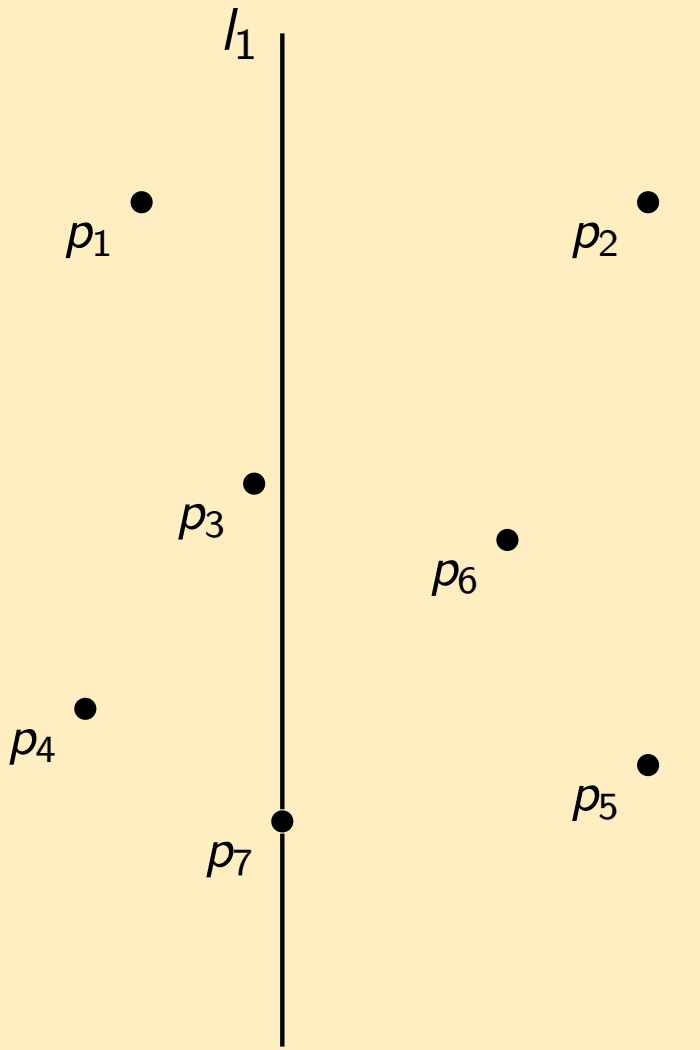
2d-Trees: Construction

Bisection



2d-Trees: Construction

Bisection

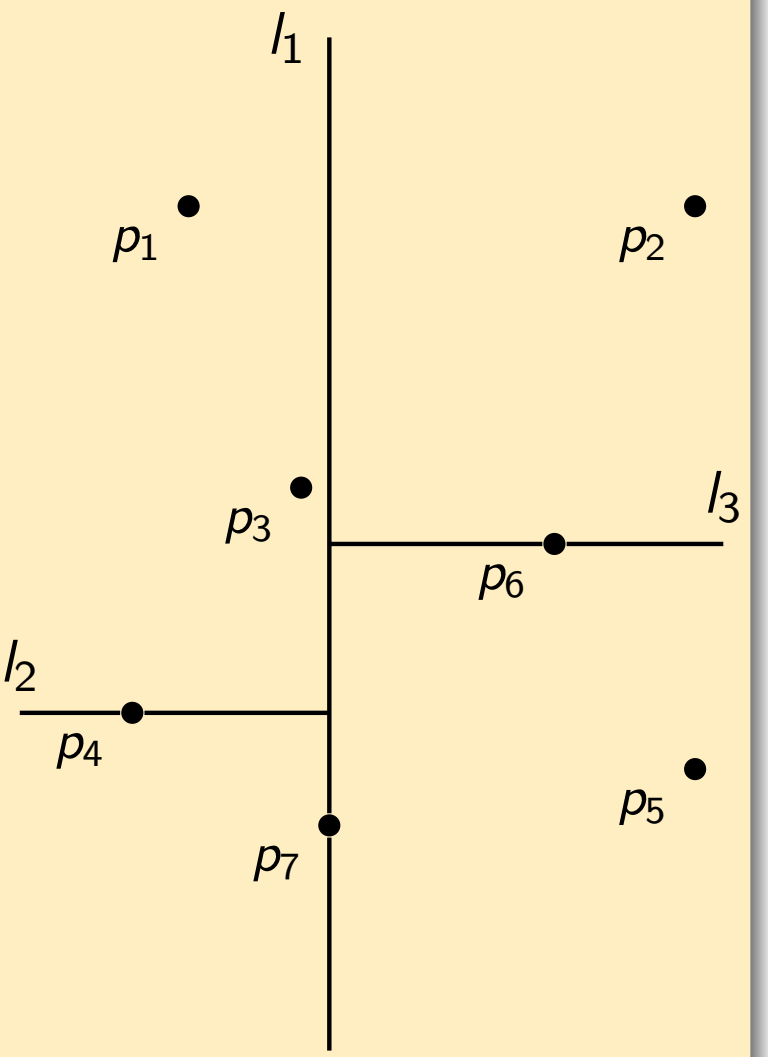


Tree

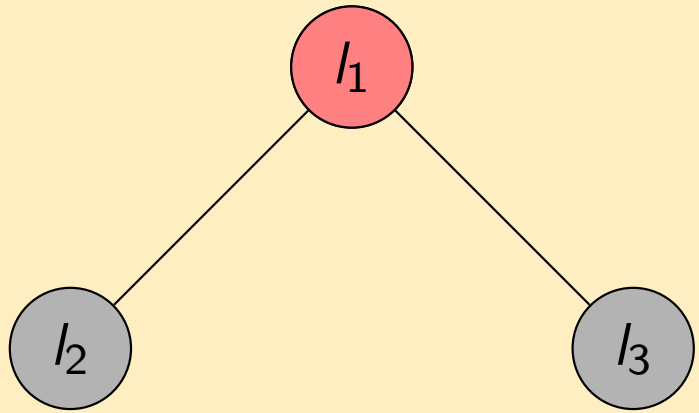
bisection by: x-axis y-axis

2d-Trees: Construction

Bisection



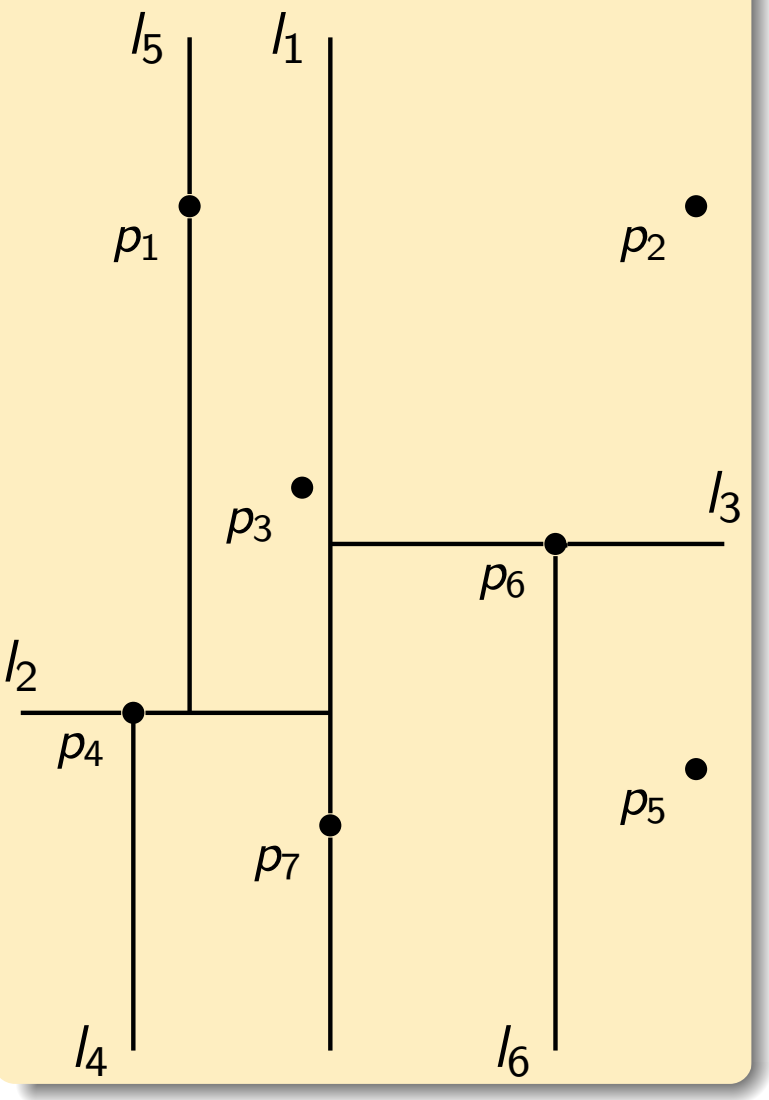
Tree



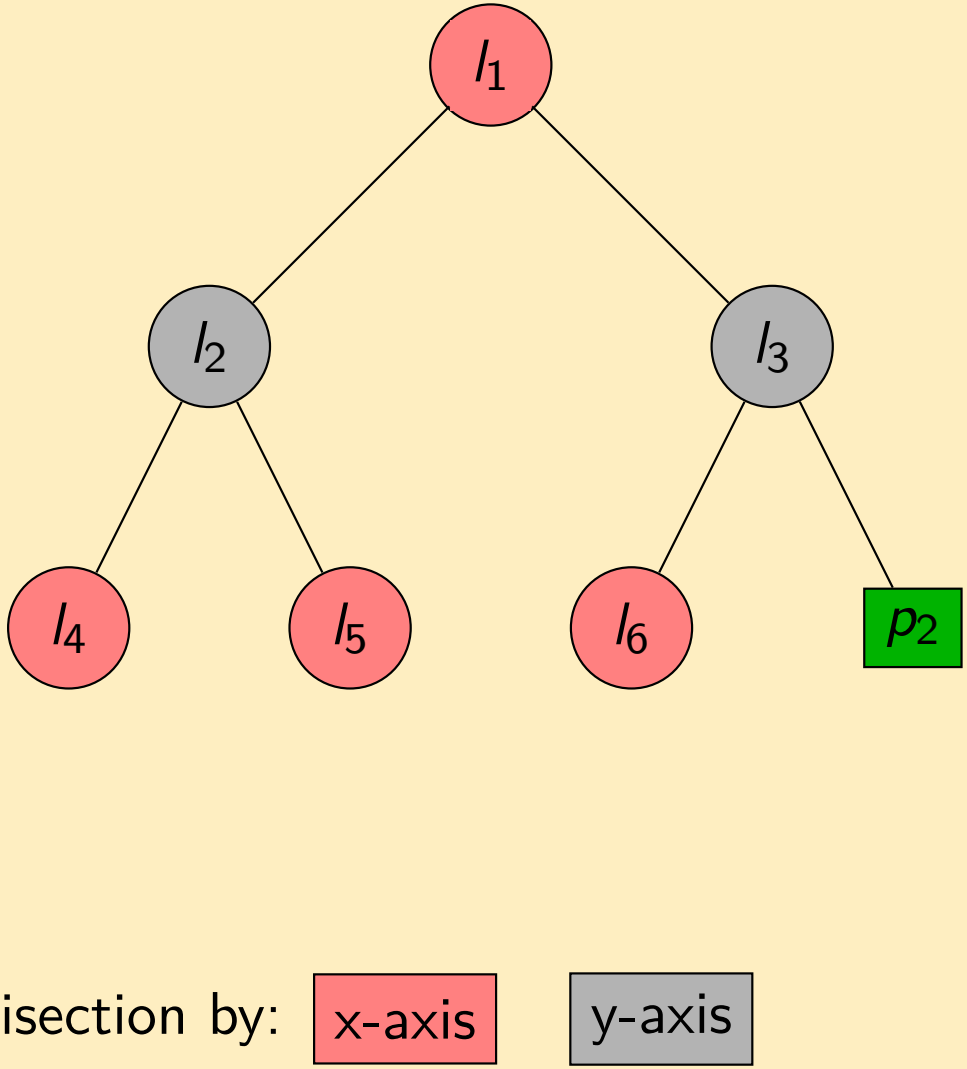
bisection by: x-axis y-axis

2d-Trees: Construction

Bisection

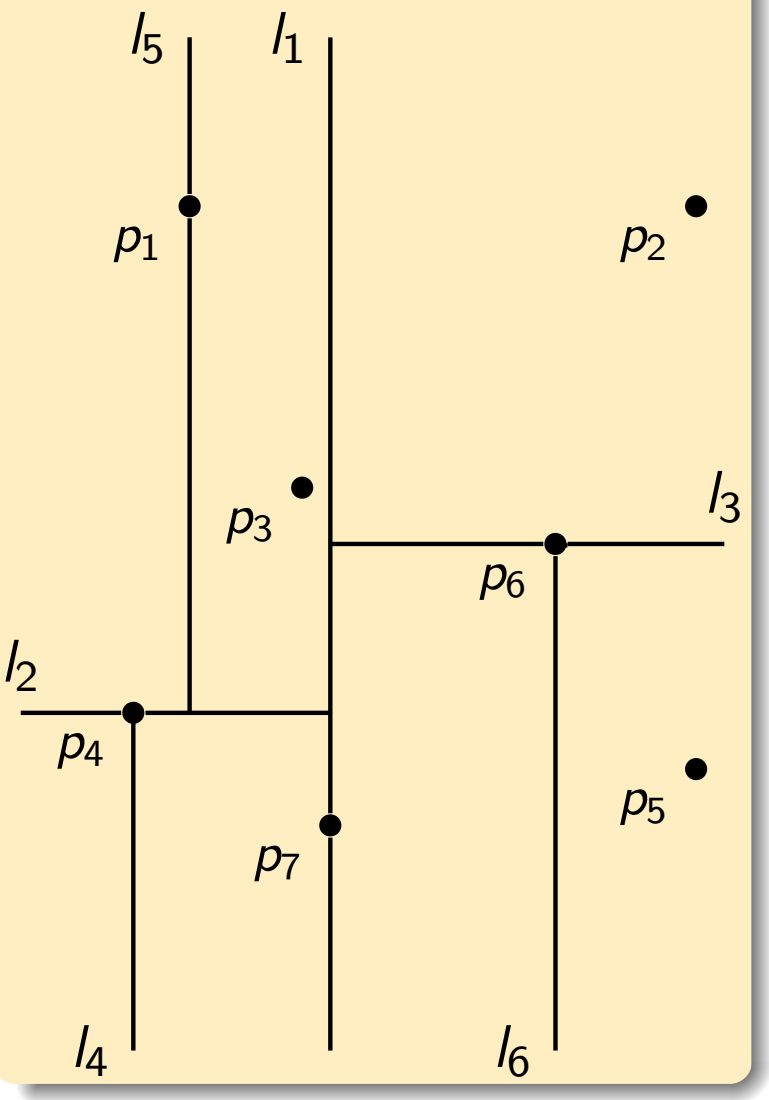


Tree

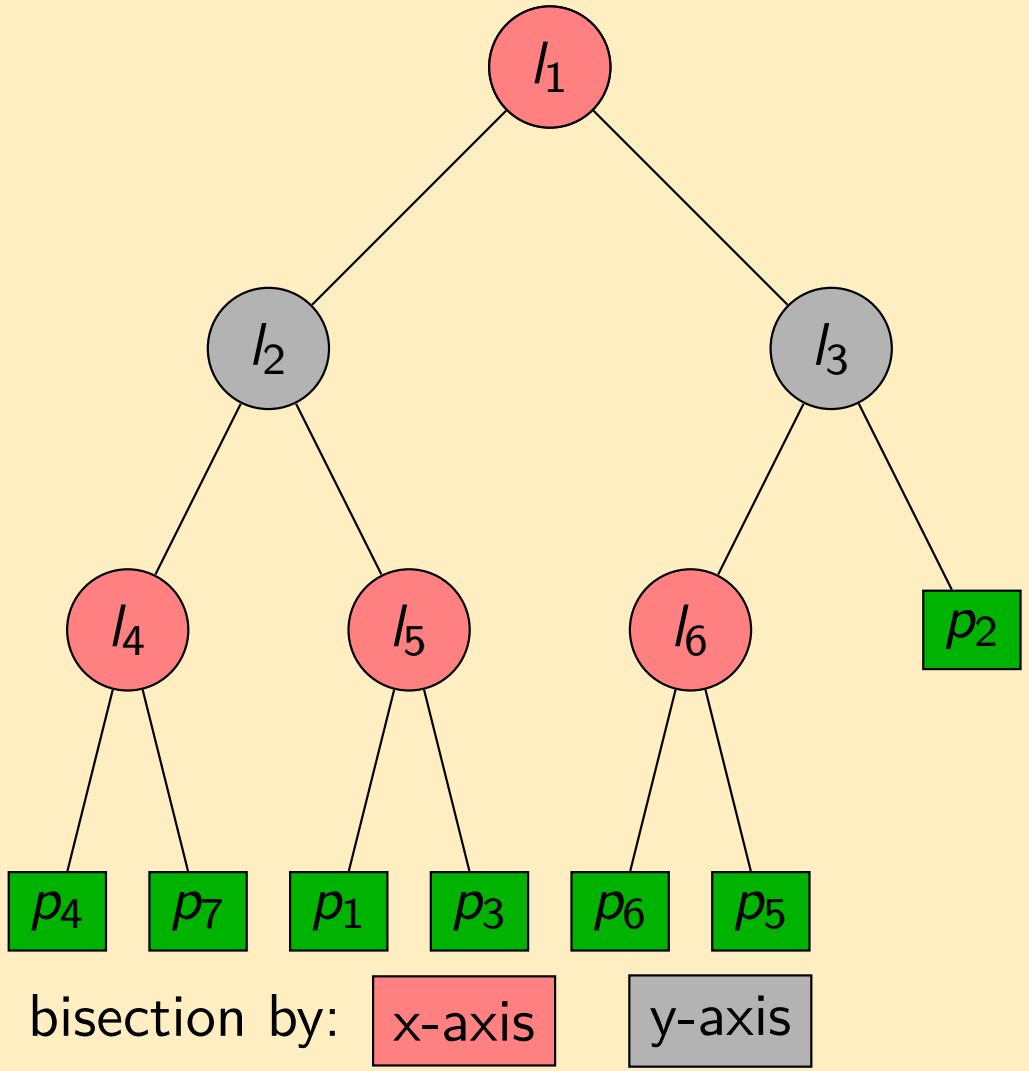


2d-Trees: Construction

Bisection



Tree



2d-Trees: Query

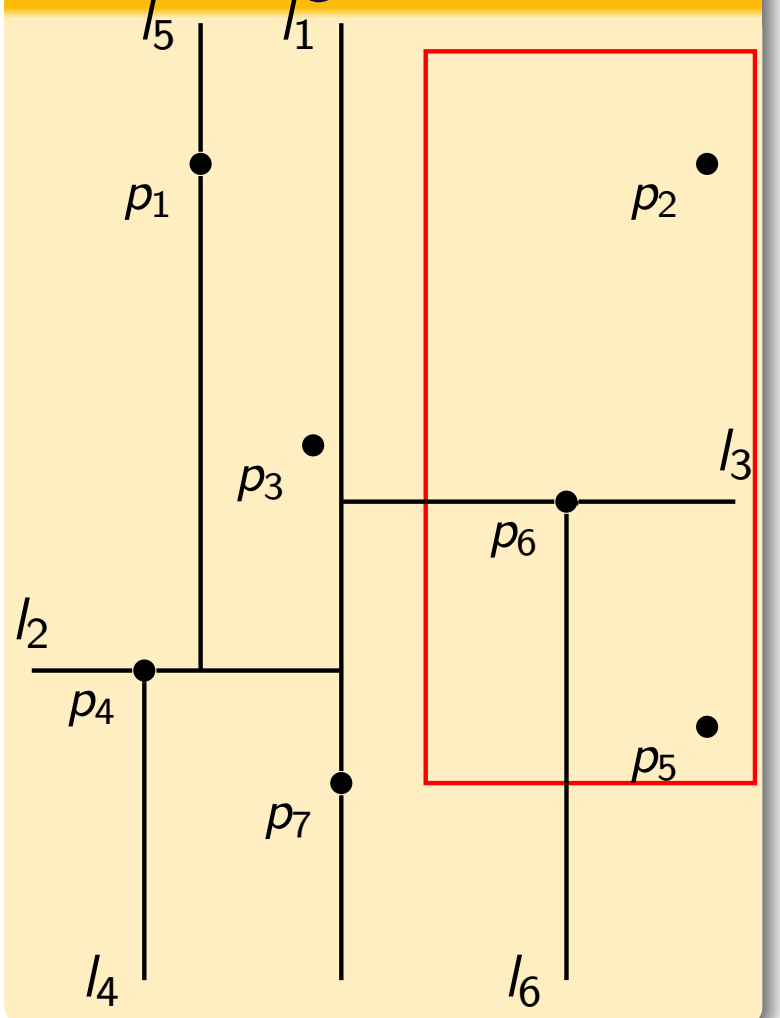
- someone gives us a rectangle $\rightarrow$ which points are in this range?
- rectangle can be even unbounded on one or more sides

