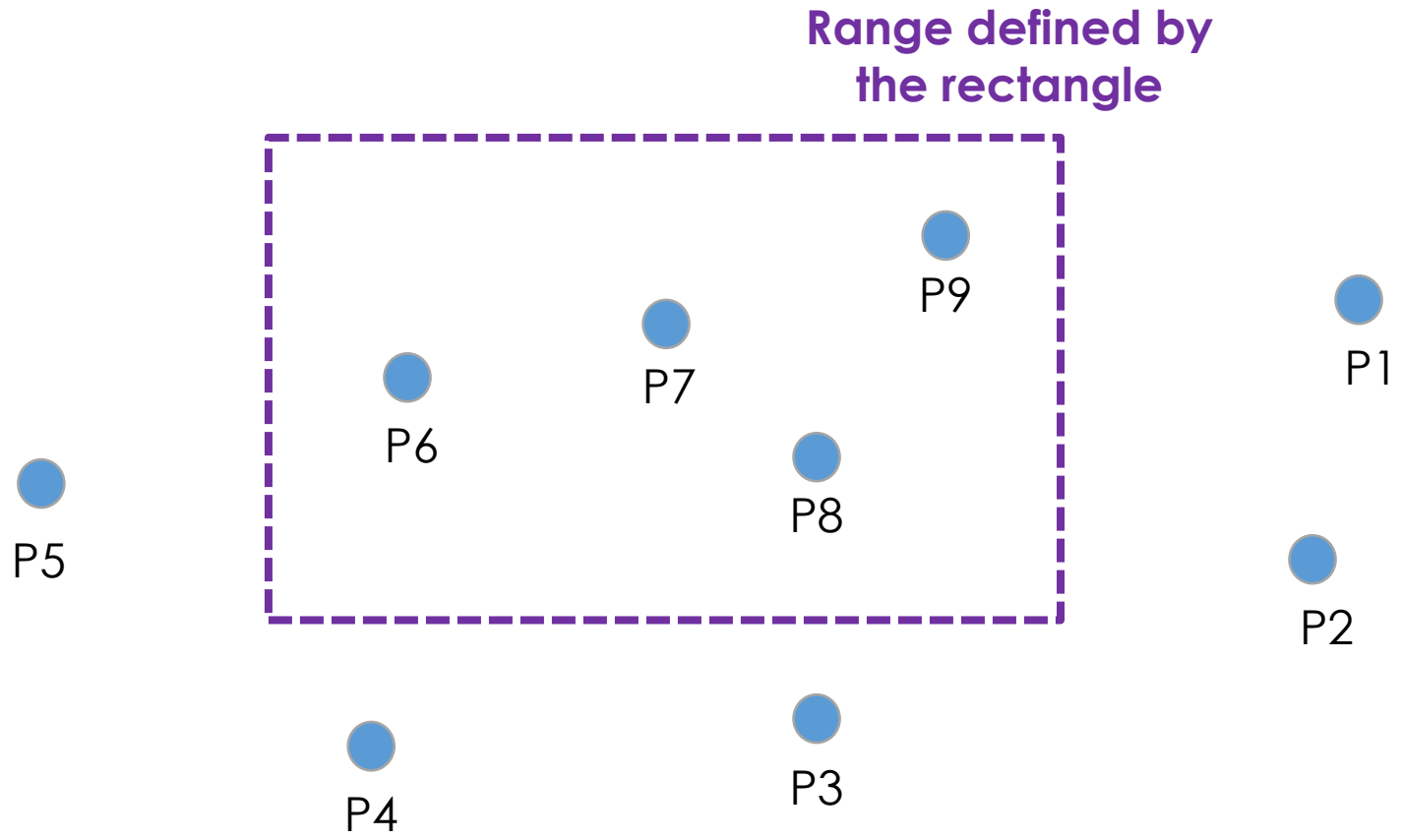
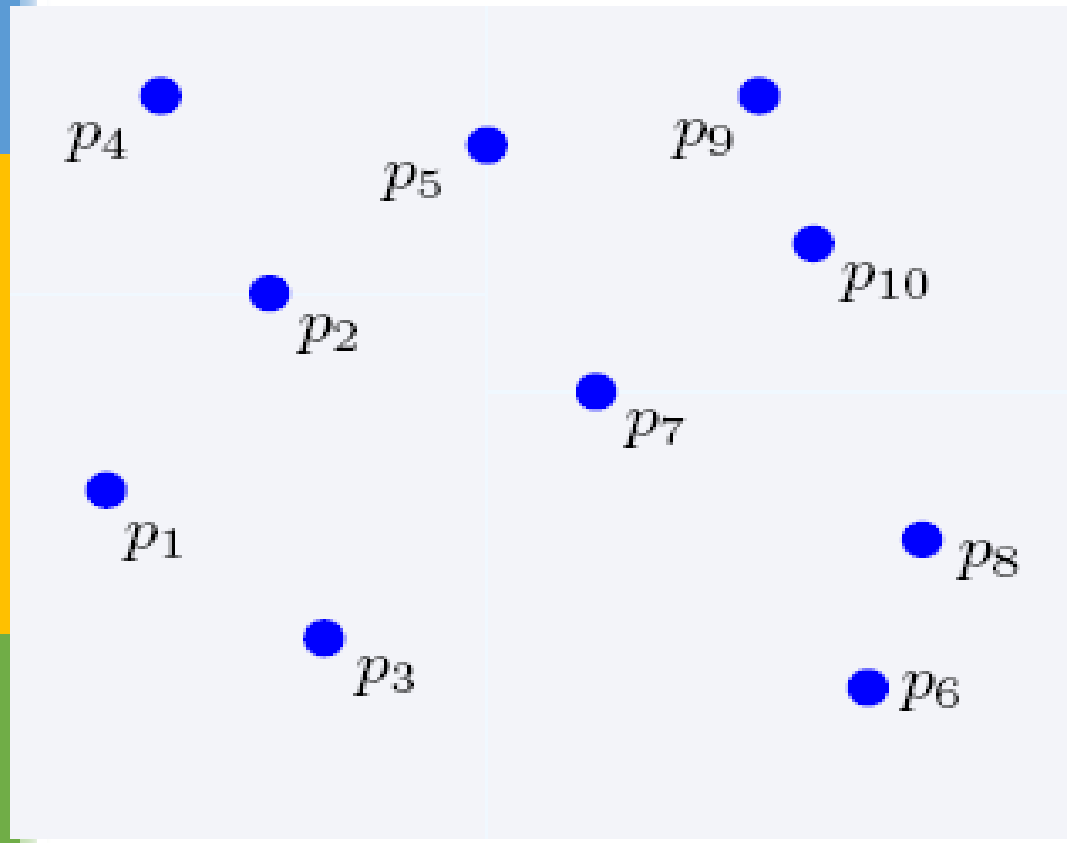
Geometric Data Structures

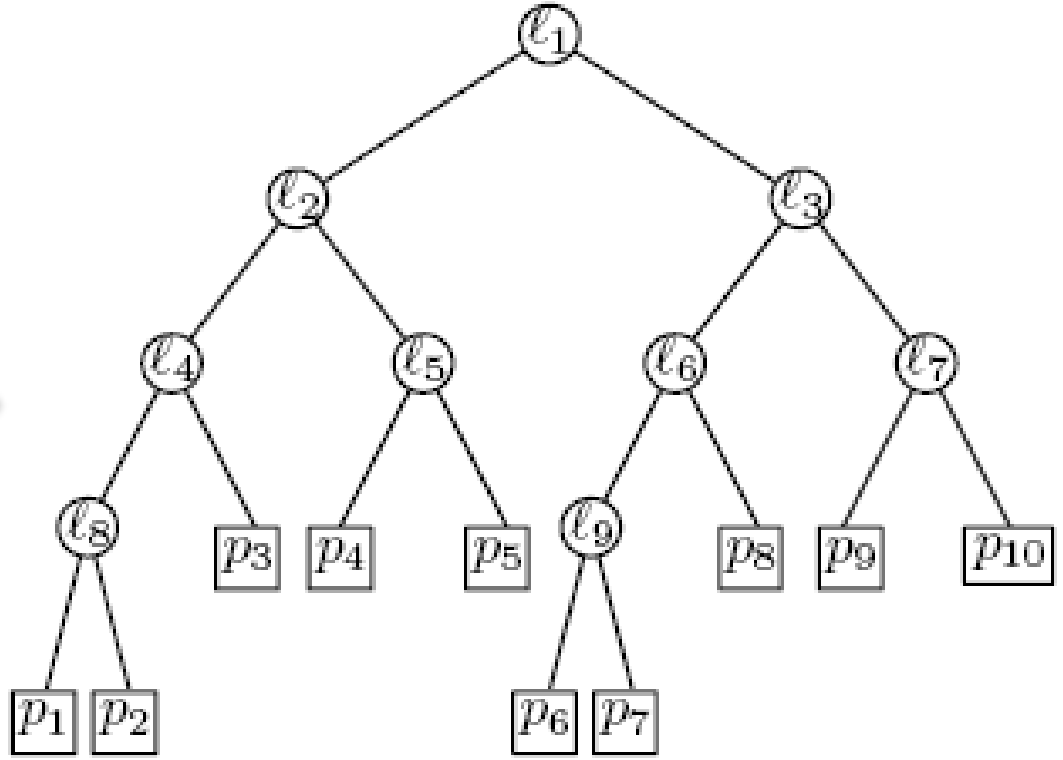
Range Queries



Retrieve all points inside the rectangle

KD Tree

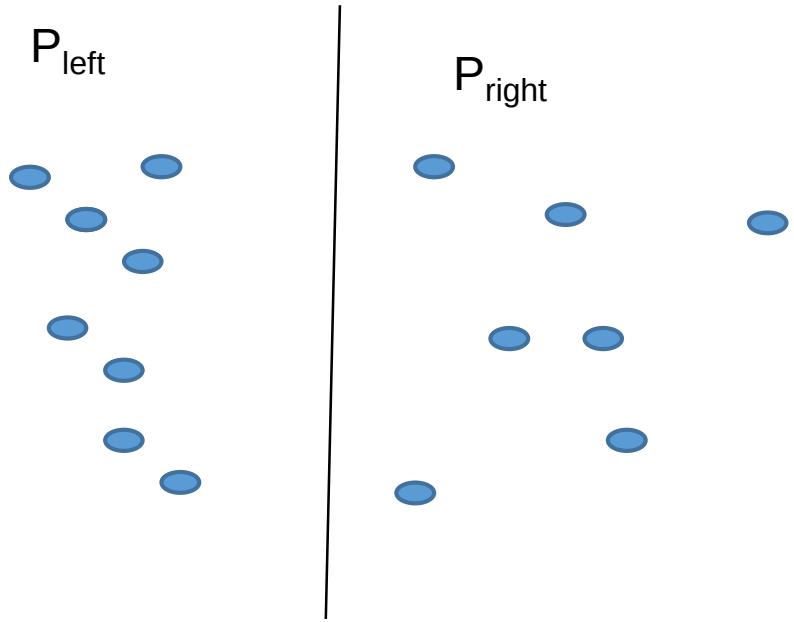


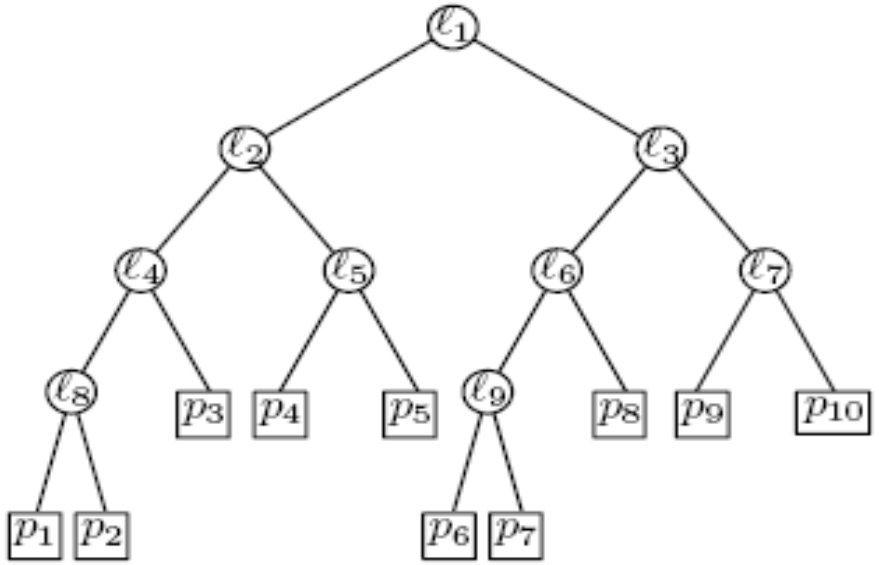


Slide adapted from Dr George Bebis Univ Nevada

KD Tree

- Every node (except leaves) represents a hyperplane that divides the space into two parts.
- Points to the left (right) of this hyperplane represent the left (right) sub-tree of that node.





Slide adapted from Dr George Bebis Univ Nevada

KD Tree

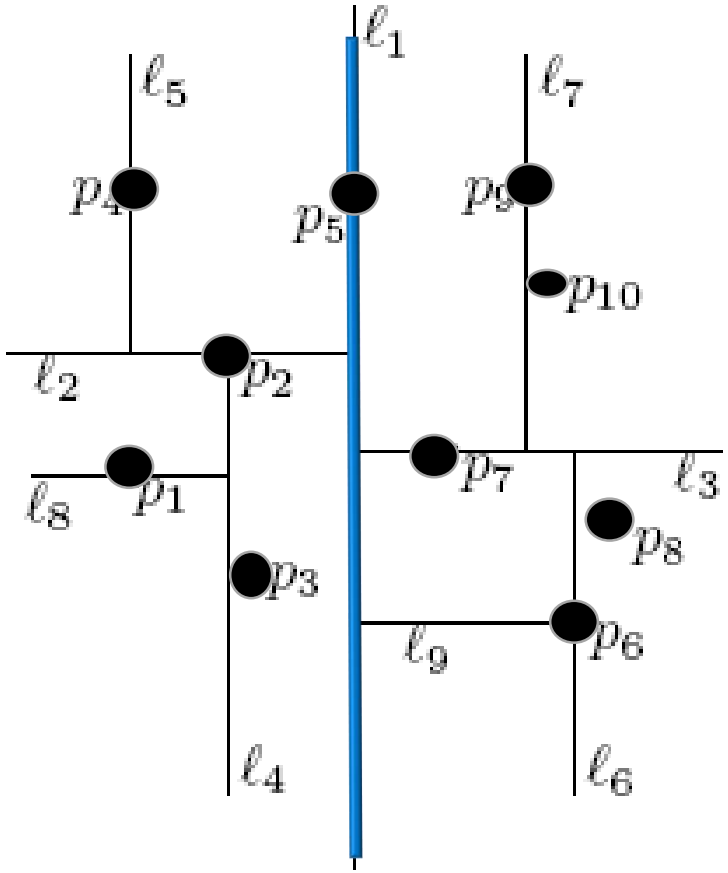
As we move down the tree, we divide the space along alternating (but not always) axis-aligned hyperplanes:

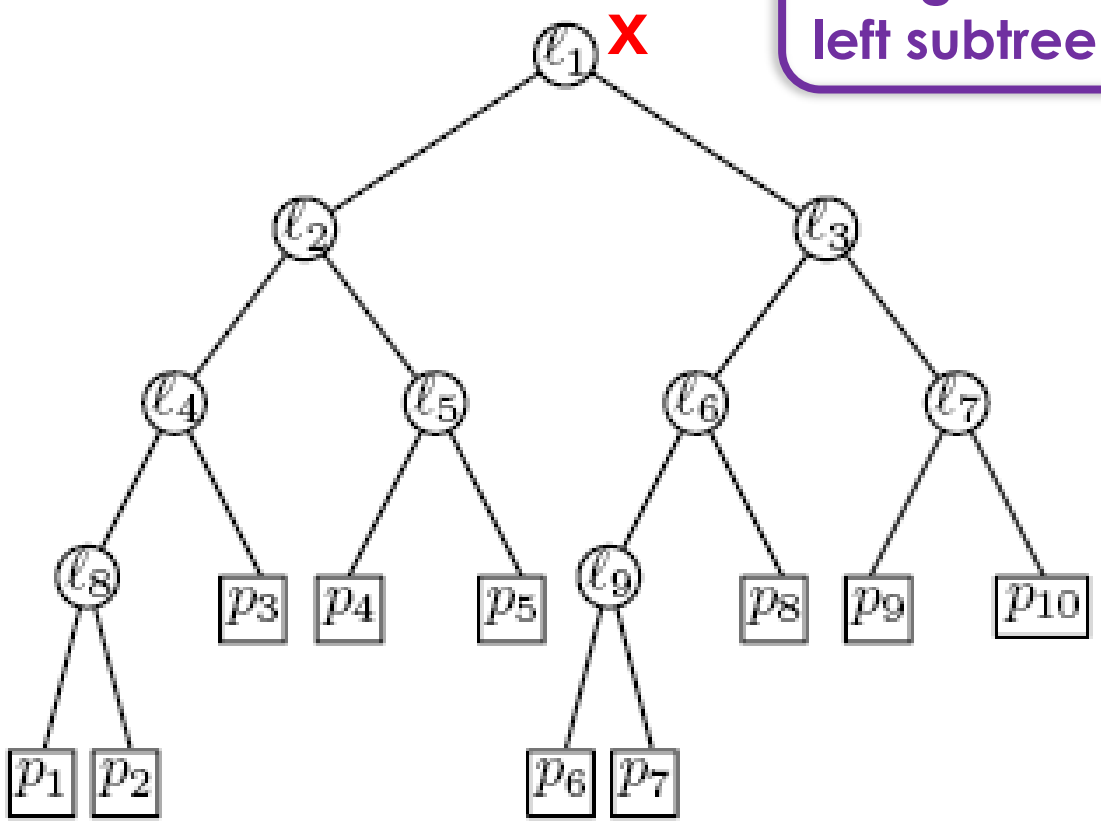
- Split by x-coordinate: split by a vertical line that has (ideally) half the points left or on, and half right.
- Split by y-coordinate: split by a horizontal line that has (ideally) half the points below or on and half above.

Slide adapted from Dr George Bebis Univ Nevada

KD Tree Construction

Split by x-coordinate: split by a vertical line that has approximately half the points left or on, and half right.



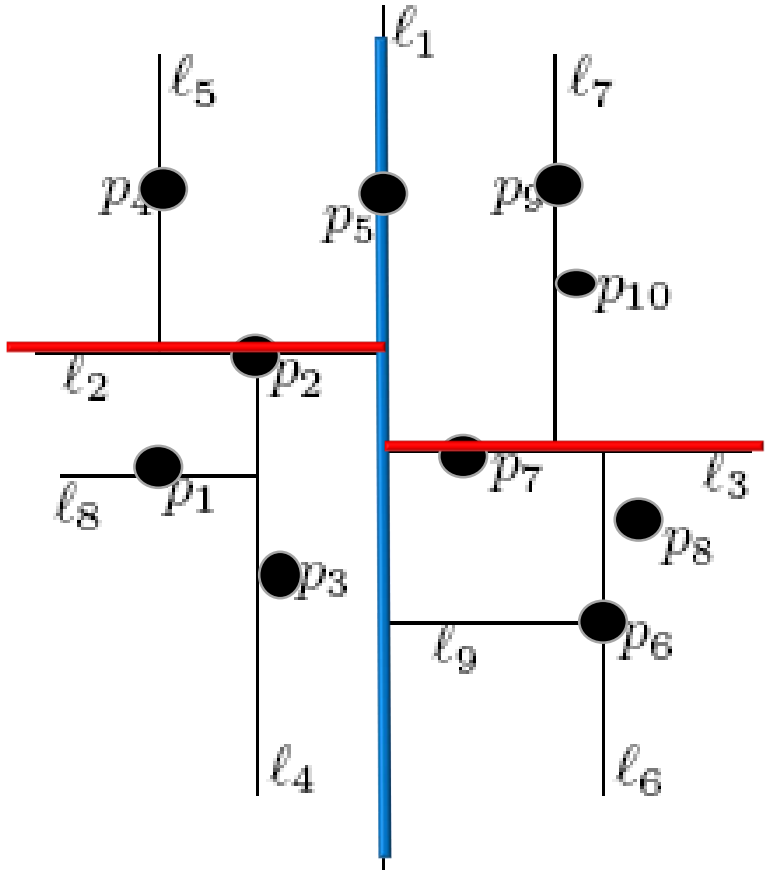


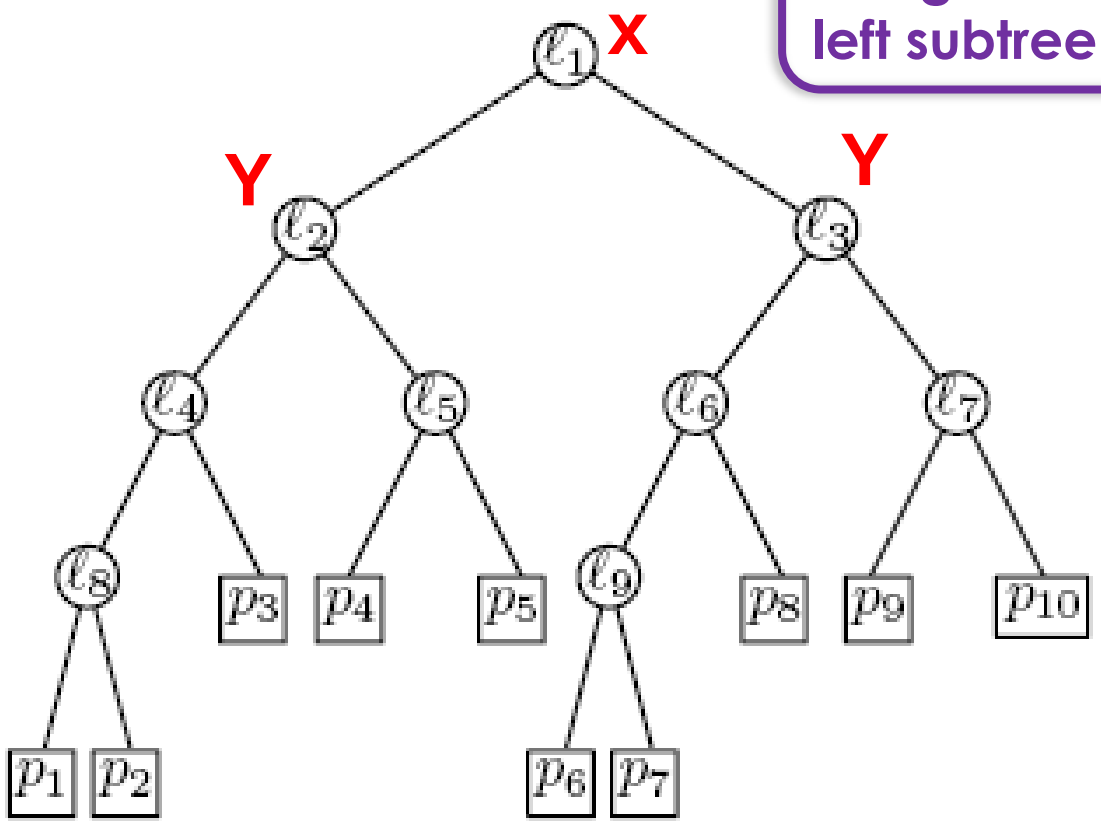
Points on the line go to the left subtree

Slide adapted from Dr George Bebis Univ Nevada

KD Tree Construction

Split by y-coordinate: split by a horizontal line that has approximately half the points left or on, and half right.



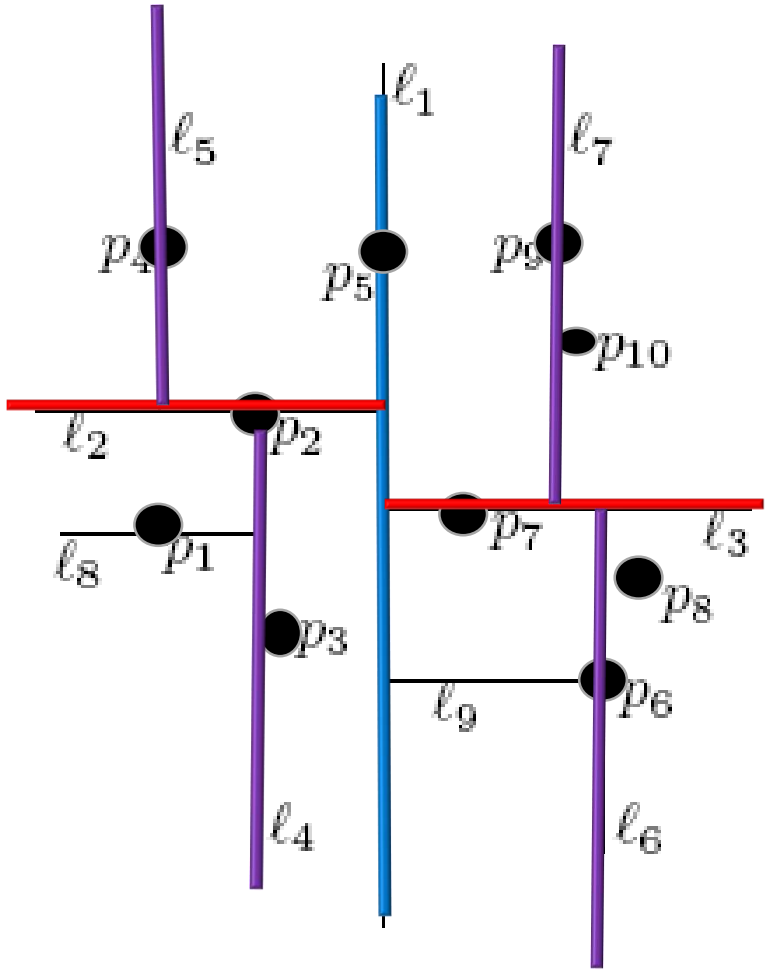


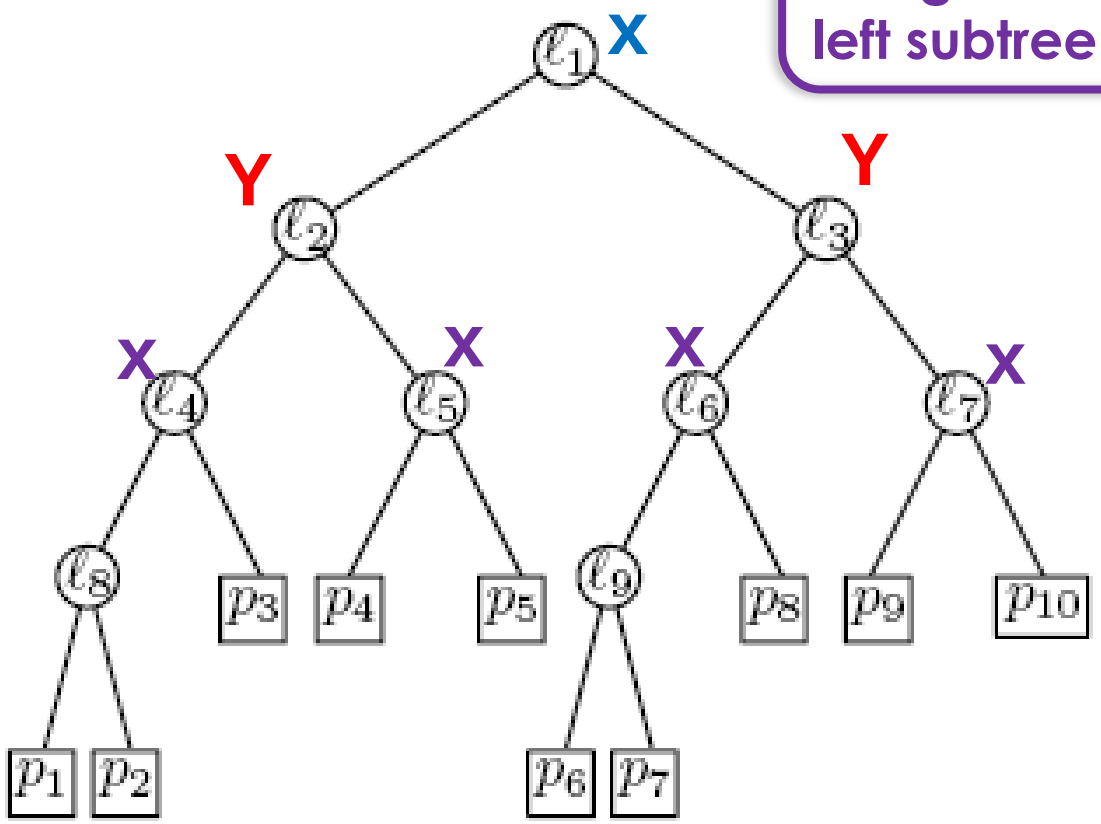
Points on the line go to the left subtree

Slide adapted from Dr George Bebis Univ Nevada

KD Tree Construction

Split by x-coordinate: split by a vertical line that has approximately half the points left or on, and half right.



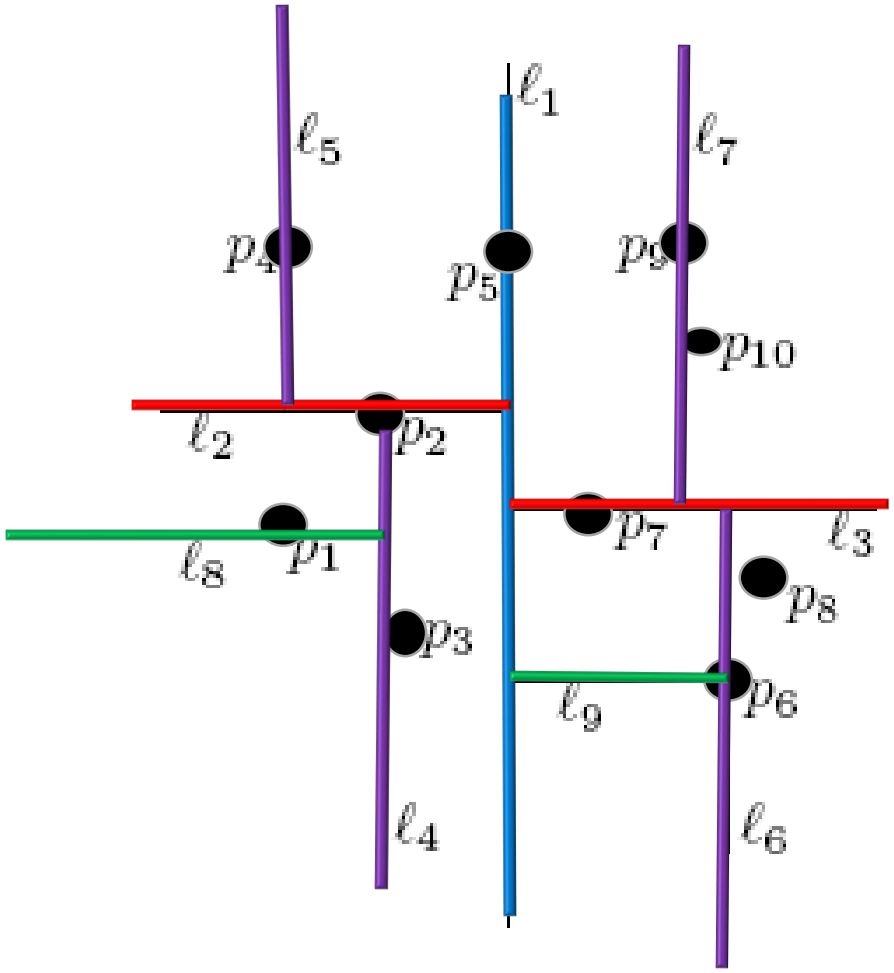


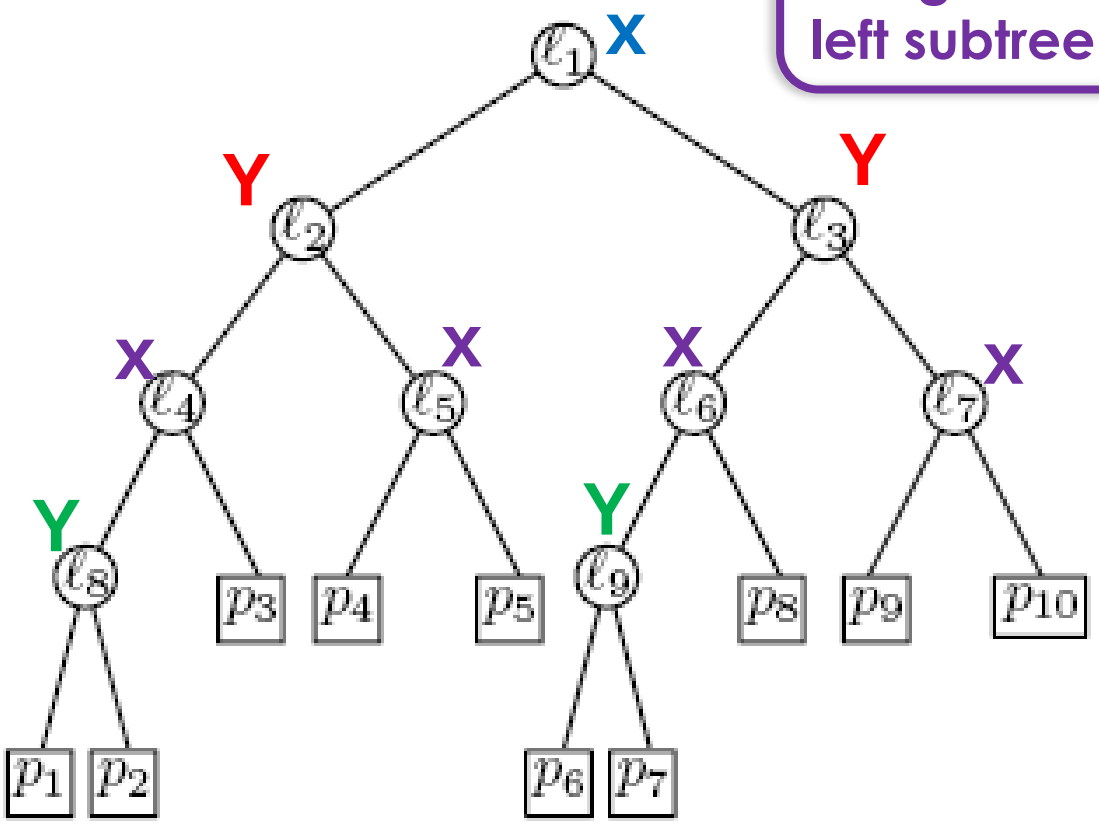
Points on the line go to the left subtree

Slide adapted from Dr George Bebis Univ Nevada

KD Tree Construction

Split by y-coordinate: split by a horizontal line that has approximately half the points left or on, and half right.



