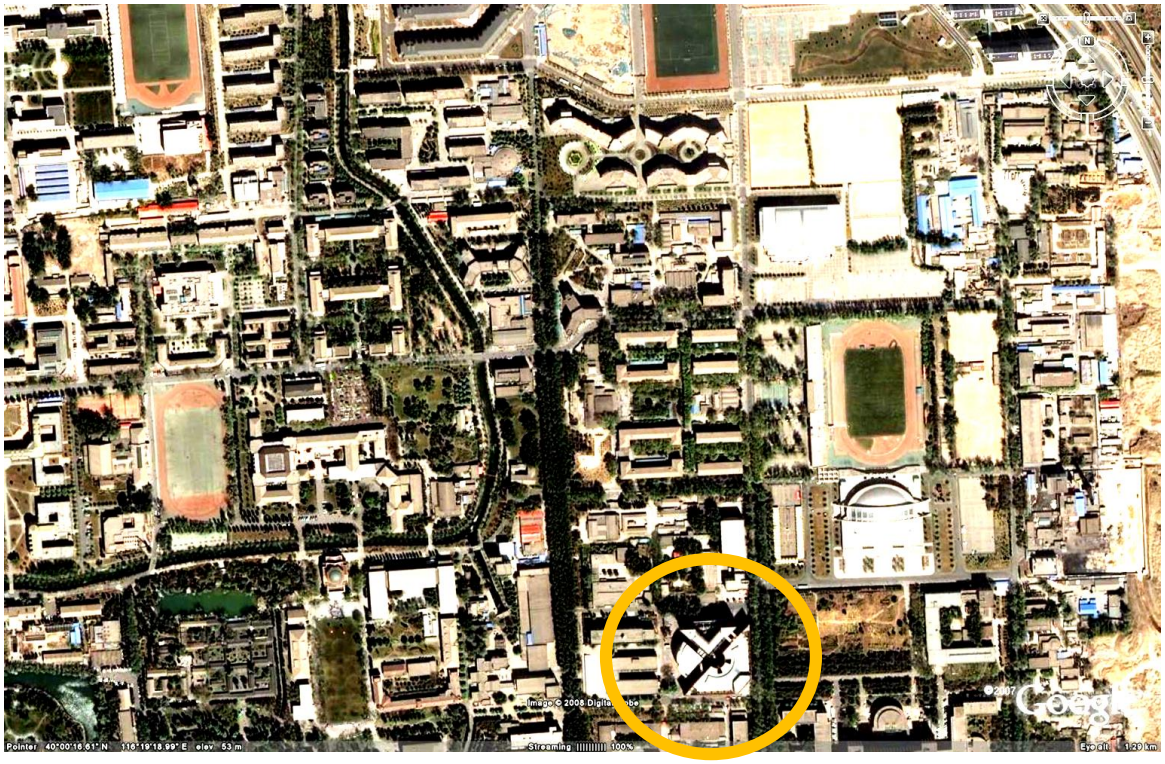
高级数据结构

Advanced Data Structure

4. Access Methods for Multidimensional Data Retrieval 2

Agenda

- Spatial access methods
  - *How to search database containing spatial objects*?
- R tree