Algorithm Search2dTree(Node n, Range r)

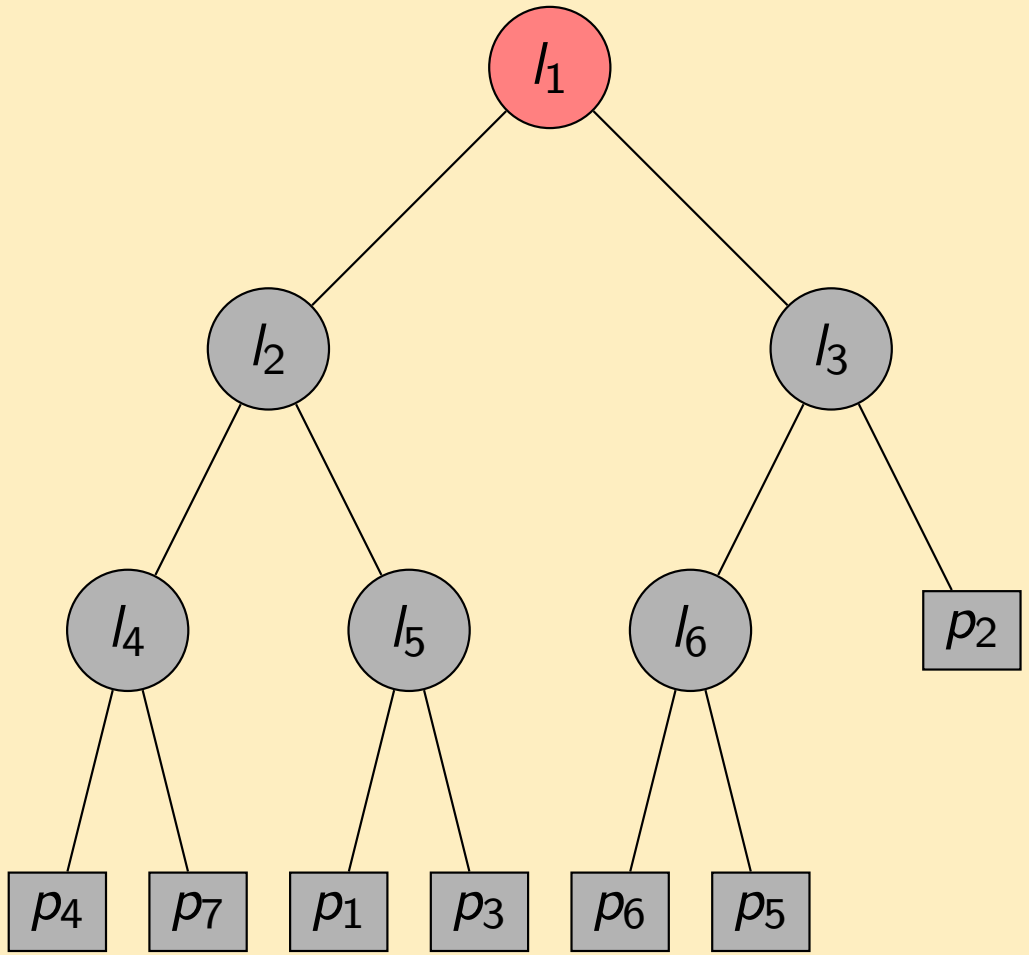
1. **if** region n is a leaf and is inside range r **then** report n
2. **else** compare the children regions $C = \{c_1, c_2\}$ and r
   - **if** c_i is contained in r **then** report all points in c_i
   - **if** c_i does not intersect r **then** report null
   - **if** c_i and r intersect **then** call Search2dTree(c_i, r)