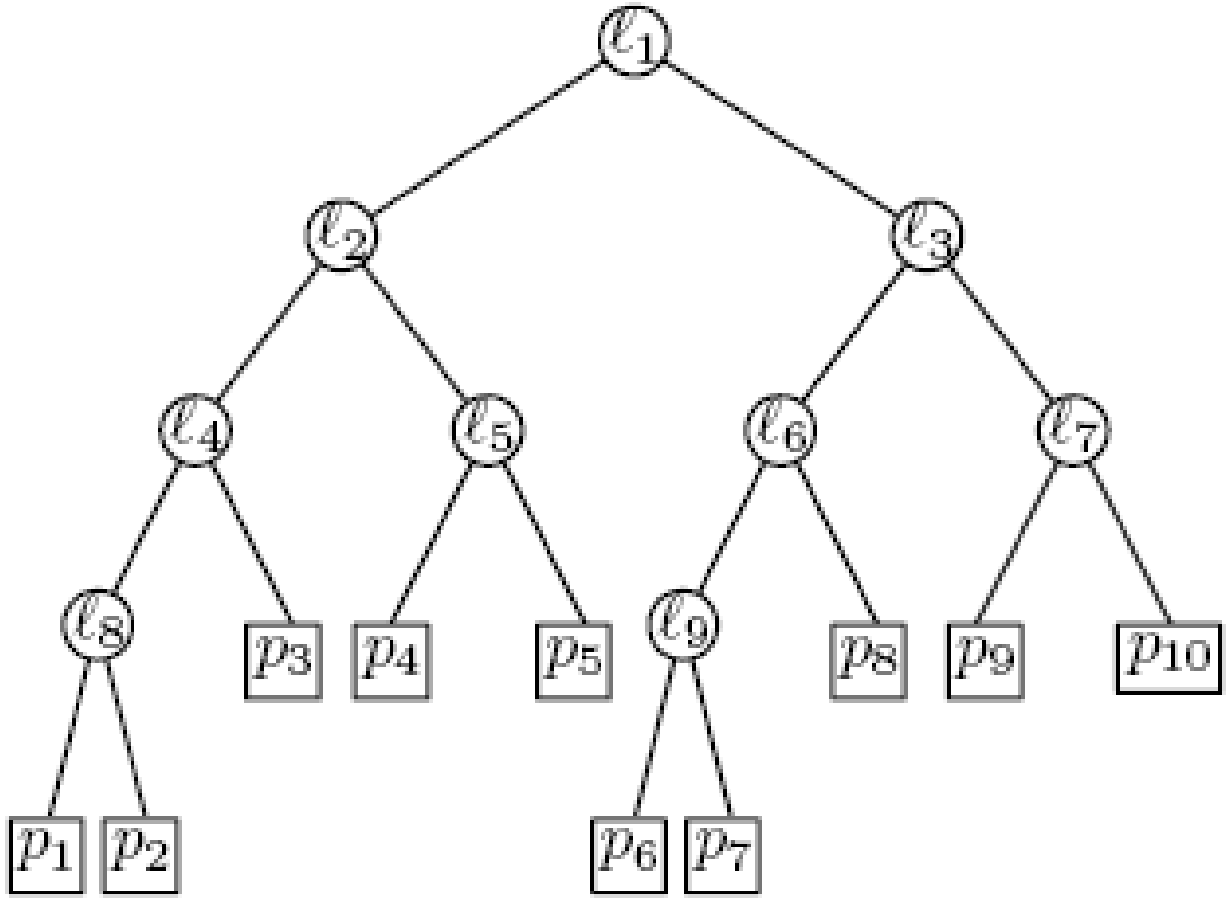


Points on the line go to the left subtree

Slide adapted from Dr George Bebis Univ Nevada

KD Tree Node Structure

- A KD-tree node has 5 fields
  - Splitting axis
  - Splitting value
  - Data
  - Left pointer
  - Right pointer



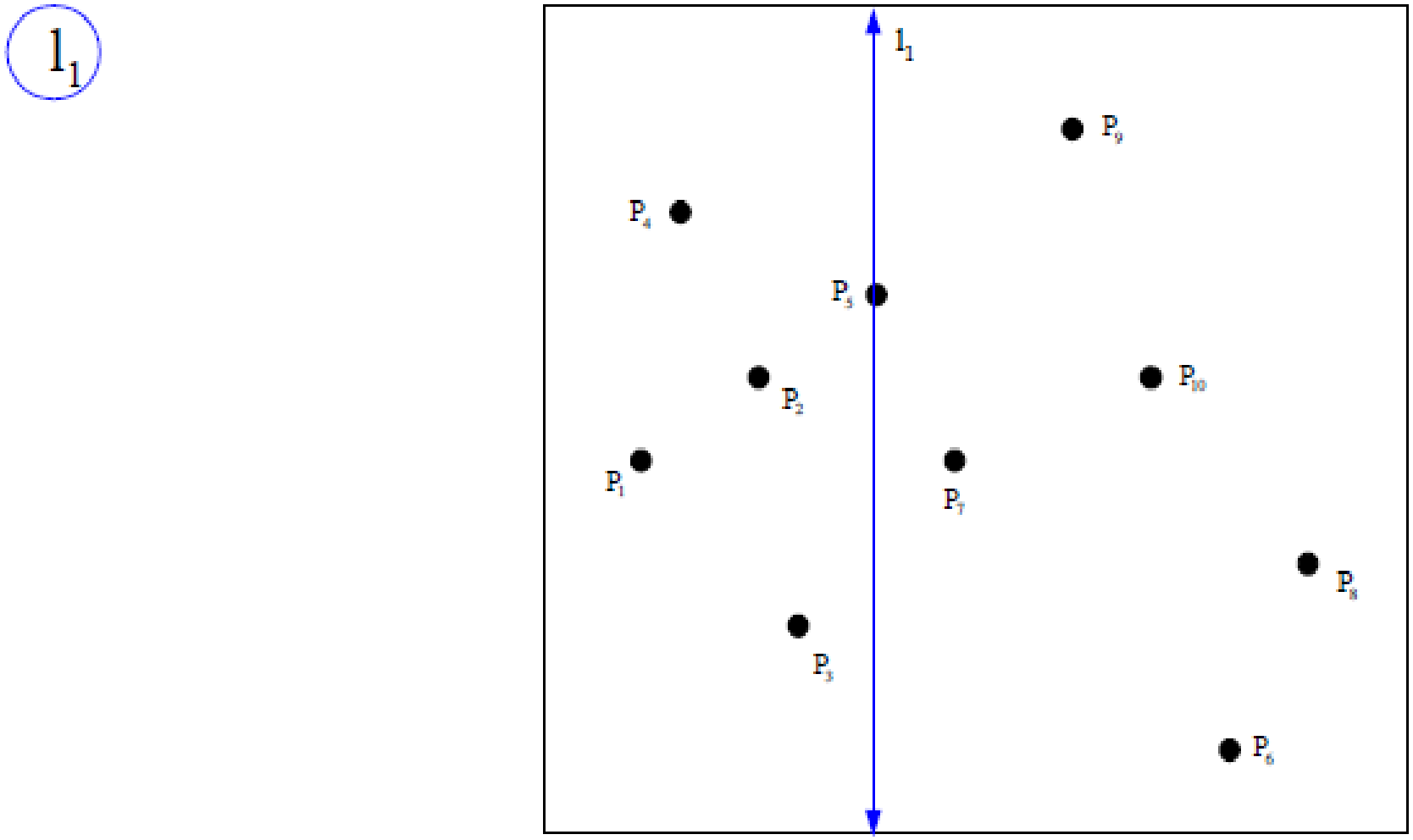
Slide adapted from Dr George Bebis Univ Nevada

KD Tree Splitting Strategies

- **Divide by finding median**
  - Assumes all the points are available ahead of time.
- **Divide perpendicular to the axis with widest spread**
  - Split axes might not alternate
- And many more….

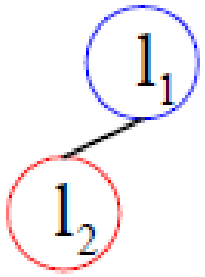
Slide adapted from Dr George Bebis Univ Nevada

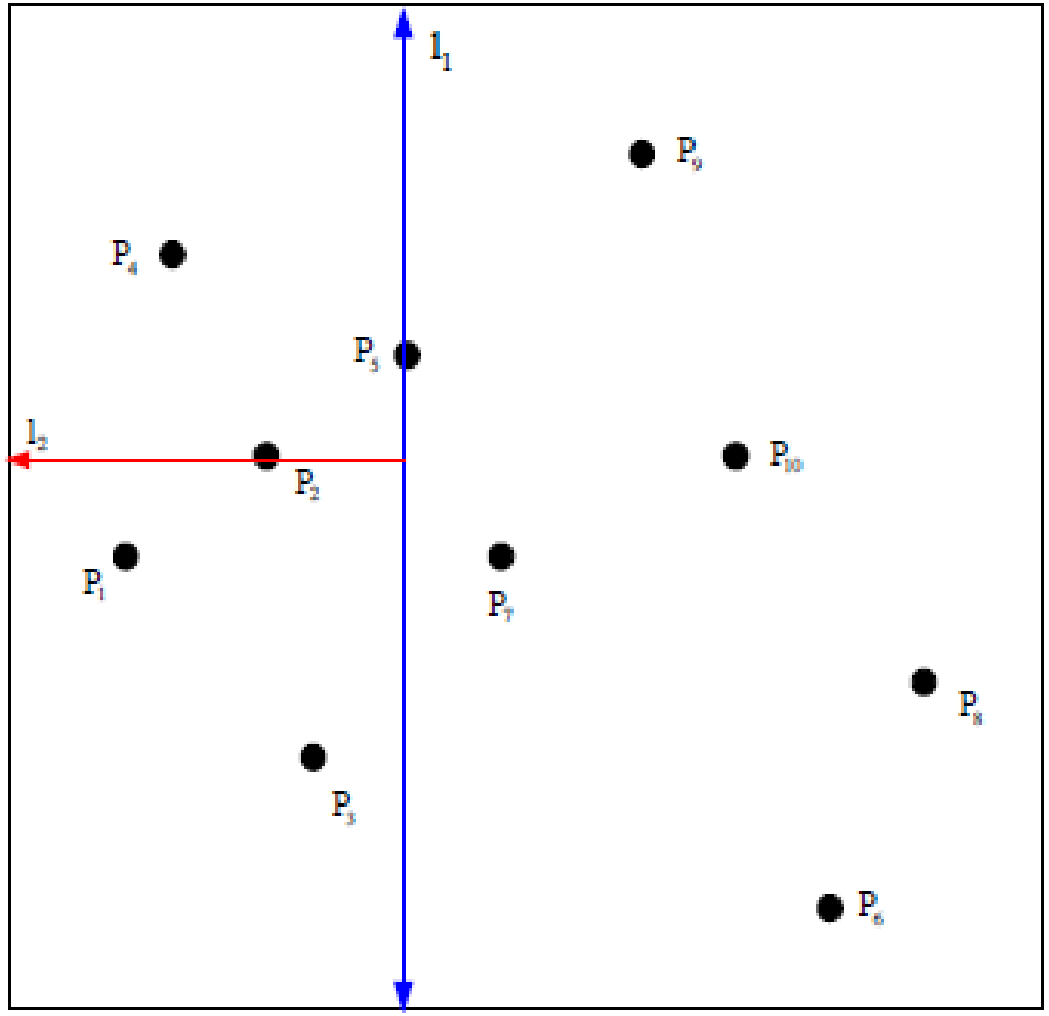
Example – using median (data stored at the leaves)



Slide adapted from Dr George Bebis Univ Nevada

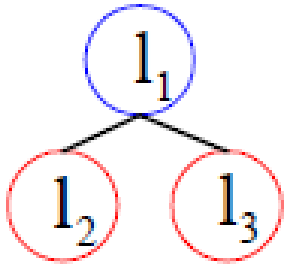
Example – using median (data stored at the leaves)

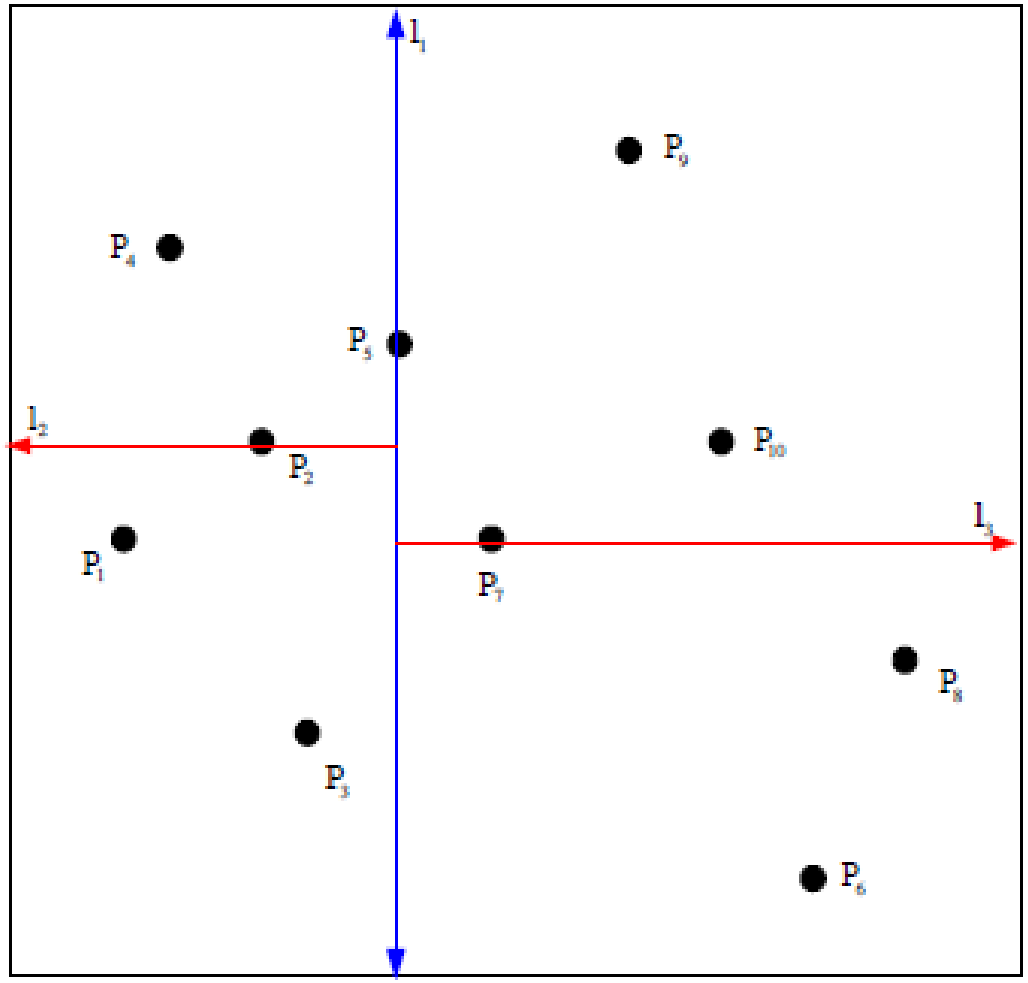




Slide adapted from Dr George Bebis Univ Nevada

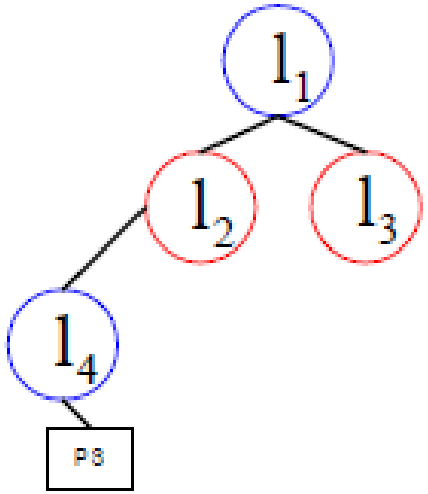
Example – using median (data stored at the leaves)

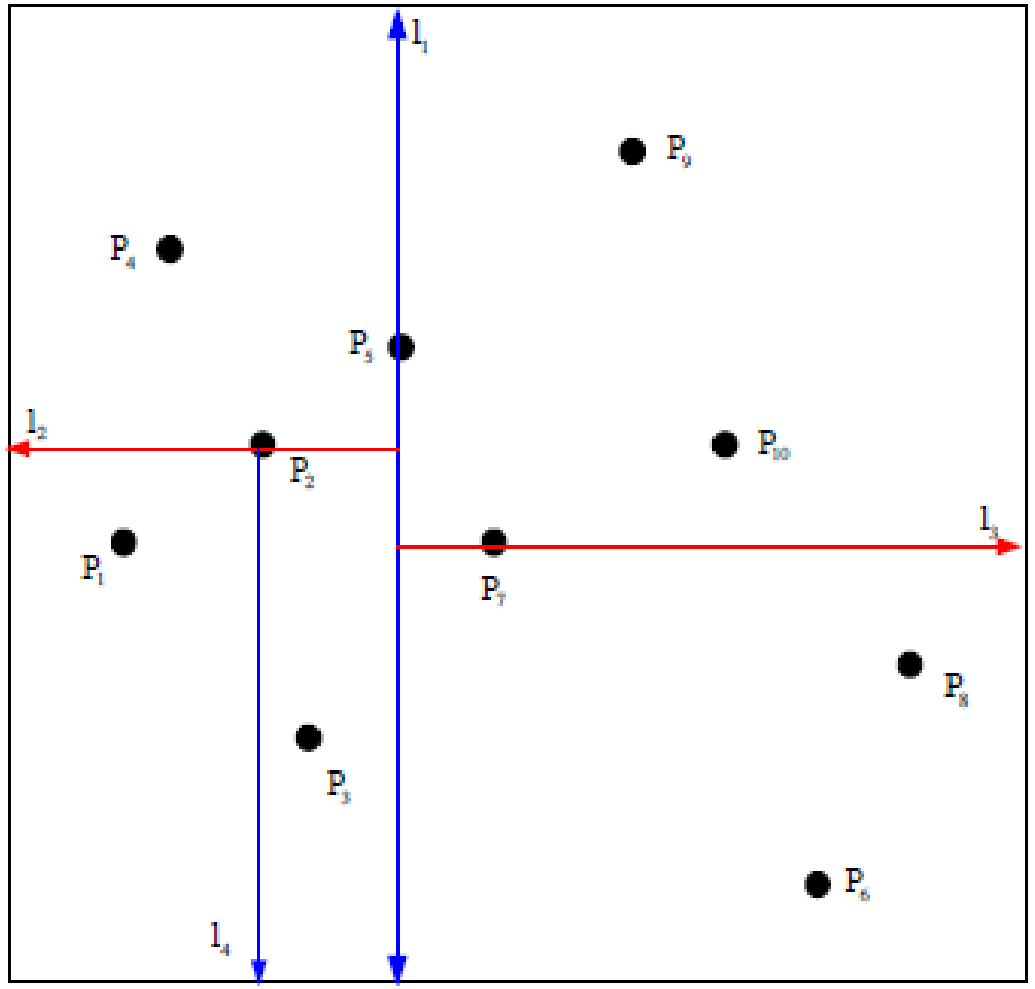




Slide adapted from Dr George Bebis Univ Nevada

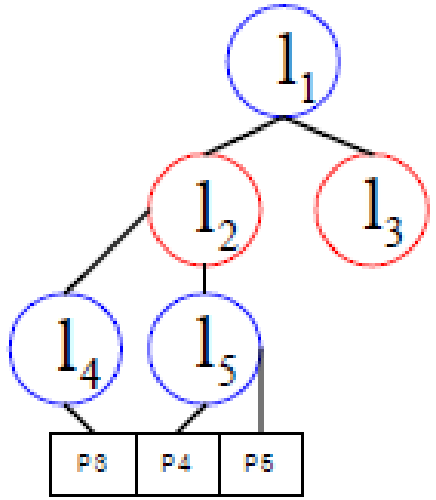
Example – using median (data stored at the leaves)

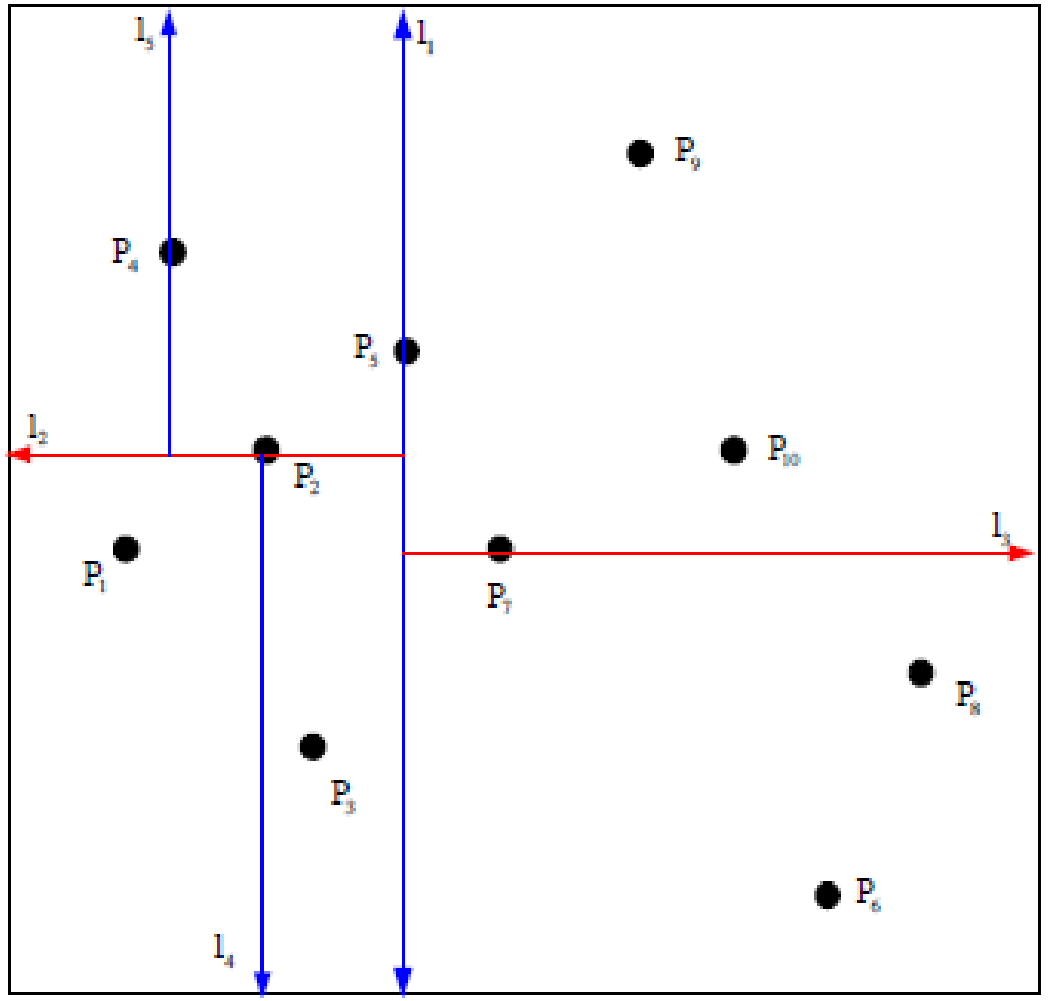




Slide adapted from Dr George Bebis Univ Nevada

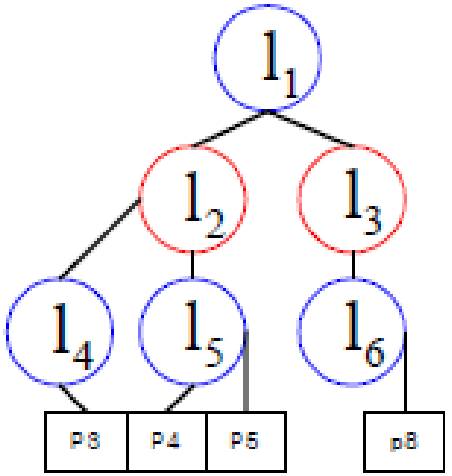
Example – using median (data stored at the leaves)

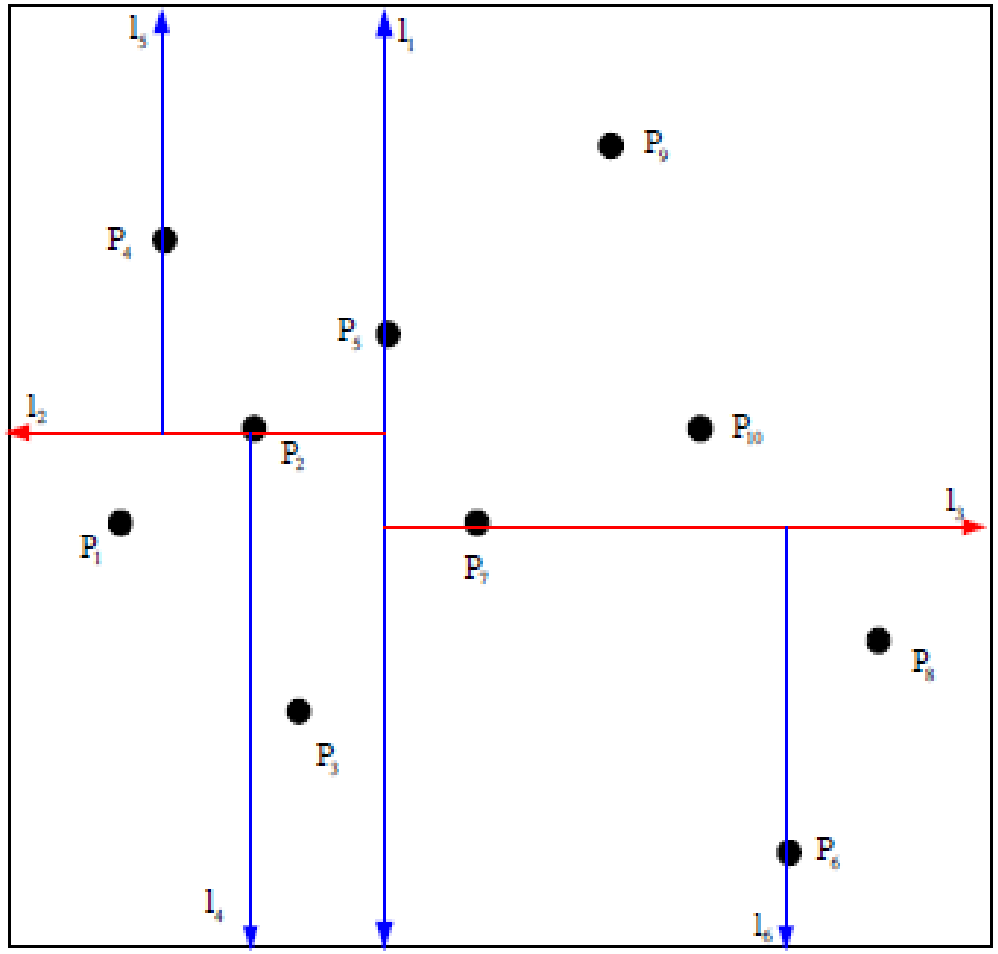




Slide adapted from Dr George Bebis Univ Nevada

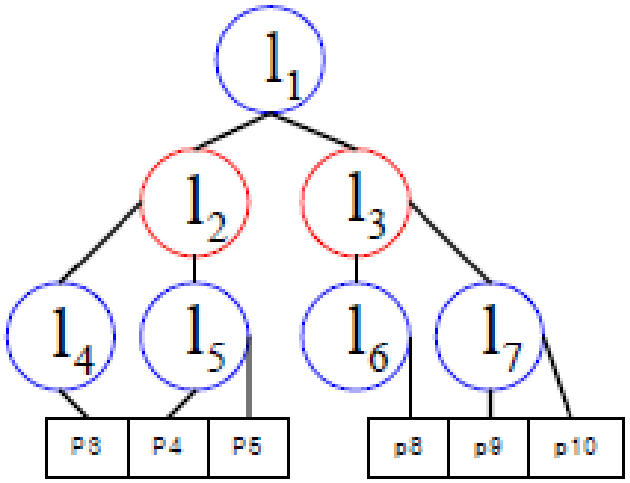
Example – using median (data stored at the leaves)