Agenda

- Spatial access methods
  - Computational model
  - General method
- R tree

How to represent real objects

- Two main approaches
  - **Raster(栅格) model**

    To divide the space into cells of
    - regular size
    - regular shape

    Each cell is assigned the value of the attribute

    Represent the objects by raster array

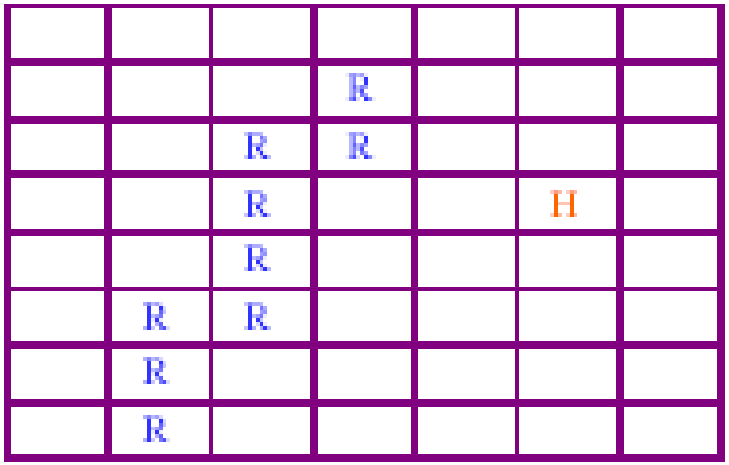
  - **Vector(向量) model**

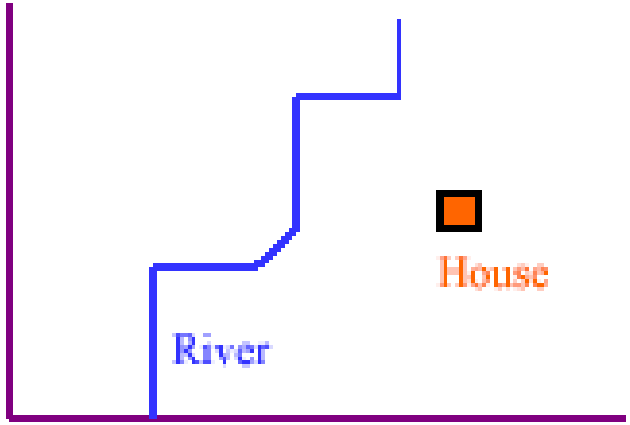
    Represent the objects by geographic positions

    Point (x,y)

    Line (x1,y1, x2,y2, ...,xn,yn)

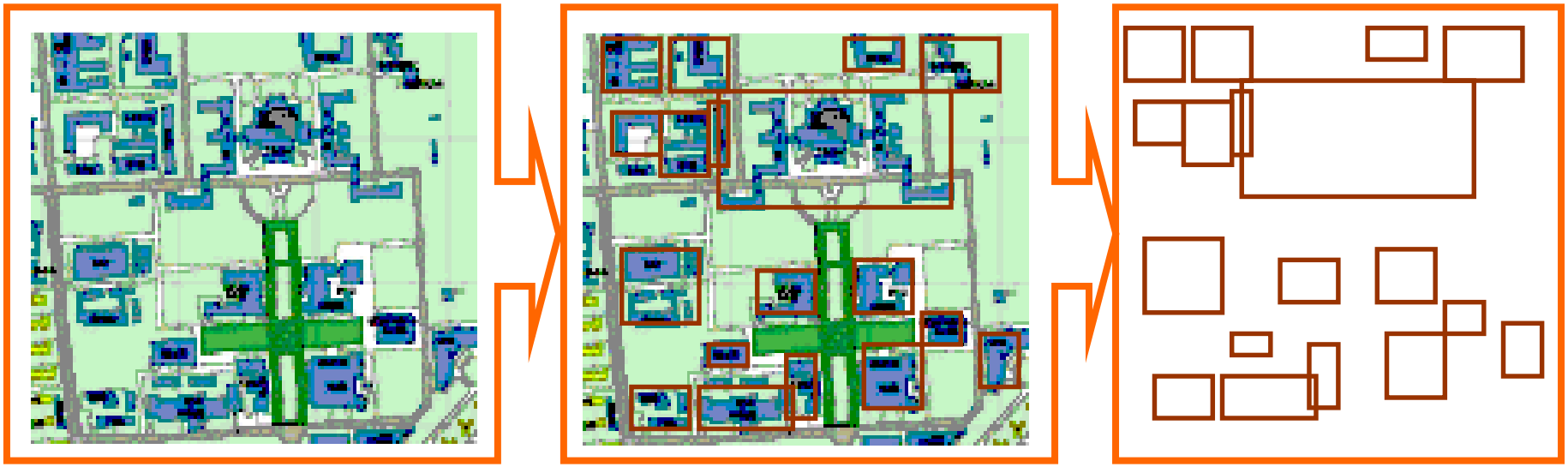
    Region (x1,y1, ...,xn,yn, x1,y1)