2d-Trees: Query, Example 1

Given Range

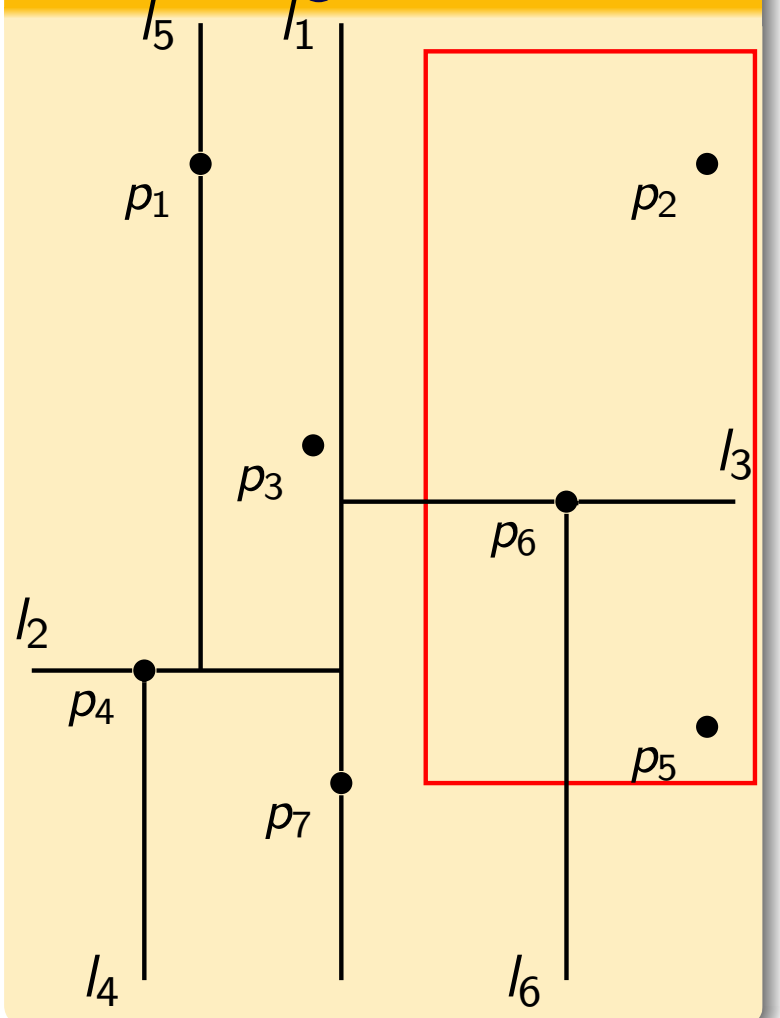


Tree

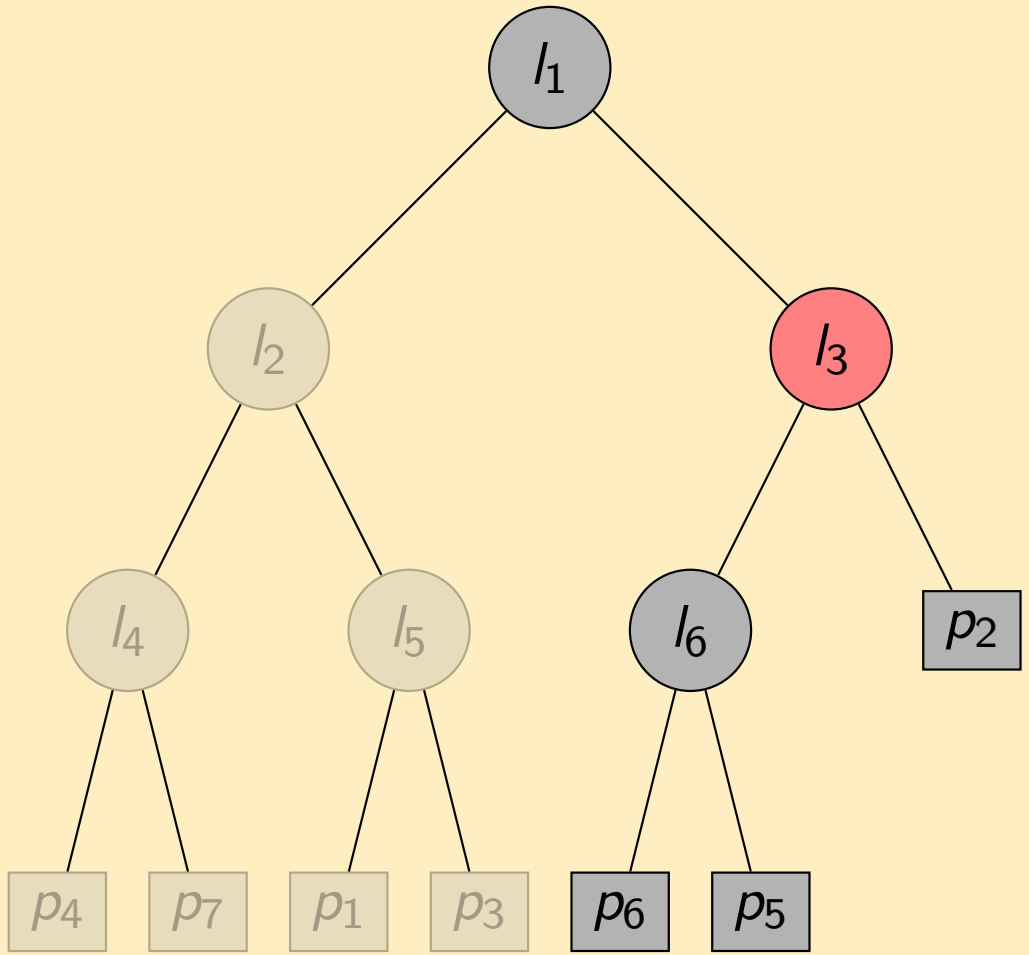


2d-Trees: Query, Example 1

Given Range

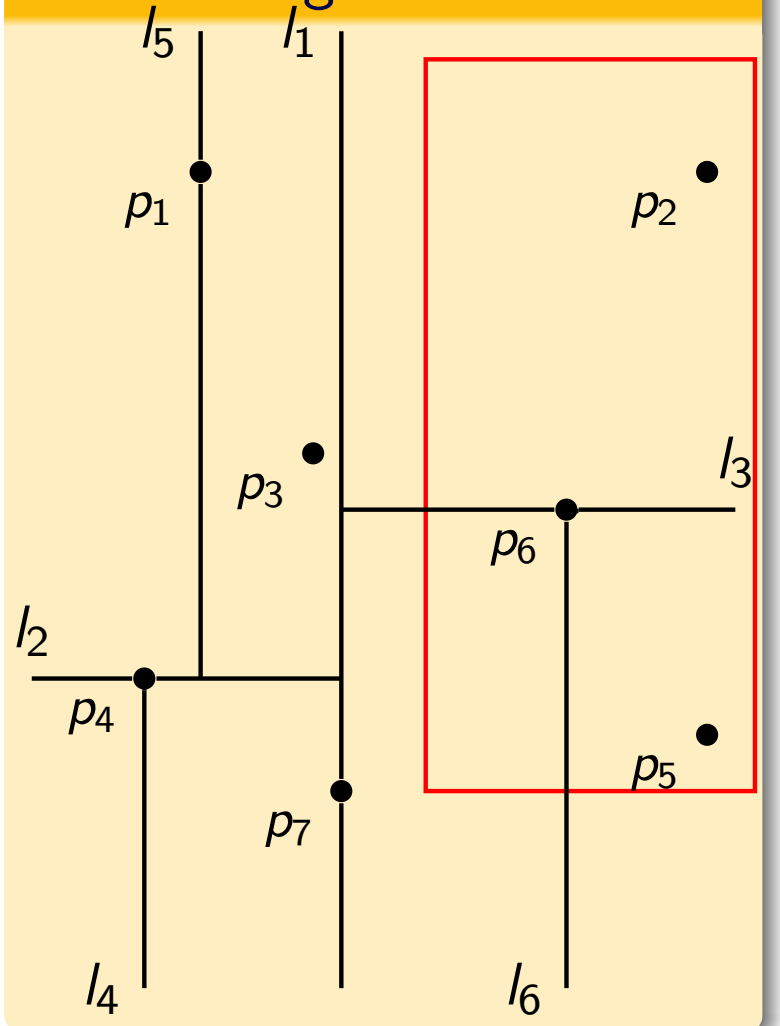


Tree

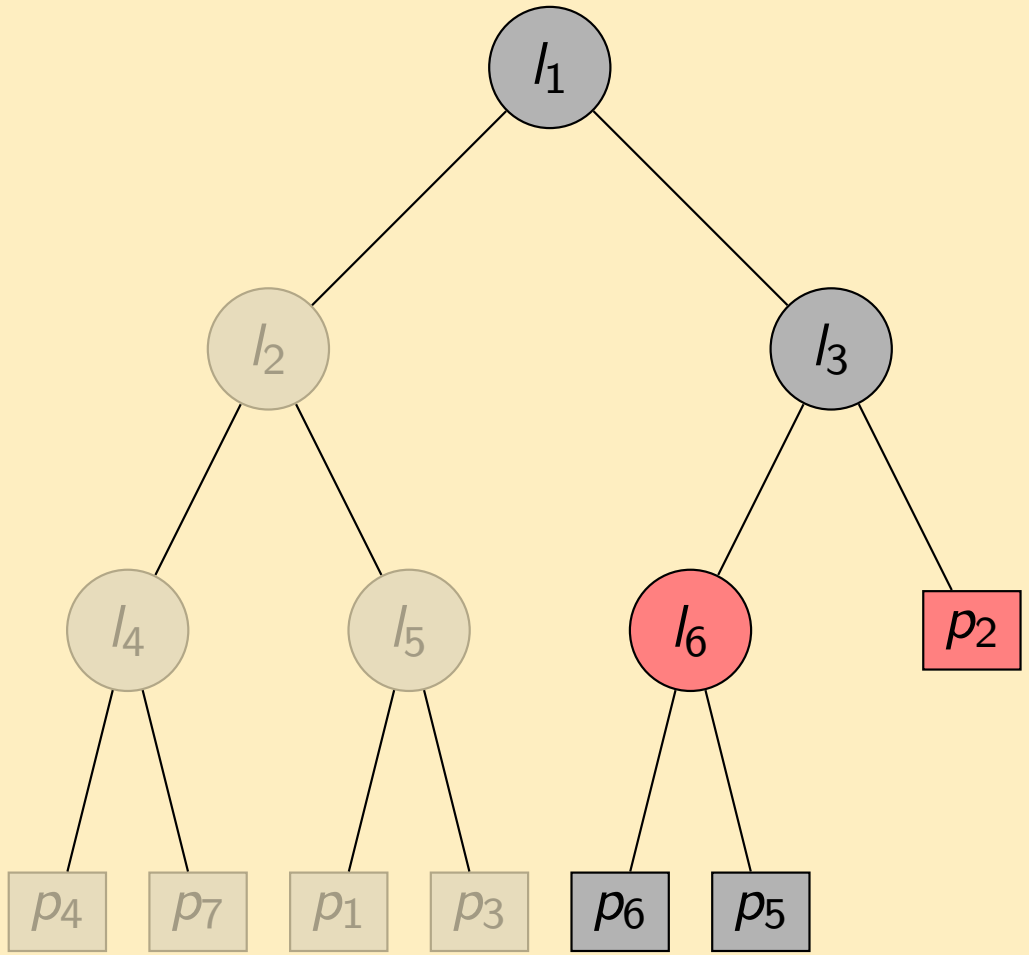


2d-Trees: Query, Example 1

Given Range

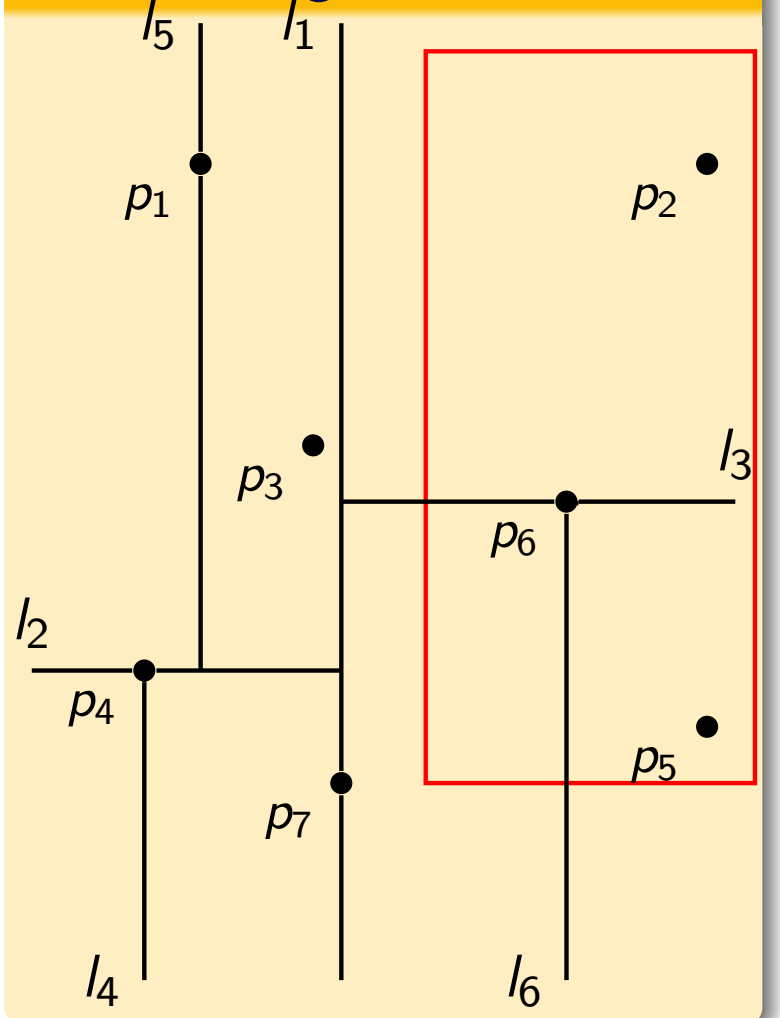


Tree

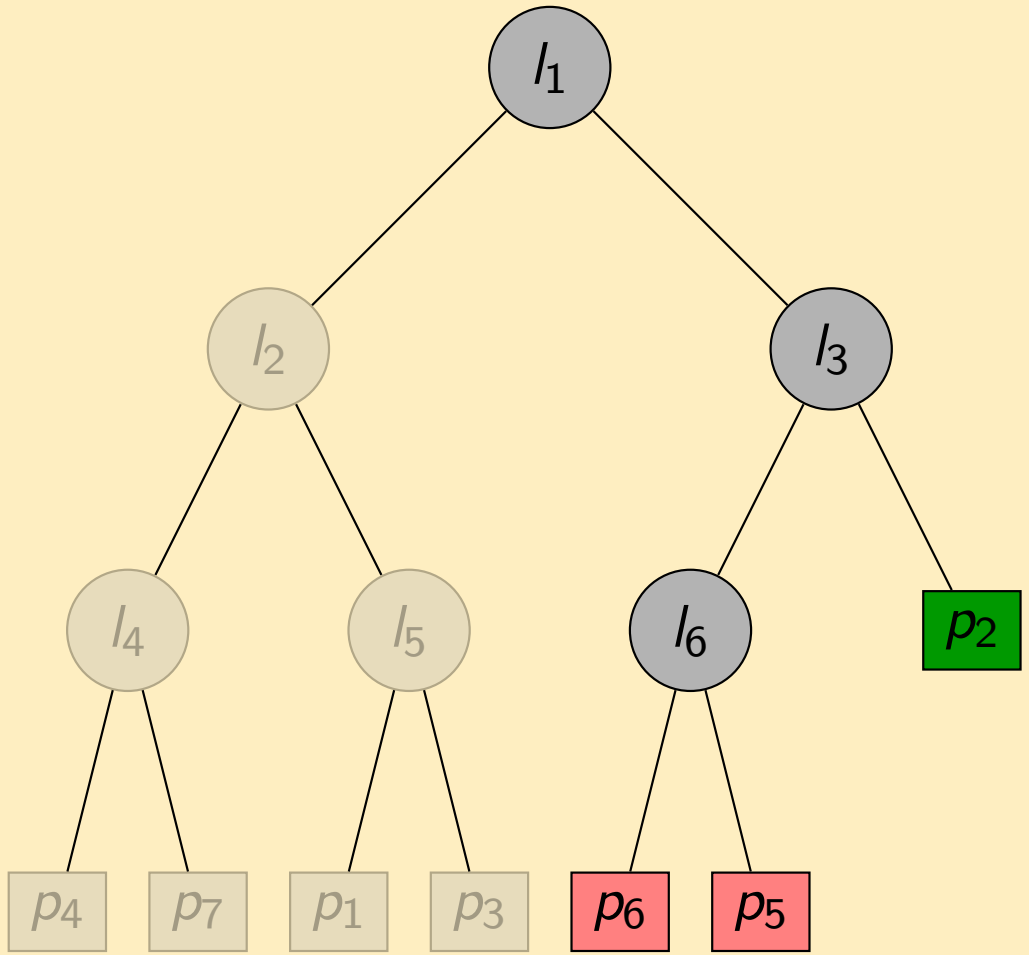


2d-Trees: Query, Example 1

Given Range

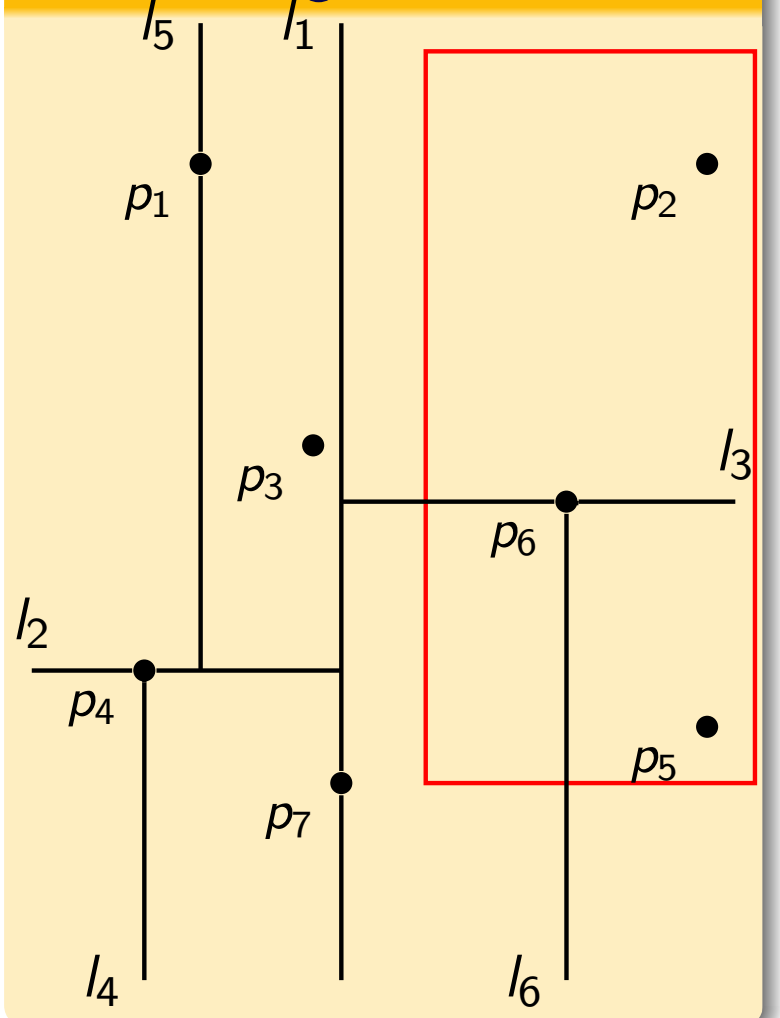


Tree

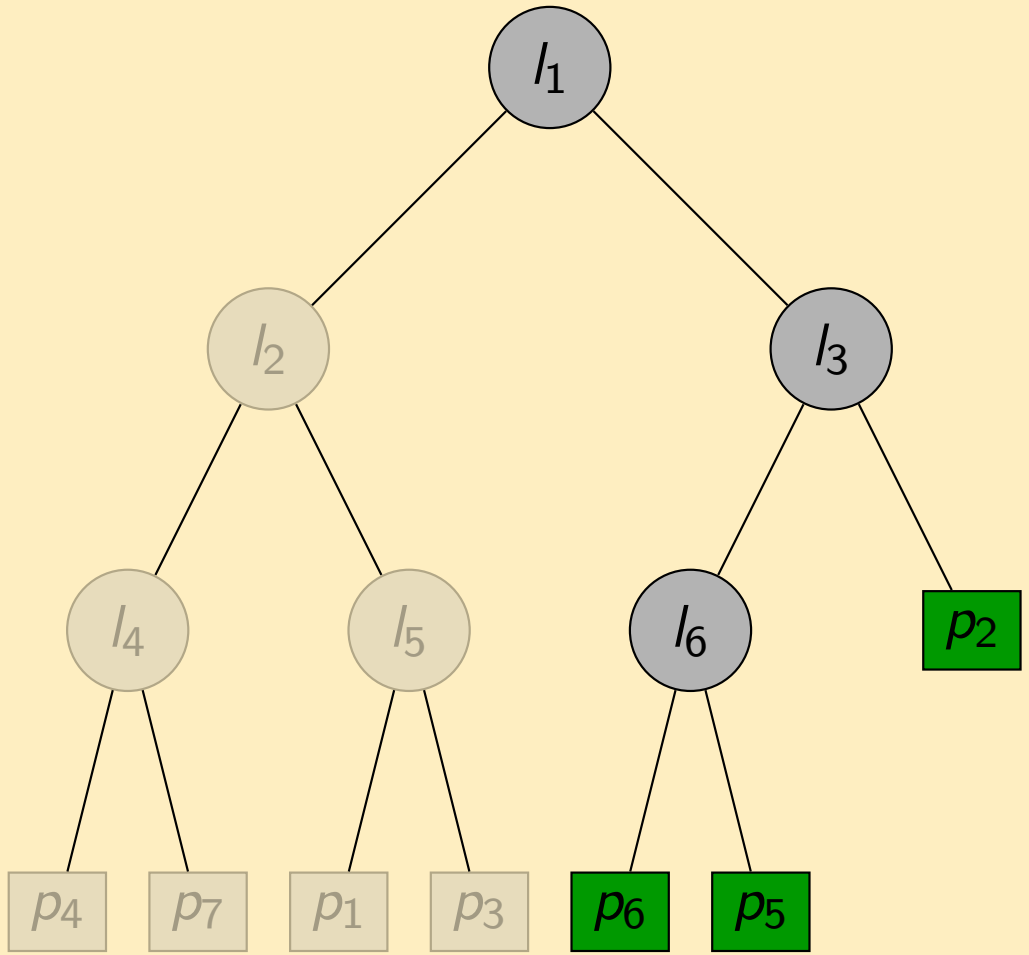


2d-Trees: Query, Example 1

Given Range

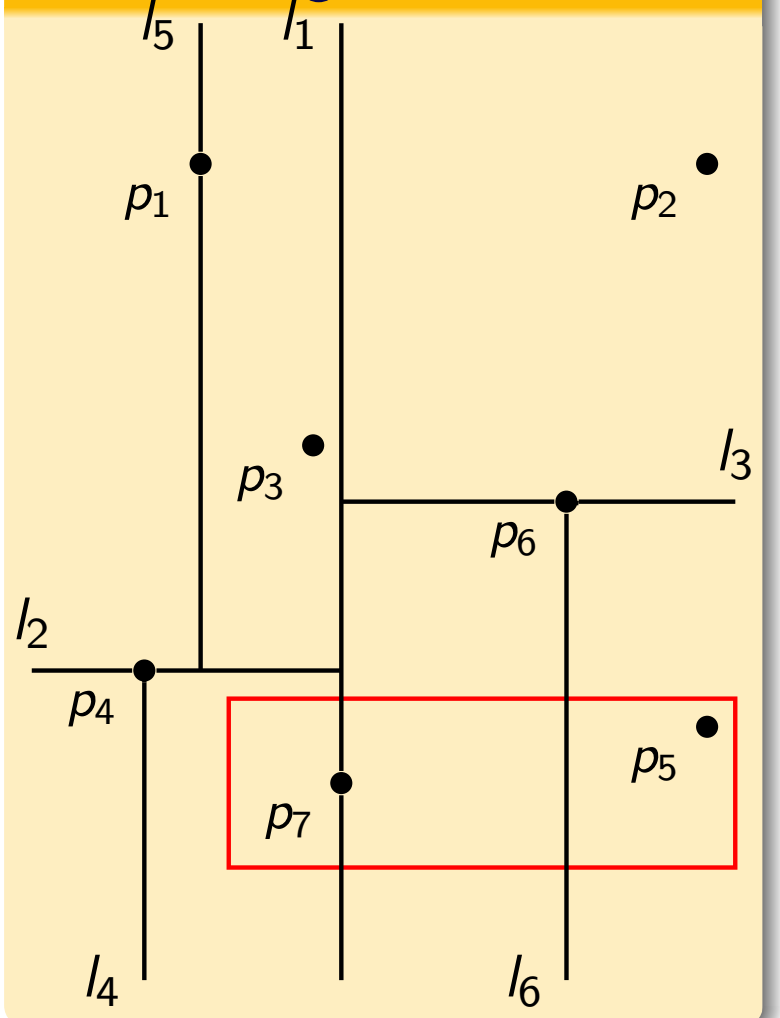


Tree

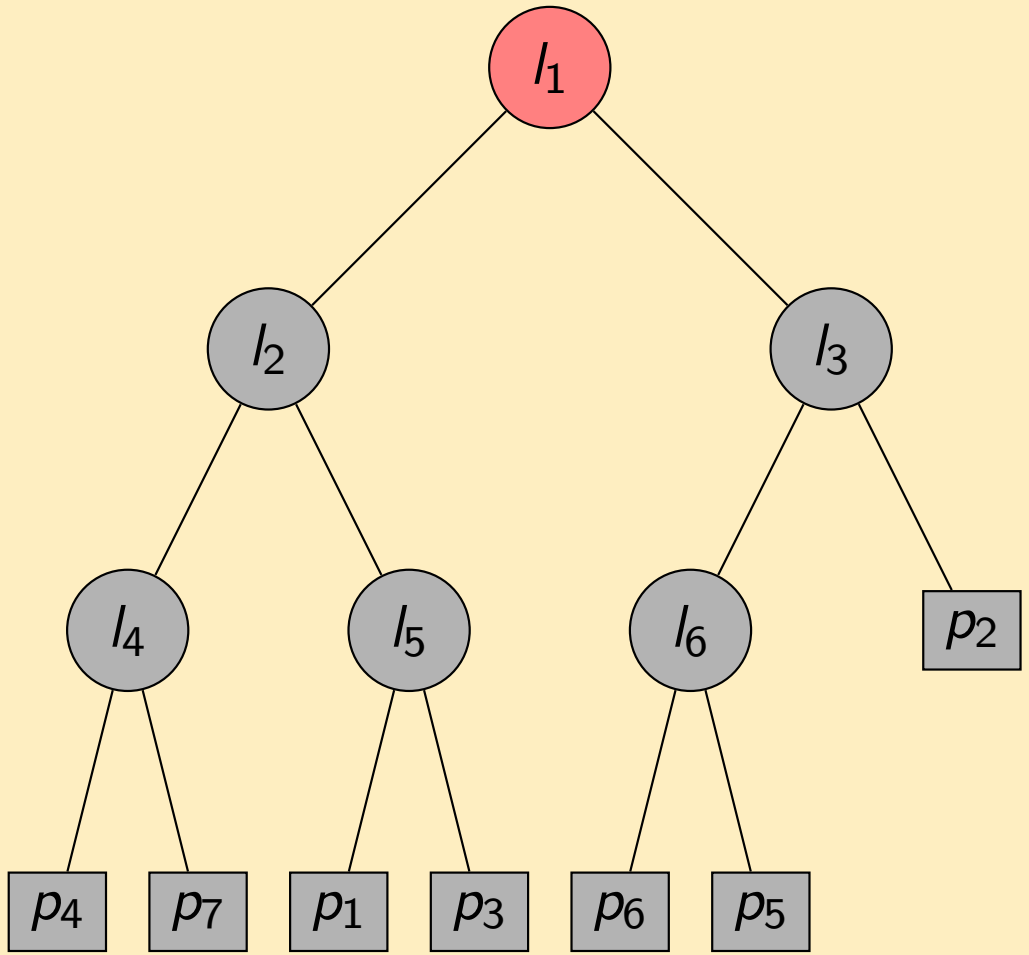


2d-Trees: Query, Example 2

Given Range

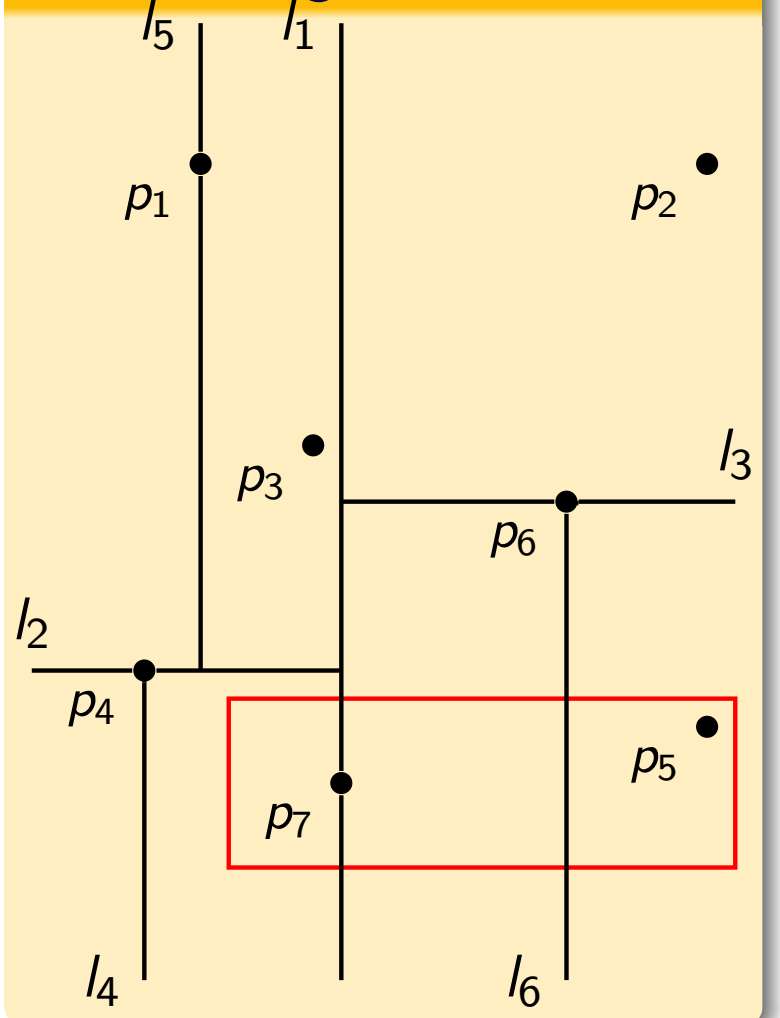


Tree

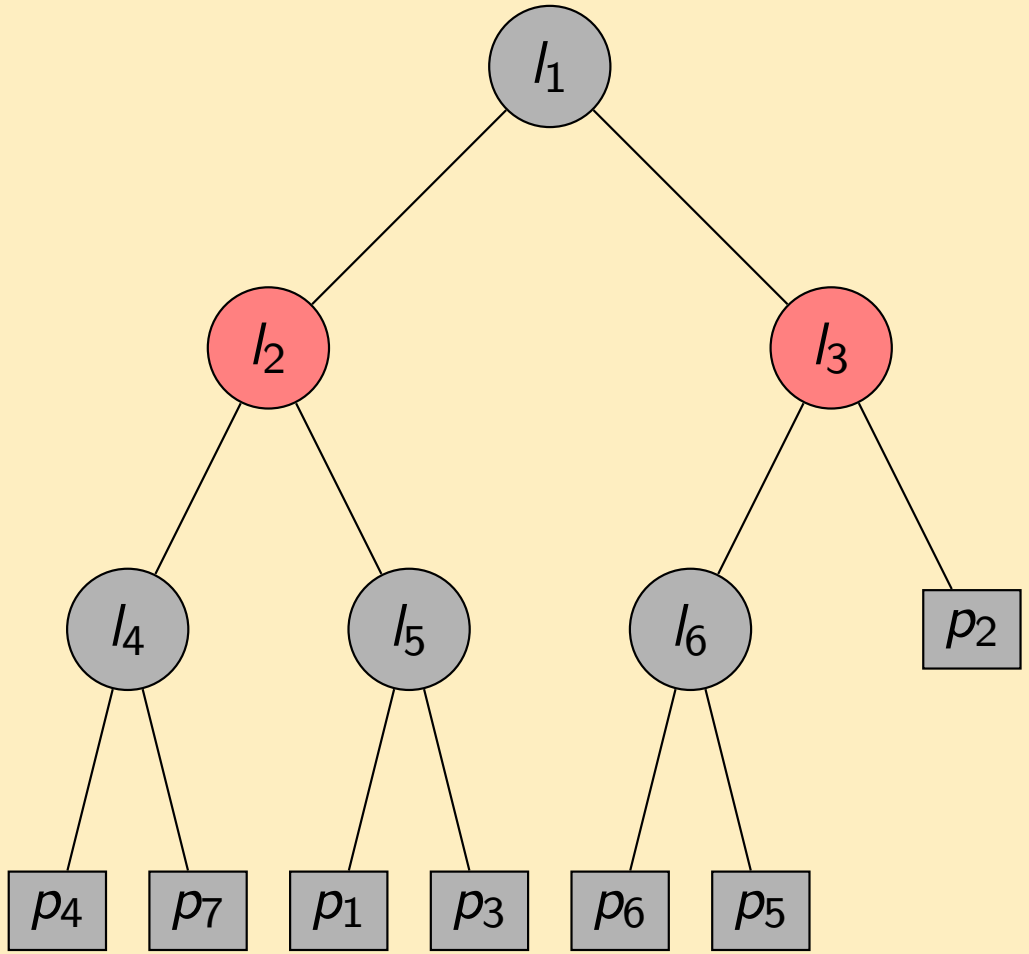


2d-Trees: Query, Example 2

Given Range

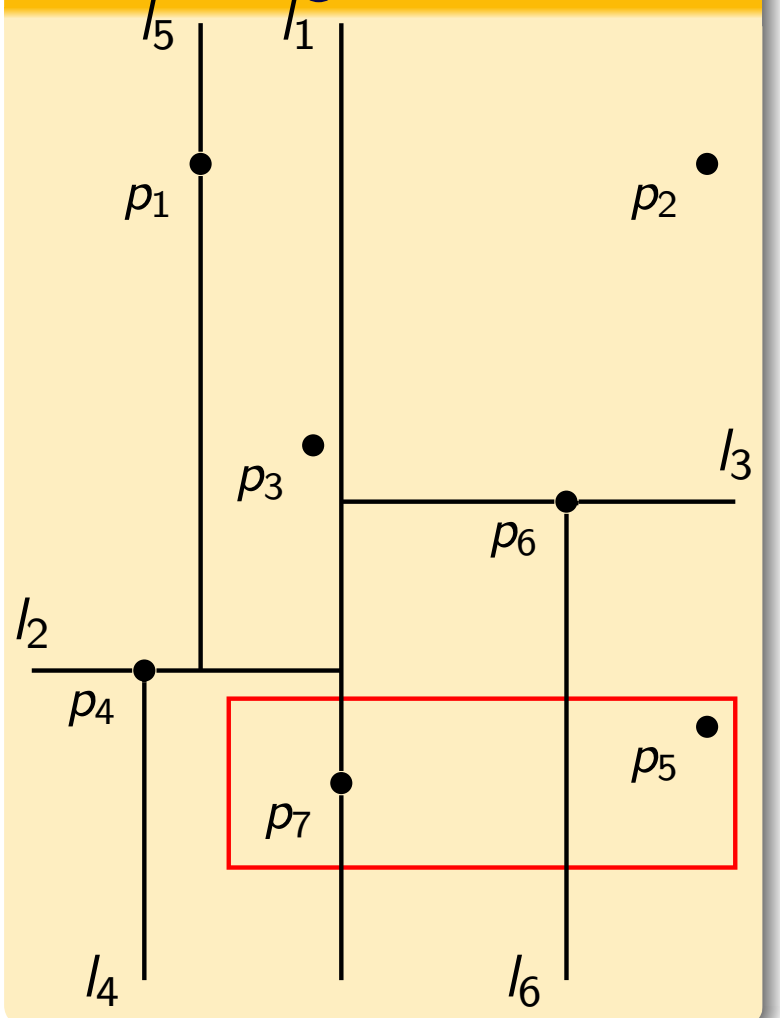


Tree

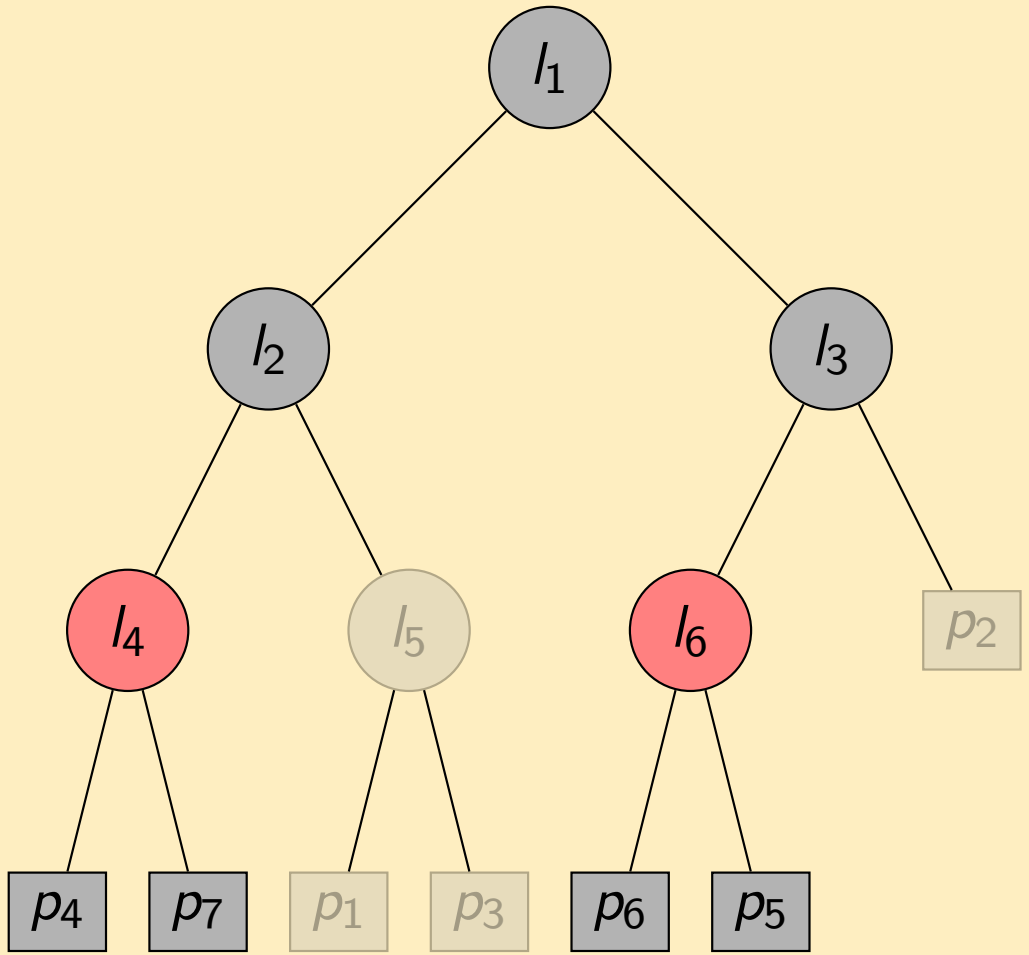


2d-Trees: Query, Example 2

Given Range

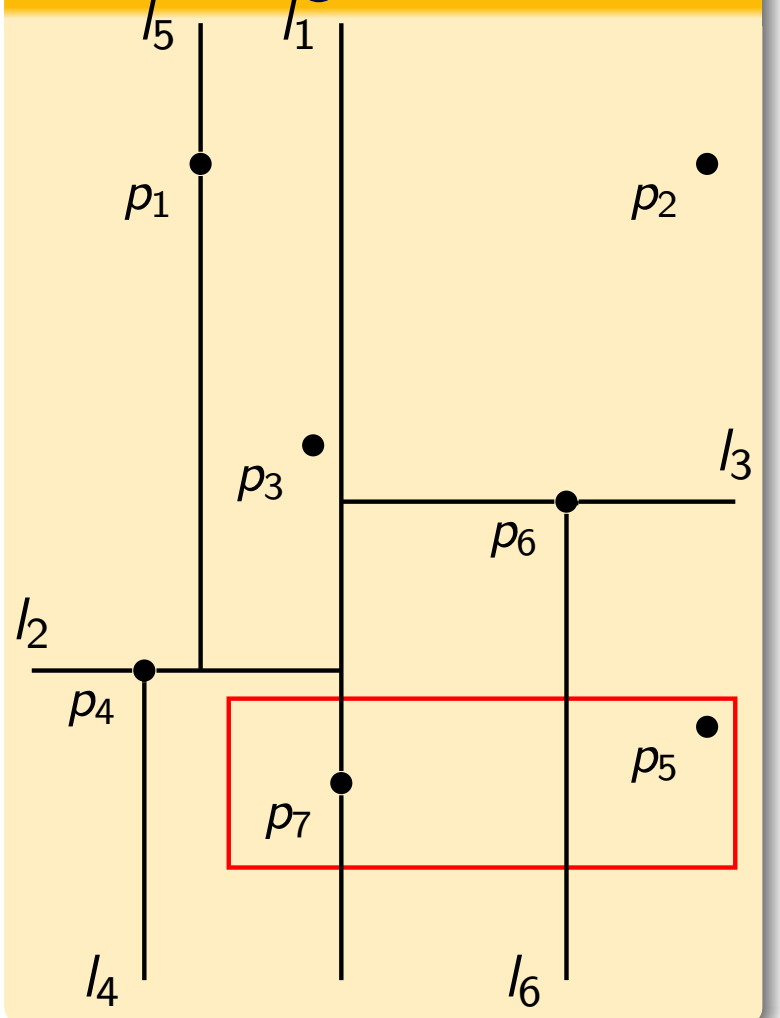


Tree

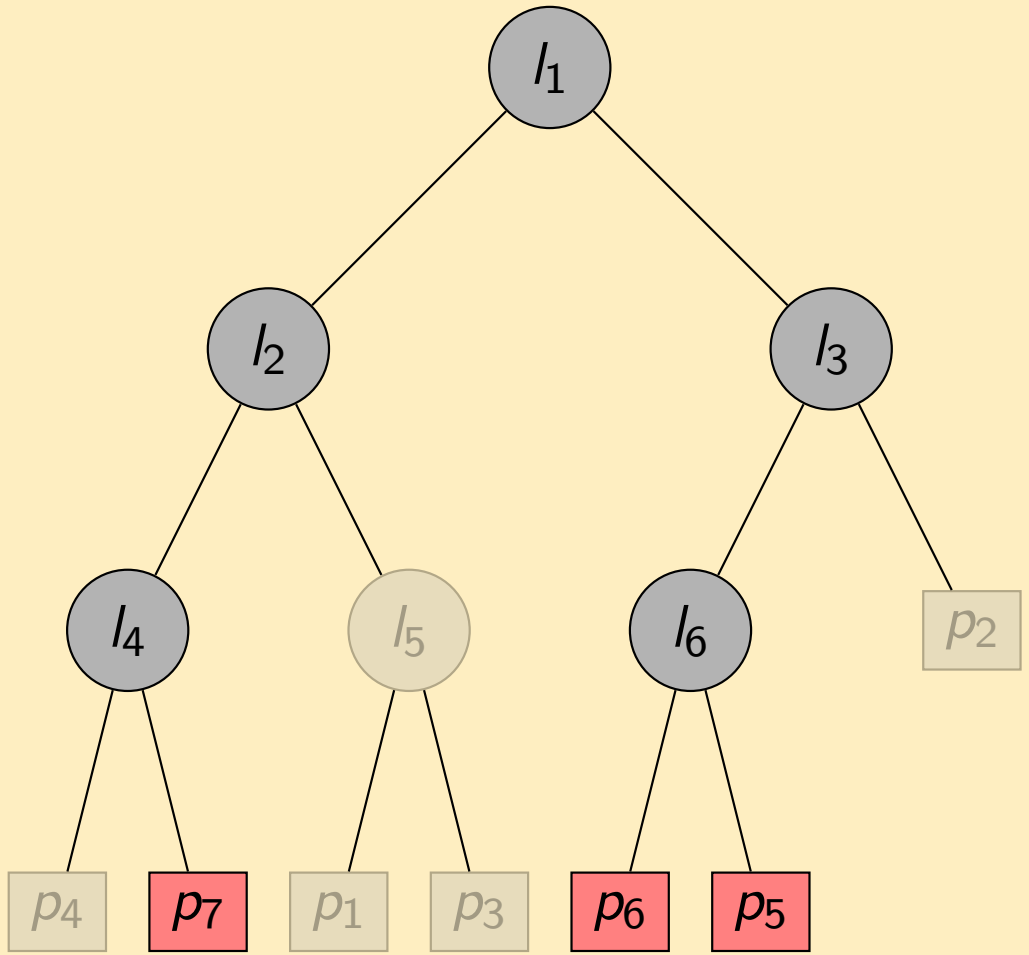


2d-Trees: Query, Example 2