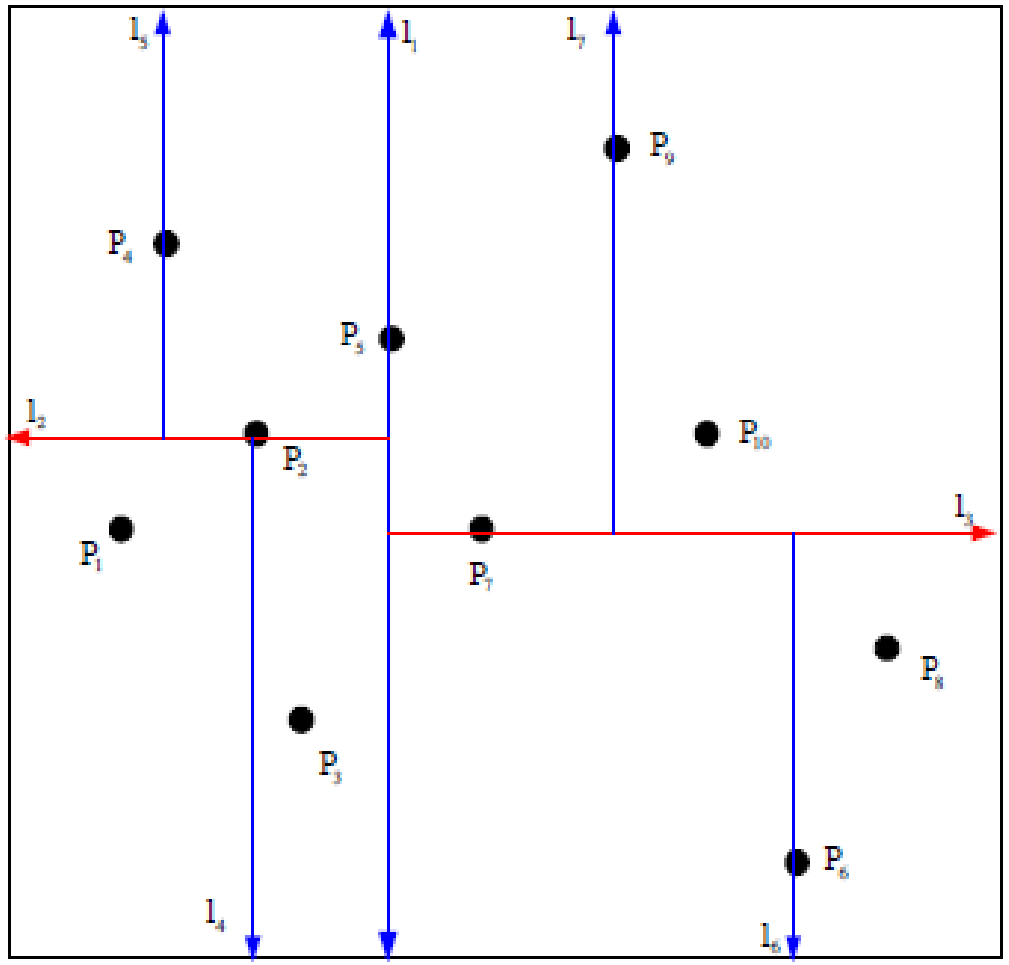




Slide adapted from Dr George Bebis Univ Nevada

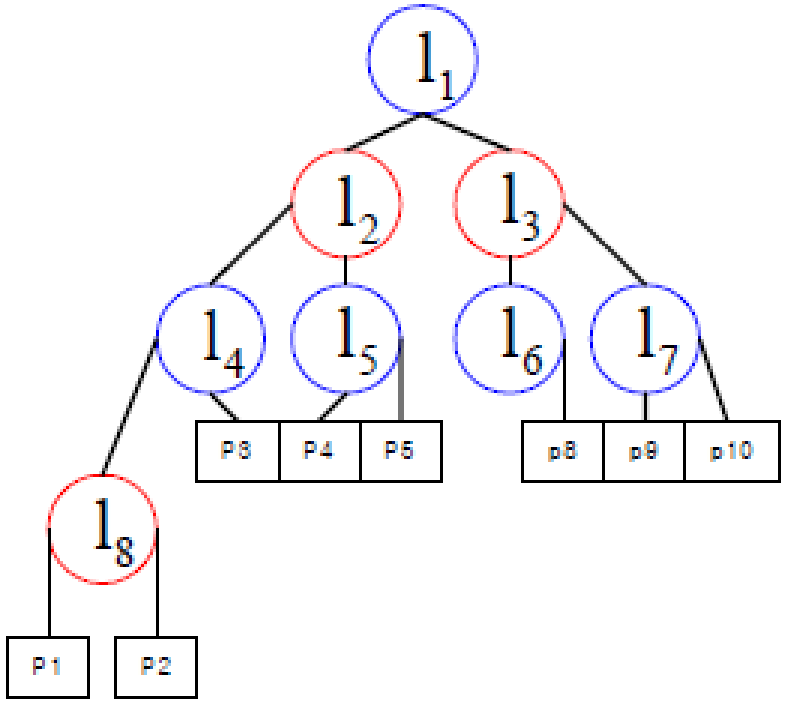
Example – using median (data stored at the leaves)

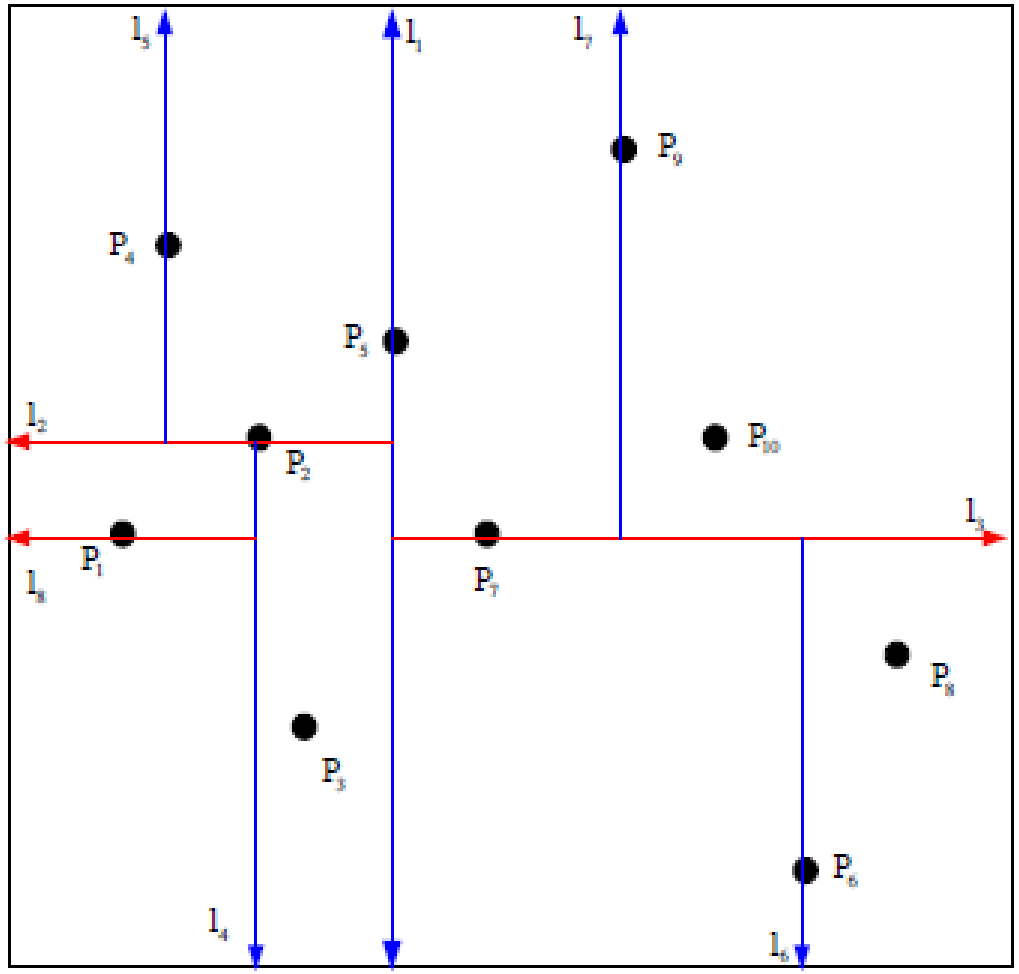




Slide adapted from Dr George Bebis Univ Nevada

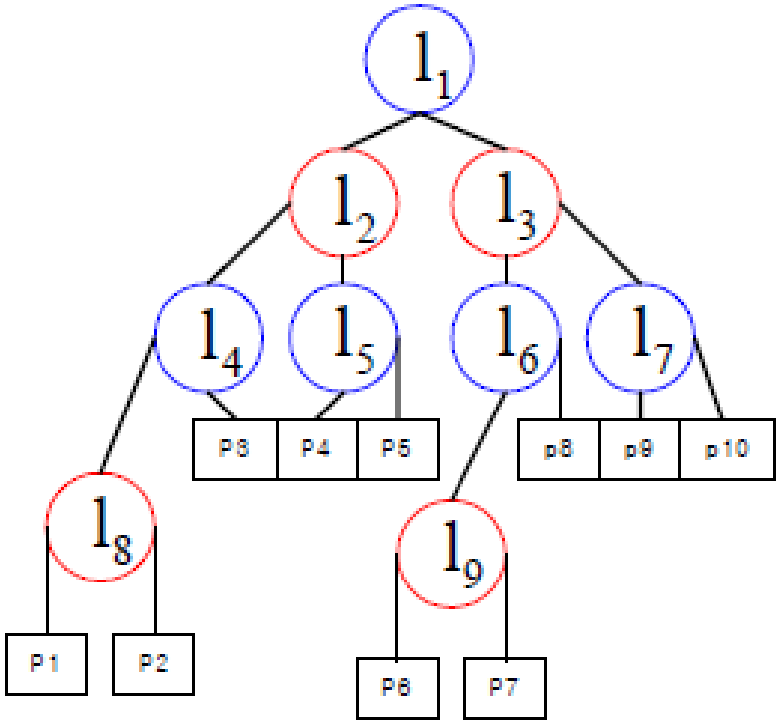
Example – using median (data stored at the leaves)

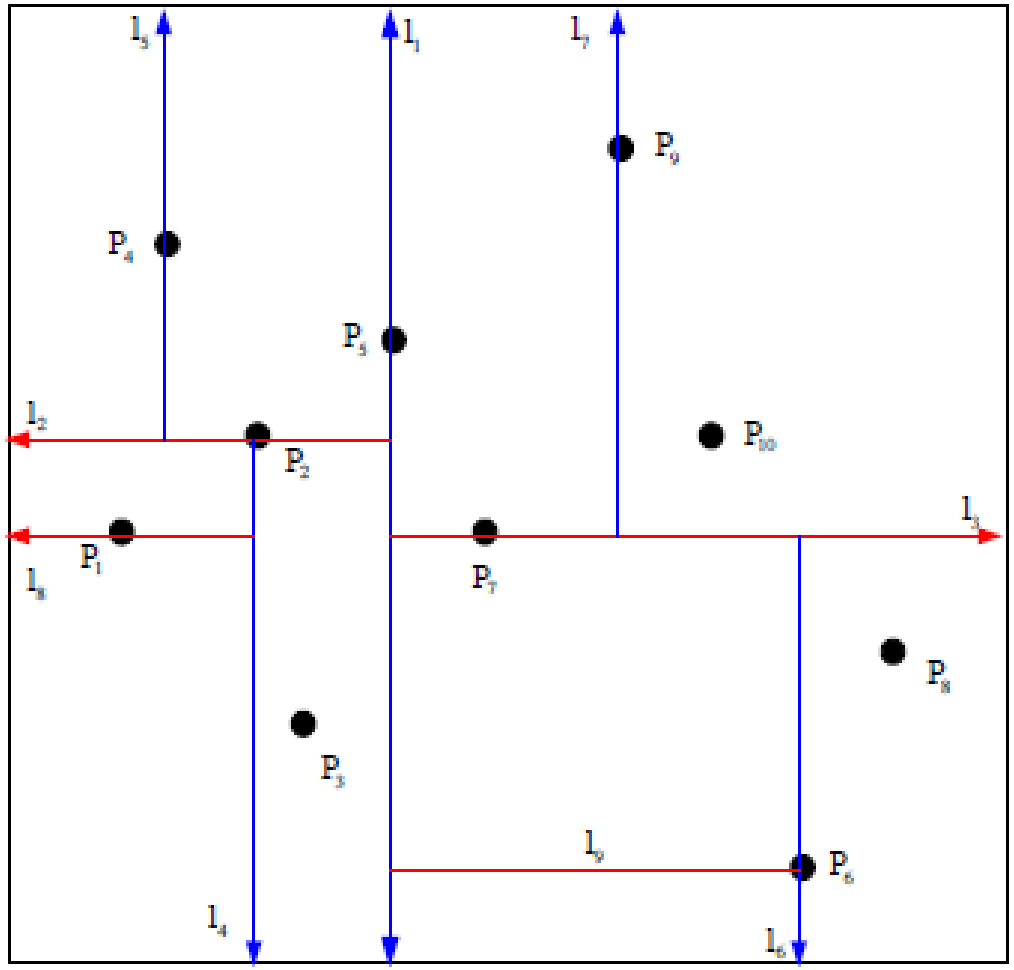




Slide adapted from Dr George Bebis Univ Nevada

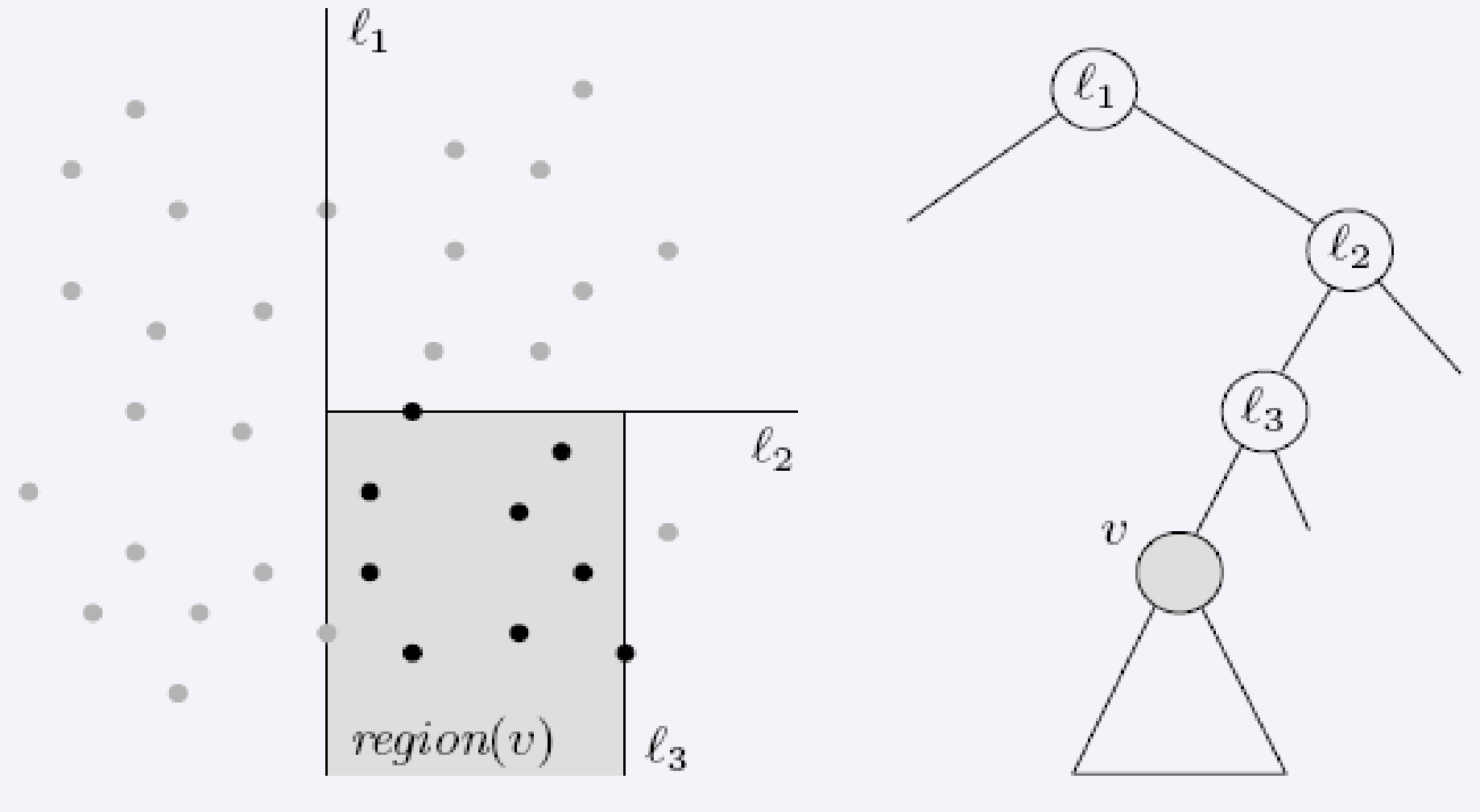
Example – using median (data stored at the leaves)





Slide adapted from Dr George Bebis Univ Nevada

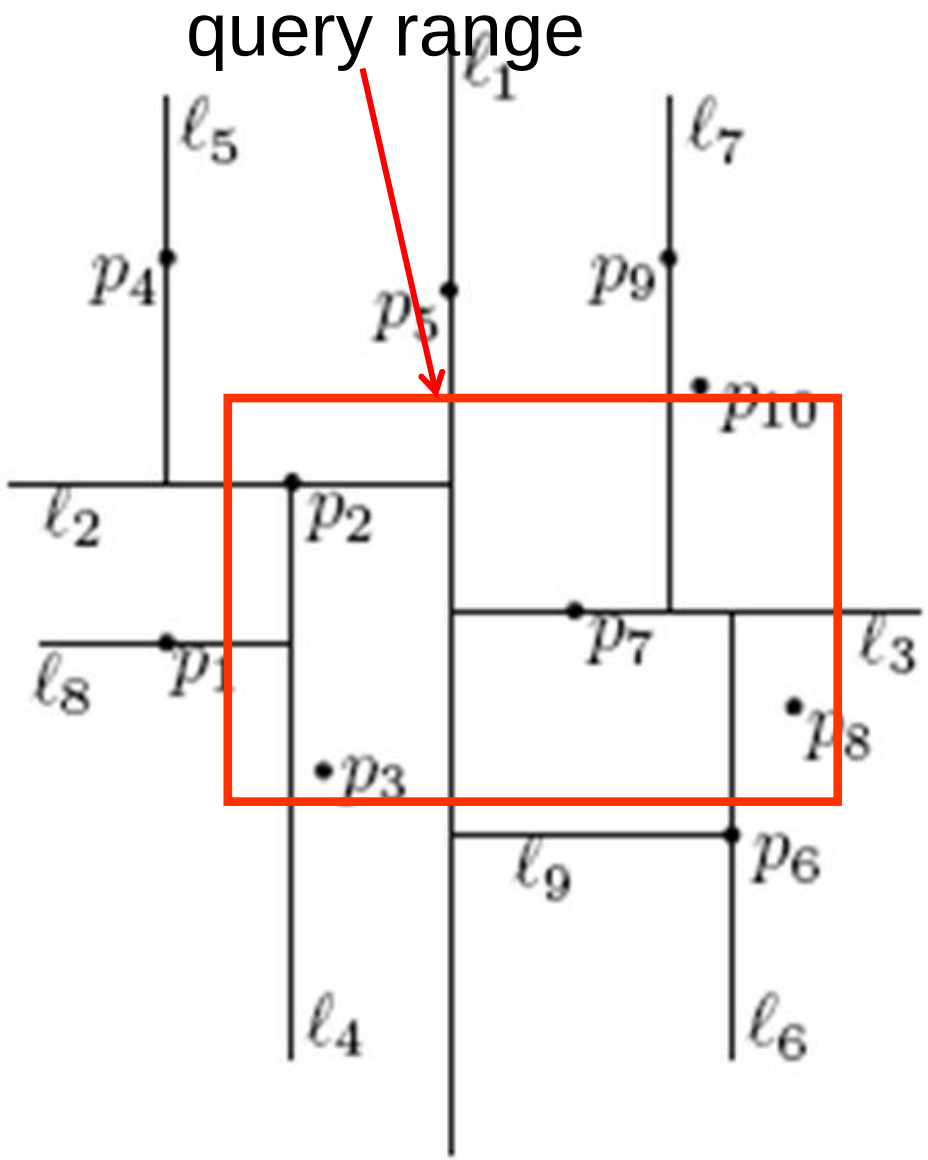
Region of node **v**



Region(v) : the subtree rooted at **v** stores the points in black dots

KD Trees – Range Search

- Need only search nodes whose region intersects query region.
  - Report all points in subtrees whose regions are entirely contained in query range.
  - If a region is partially contained in the query range check points.



Slide adapted from Dr George Bebis Univ Nevada

KD-tree: range queries

- Recursive procedure starting from **v = root**

SearchKDtree (v, query rectangle R)

1. **If v** is a leaf, then report the points stored in **v** if it lies in **R**
2. **Else**

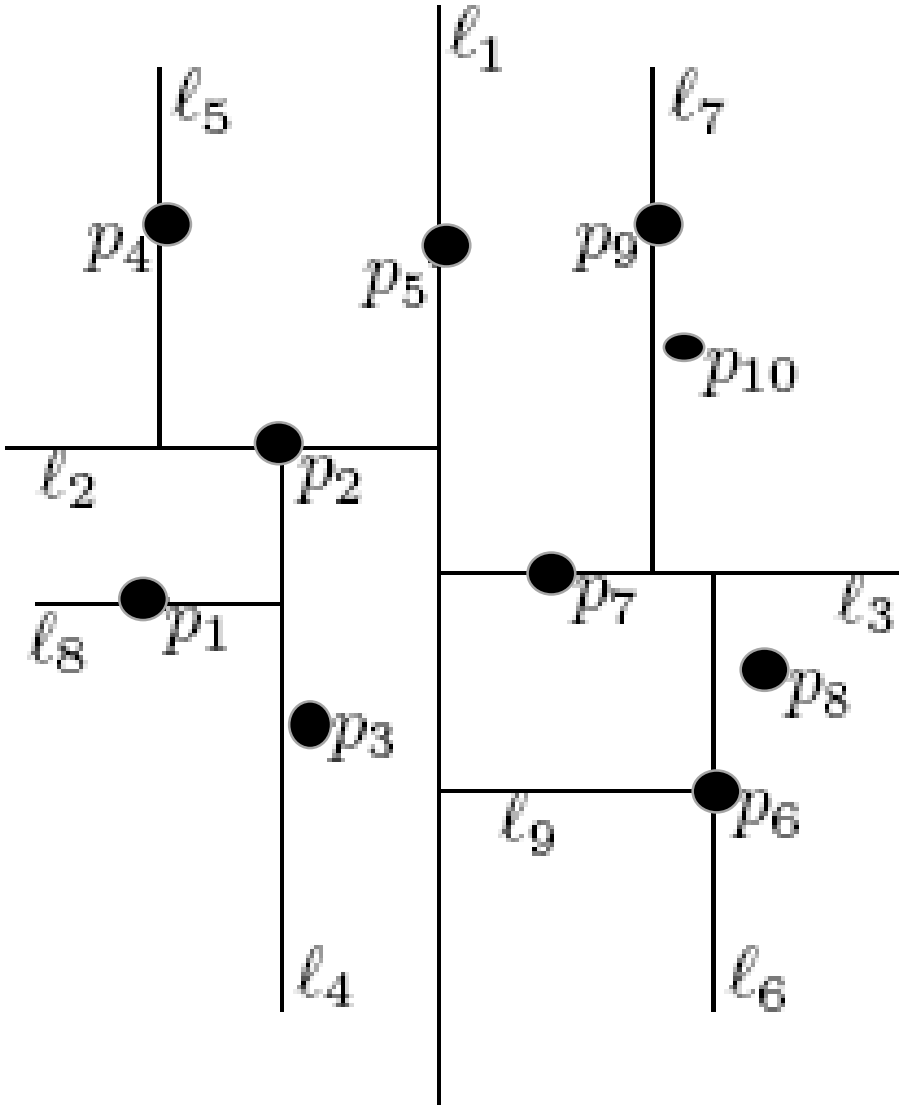
   2.1 if Region(lc(v)) is contained in **R**, then report all points in the subtree(v).

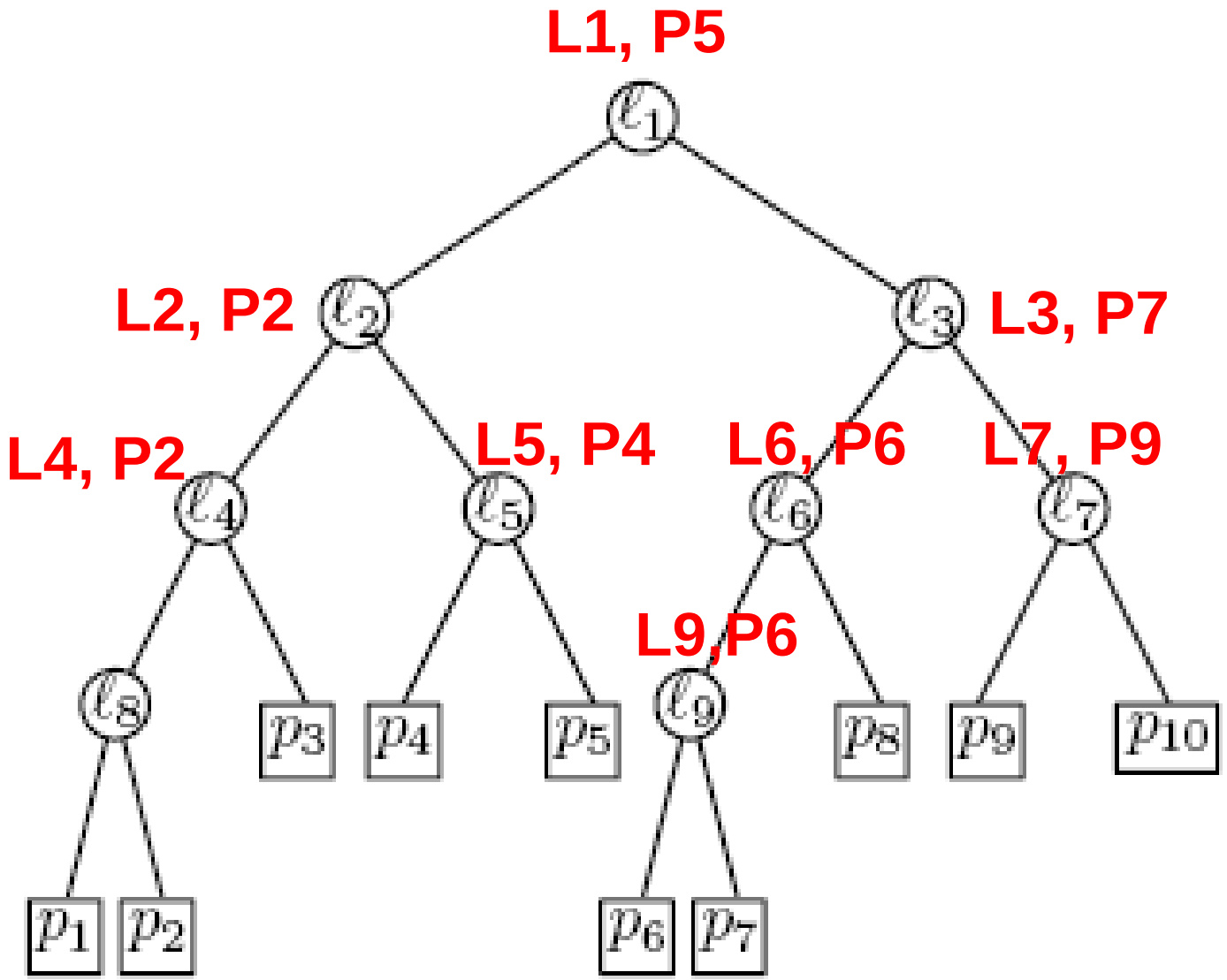
   2.2 Else If Region(lc(v)) intersects with **R**, then SearchKDtree(lc(v), **R**)

   2.3 If Region(rc(v)) is contained in R, then report all points in the subtree(v). intersects **R**

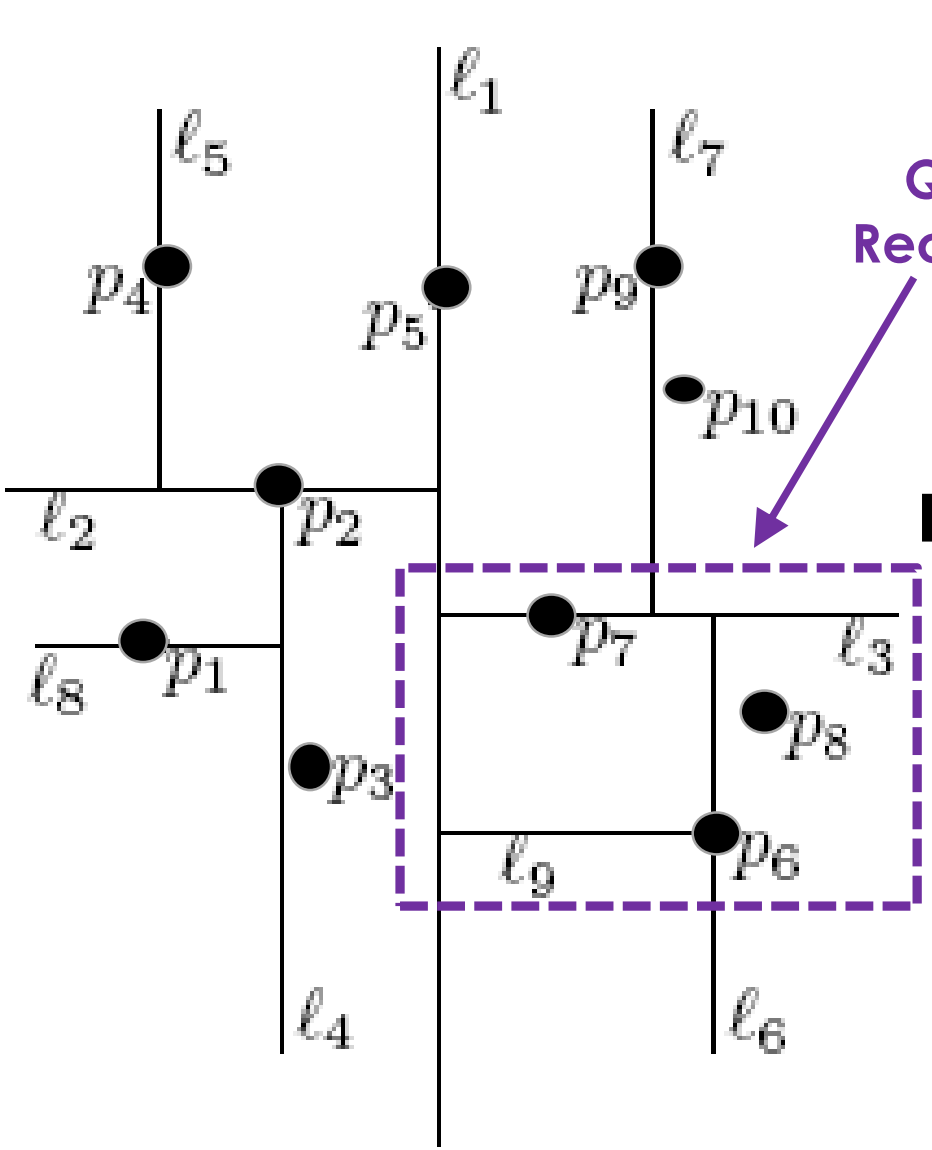
   2. 4 Else if Region(rc(v)) intersects **R**, then SearchKDtree(rc(v), **R**)