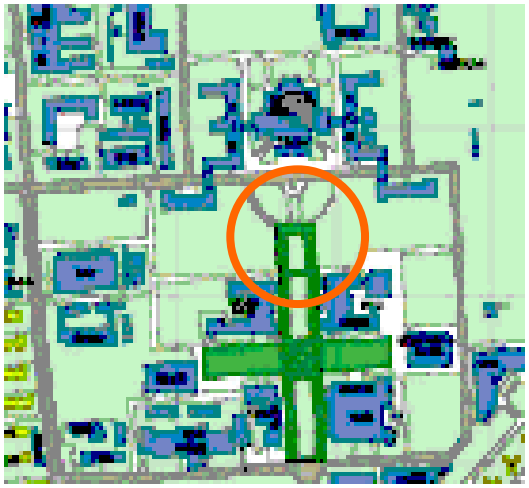
How to represent real objects - General strategy

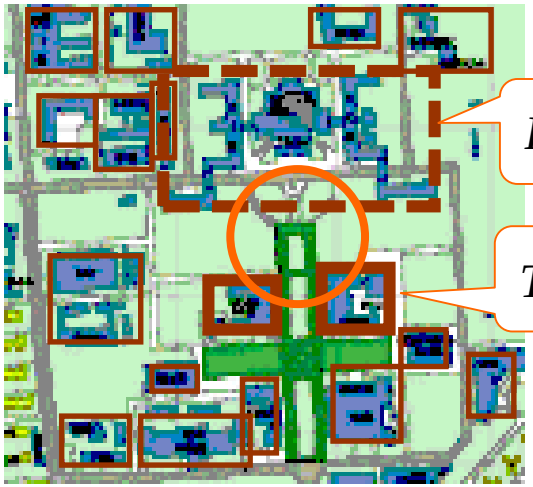
- Real object is too complex
- Using approximations instead of the exact shapes?
  - Minimum Bounding Rectangle (MBR)

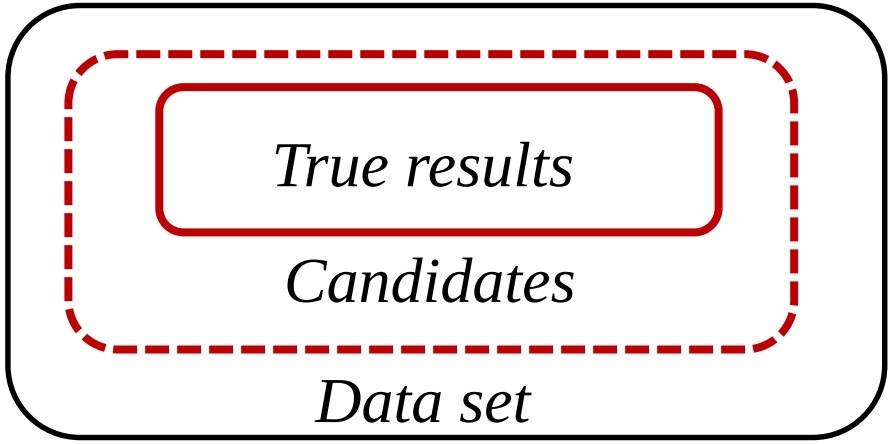


How to represent real objects - General strategy

- How to use MBR in searching?
- **Progressive refinement**
  - **Filter step**: based on objects' approximations → output the candidate set (quick)
  - **Refinement step**: comparison of objects' actual shapes → output the true answer (slow)