Given Range

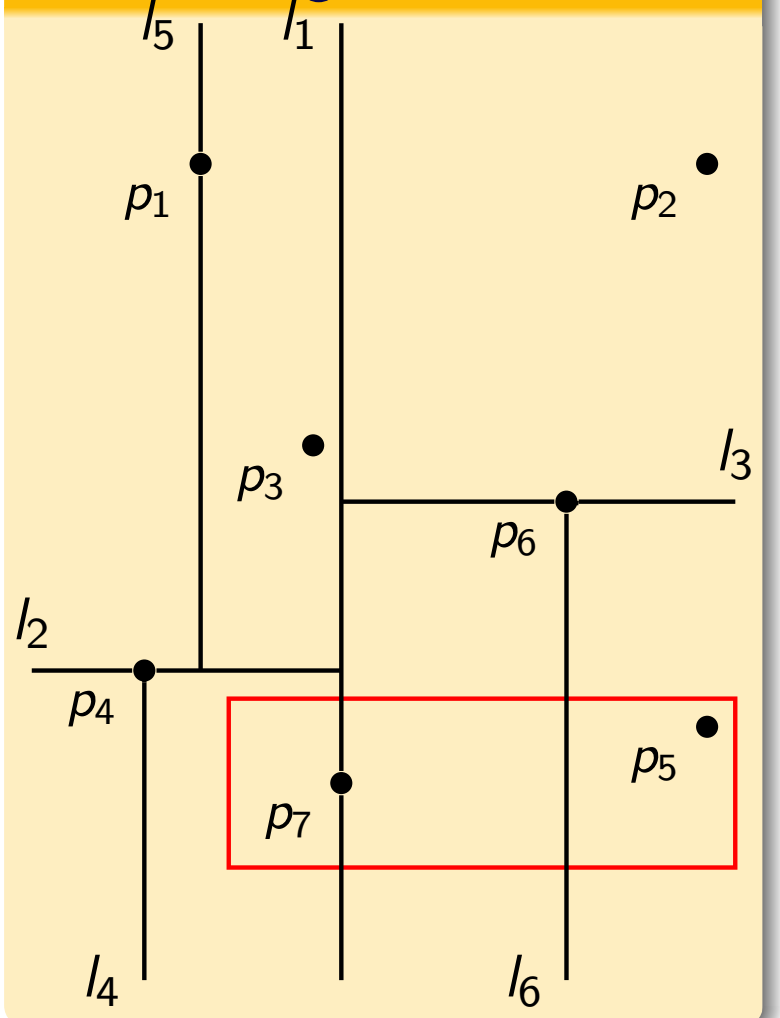


Tree

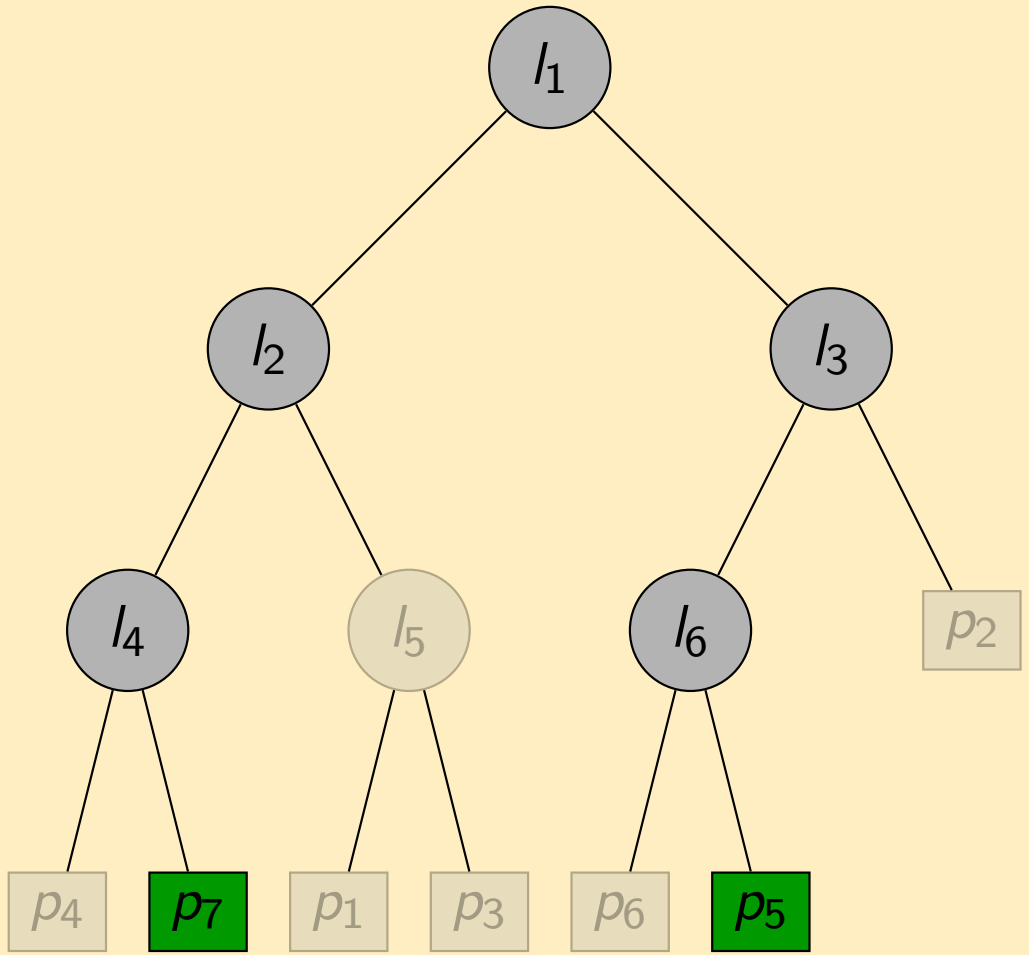


2d-Trees: Query, Example 2

Given Range



Tree



2d-Trees: Complexity

Building the 2d-Tree

Time complexity is expressed by recurrence:

$$T(n) = \begin{cases} O(1) & \text{if } n = 1 \\ O(n) + 2T([\frac{n}{2}]) & \text{if } n > 1 \end{cases}$$

Time complexity: $O(n \log n)$

Storage complexity: $O(n)$

Query in 2d-Tree

$$Q(n) = \begin{cases} O(1), & \text{if } n = 1 \\ O(n) + 2Q([\frac{n}{4}]), & \text{if } n > 1 \end{cases}$$

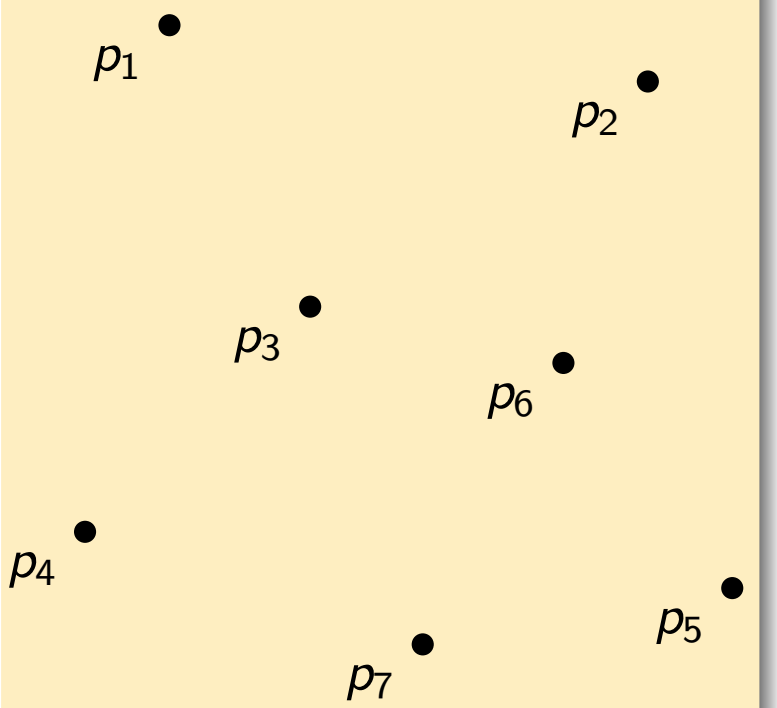
Time complexity: $O(\sqrt{n} + k)$, $k \ldots$ number of reported points

Using Range Trees to Solve Range Search Problems

- kd-tree search time can be improved upon for the cost of additional storage with Range Trees
- multidimensional Range Searching is composed of 1D sub-queries
  - in 2D, we construct ordinary binary search tree according to the x-axis, i.e. the first-level tree
  - for every sub-tree we construct the second-level tree according to the y-axis
- Figure: on the blackboard