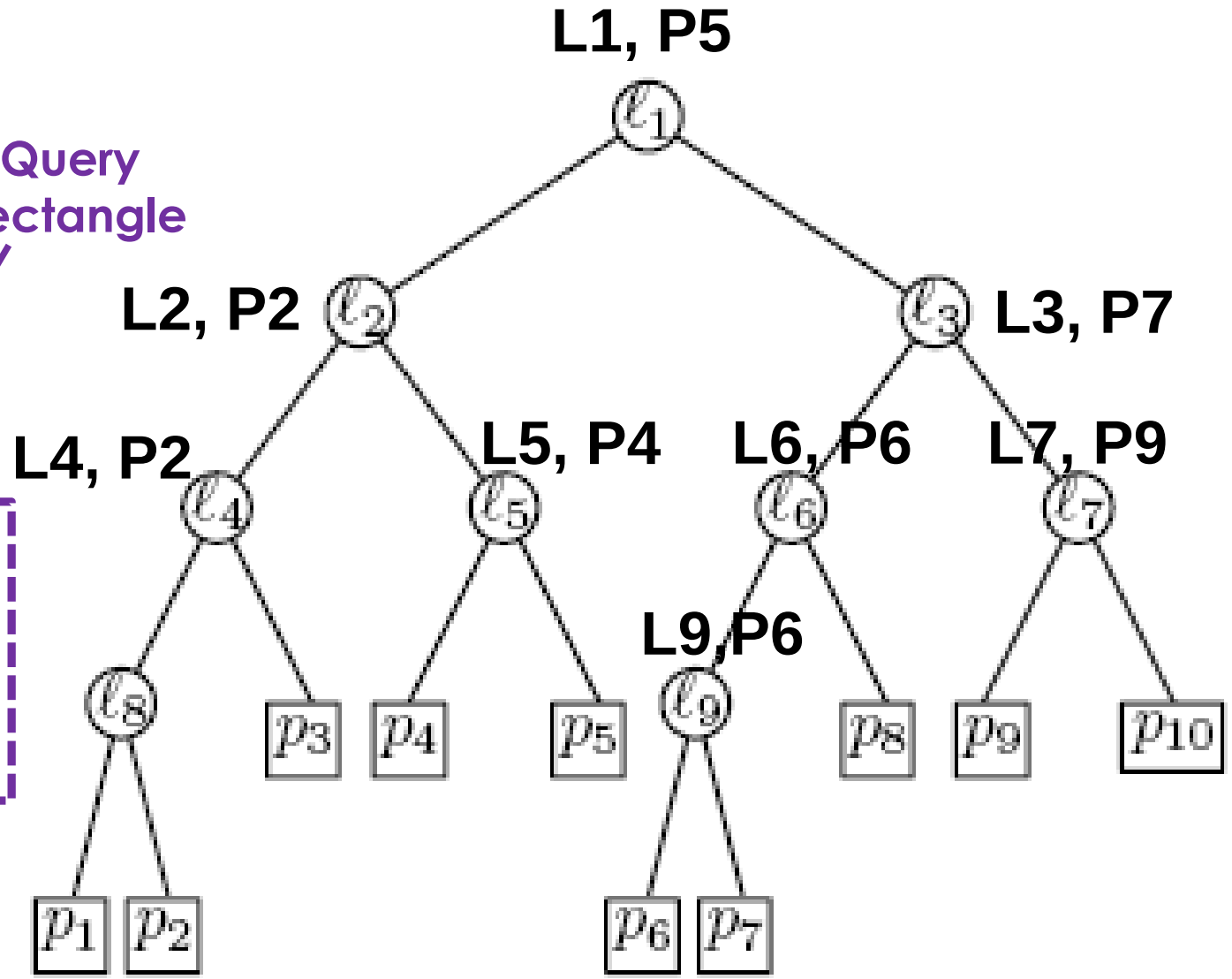
KD-tree: Range Query Example



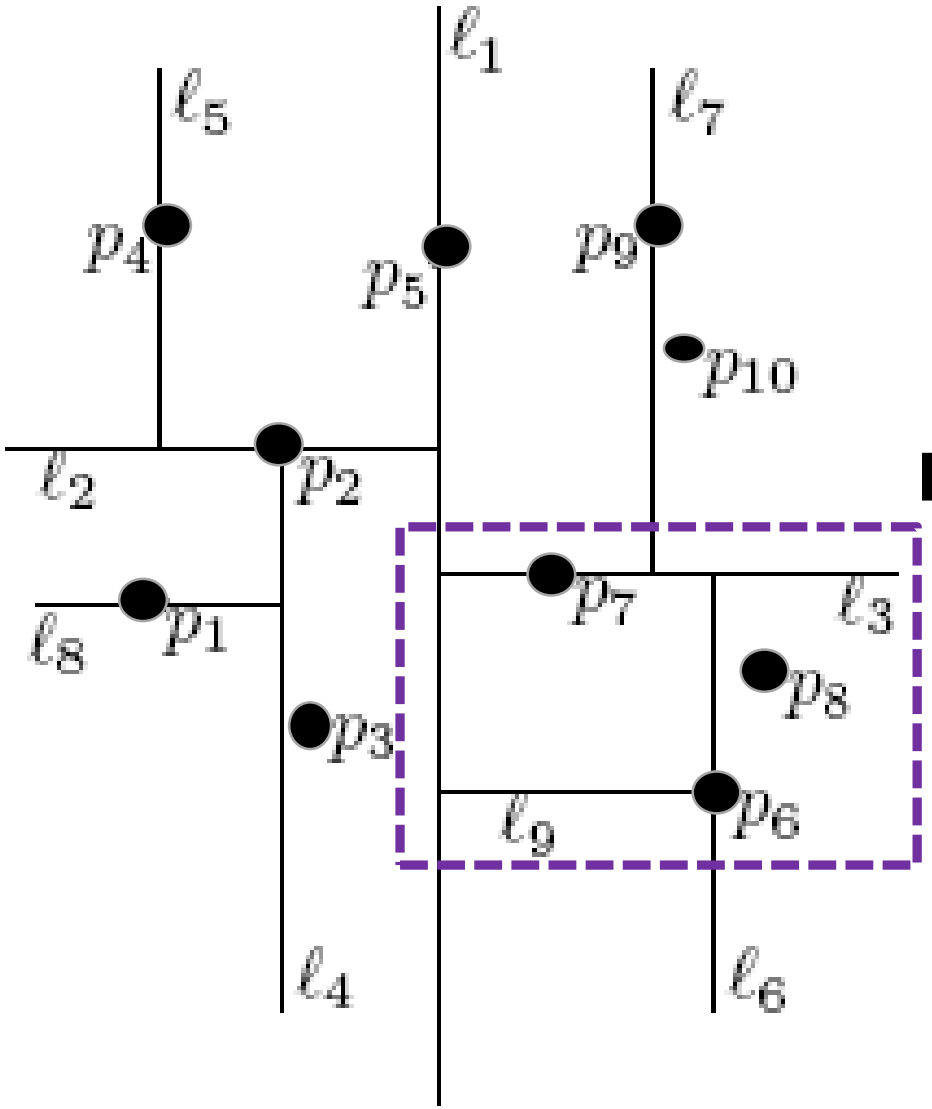


KD-tree: Range Query Example

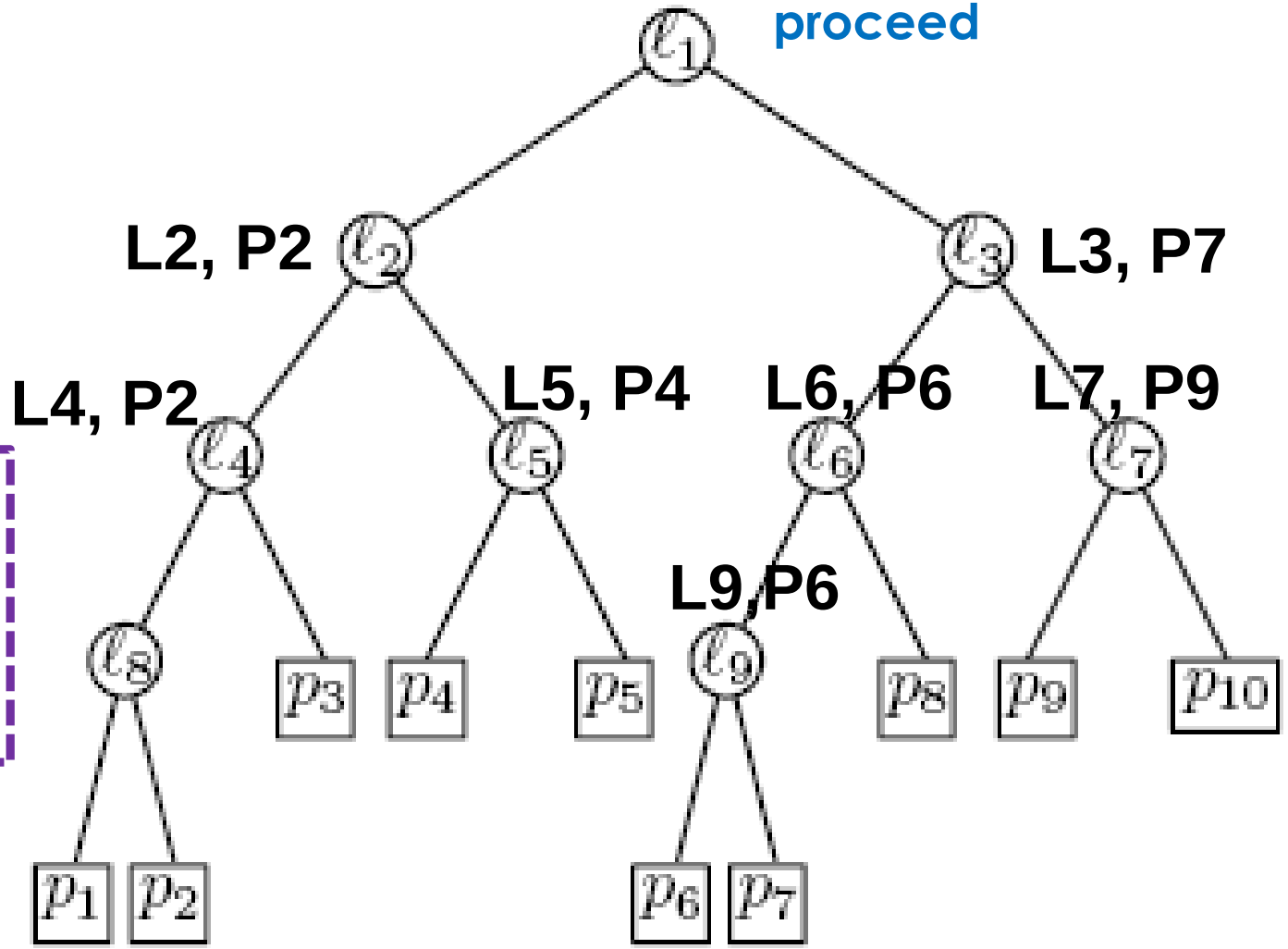




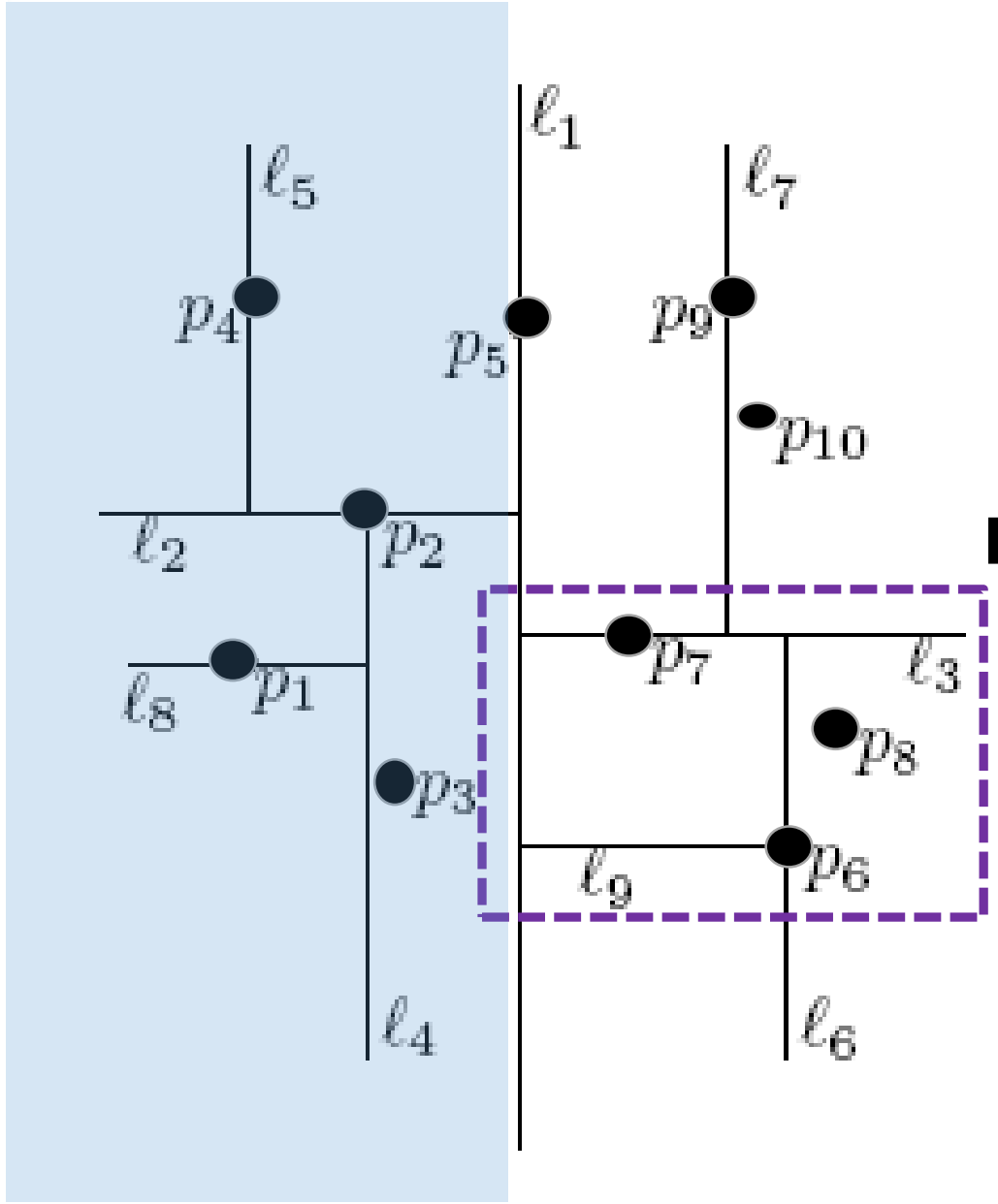
KD-tree: Range Query Example



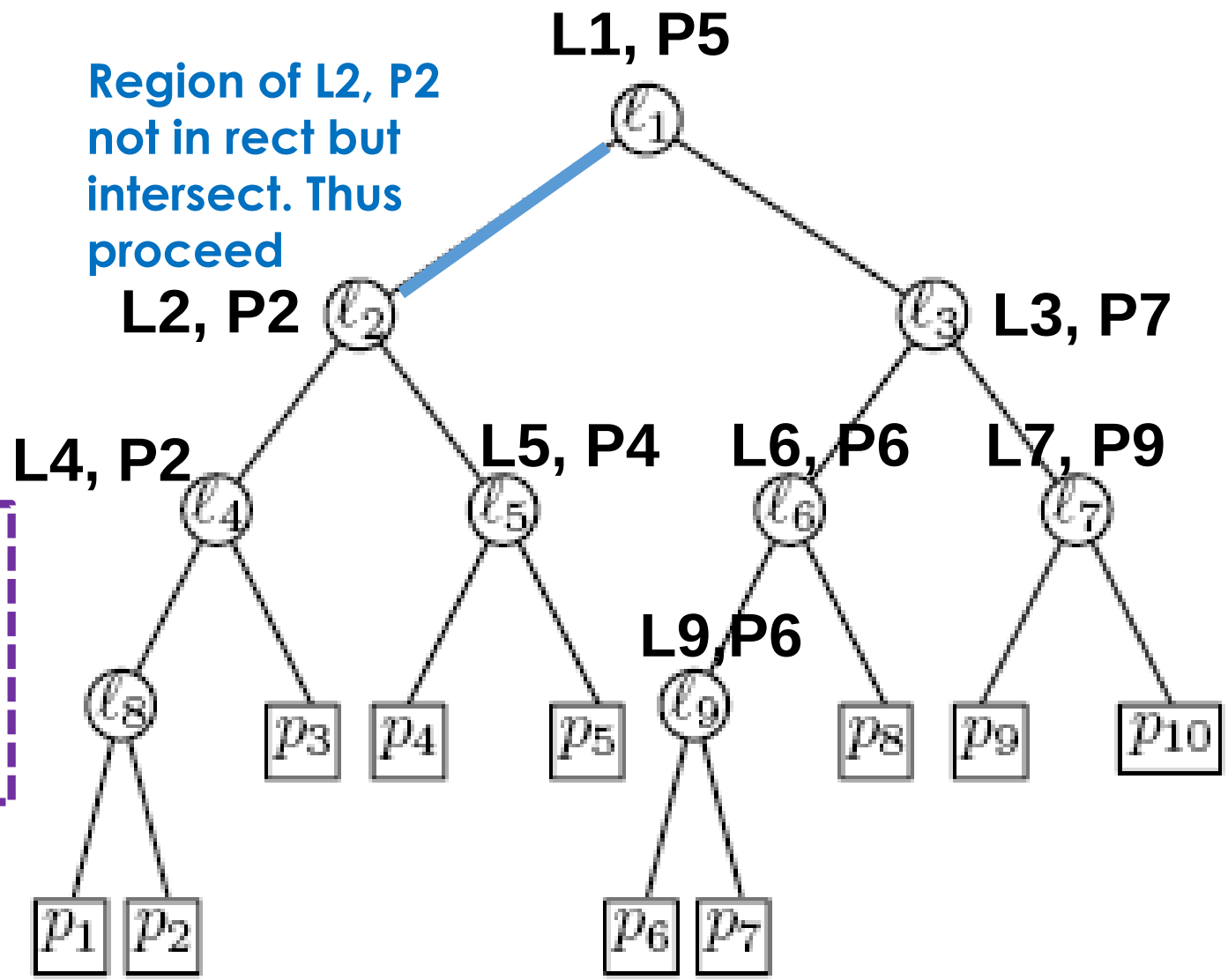
Intersection with region of P5. Thus proceed



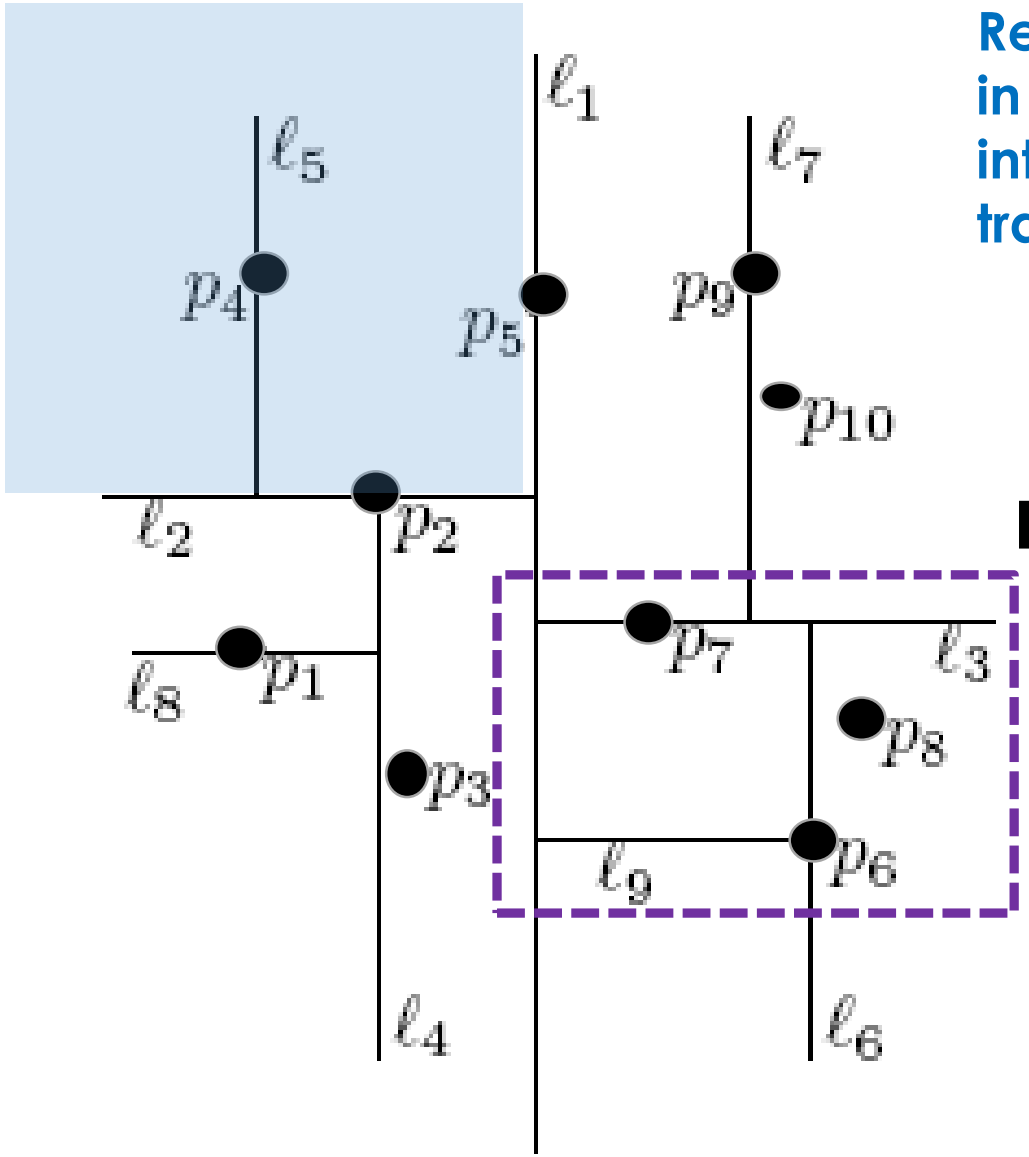
KD-tree: Range Query Example



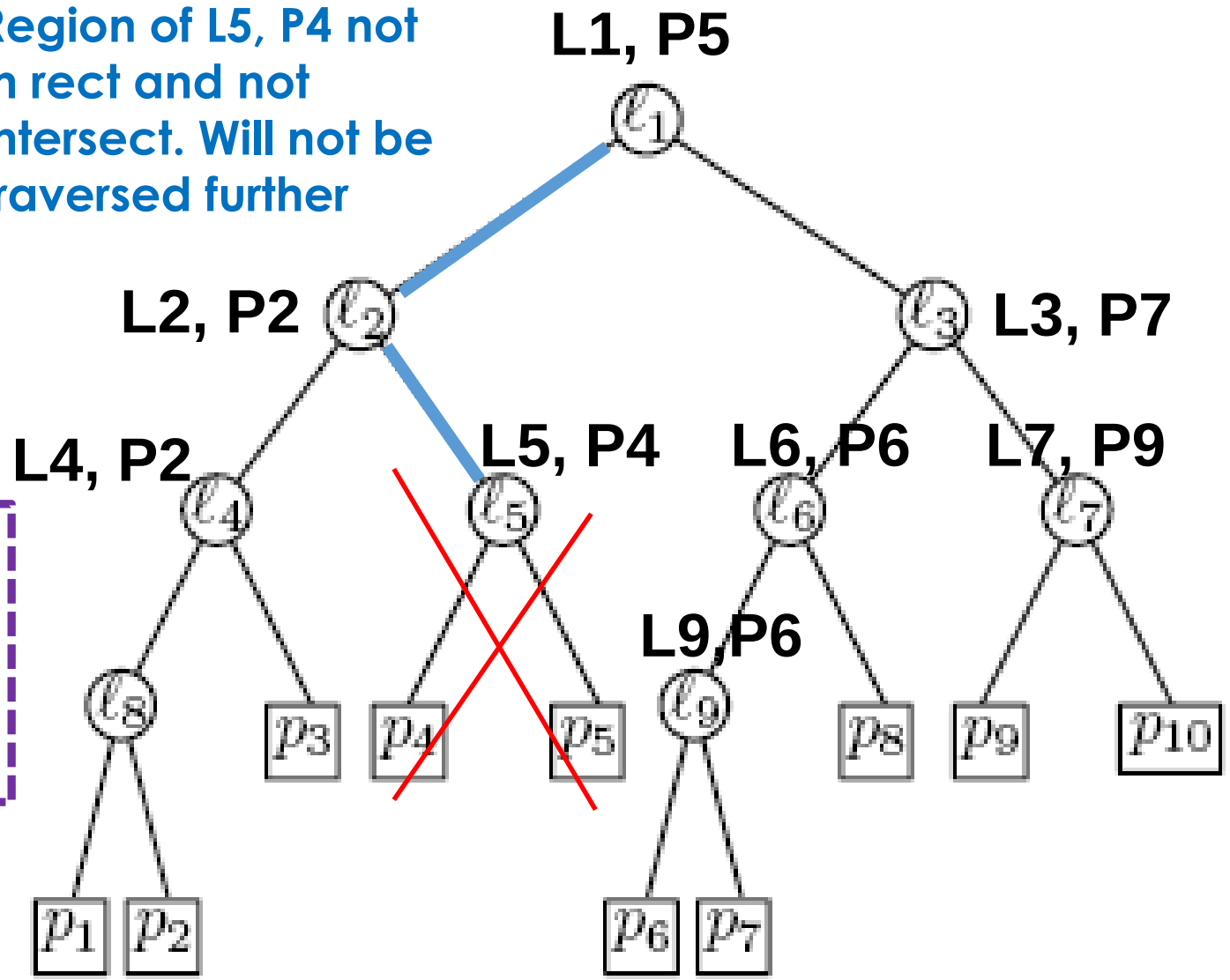
Region of L2, P2 not in rect but intersect. Thus proceed



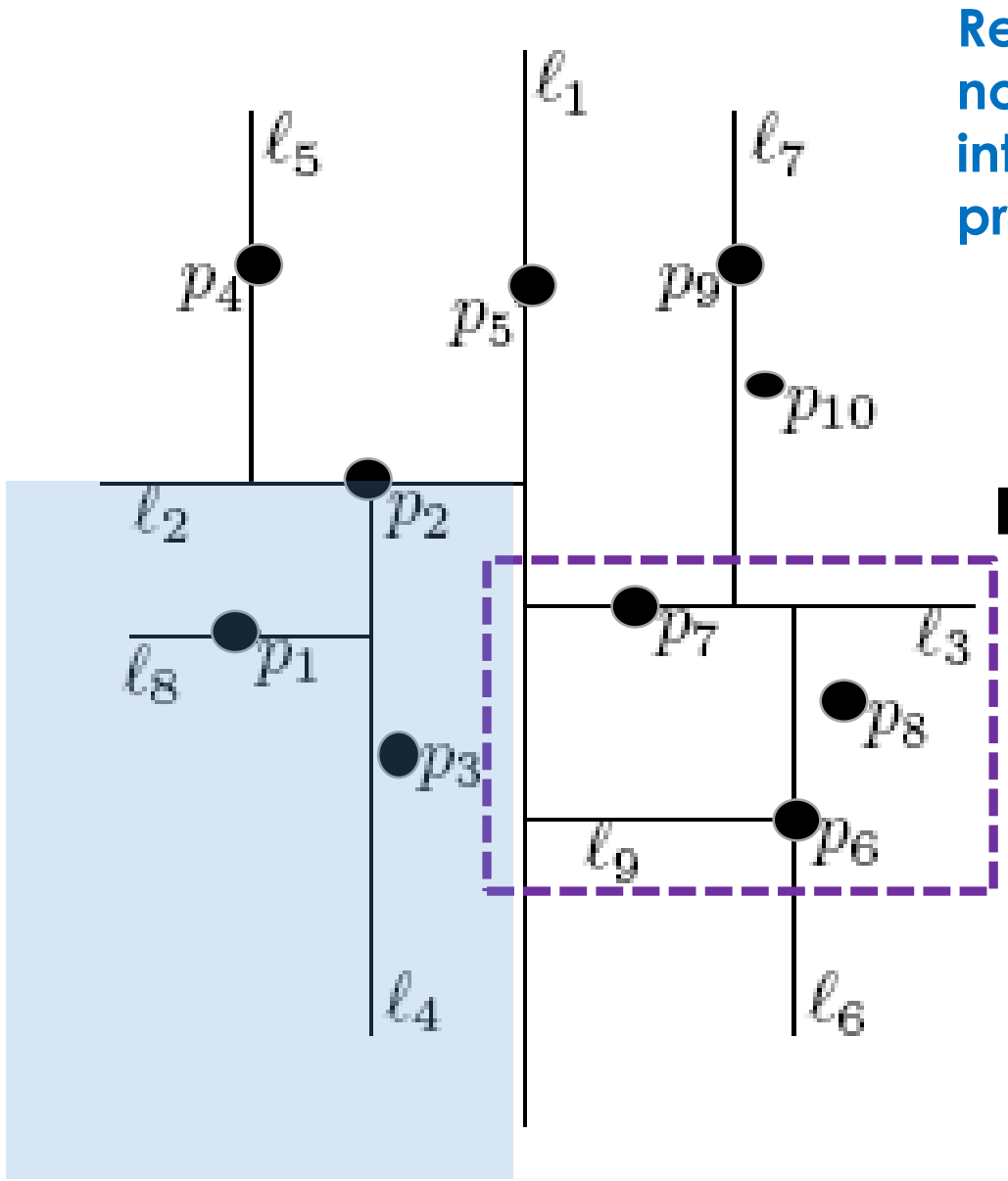
KD-tree: Range Query Example



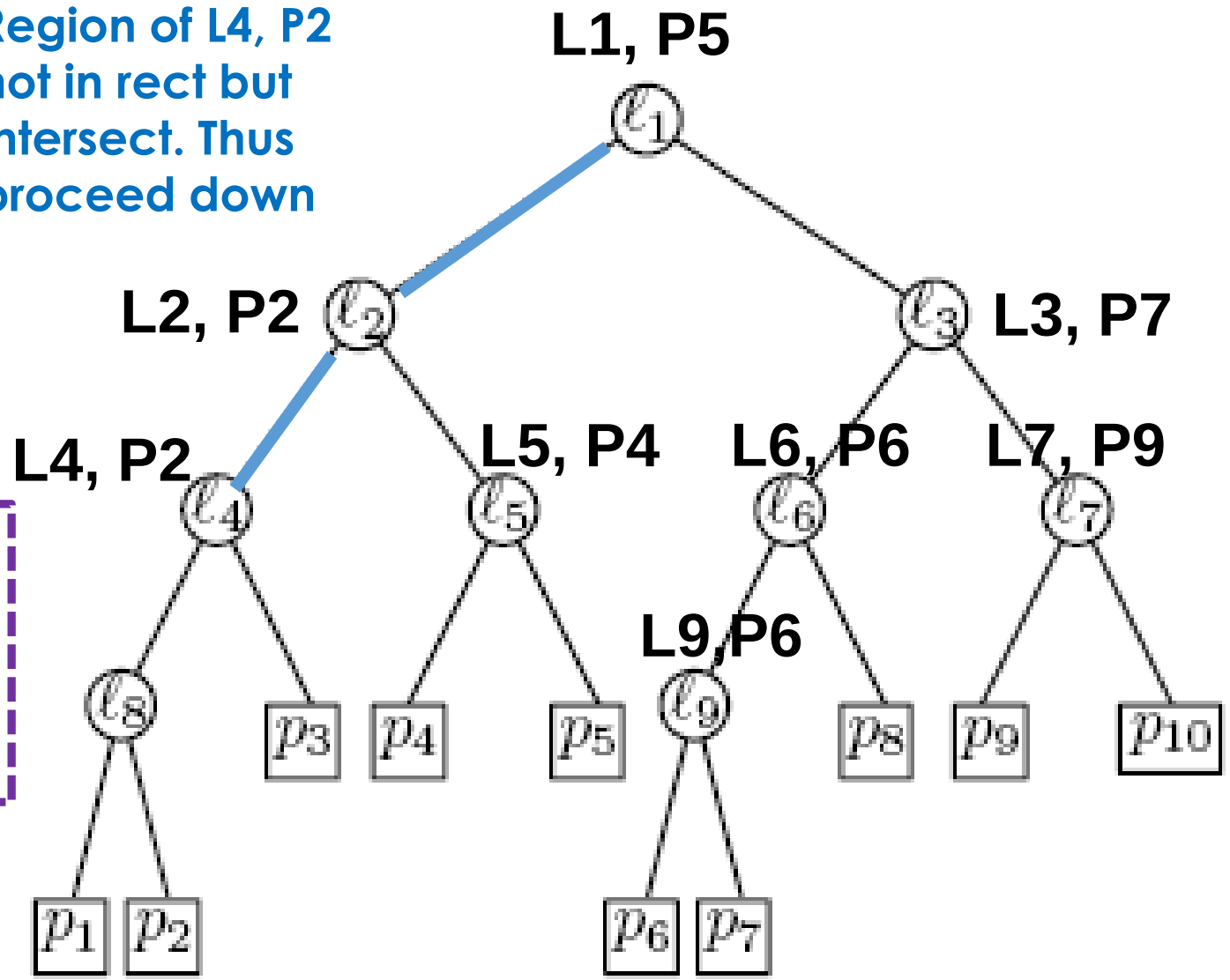
Region of L5, P4 not in rect and not intersect. Will not be traversed further