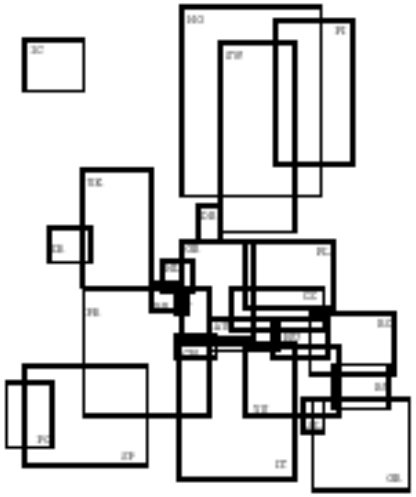


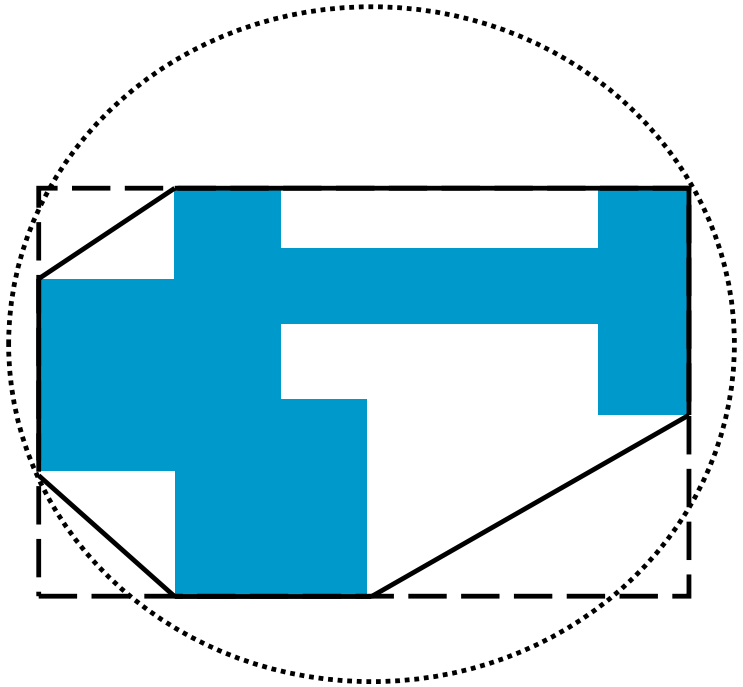




Other approximate bounding shapes

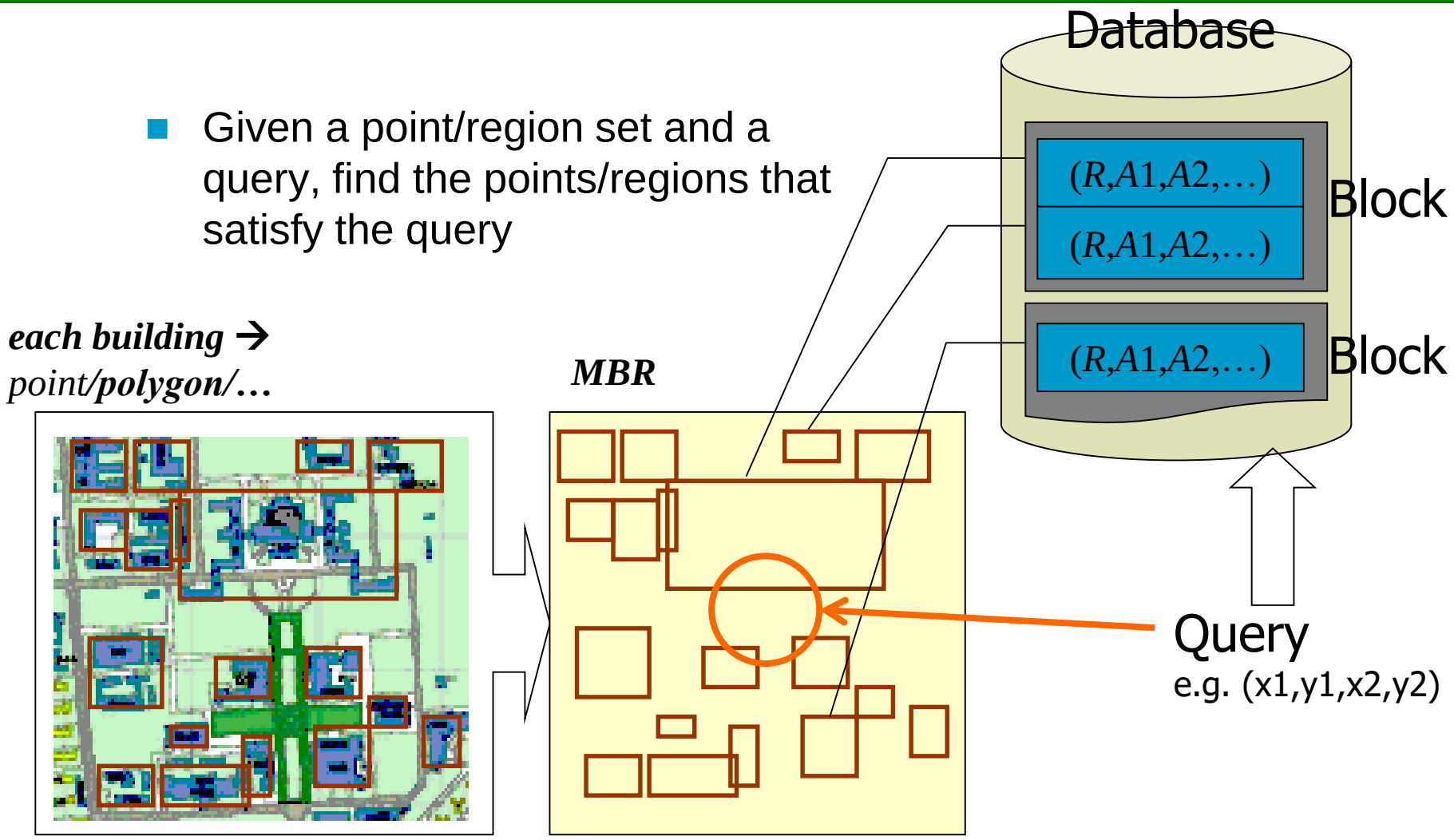
- MBR
- Encompassing circle
- Convex hull(凸包)