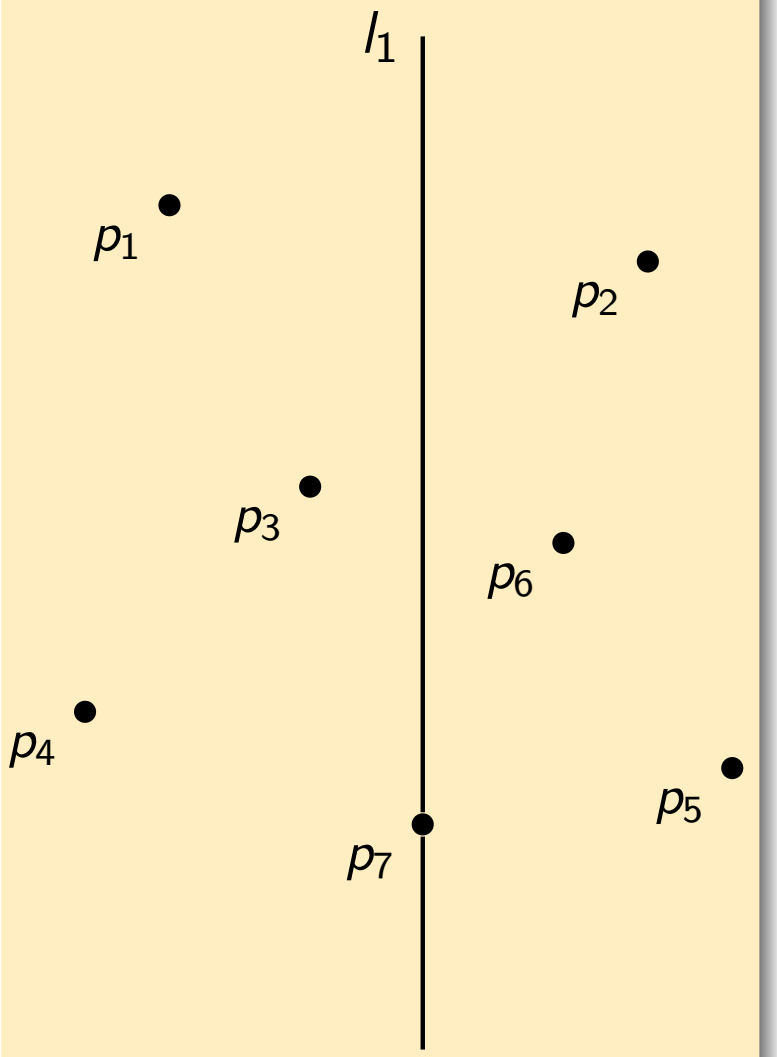
2D Range Tree: Building 1st-Level Tree

Bisection



2D Range Tree: Building 1st-Level Tree

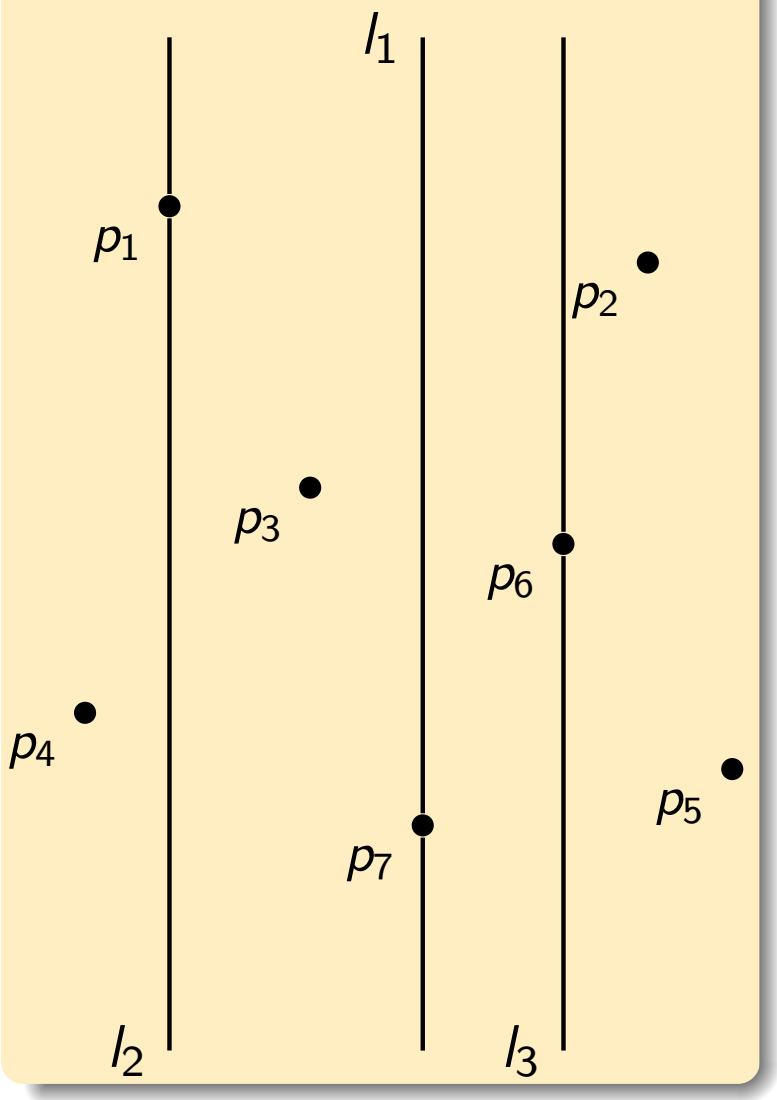
Bisection



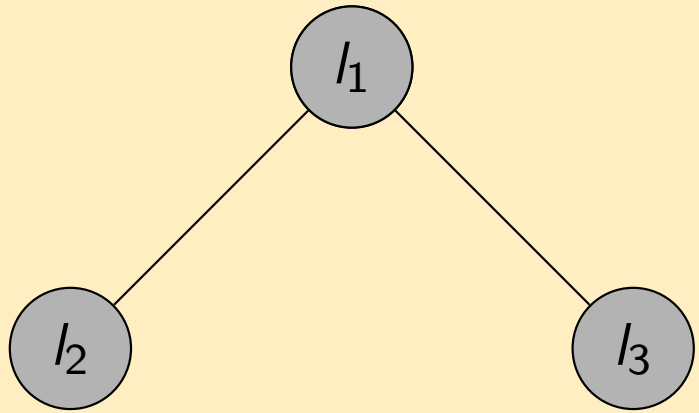
Tree

2D Range Tree: Building 1st-Level Tree

Bisection

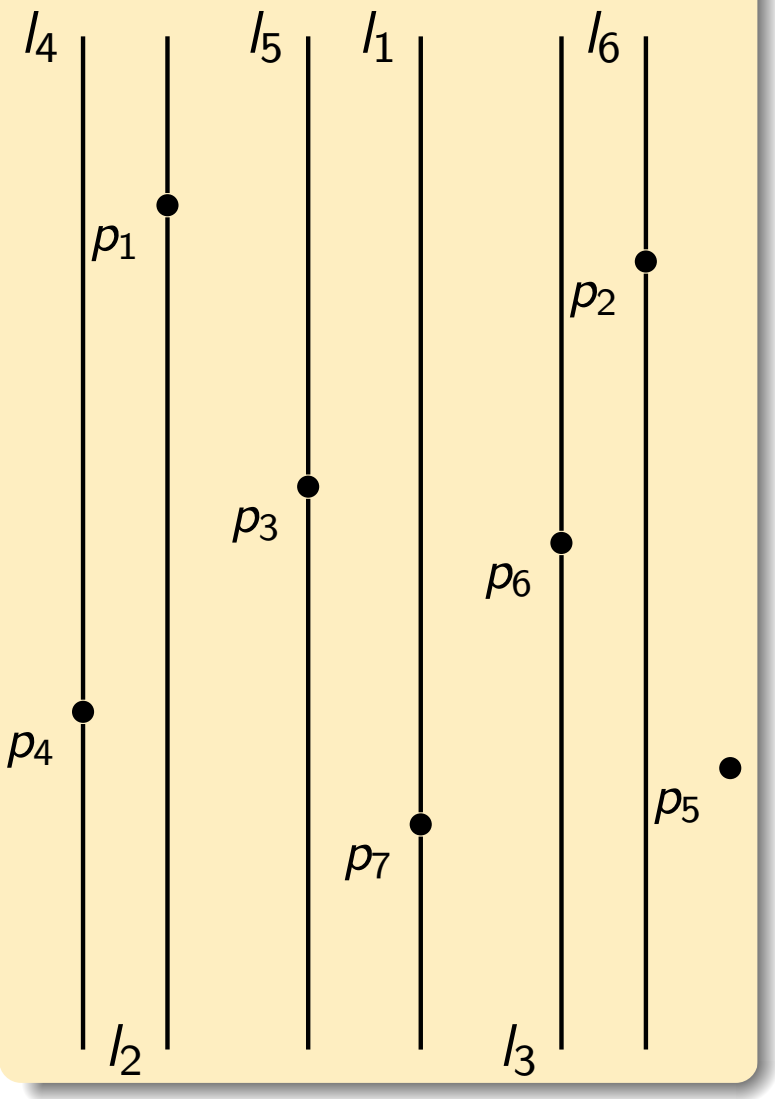


Tree

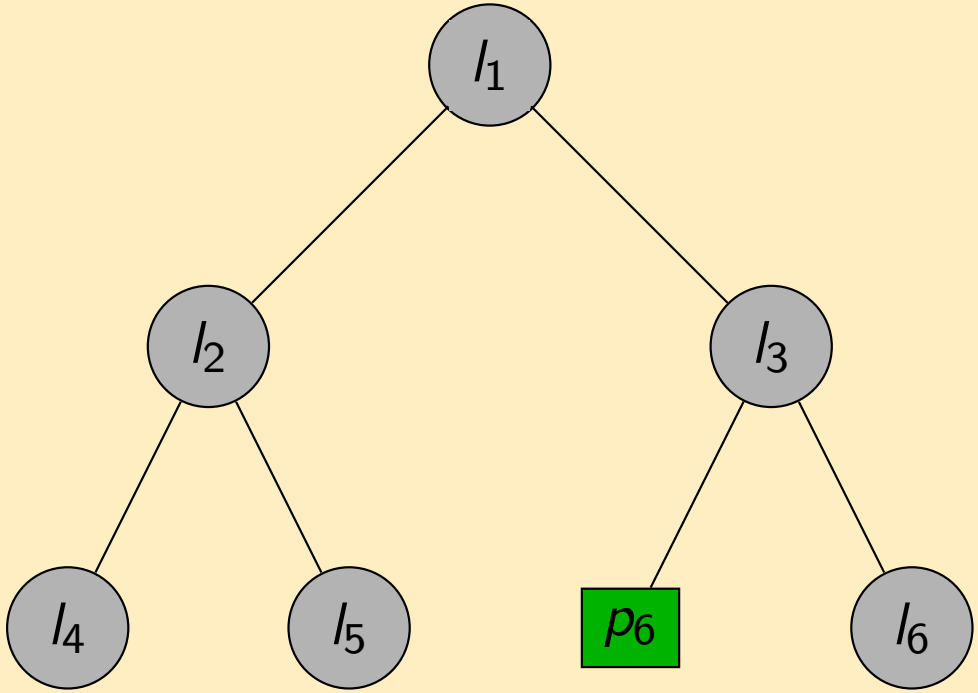


2D Range Tree: Building 1st-Level Tree

Bisection

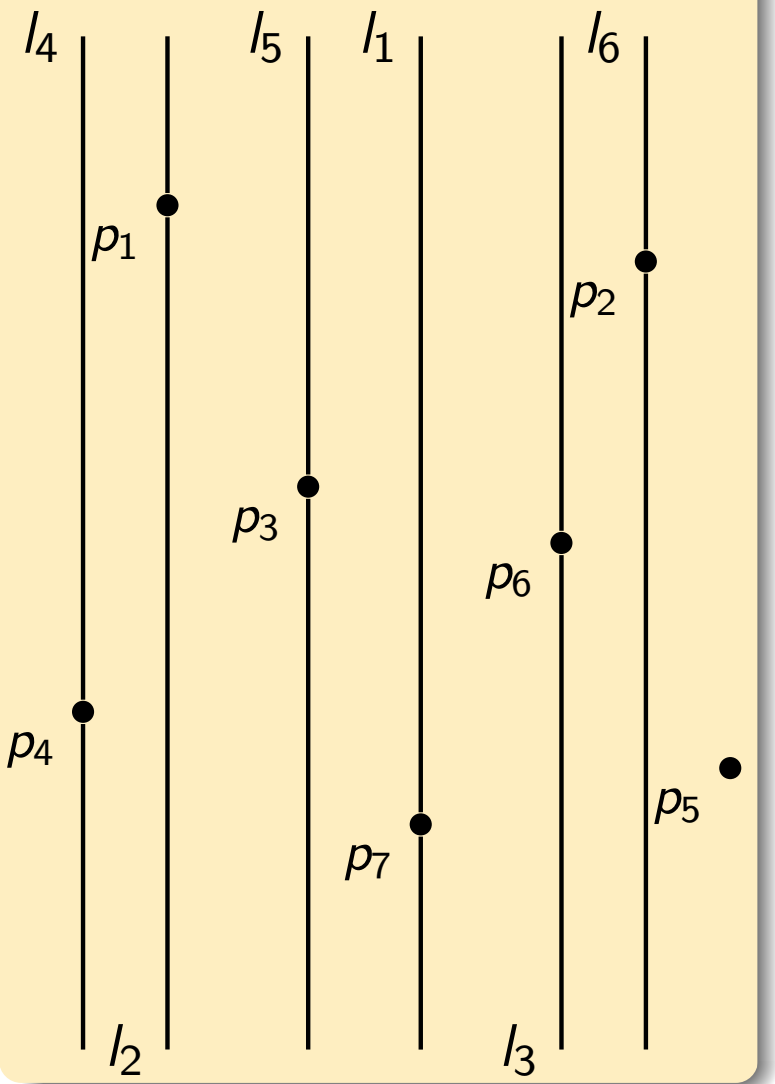


Tree

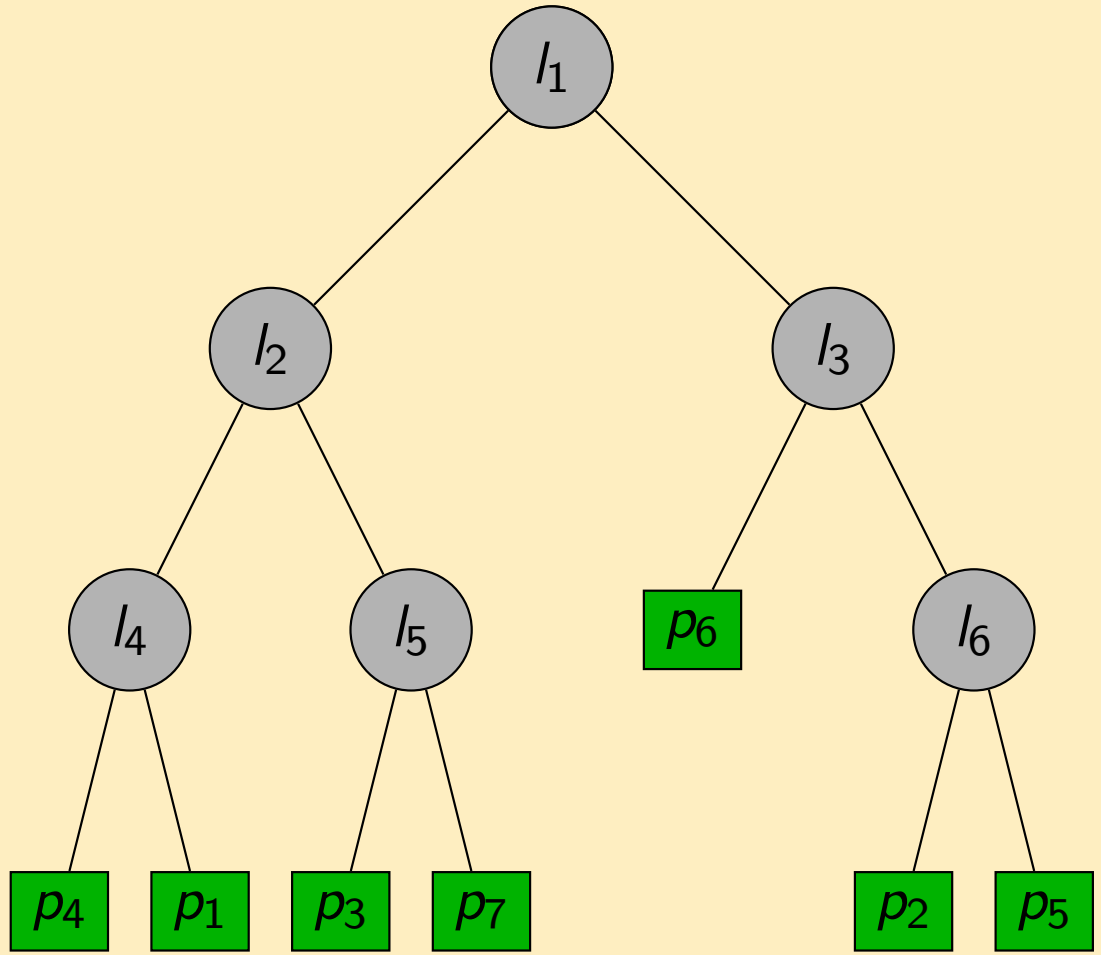


2D Range Tree: Building 1st-Level Tree

Bisection

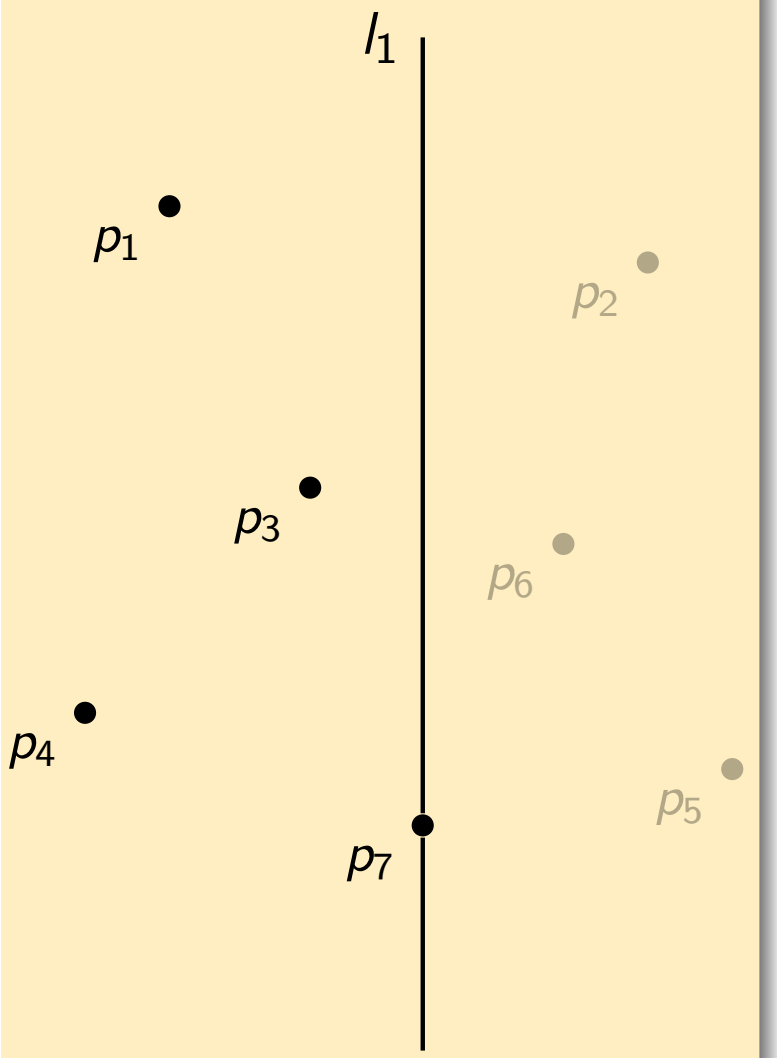


Tree

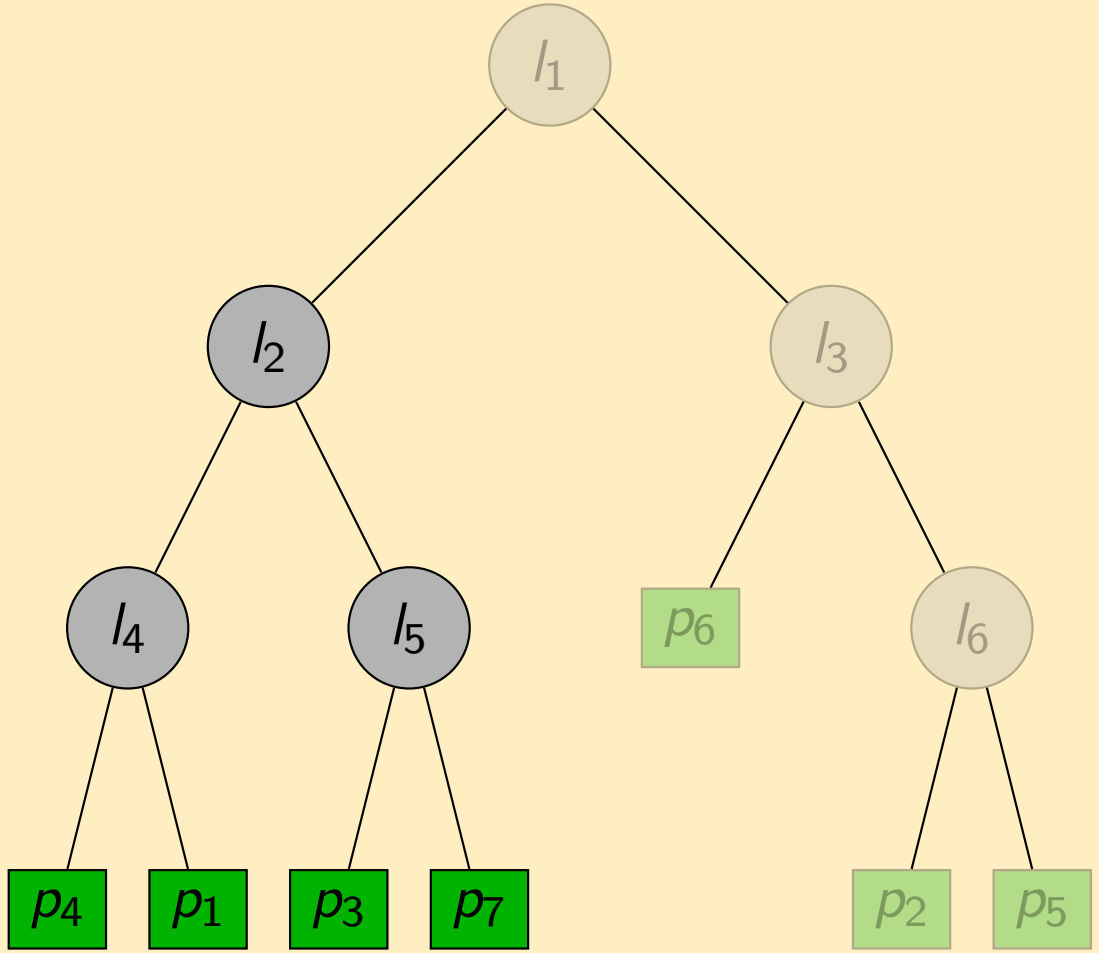


2D Range Tree: Building 2nd-Level Tree for l_2 node

Bisection

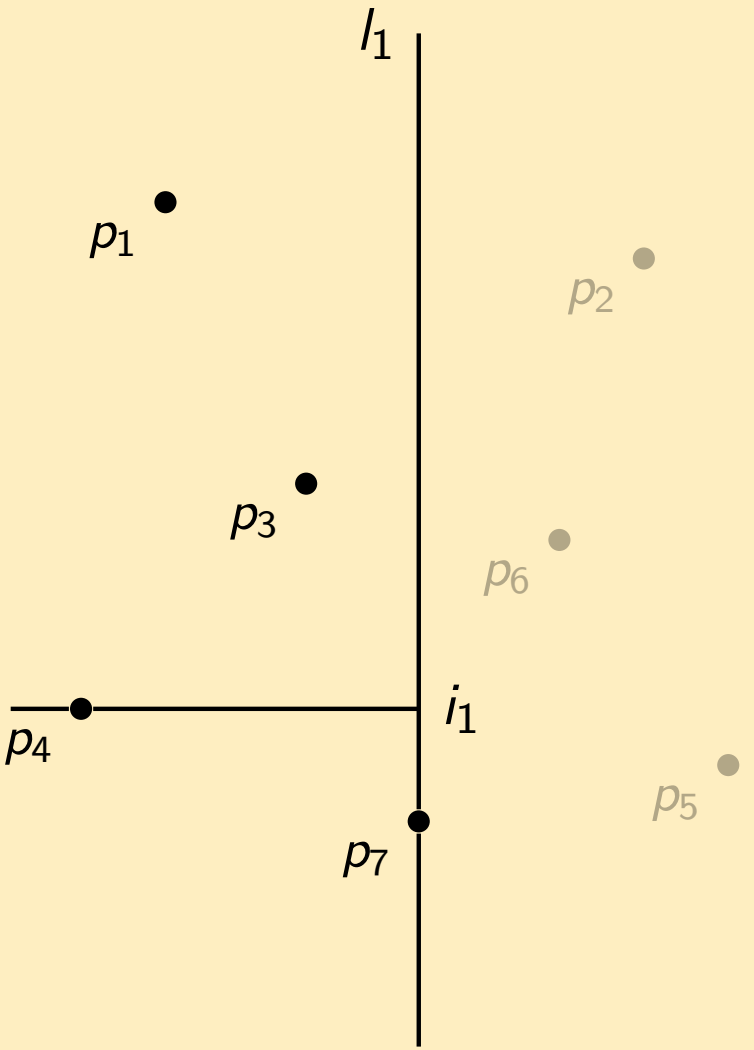


2nd-Level Tree