KD-tree: Range Query Example

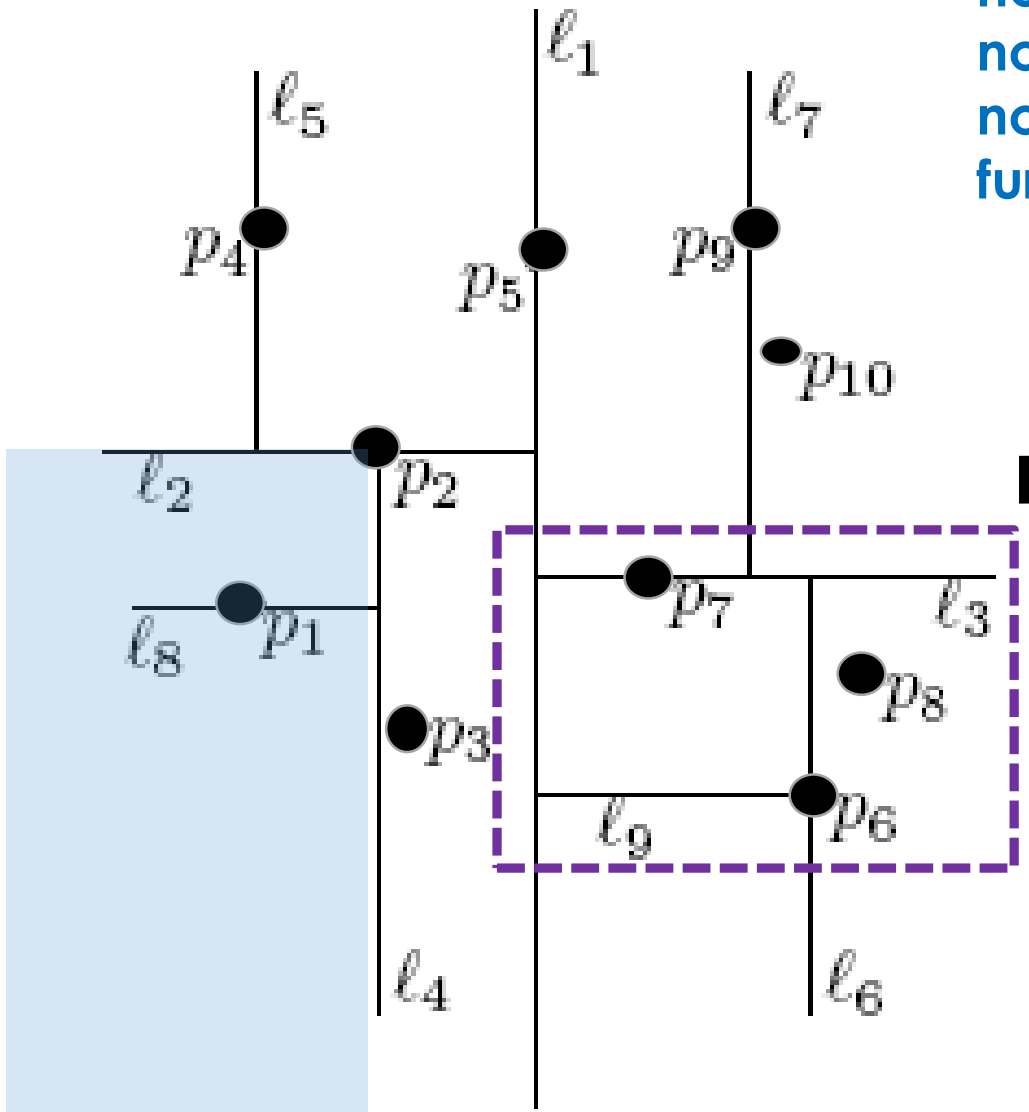


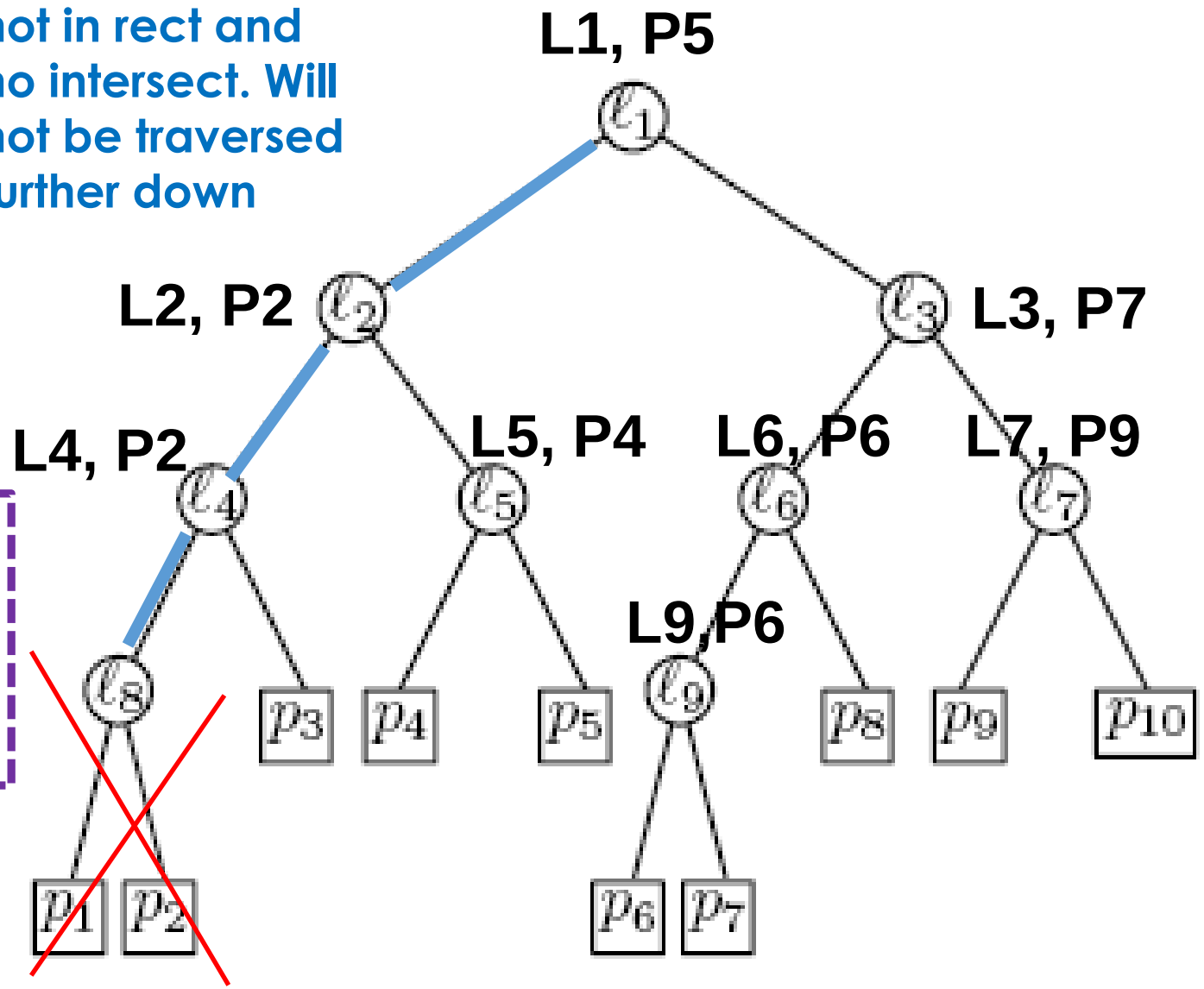
Region of L4, P2 not in rect but intersect. Thus proceed down



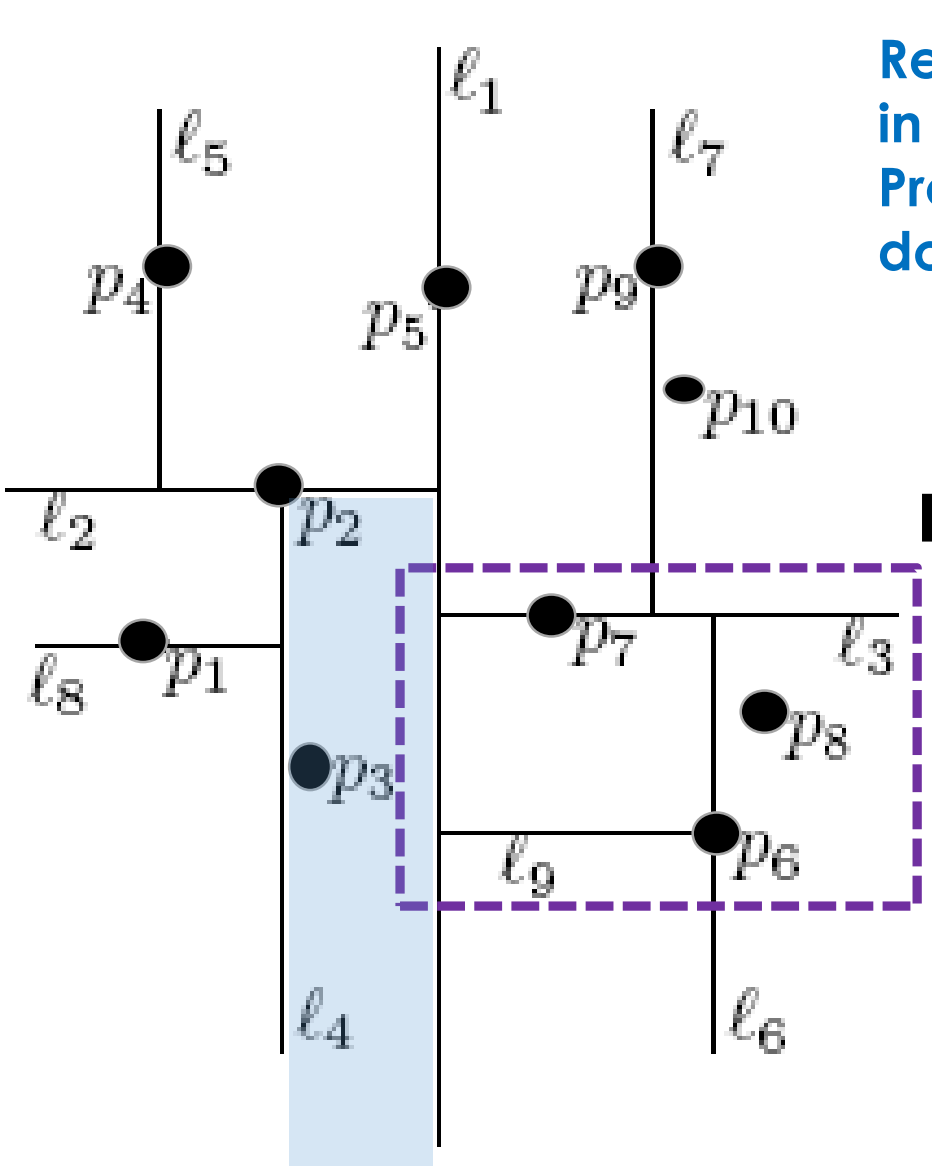
KD-tree: Range Query Example

Region of L8, P1 not in rect and no intersect. Will not be traversed further down

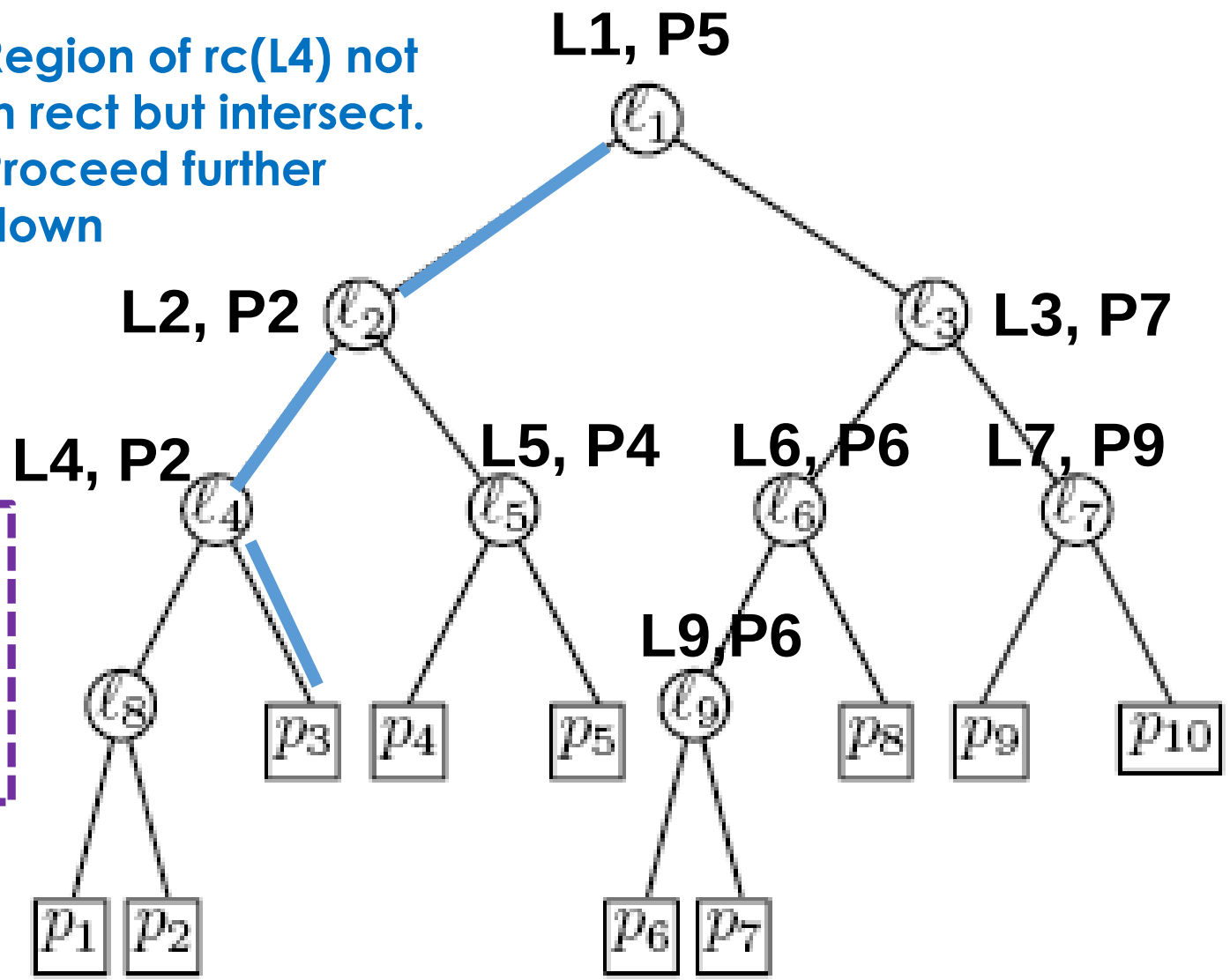




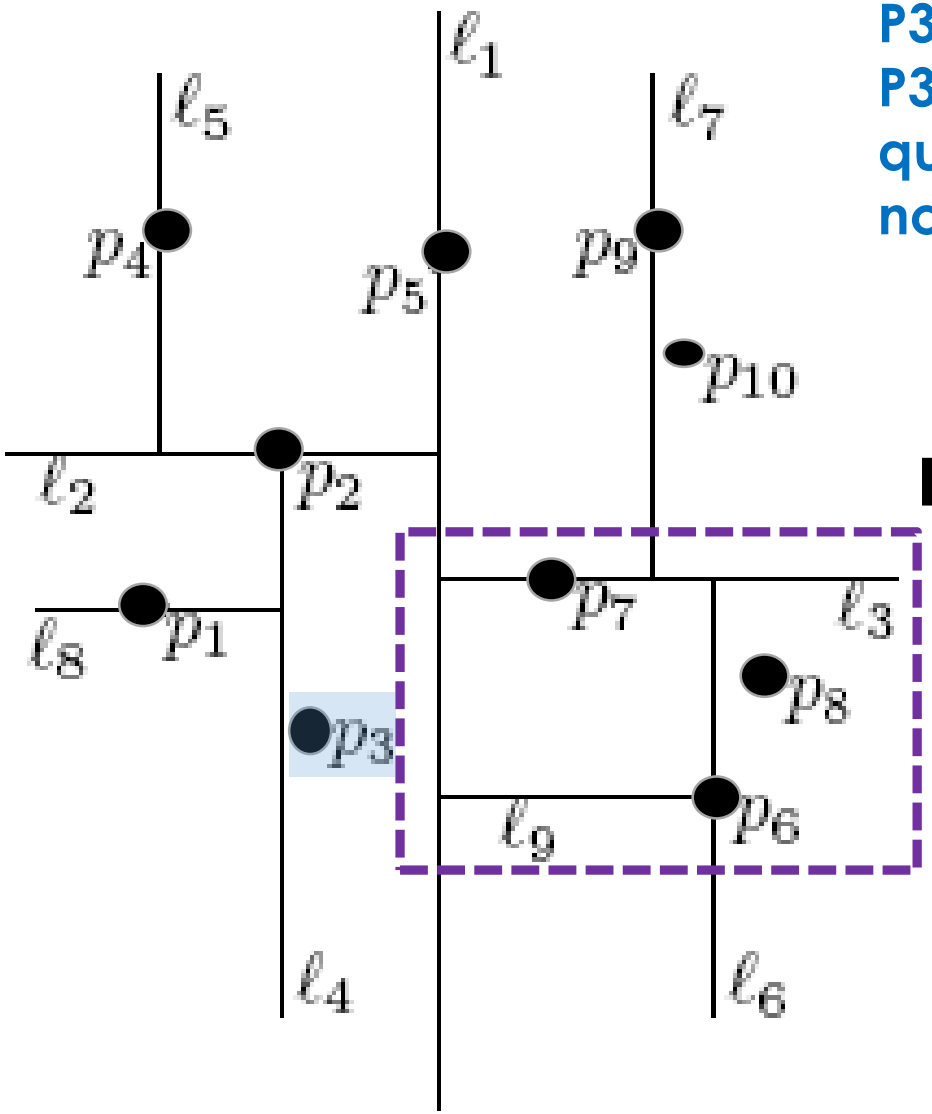
KD-tree: Range Query Example



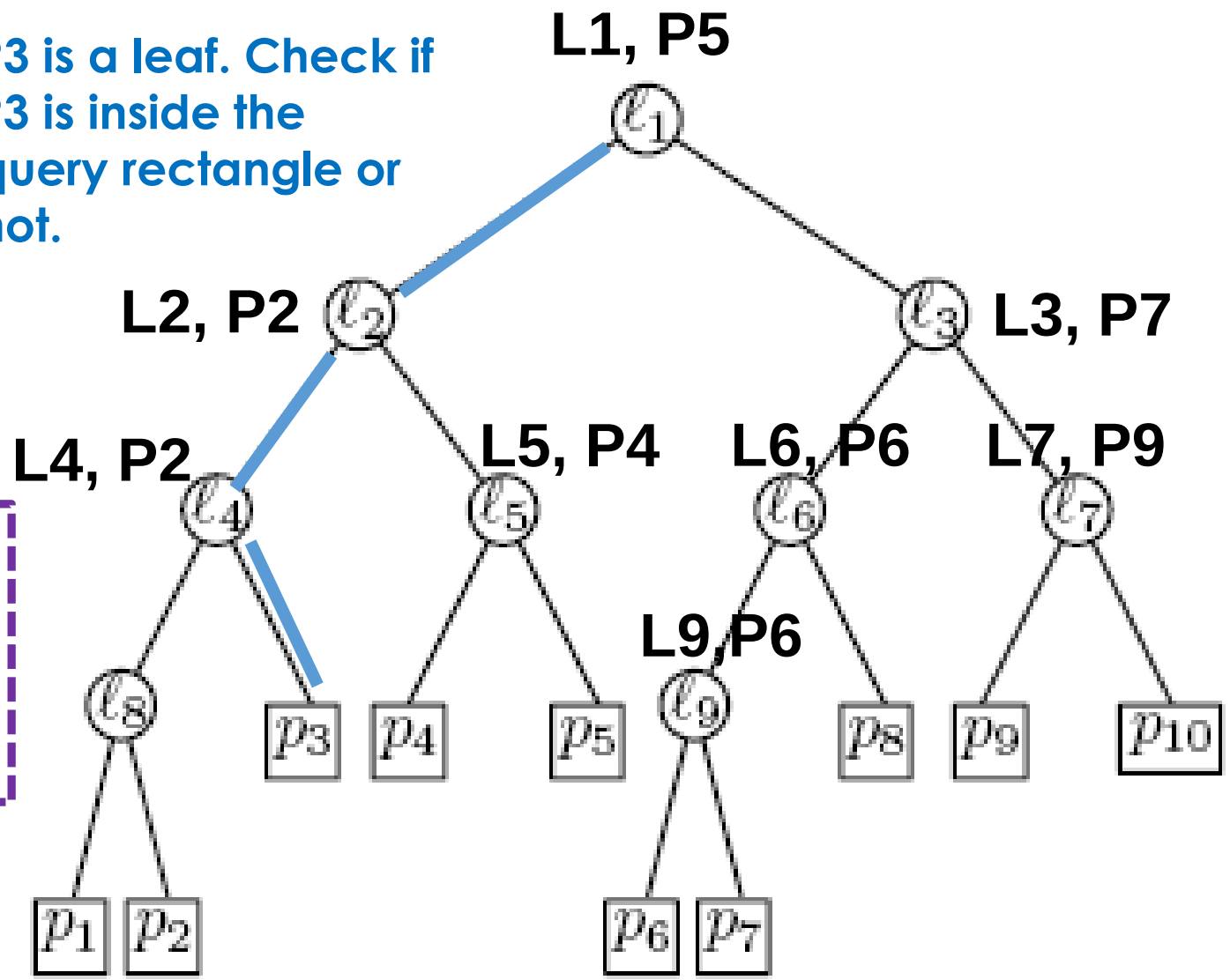
Region of rc(L4) not in rect but intersect. Proceed further down



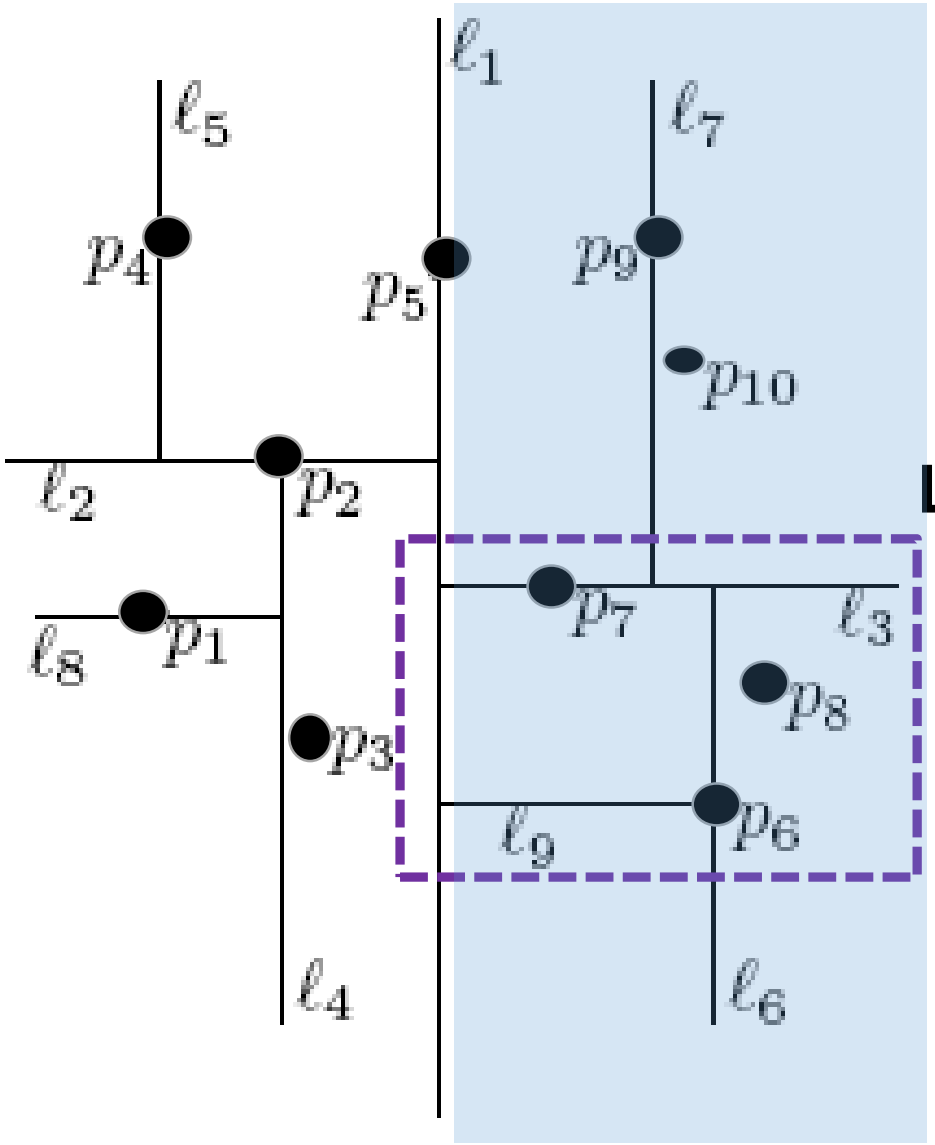
KD-tree: Range Query Example



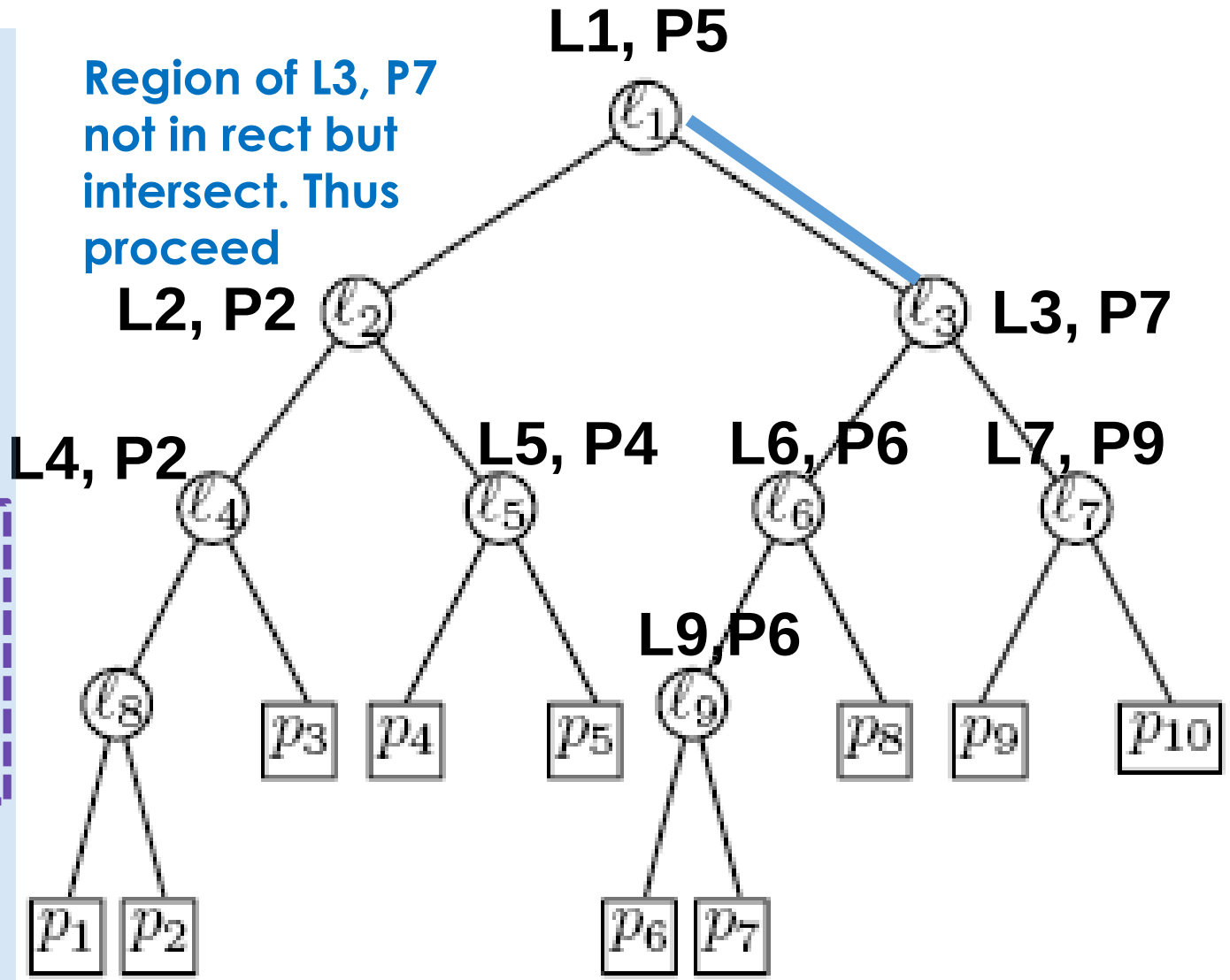
P3 is a leaf. Check if P3 is inside the query rectangle or not.