2D Range Tree: Building 2nd-Level Tree for l_2 node

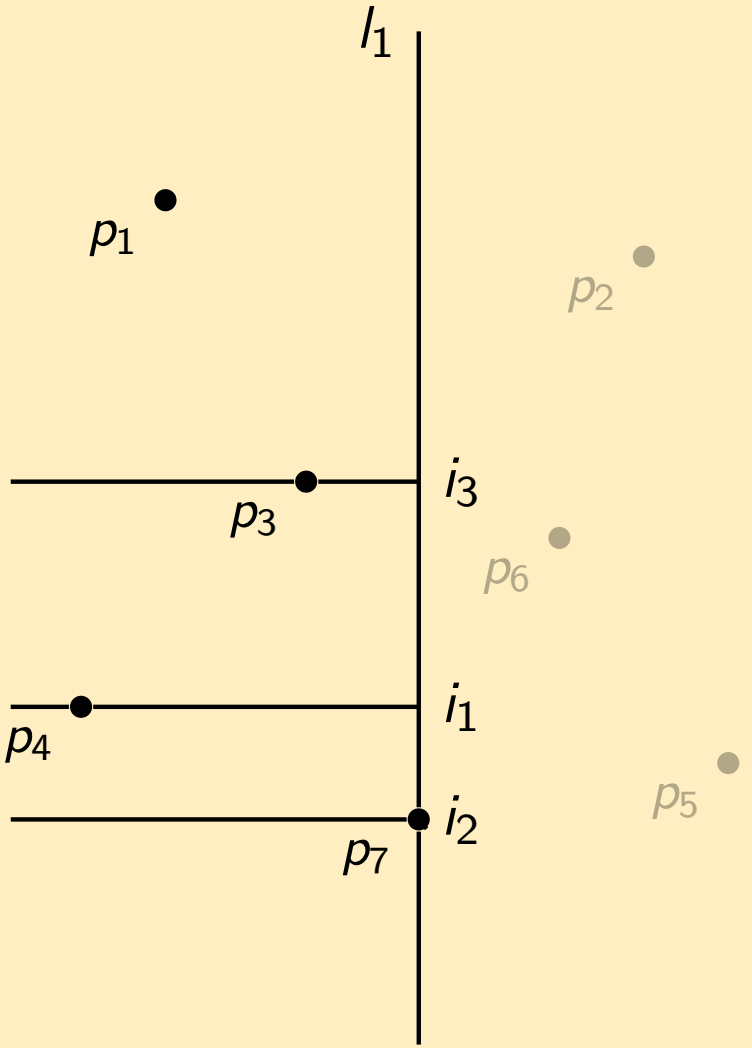
Bisection



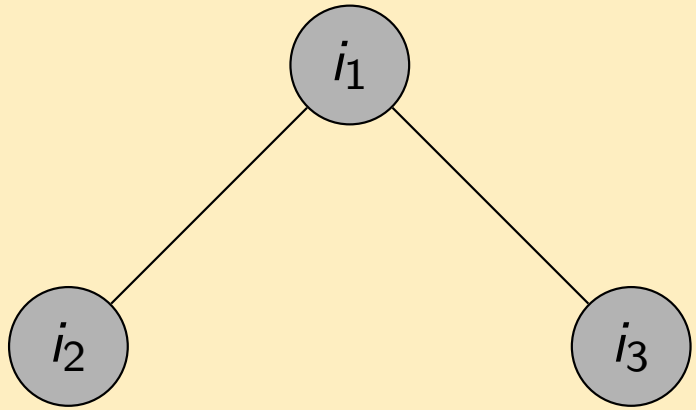
2nd-Level Tree

2D Range Tree: Building 2nd-Level Tree for l_2 node

Bisection

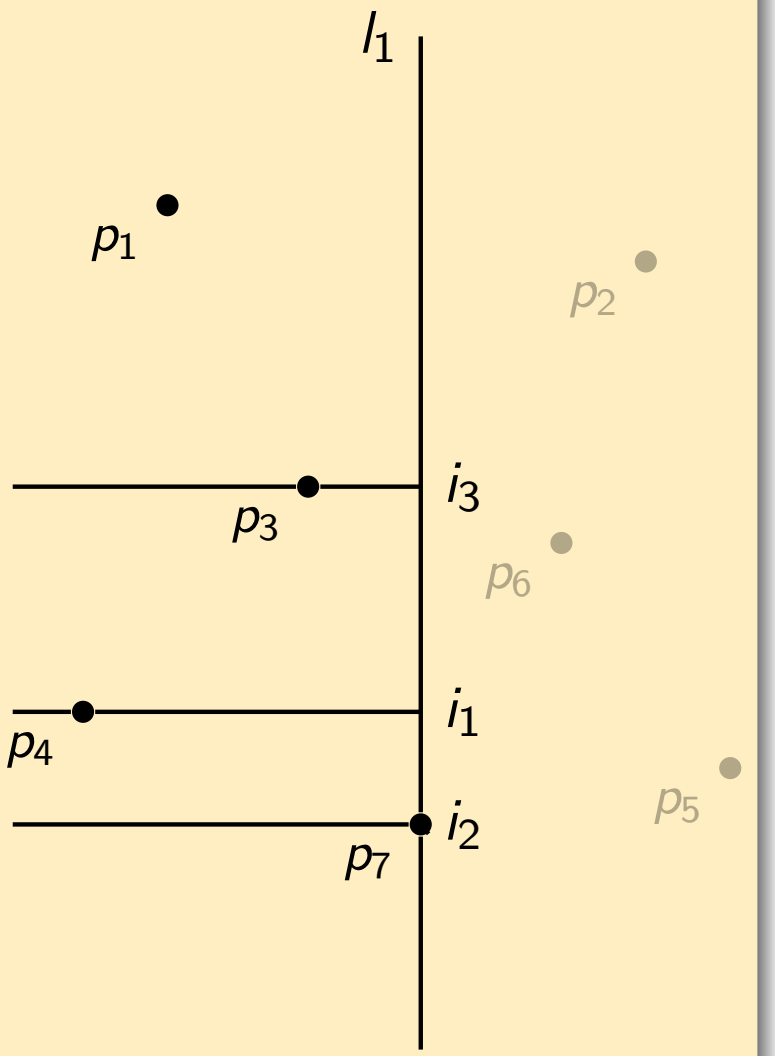


2nd-Level Tree

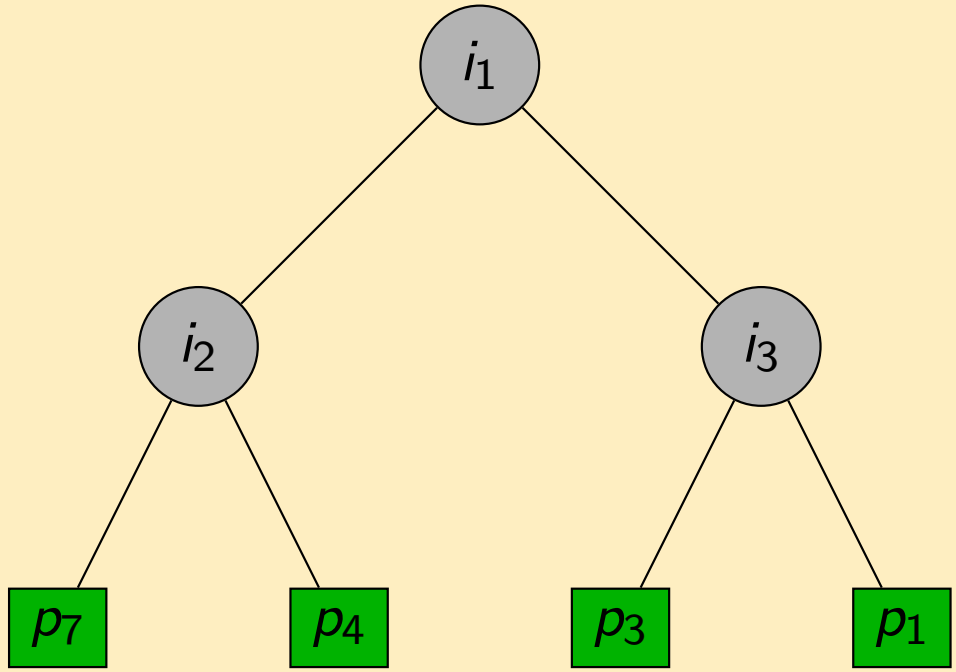


2D Range Tree: Building 2nd-Level Tree for l_2 node

Bisection



2nd-Level Tree

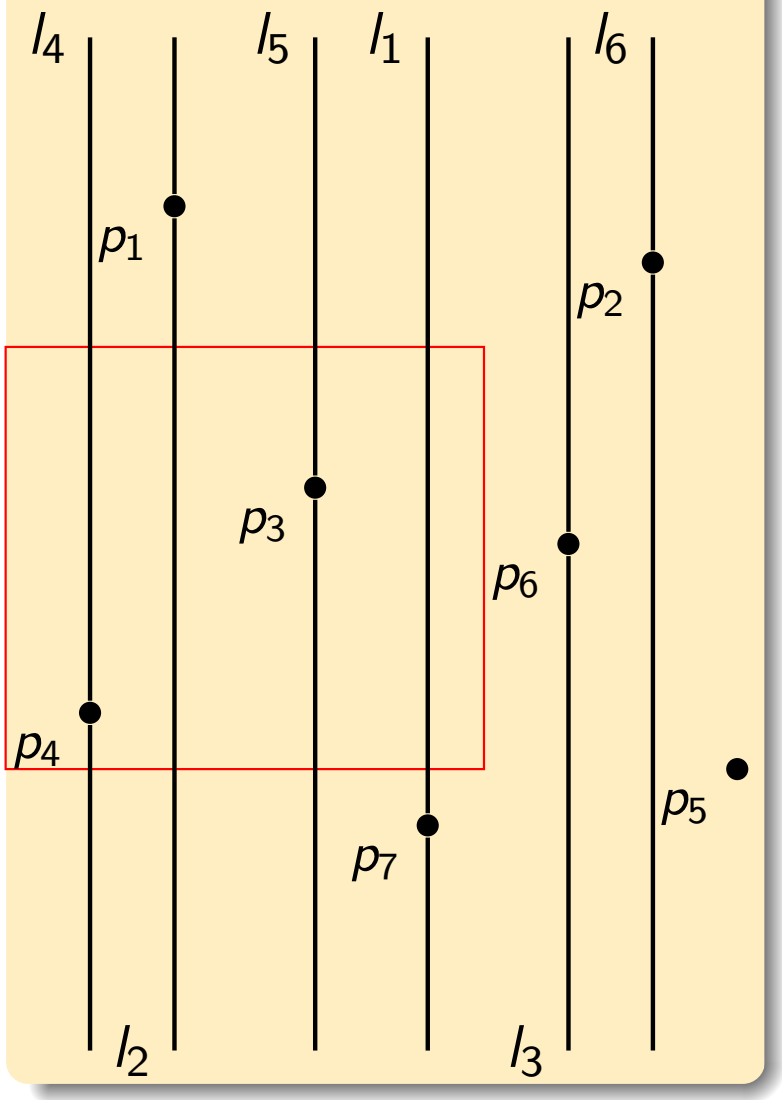


2D Range Tree: Query

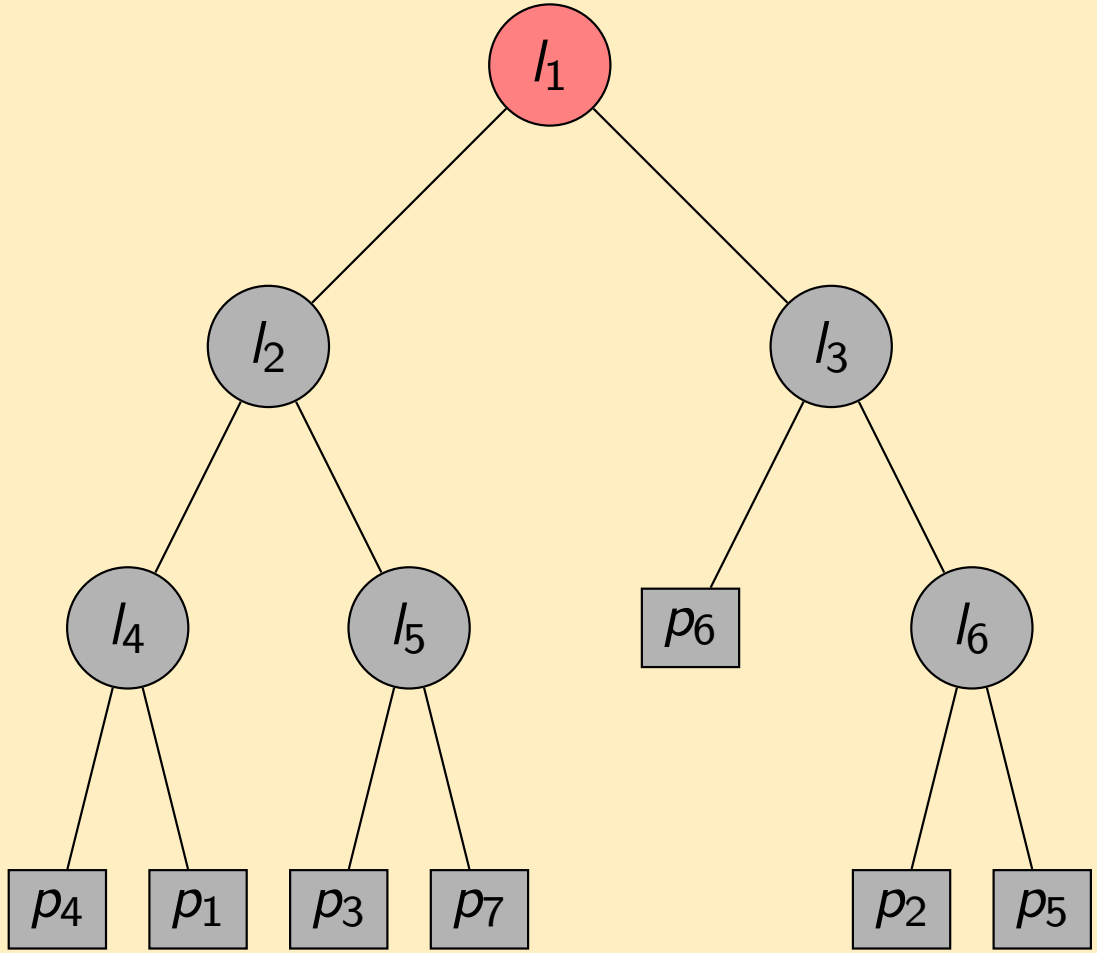
- rectangle is determined by points $[x, y]$ and $[x', y']$
- the problem is split to 1-dimensional searching:
  1. find canonical subset of points contained in the the region (x, x') (1st-level tree)
  2. search for the points from the range (y, y') in the 2nd level sub-tree corresponding to the subset obtained in 1.