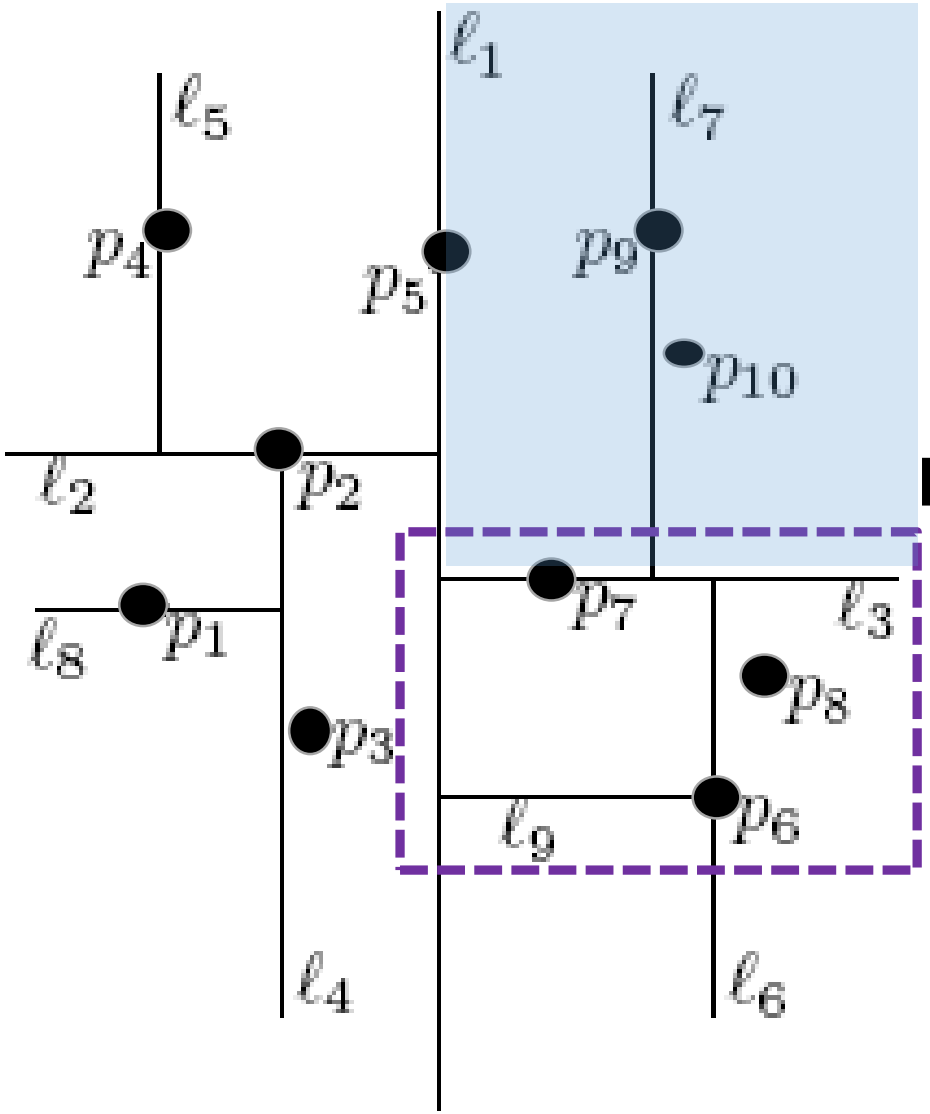
KD-tree: Range Query Example



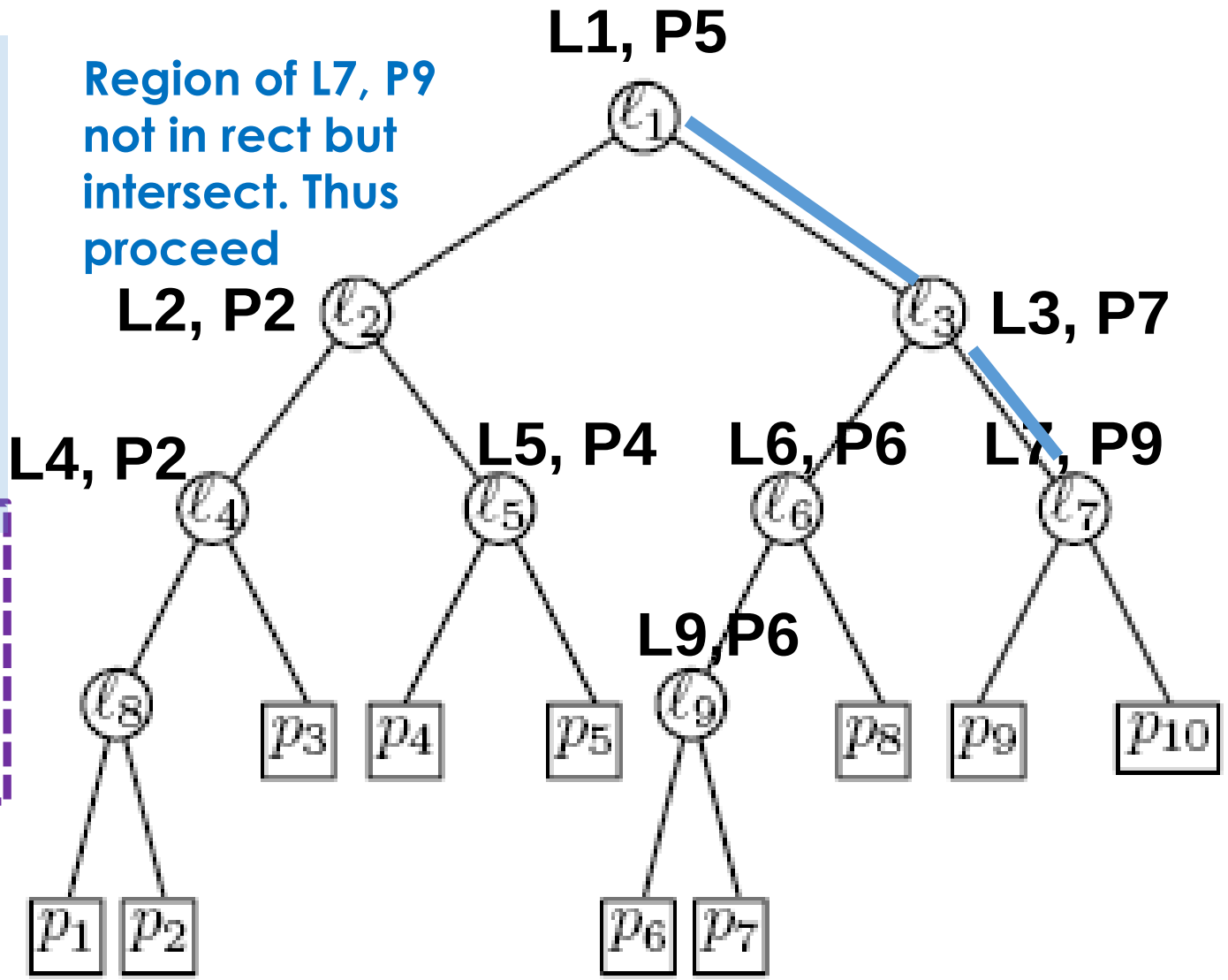
Region of L3, P7 not in rect but intersect. Thus proceed



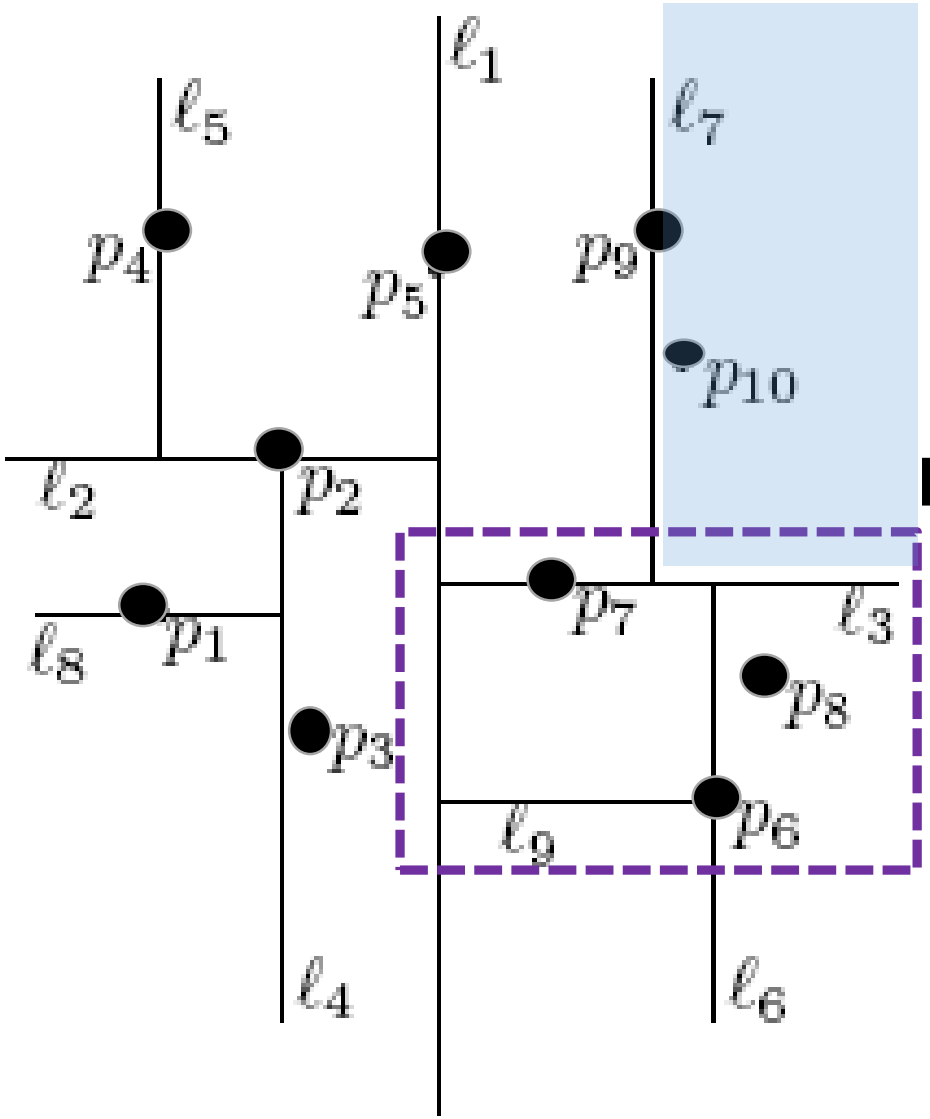
KD-tree: Range Query Example



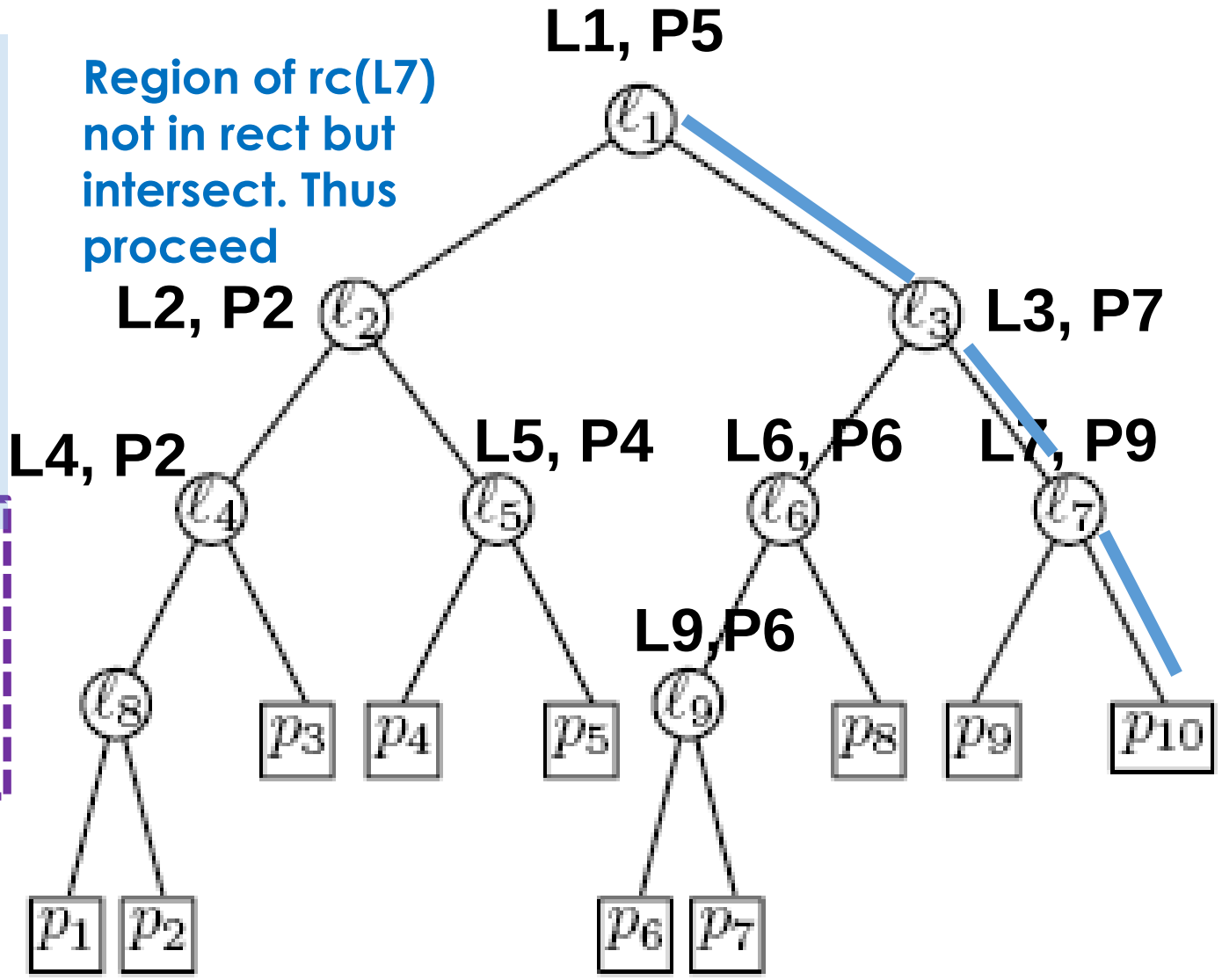
Region of L7, P9 not in rect but intersect. Thus proceed



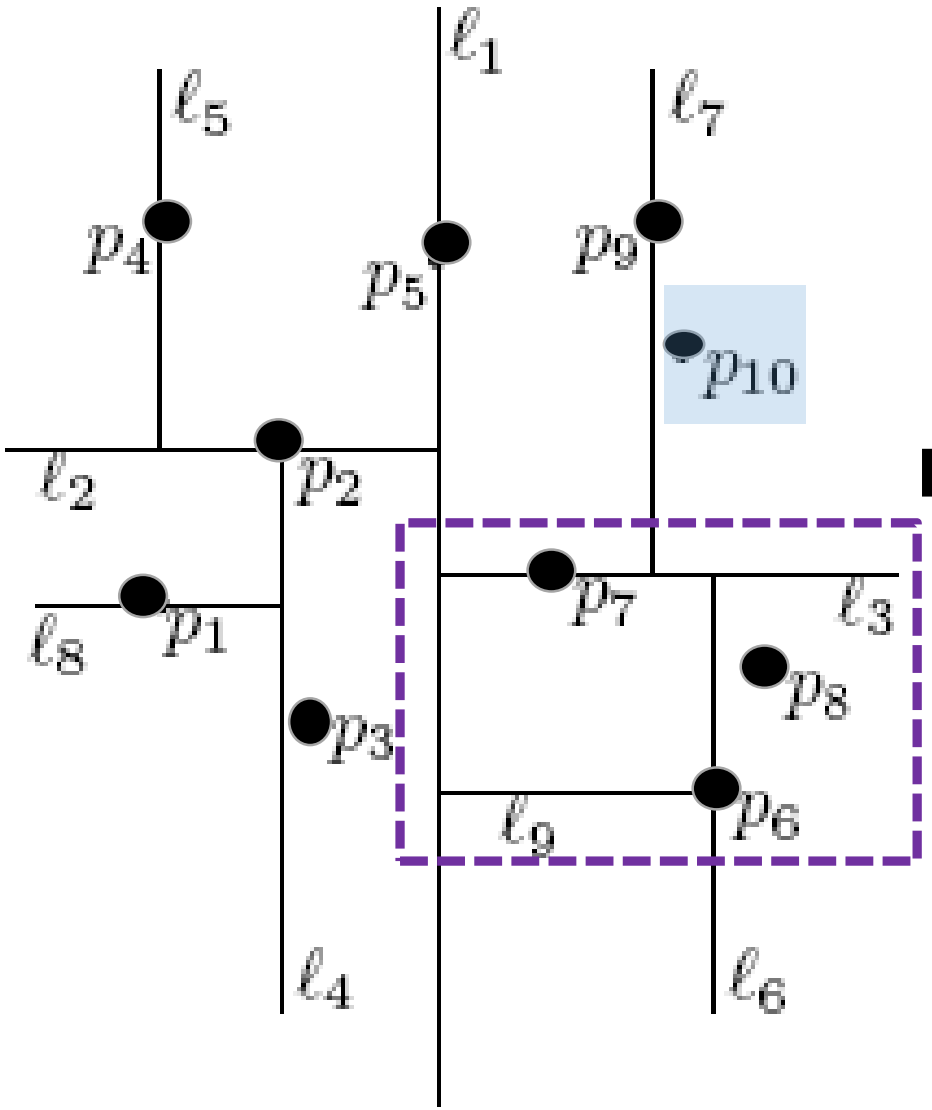
KD-tree: Range Query Example



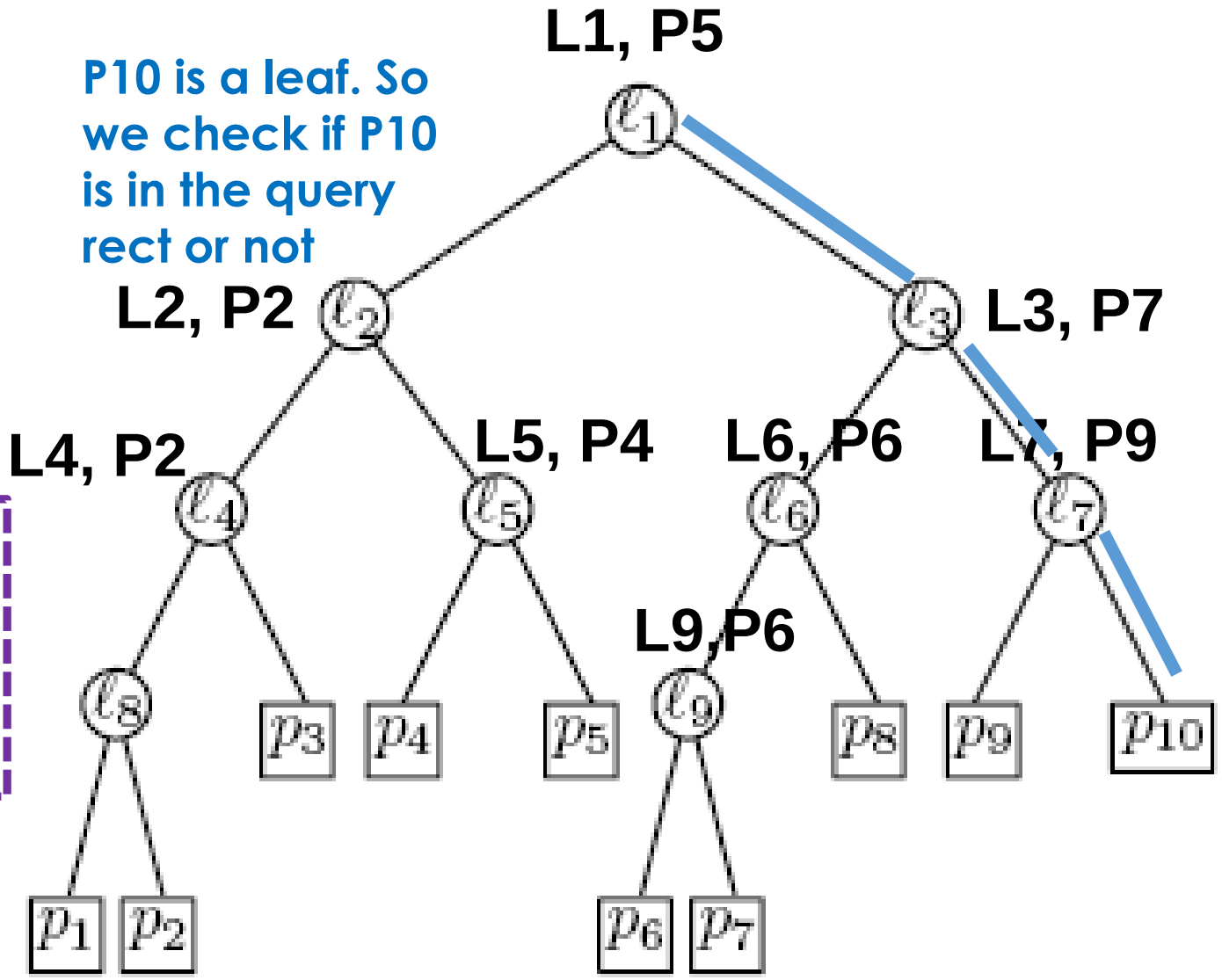
Region of rc(L7) not in rect but intersect. Thus proceed



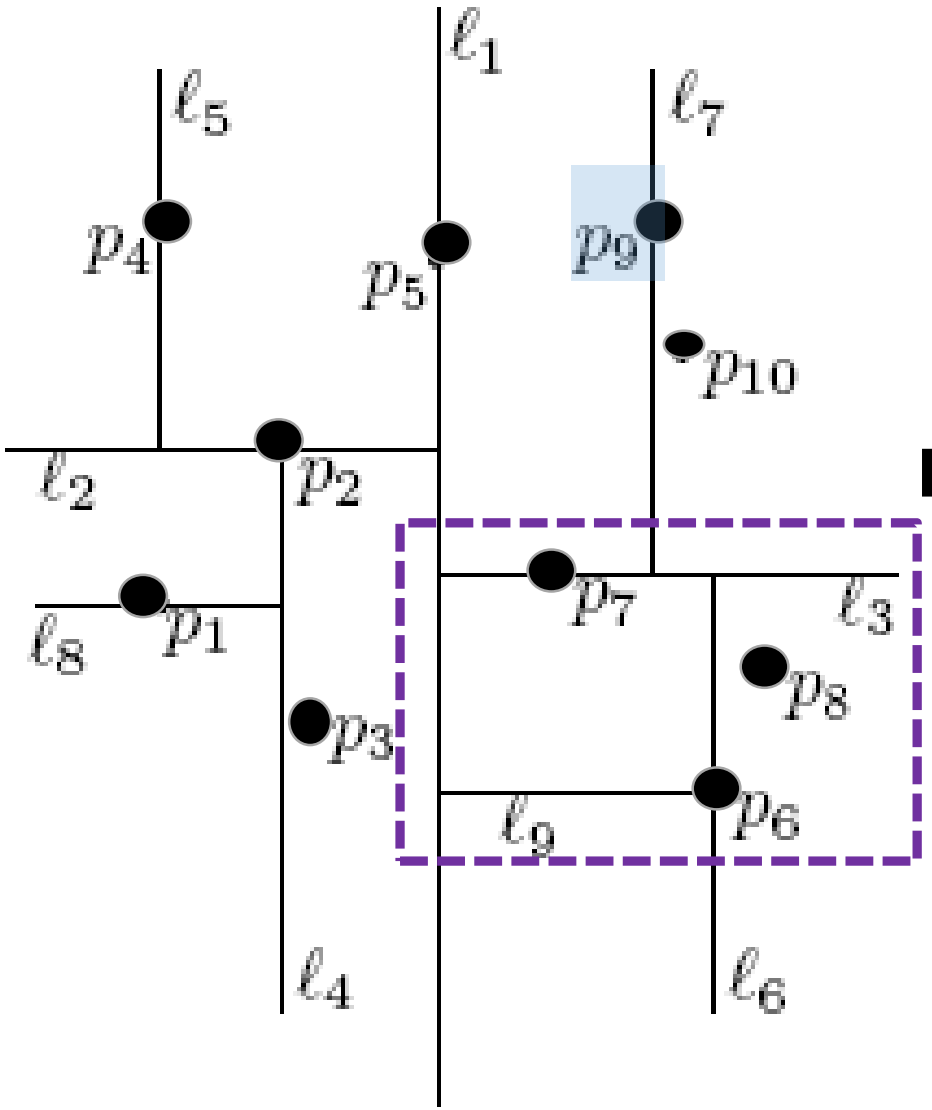
KD-tree: Range Query Example



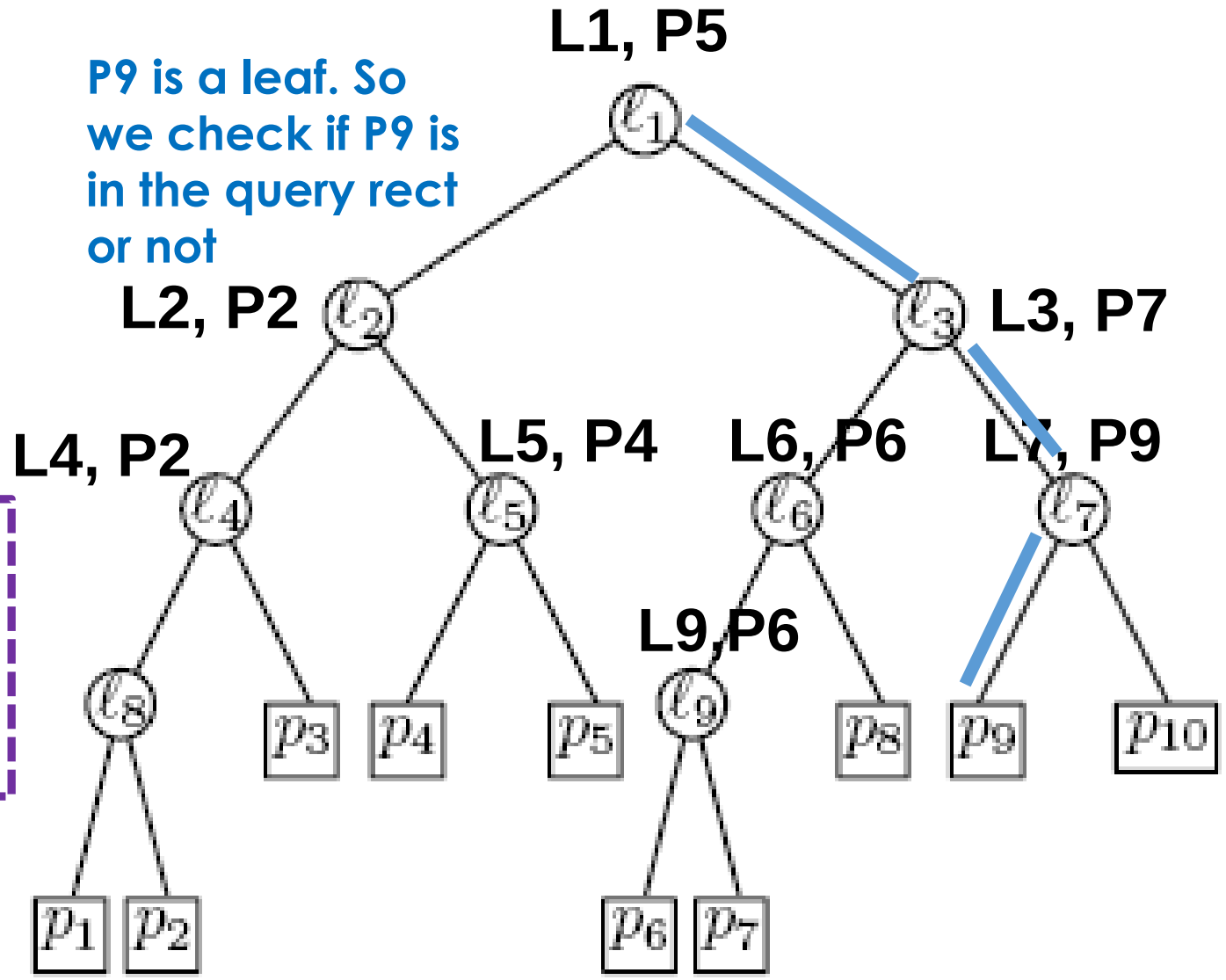
P10 is a leaf. So we check if P10 is in the query rect or not



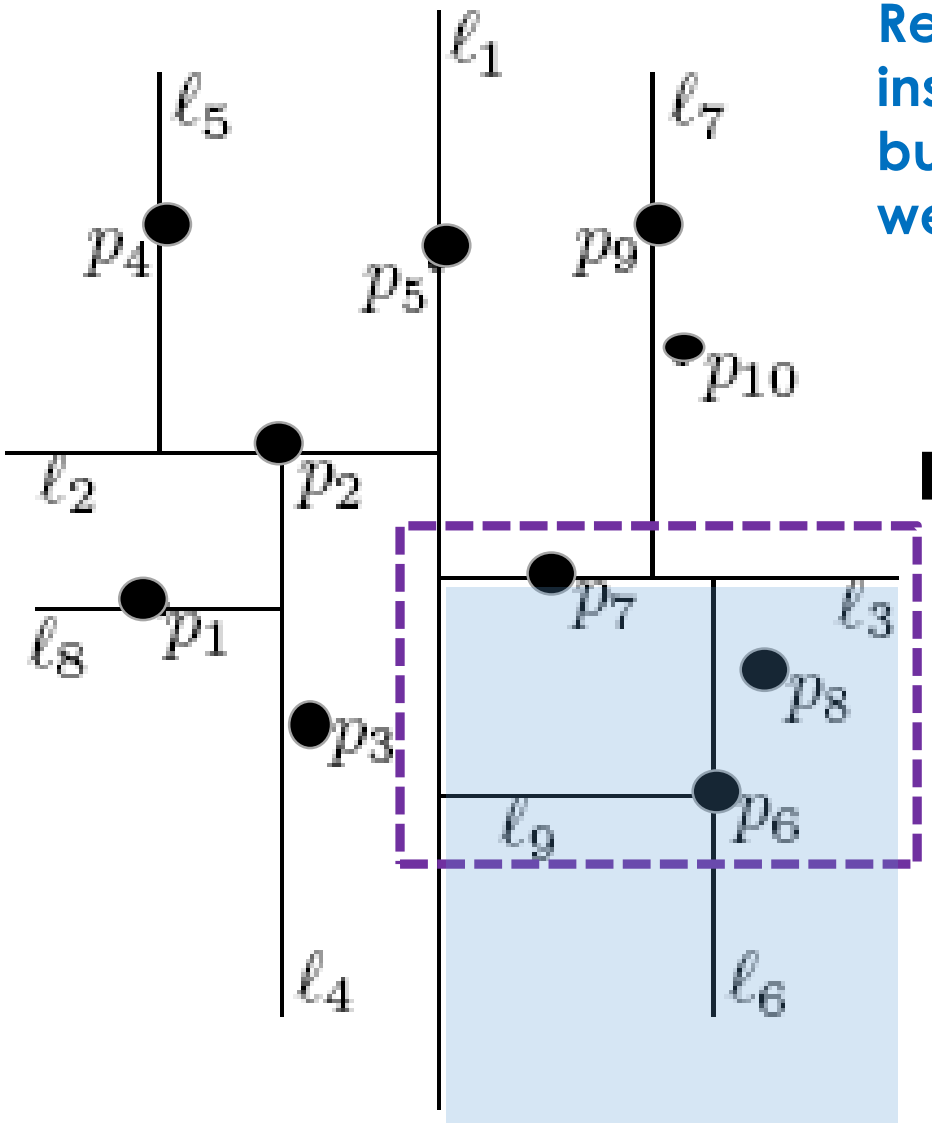
KD-tree: Range Query Example



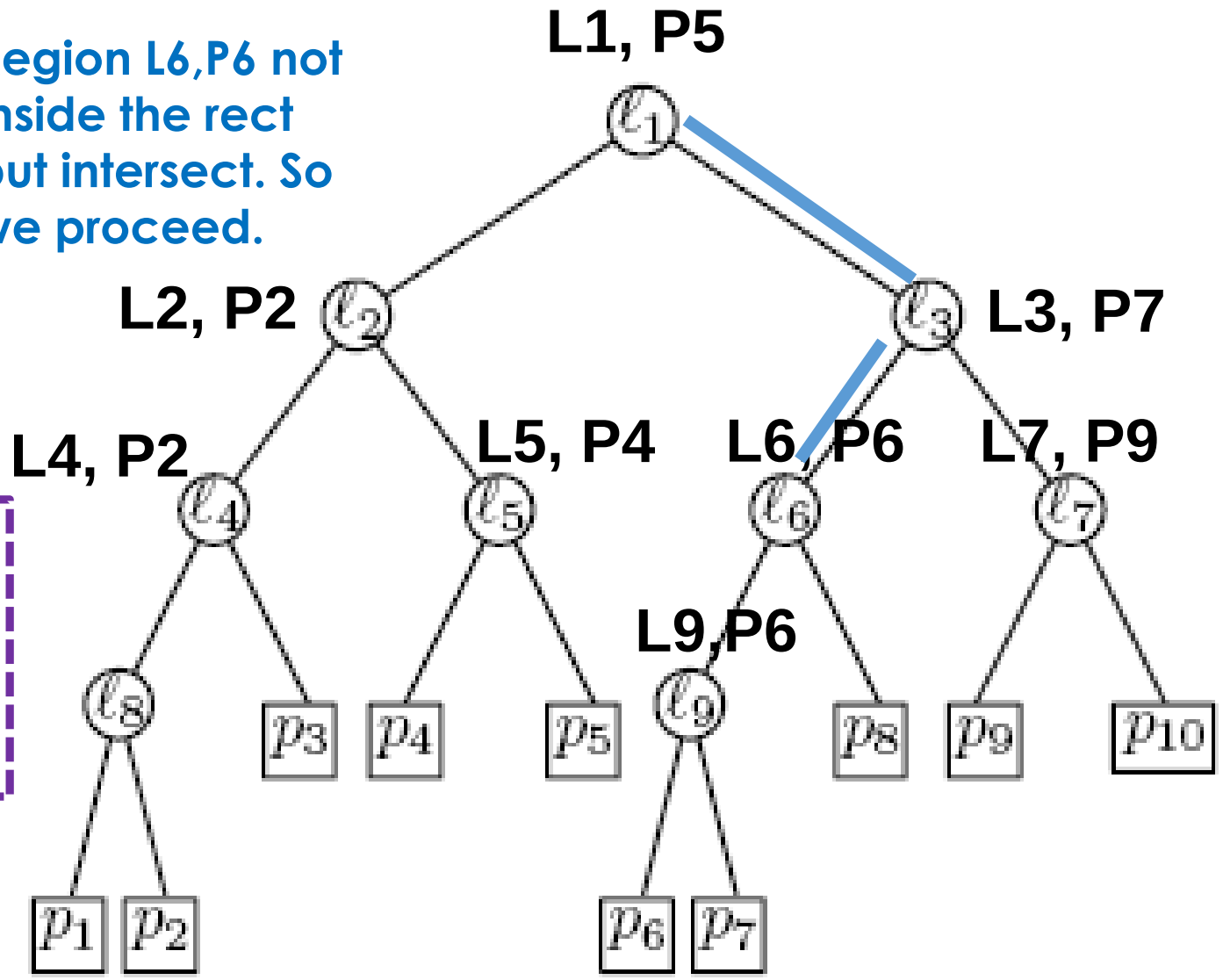
P9 is a leaf. So we check if P9 is in the query rect or not



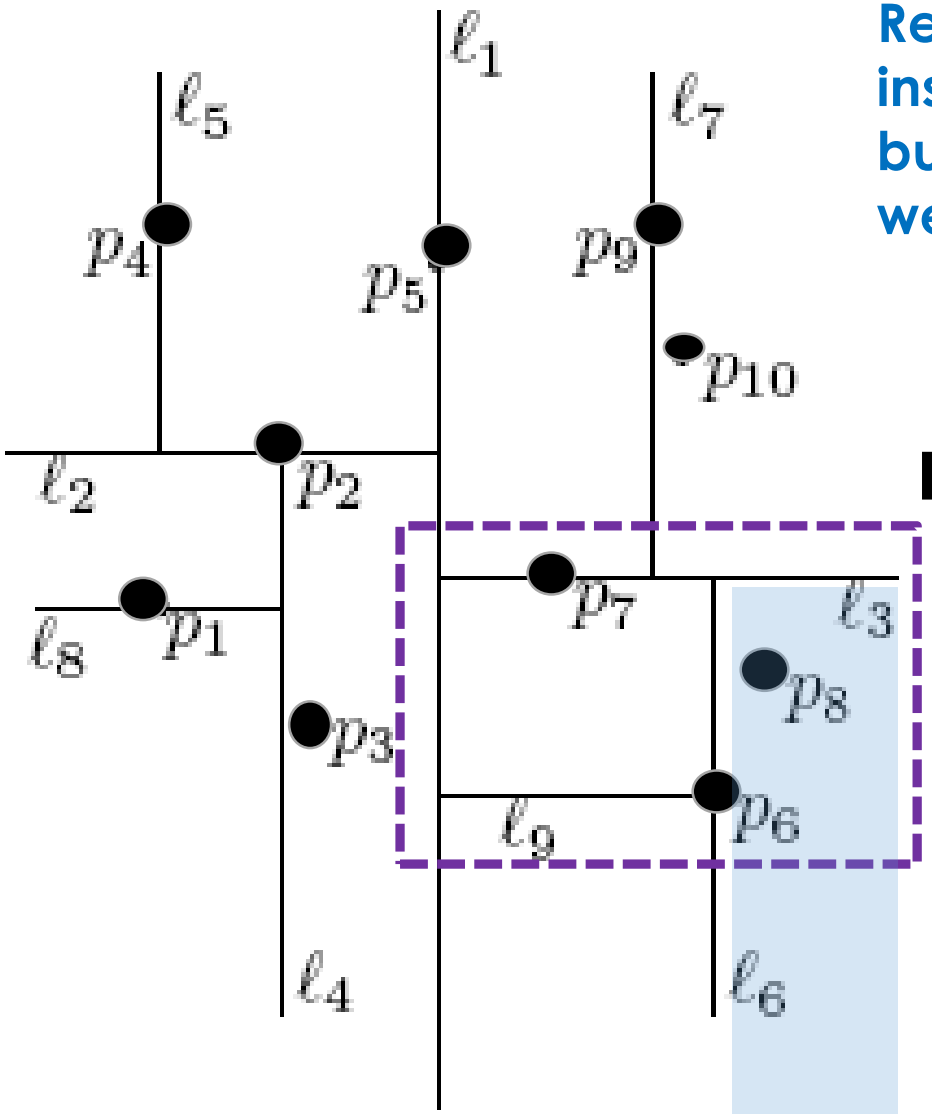
KD-tree: Range Query Example



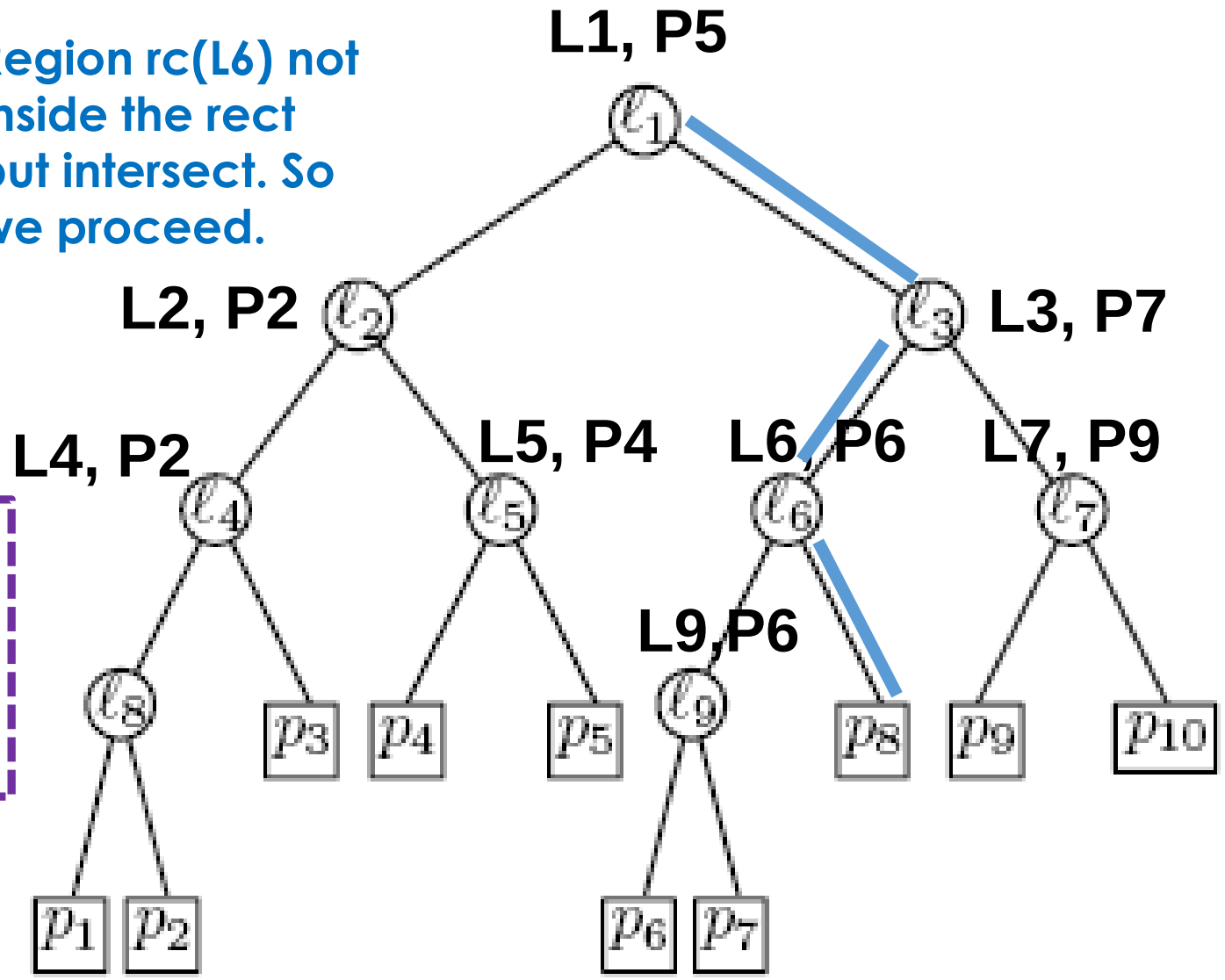
Region L6,P6 not inside the rect but intersect. So we proceed.



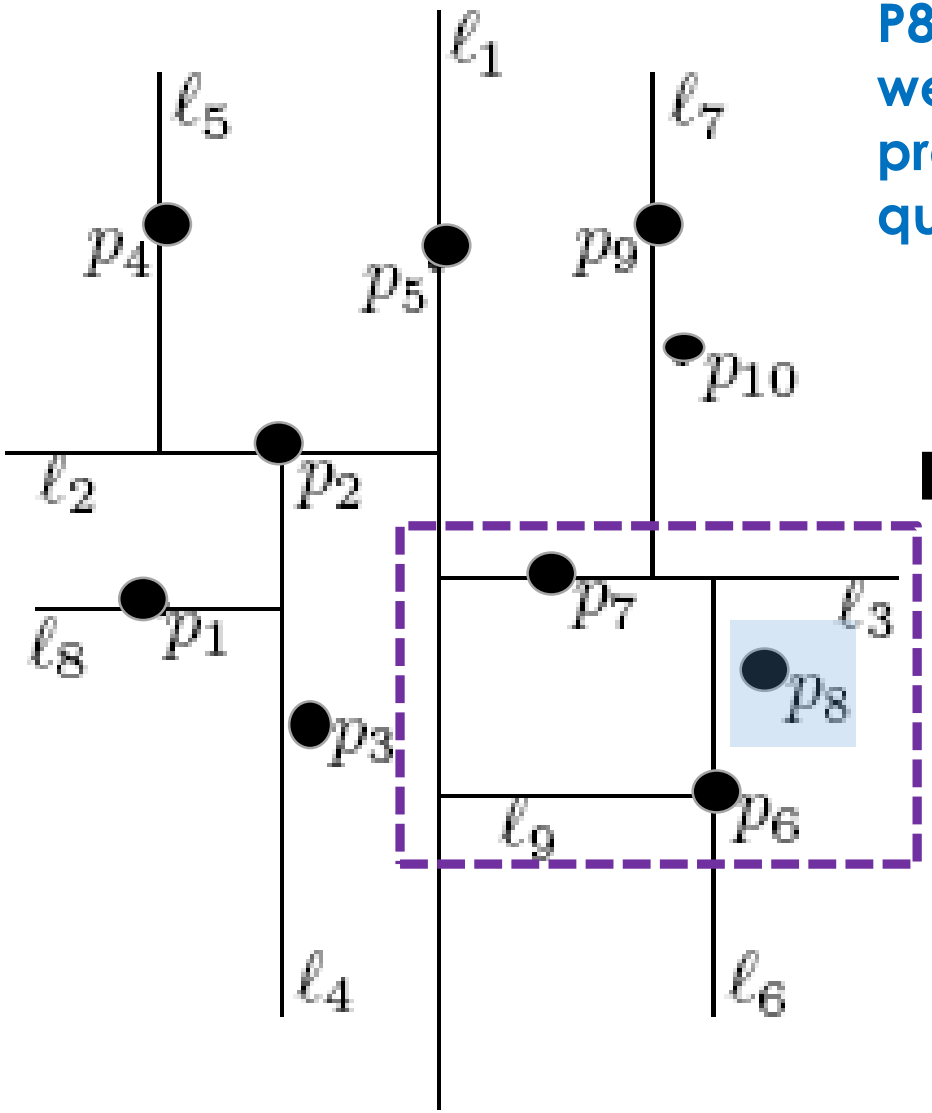
KD-tree: Range Query Example



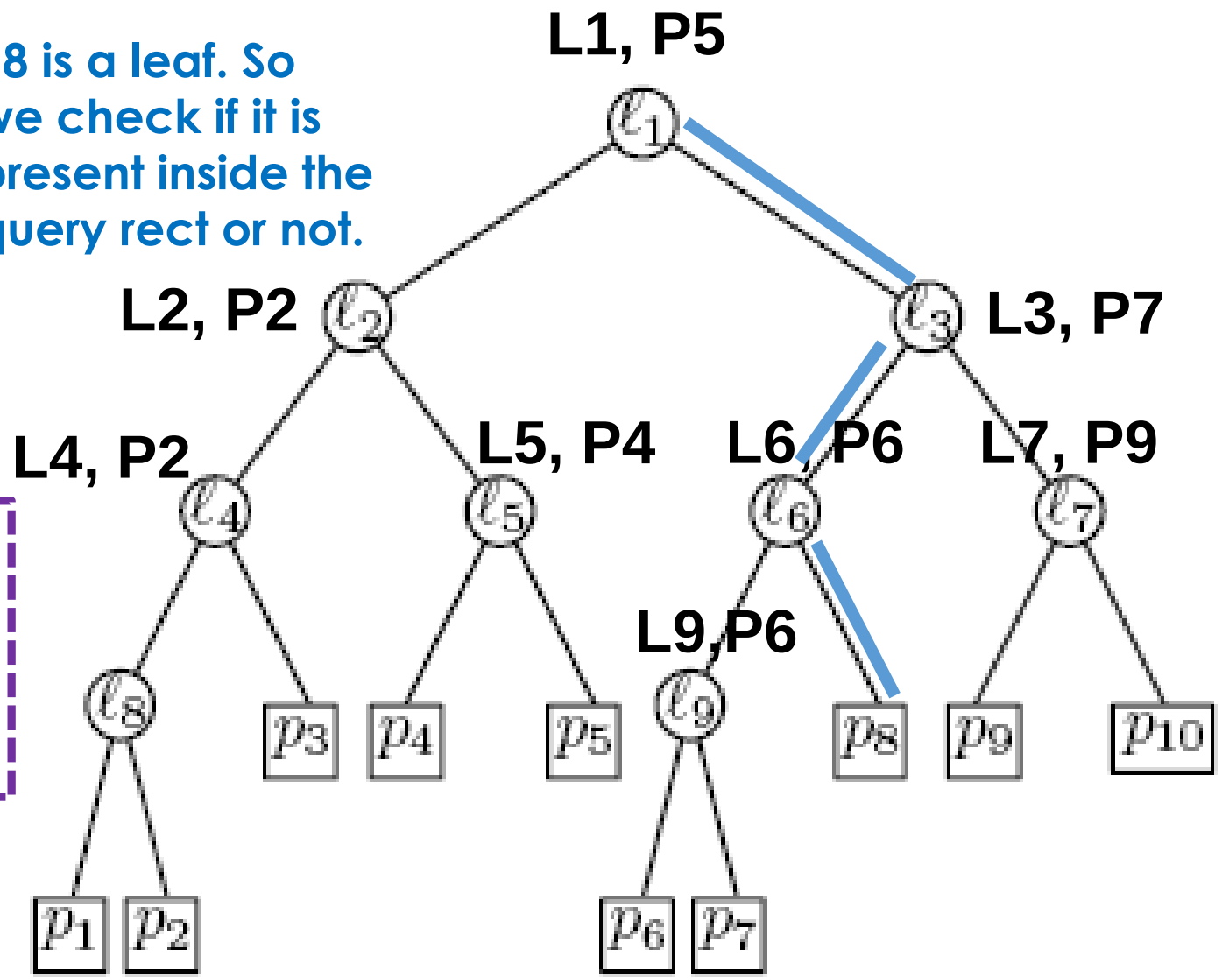
Region rc(L6) not inside the rect but intersect. So we proceed.



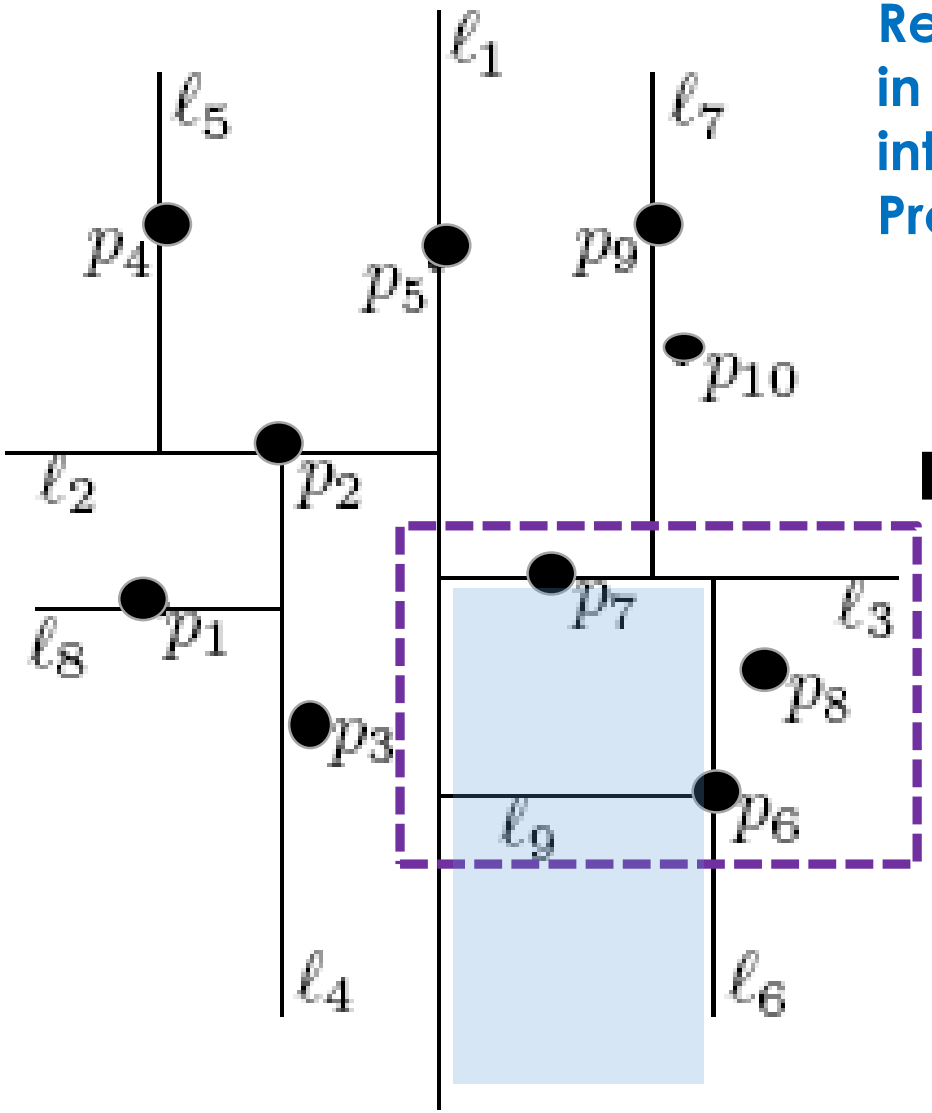
KD-tree: Range Query Example



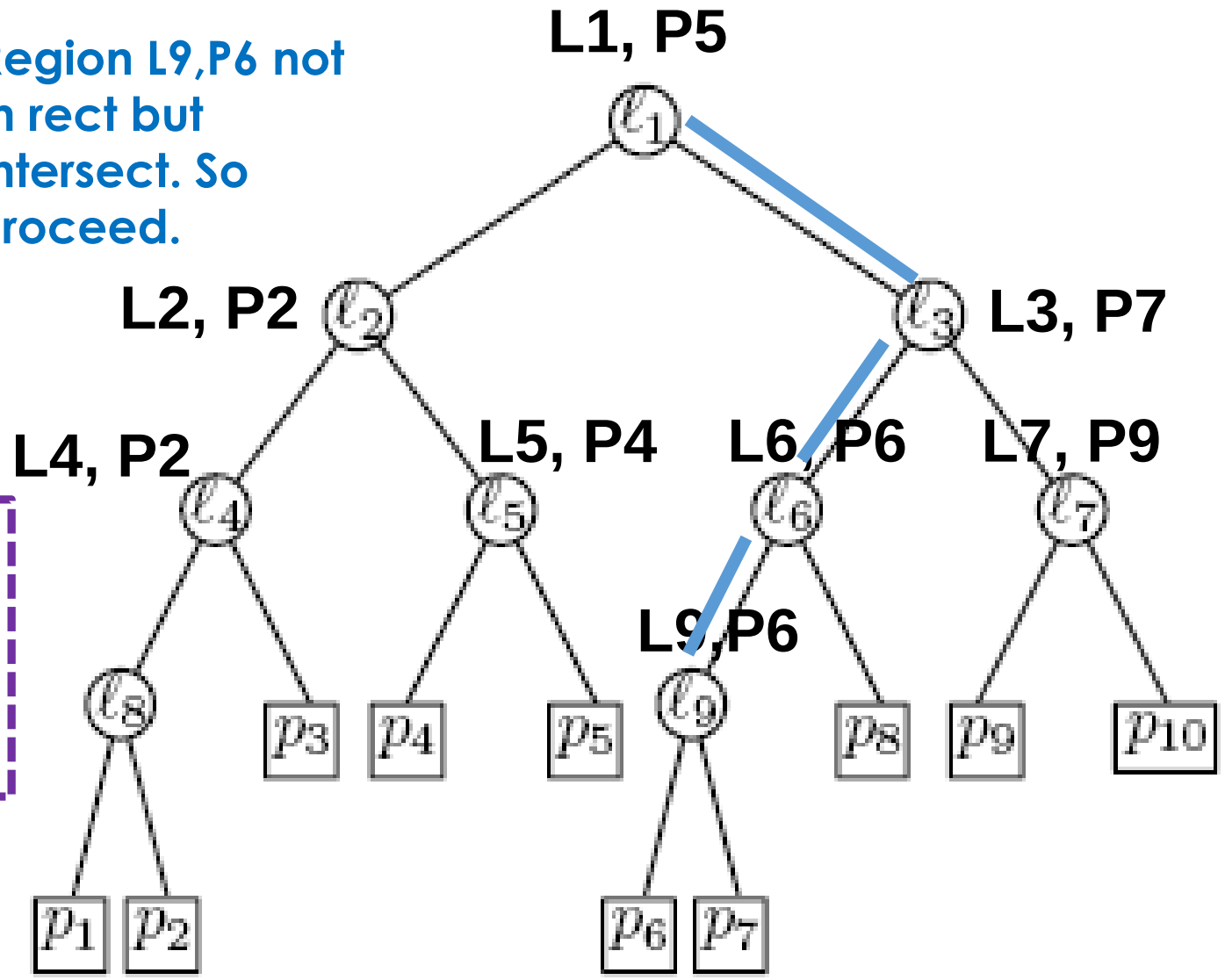
P8 is a leaf. So we check if it is present inside the query rect or not.



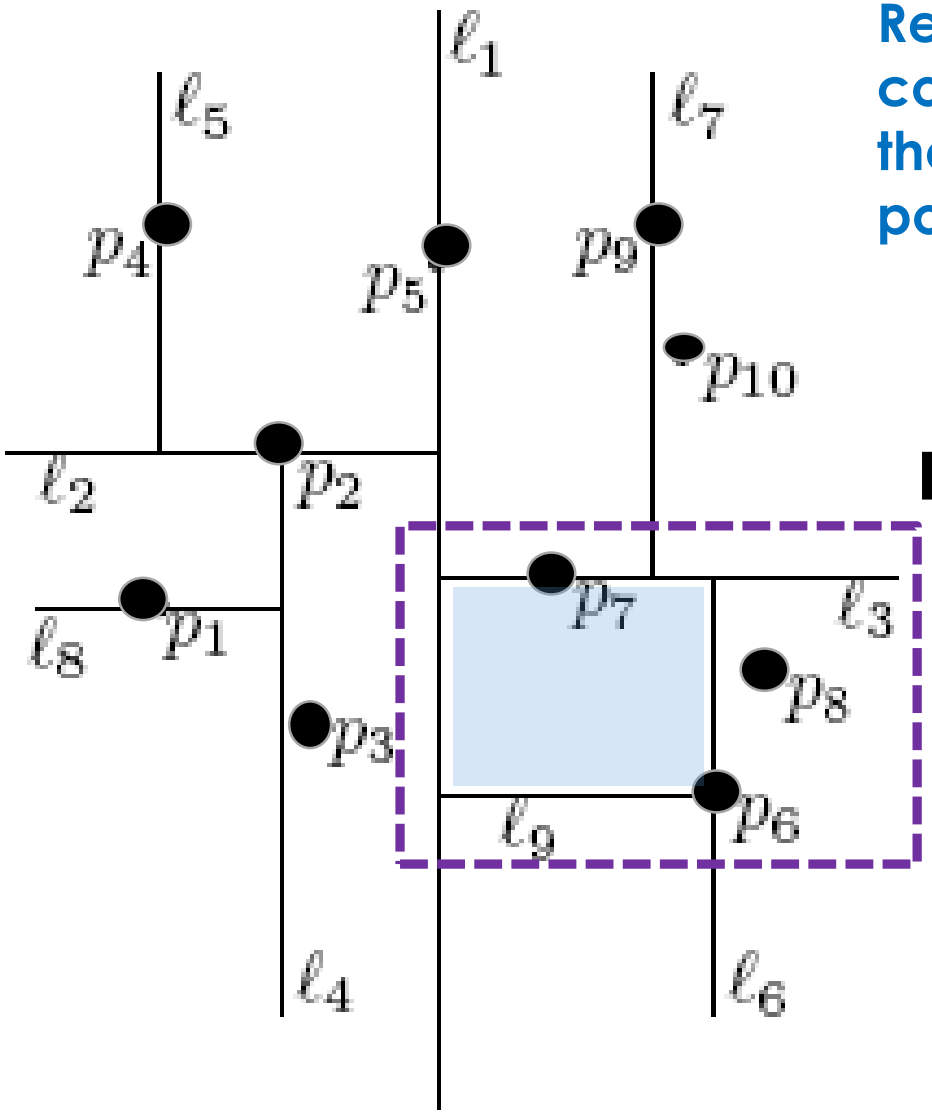
KD-tree: Range Query Example



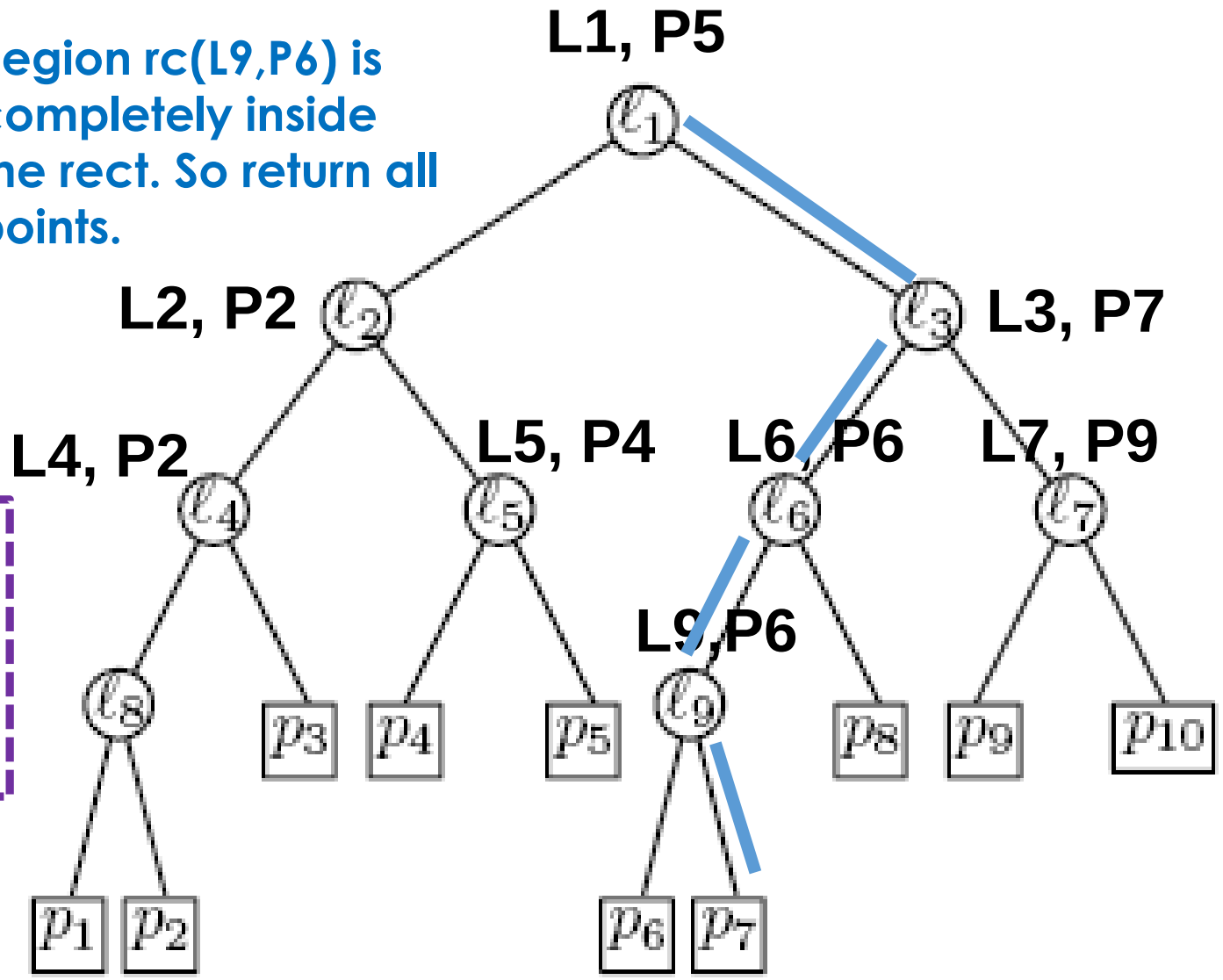
Region L9,P6 not in rect but intersect. So Proceed.



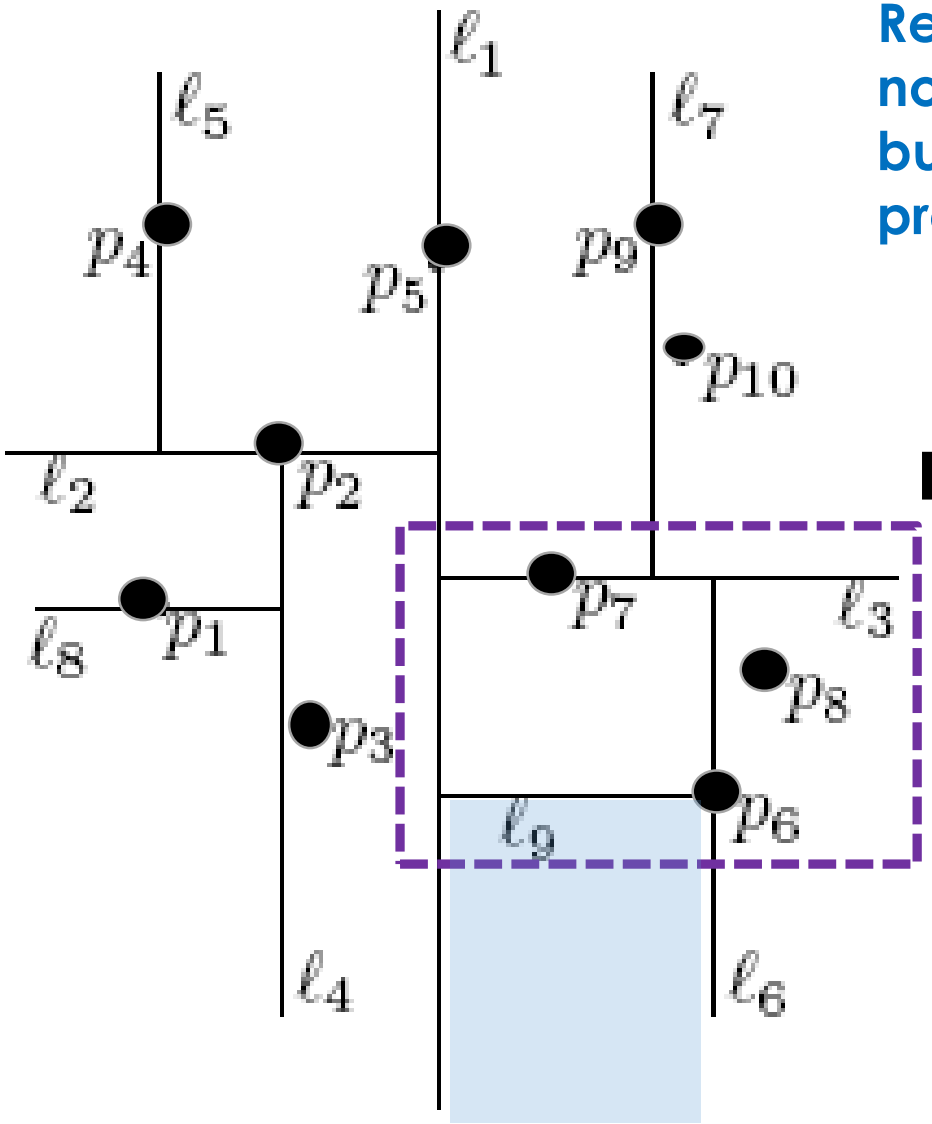
KD-tree: Range Query Example



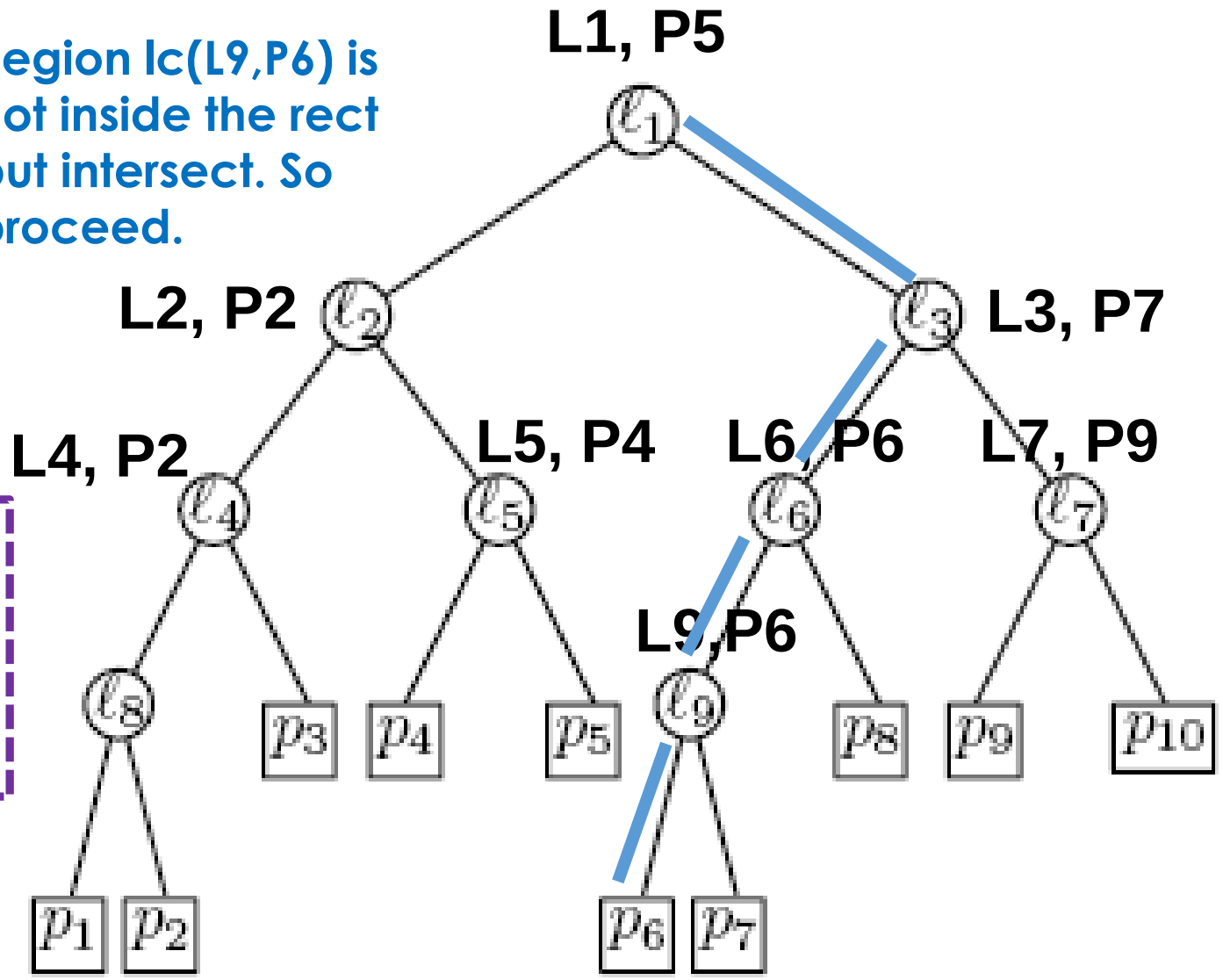
Region rc(L9,P6) is completely inside the rect. So return all points.