2D Range Tree: Query in 1st-Level Tree

Bisection

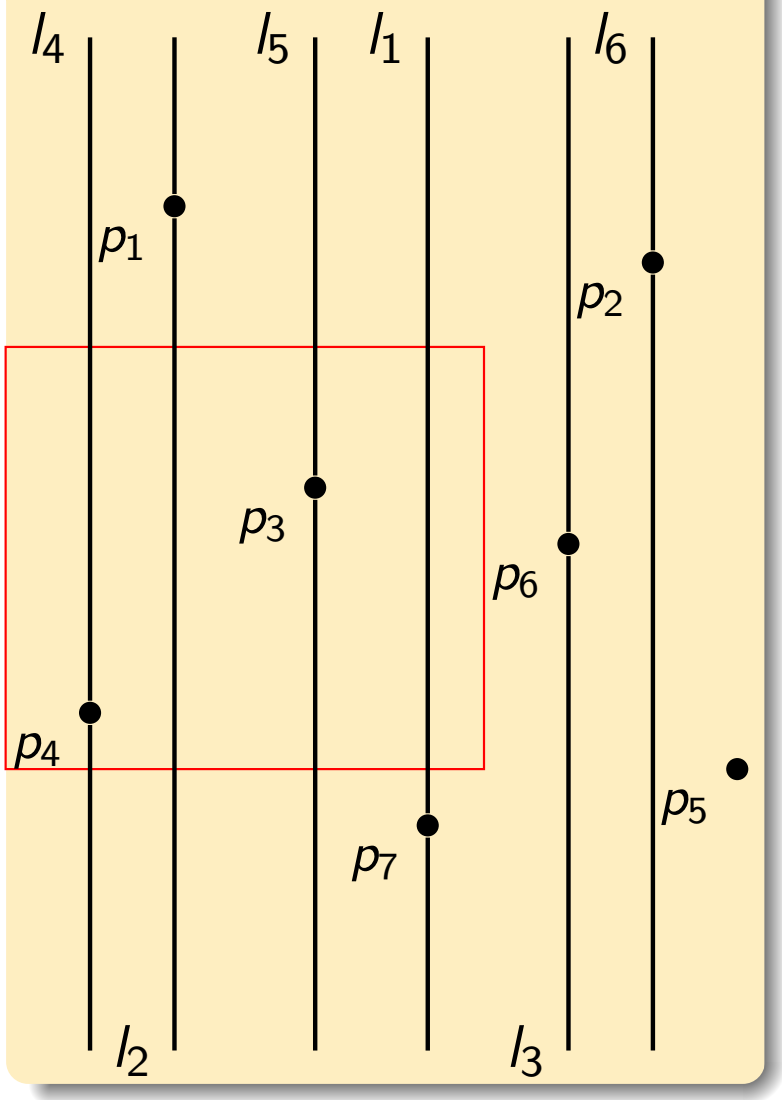


Tree

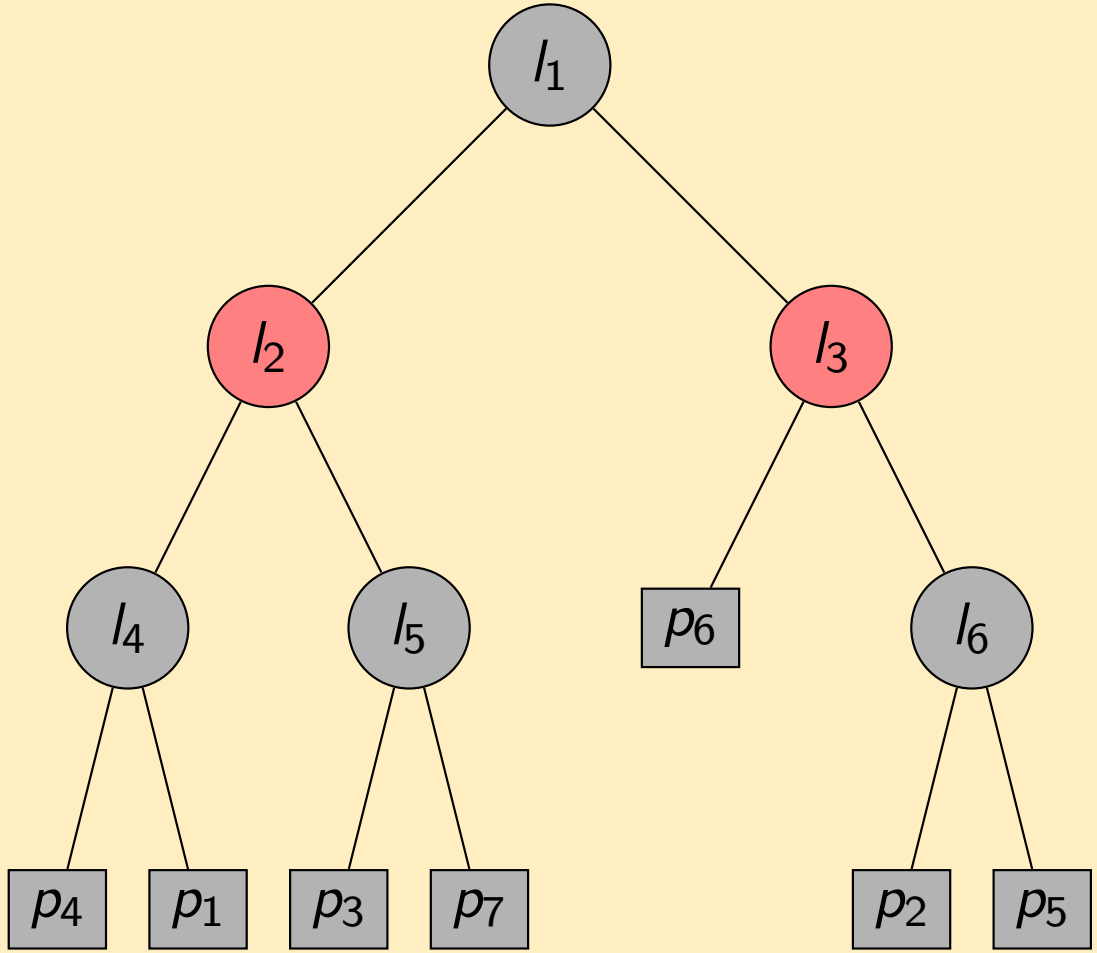


2D Range Tree: Query in 1st-Level Tree

Bisection

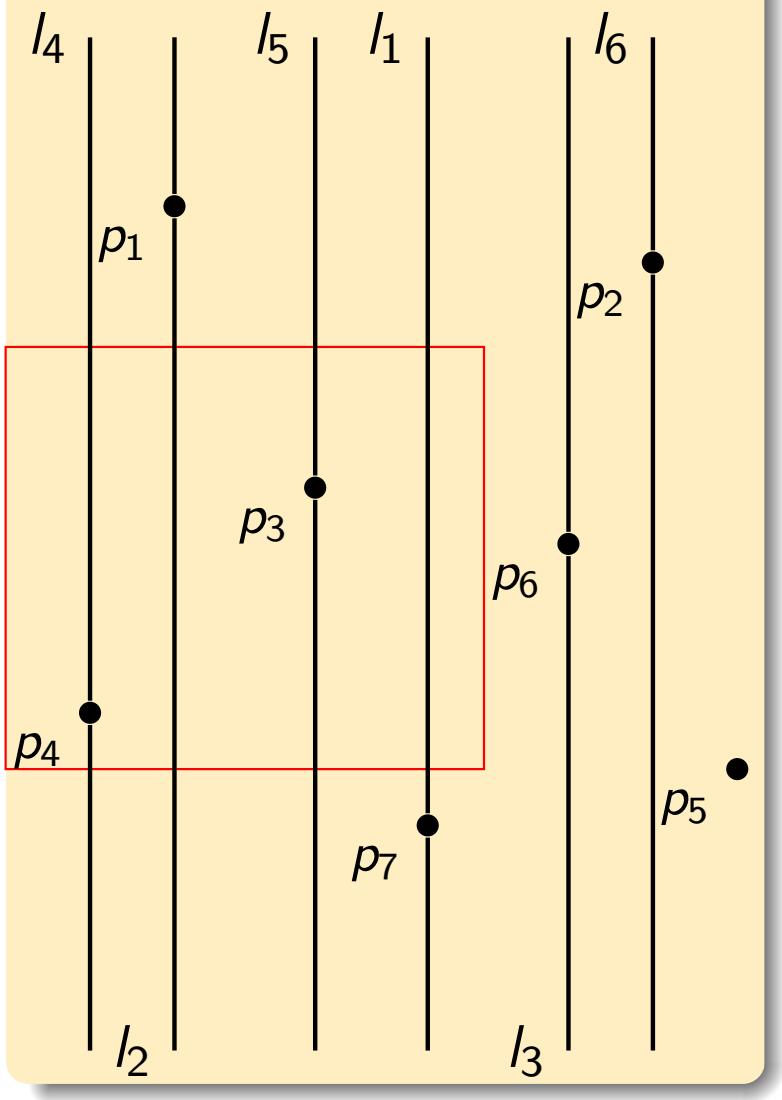


Tree

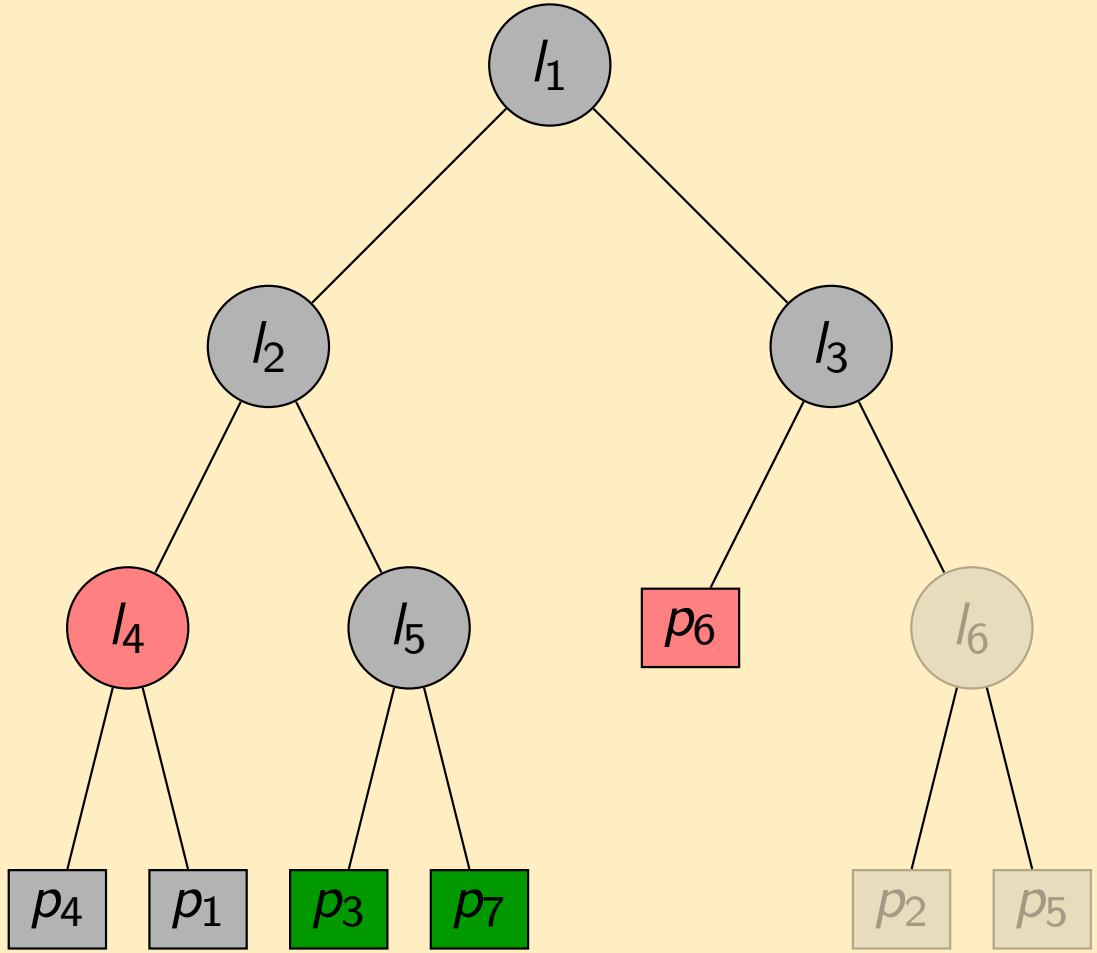


2D Range Tree: Query in 1st-Level Tree

Bisection

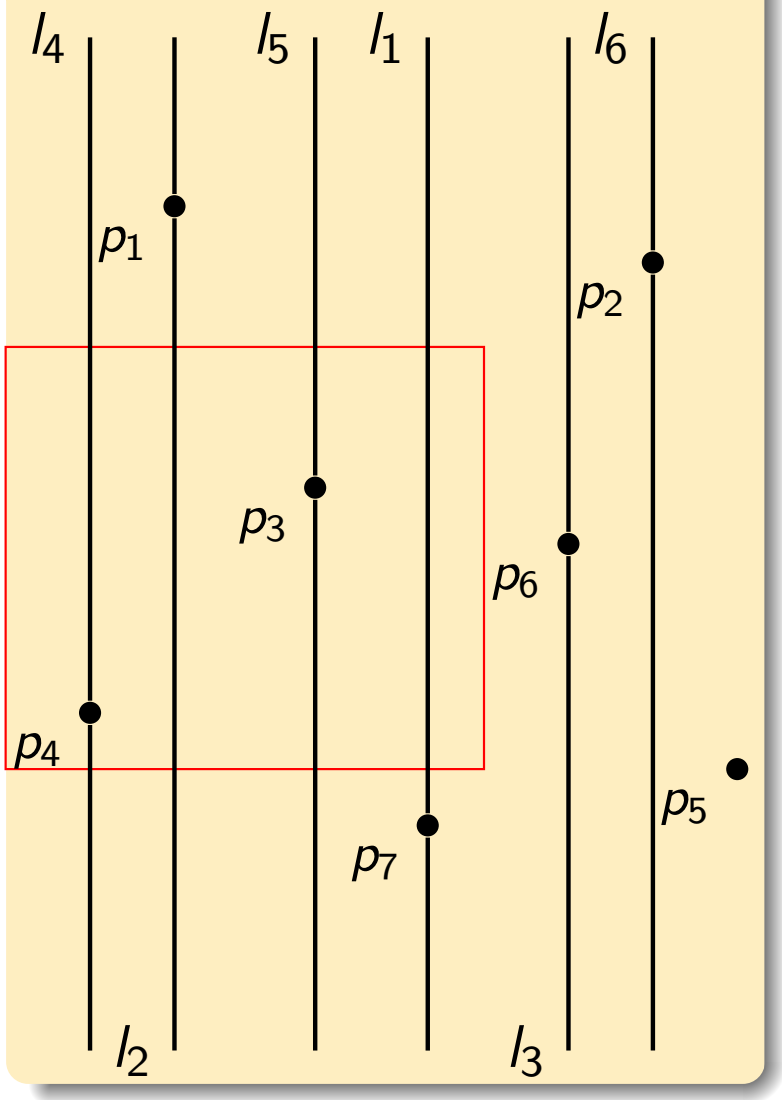


Tree

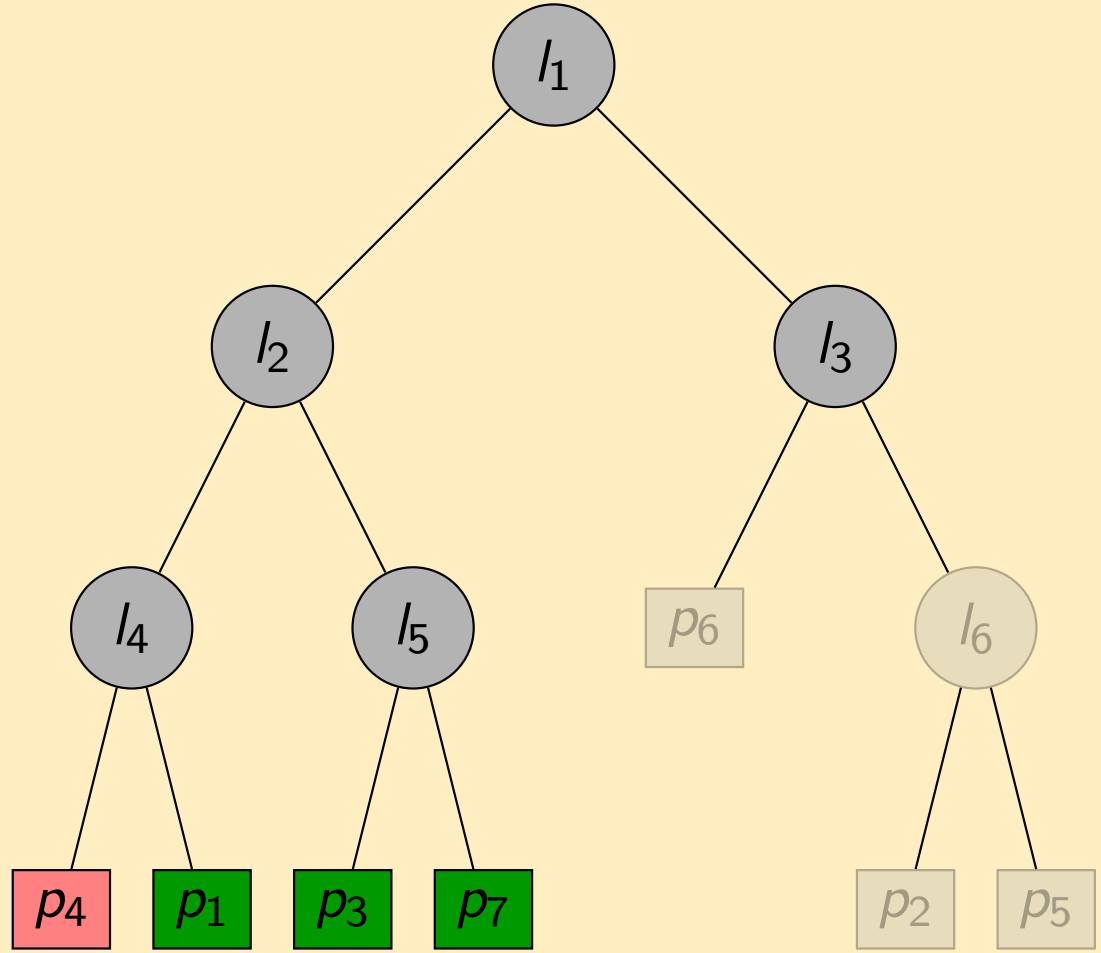


2D Range Tree: Query in 1st-Level Tree

Bisection

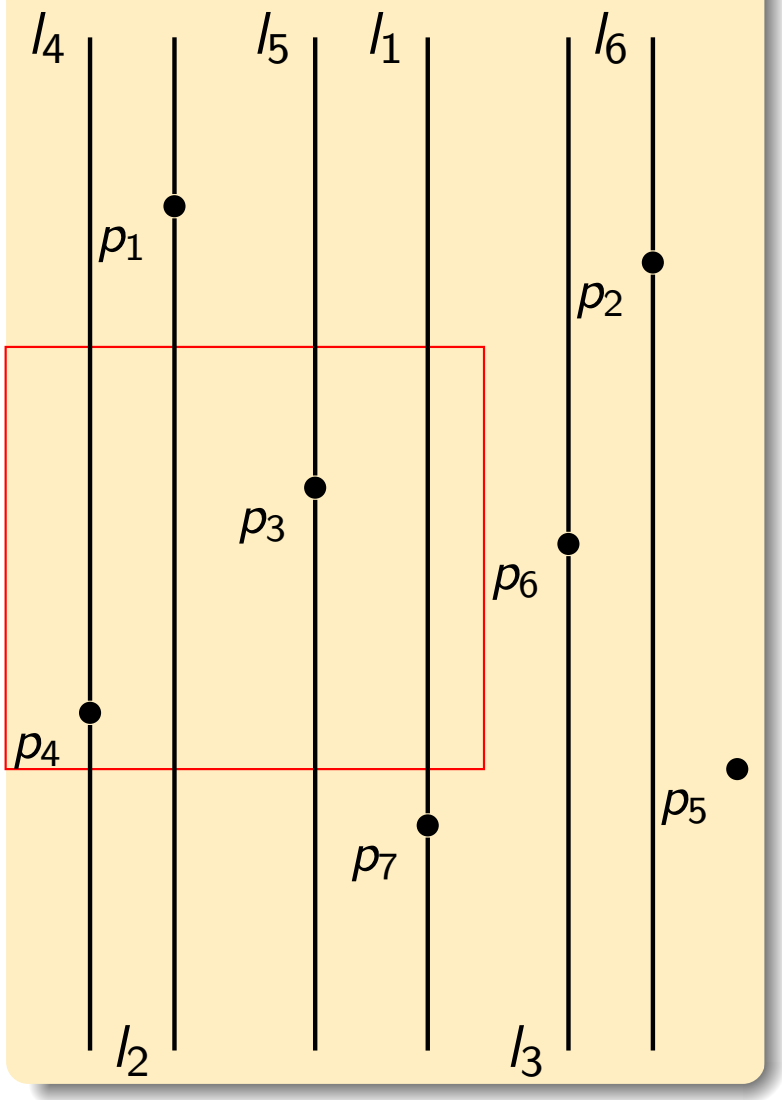


Tree

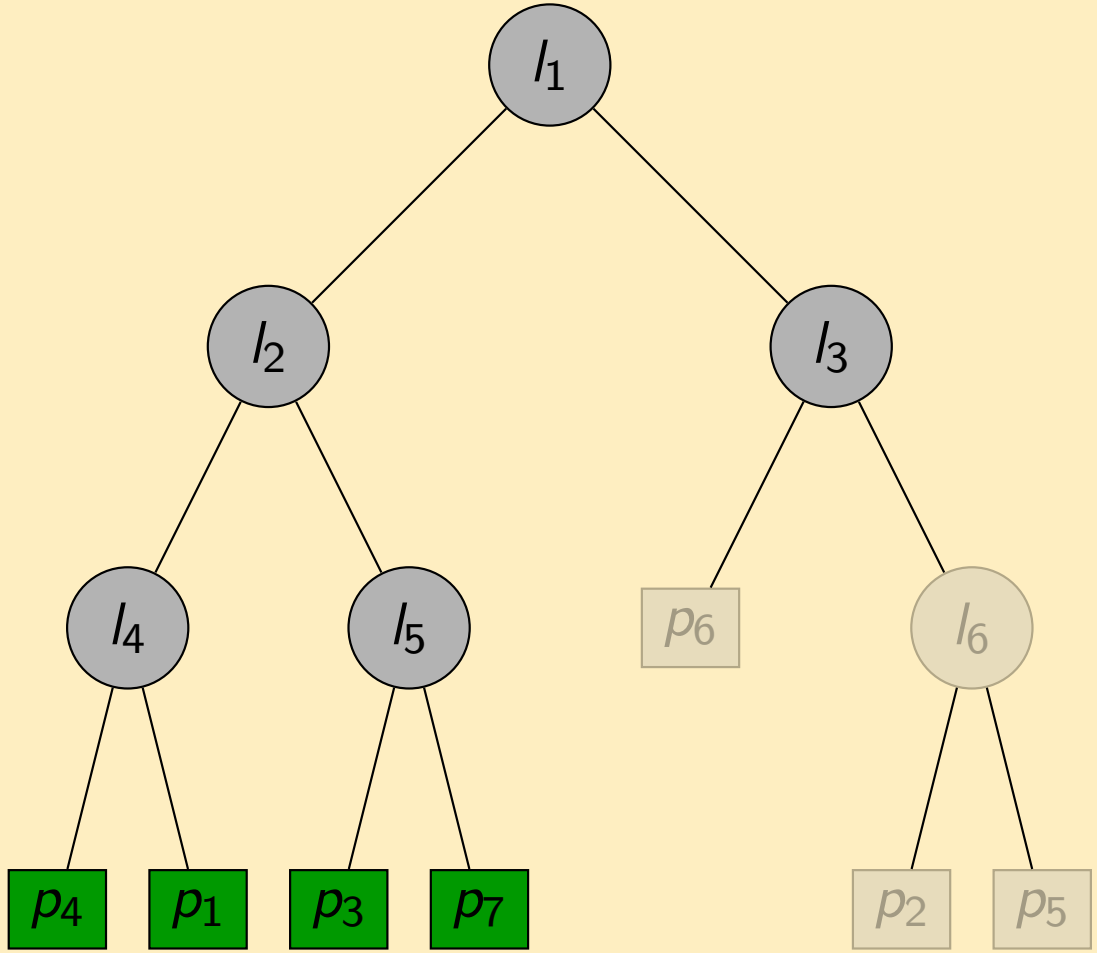


2D Range Tree: Query in 1st-Level Tree

Bisection

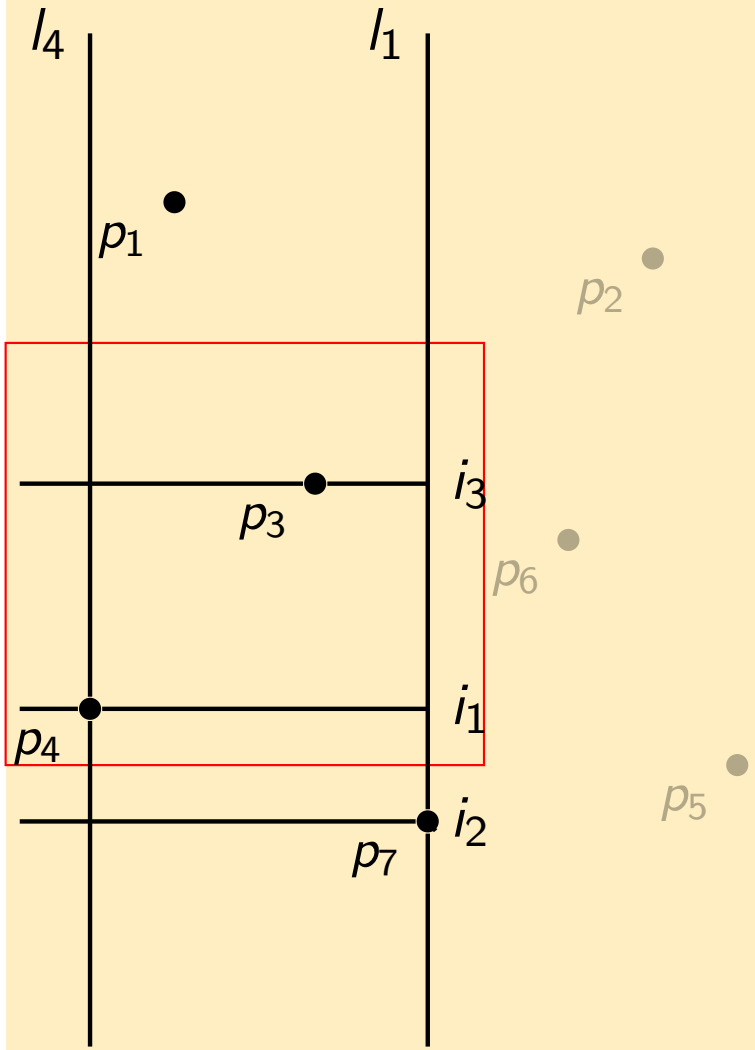


Tree

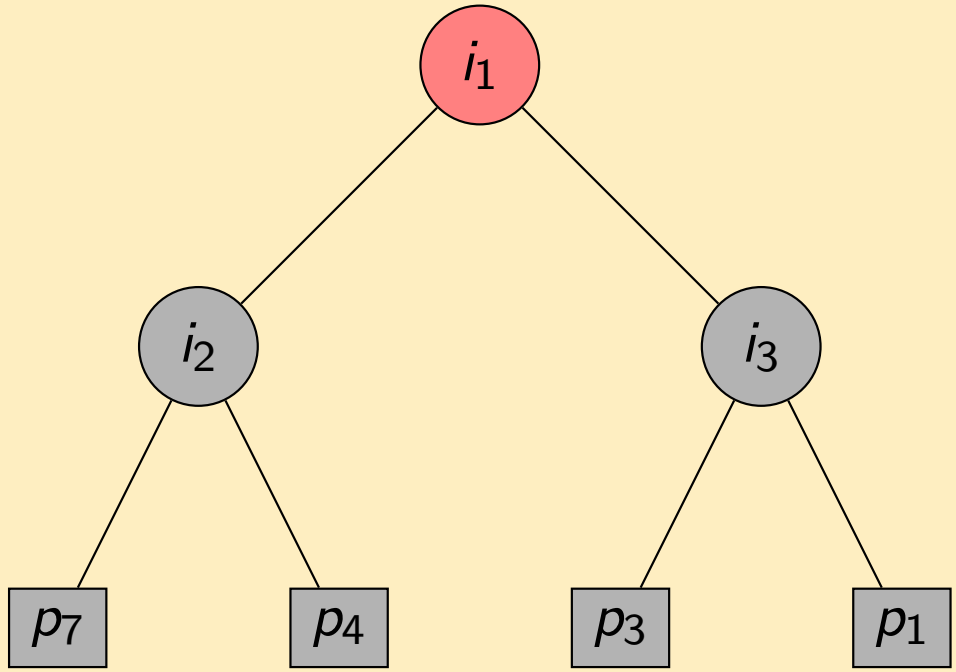


2D Range Tree: Query in the 2nd-Level Tree

Bisection

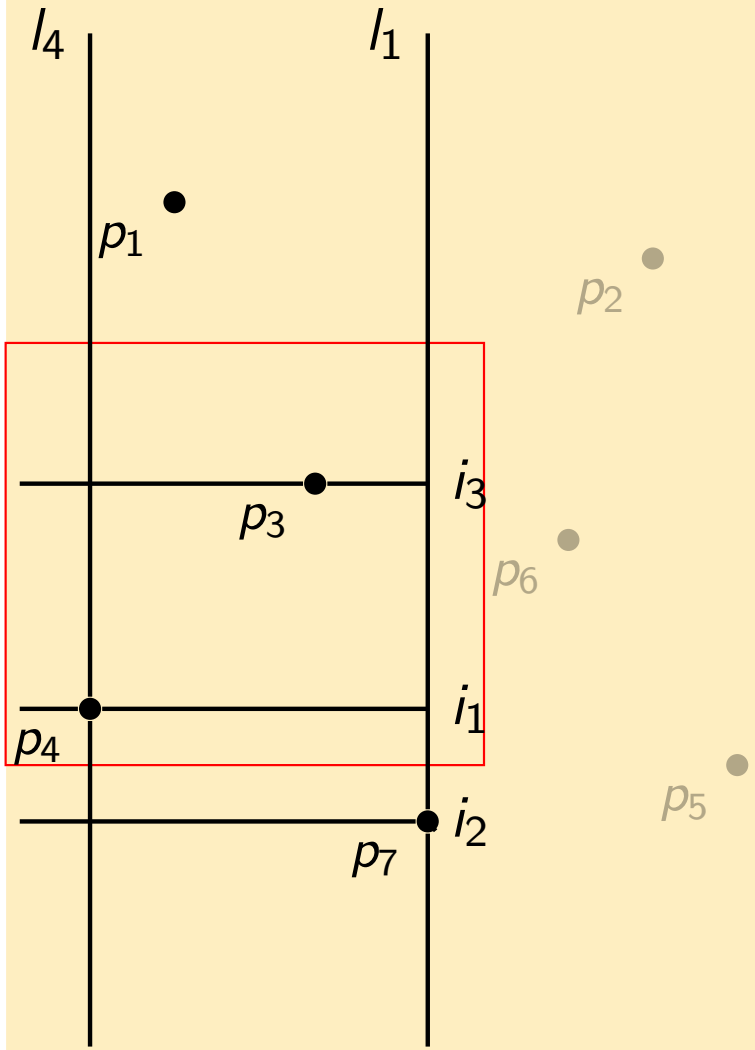


2nd-Level Tree

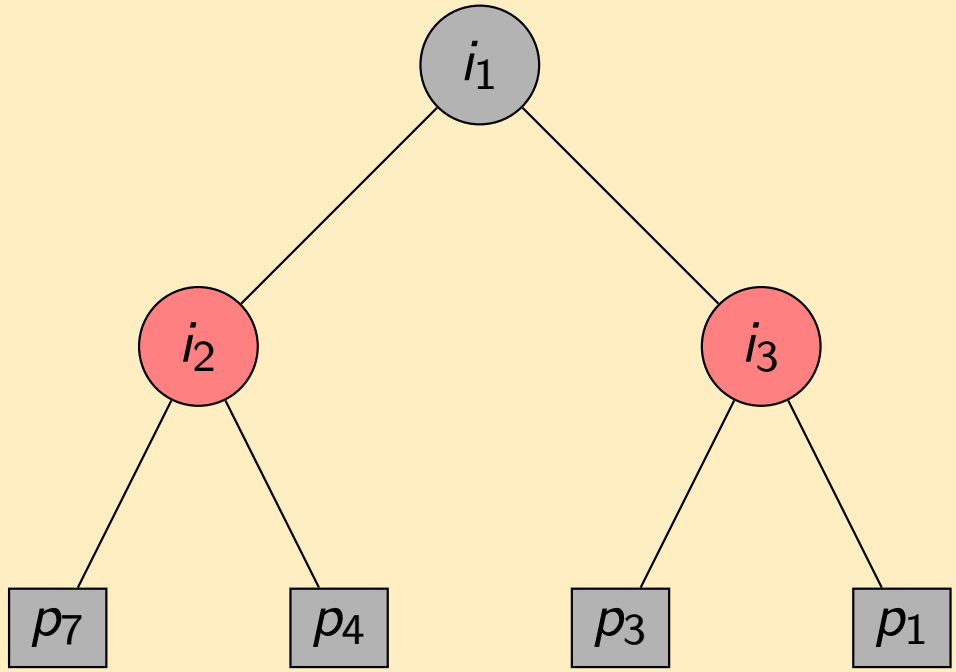


2D Range Tree: Query in the 2nd-Level Tree

Bisection

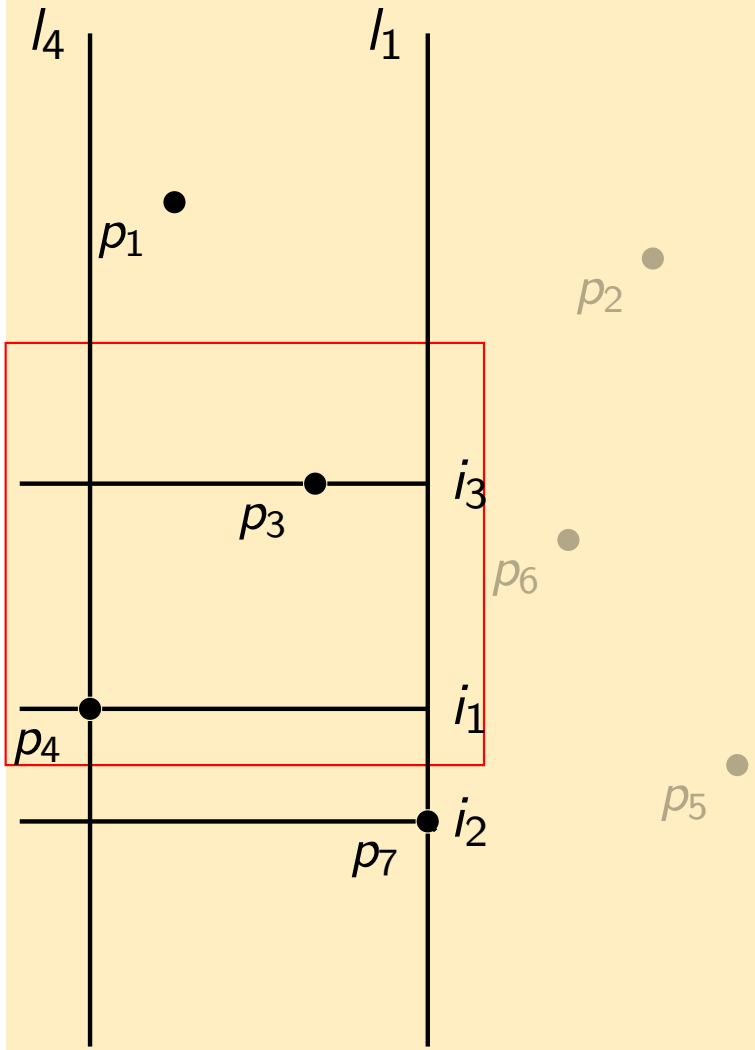


2nd-Level Tree

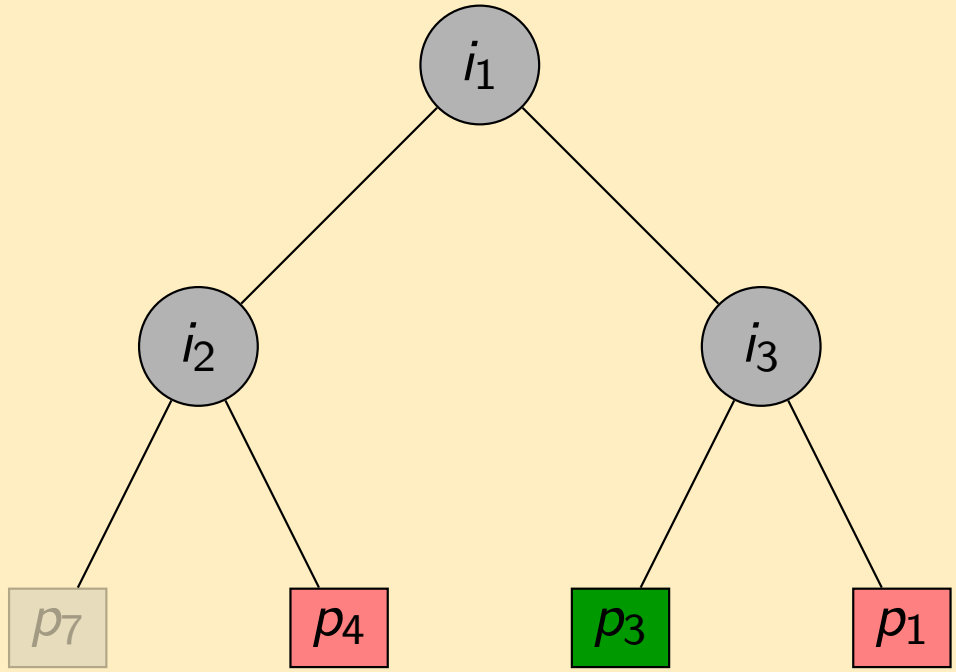


2D Range Tree: Query in the 2nd-Level Tree

Bisection

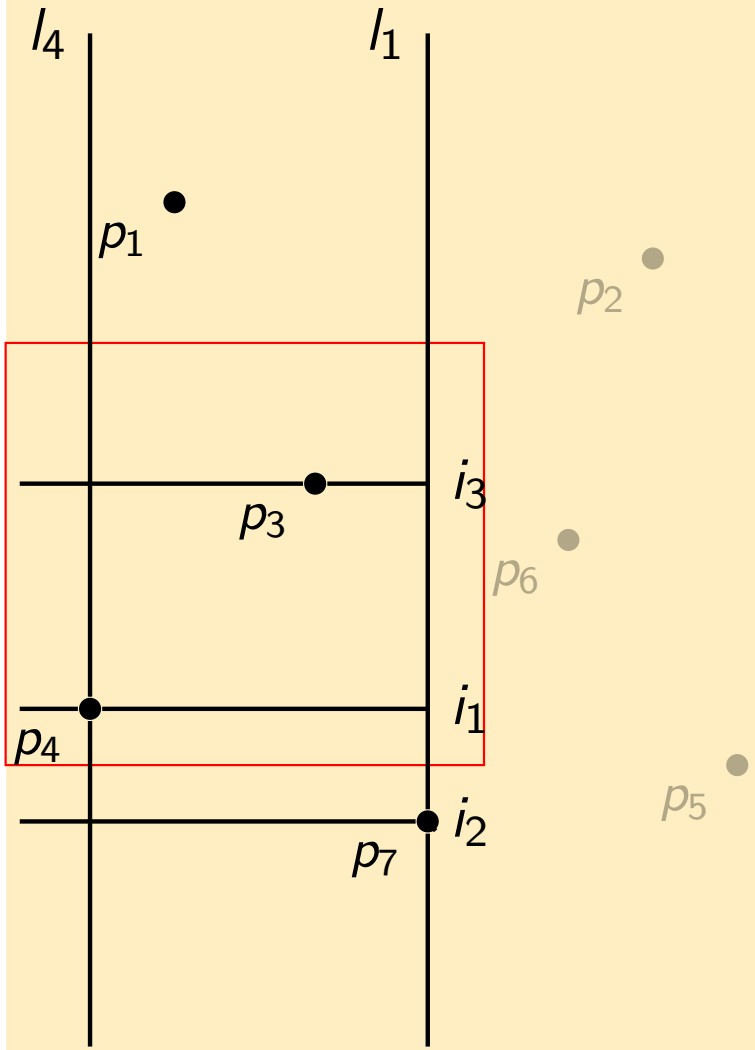


2nd-Level Tree

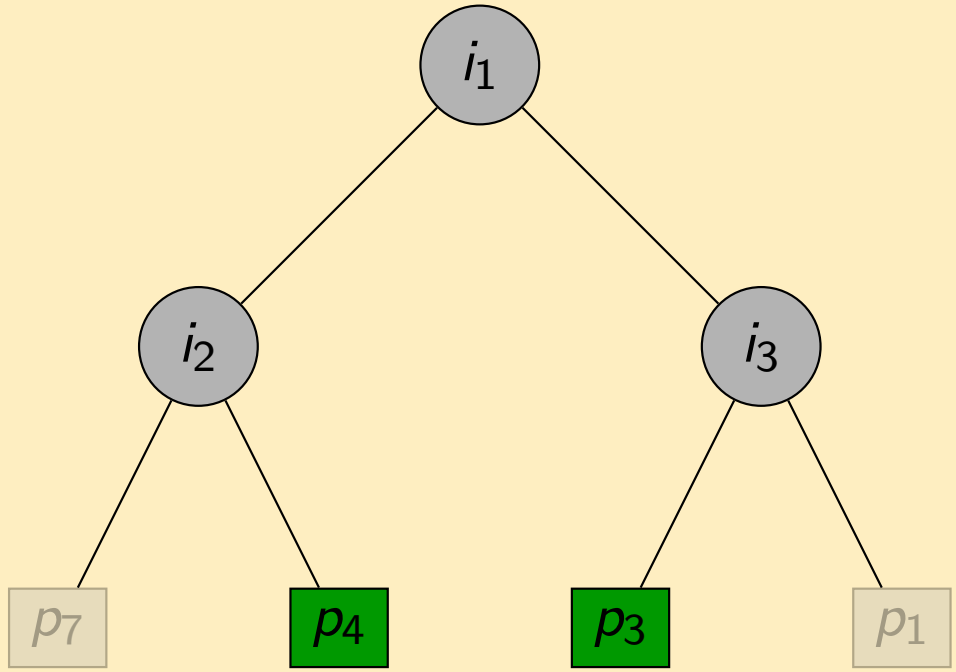


2D Range Tree: Query in the 2nd-Level Tree

Bisection



2nd-Level Tree



2D Range Tree: Complexity

Building the 2d-Tree

Time complexity: $O(n \log n)$

Storage complexity: $O(n \log n)$

Query in 2D Range Tree

Time complexity: $O(\log^2 n + k)$, k . . . number of reported points

References

F. P. Preparata and M. I. Shamos, *Computational geometry: an introduction*. New York, NY, USA: Springer-Verlag New York, Inc., 1985, ISBN: 0-387-96131-3.

M. d. Berg, O. Cheong, M. v. Kreveld, and M. Overmars, *Computational Geometry: Algorithms and Applications*, 3rd ed. Santa Clara, CA, USA: Springer-Verlag TELOS, 2008, ISBN: 3540779736, 9783540779735.

OI-OPPA. European Social Fund
Prague & EU: We invest in your future.