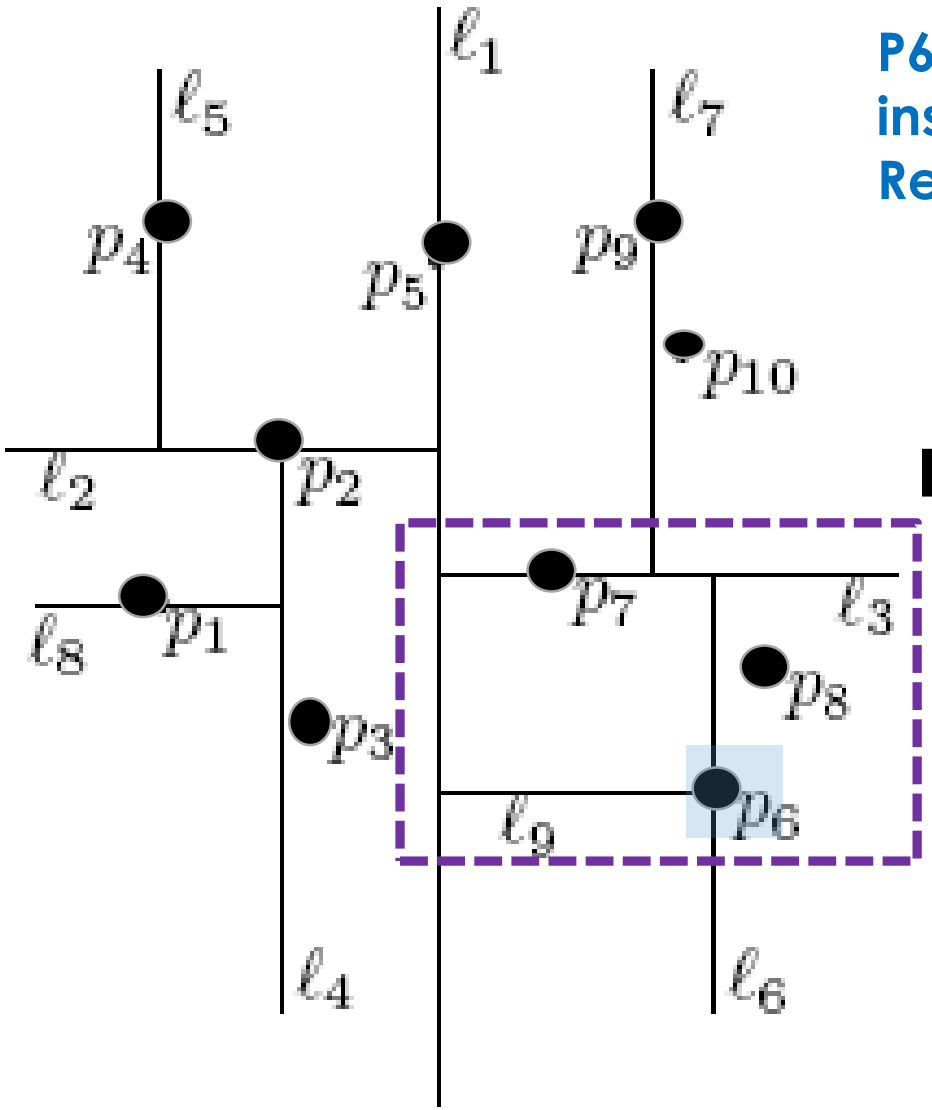
KD-tree: Range Query Example



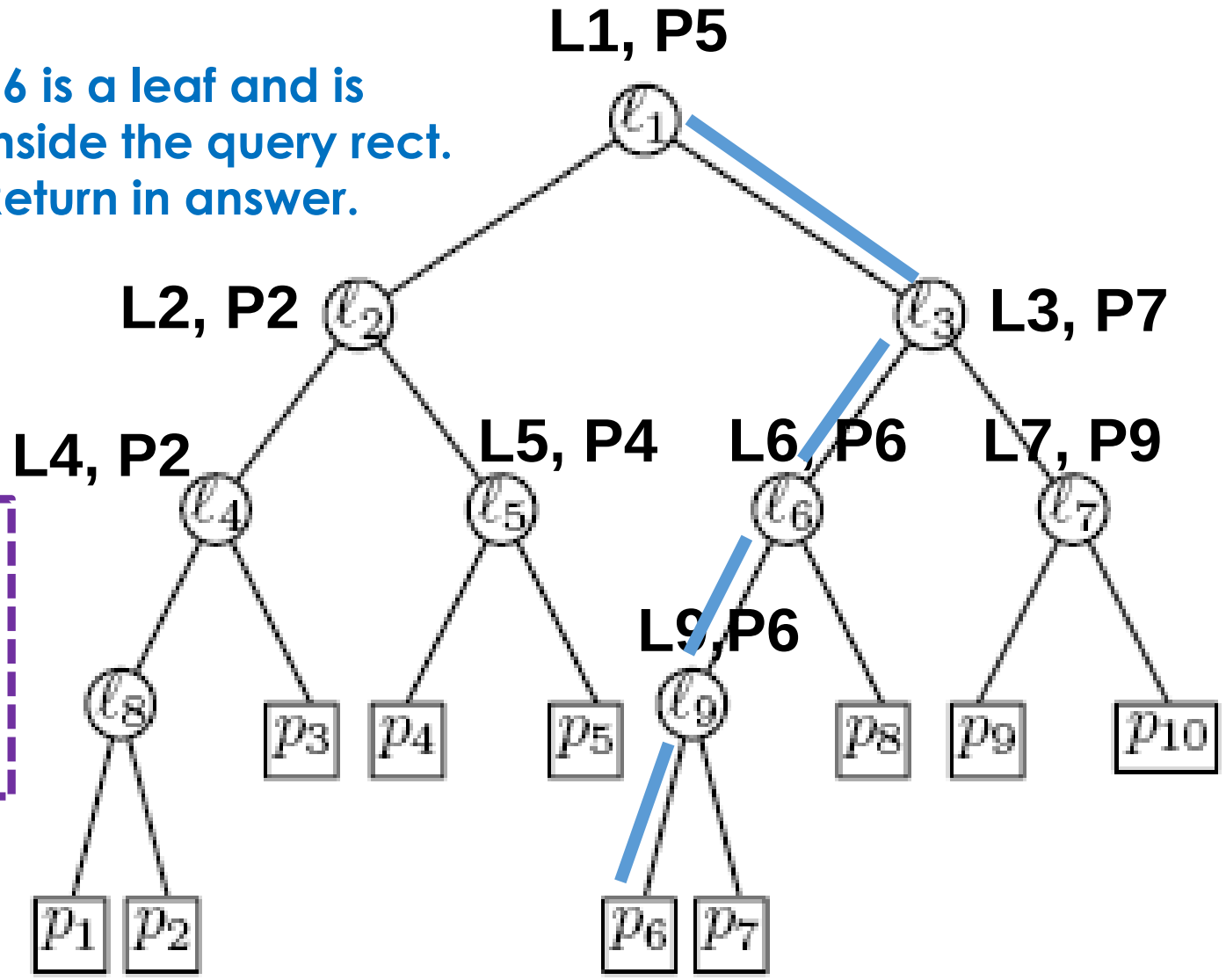
Region lc(L9,P6) is not inside the rect but intersect. So proceed.



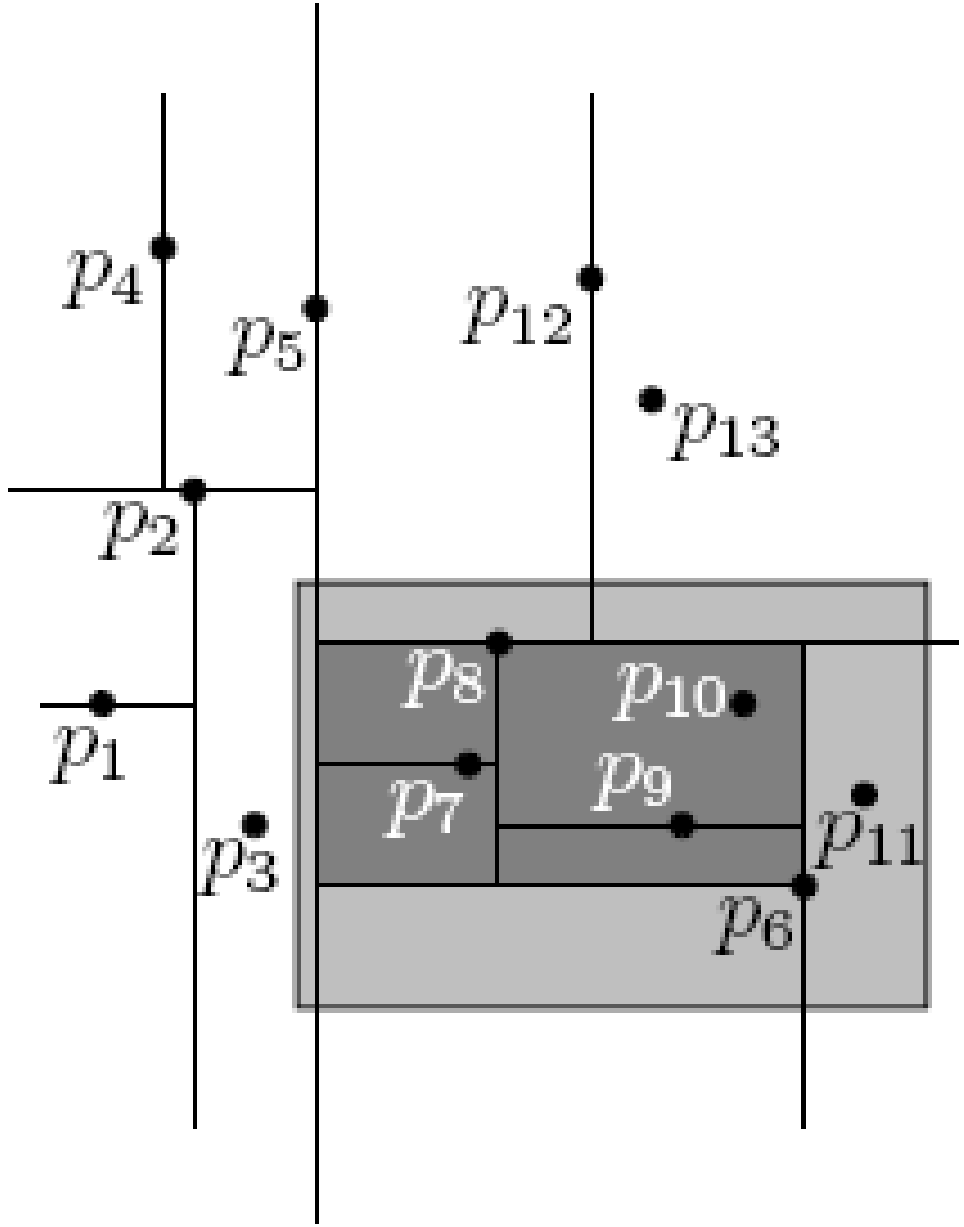
KD-tree: Range Query Example

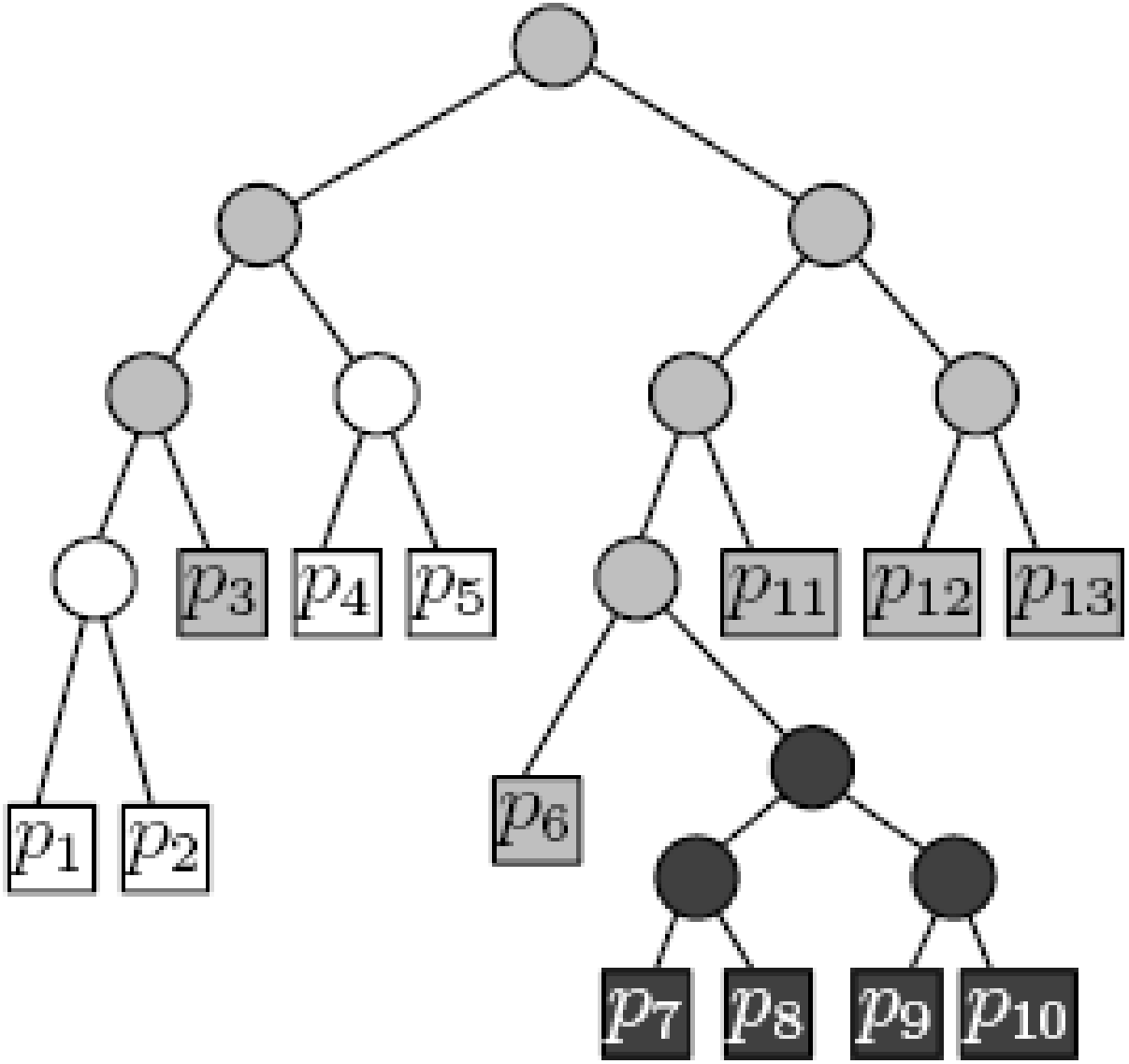


P6 is a leaf and is inside the query rect. Return in answer.



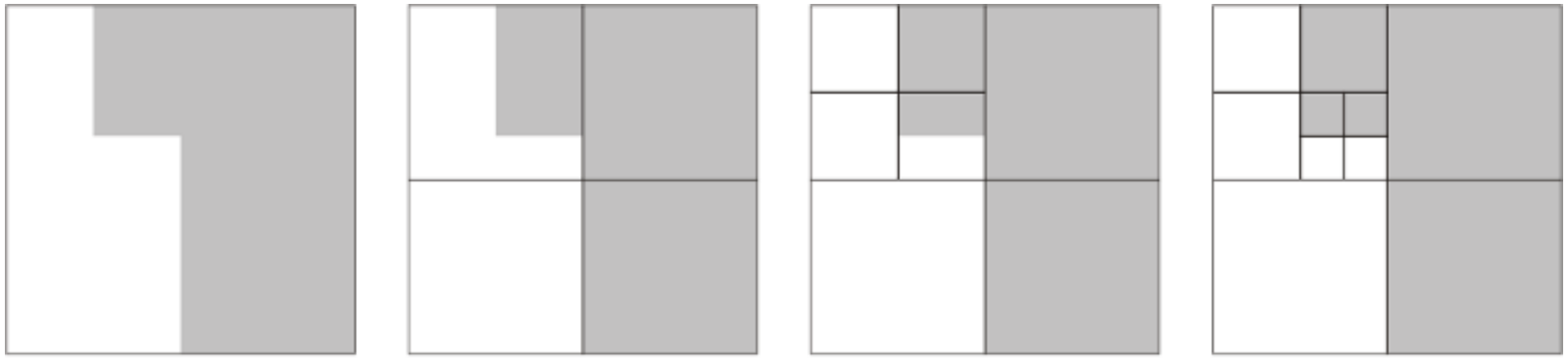
KD-tree: Range Query Another Example





Region Quadtrees

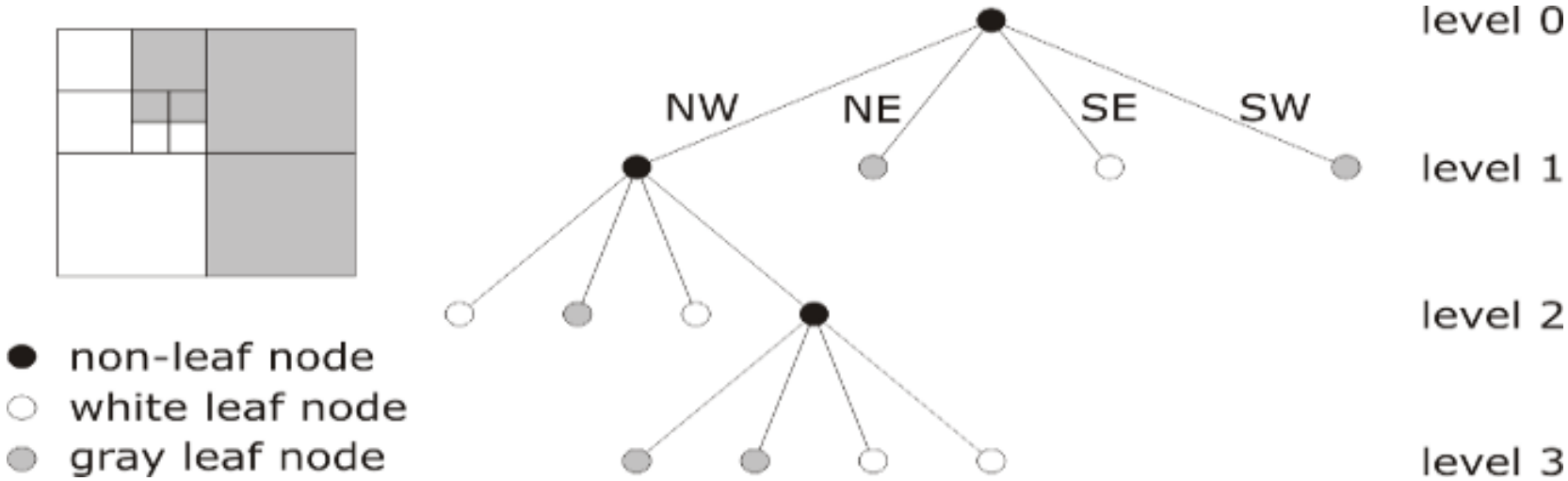
- ***Quadtree*** is a tree structure where every non-leaf node has exactly four descendents
- ***Region quadtrees*** recursively subdivide non-homogenous square arrays of cells into four equal sized quadrants
- Decomposition continues until all squares bound homogenous regions



Some slides borrowed from "GIS a computational perspective: second edition" by M. Worboys CRC press 2004.

Region Quadtrees

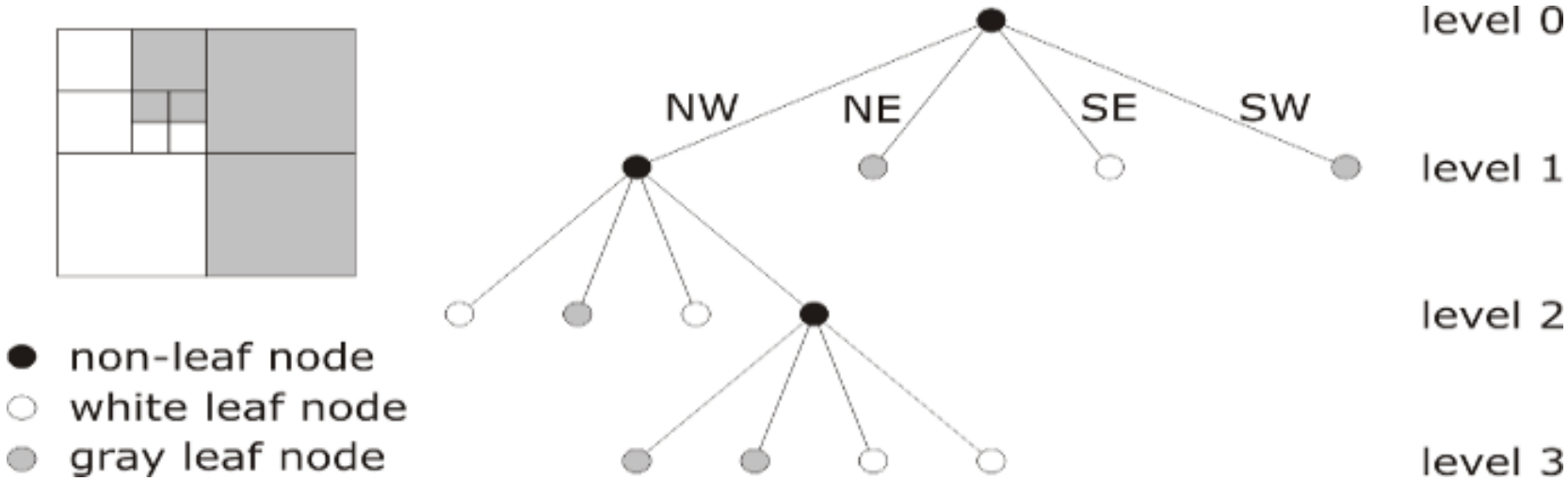
- Quadtrees take full advantage of the spatial structure, adapt to variable spatial detail
- Inefficient for highly inhomogeneous rasters
- Very sensitive to changes in the embedding space (e.g., translation, rotation)



Some slides borrowed from "GIS a computational perspective: second edition" by M. Worboys CRC press 2004.

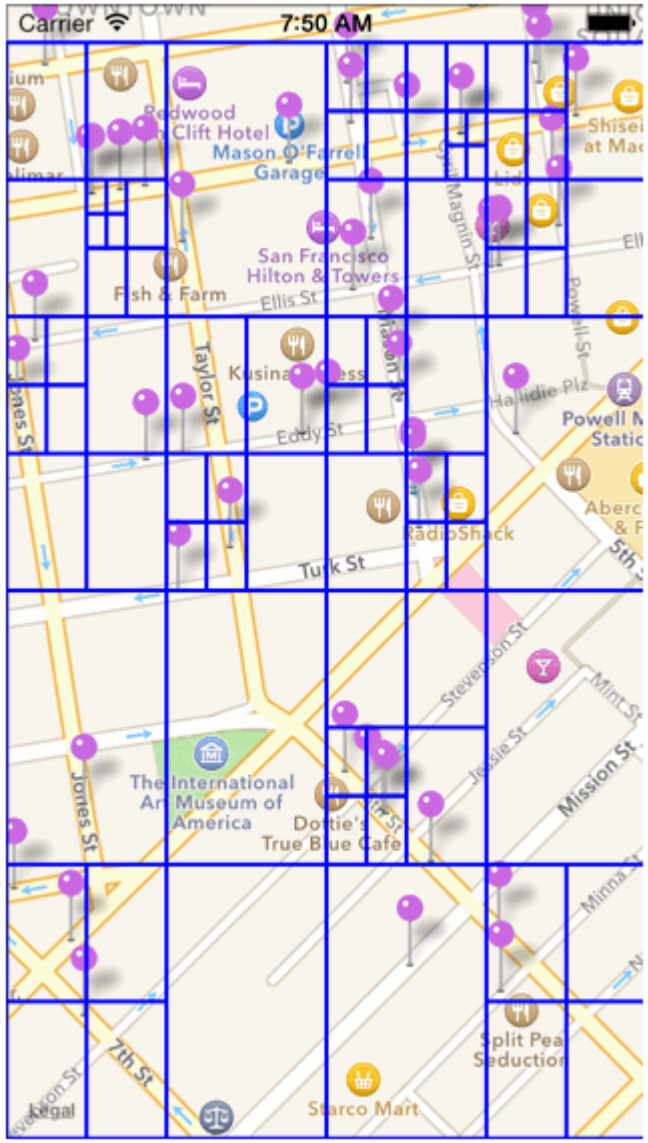
Region Quadtrees

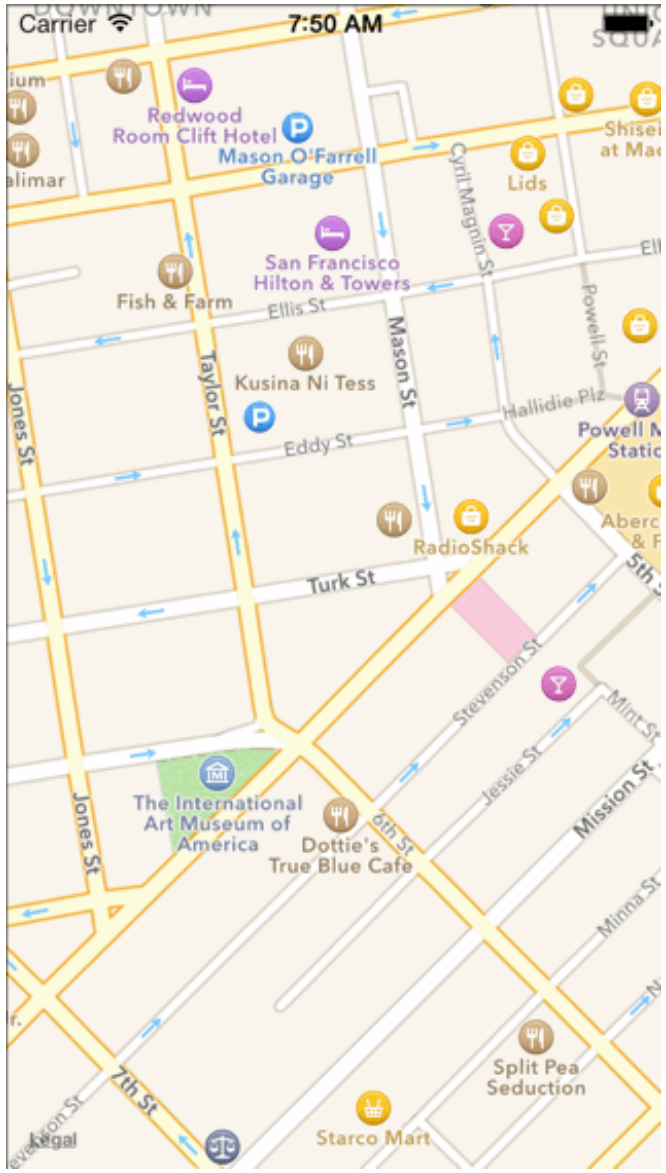
- Quadtrees take full advantage of the spatial structure, adapt to variable spatial detail
- Inefficient for highly inhomogeneous rasters
- Very sensitive to changes in the embedding space (e.g., translation, rotation)
- More useful for representing/compressing raster data.



Some slides borrowed from "GIS a computational perspective: second edition" by M. Worboys CRC press 2004.

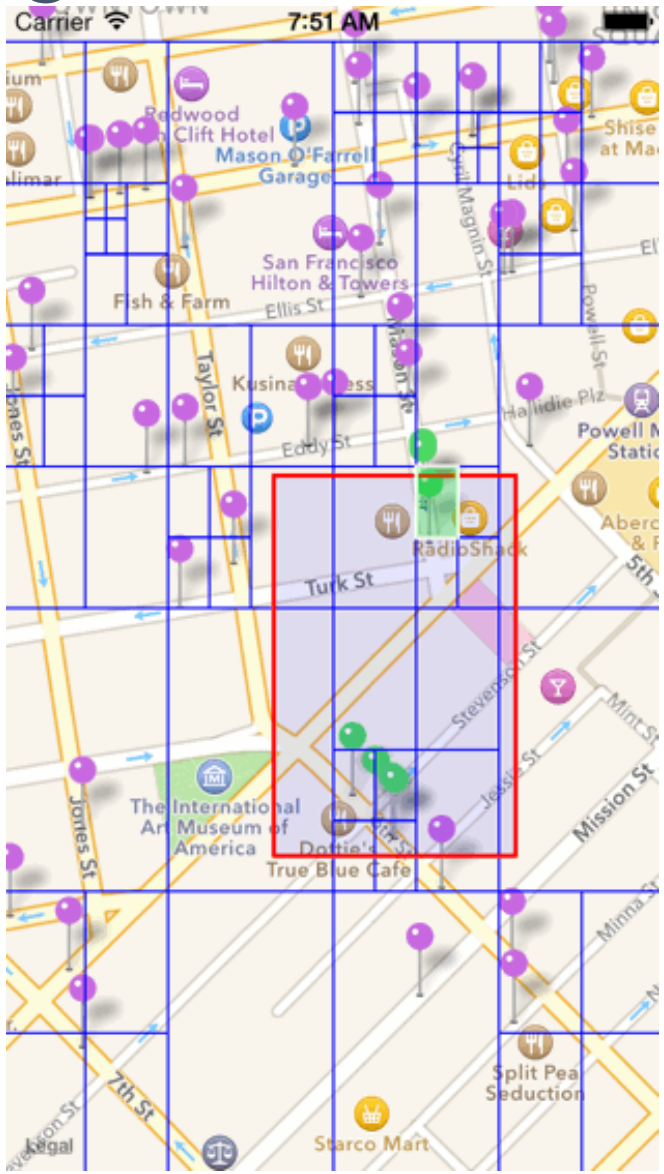
Region Quadtrees for points Insertion Animation





Animation available at this link: https://robots.thoughtbot.com/how-to-handle-large-amounts-of-data-on-maps

Region Quadtrees for points Range Query Example



Animation available at this link: https://robots.thoughtbot.com/how-to-handle-large-amounts-of-data-on-maps