Spatial access methods (SAM) - Problem

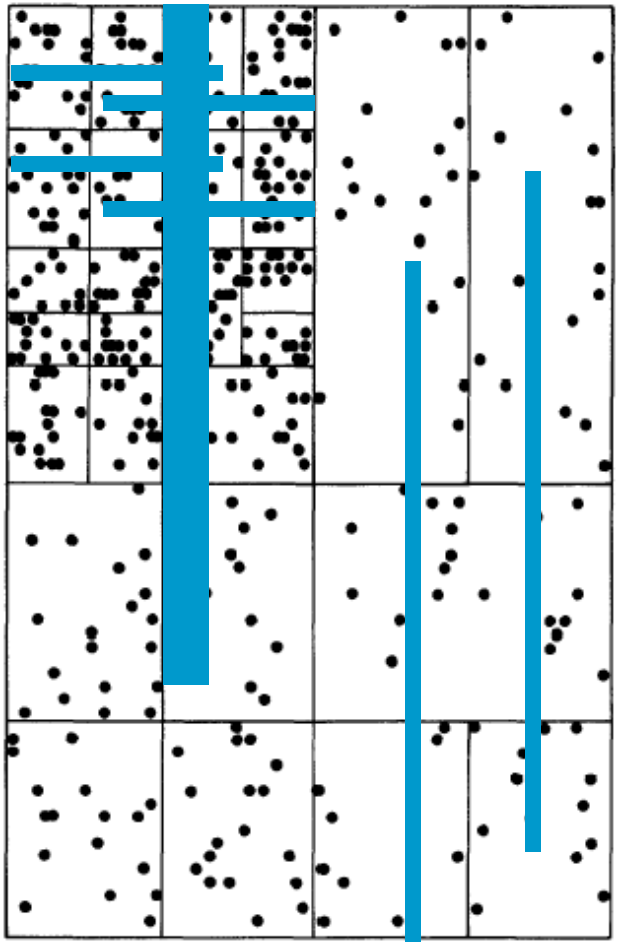
- Given a point/region set and a query, find the points/regions that satisfy the query

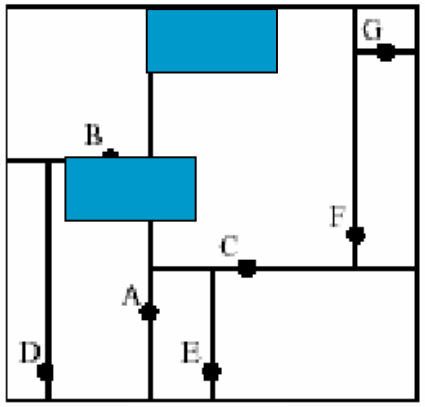


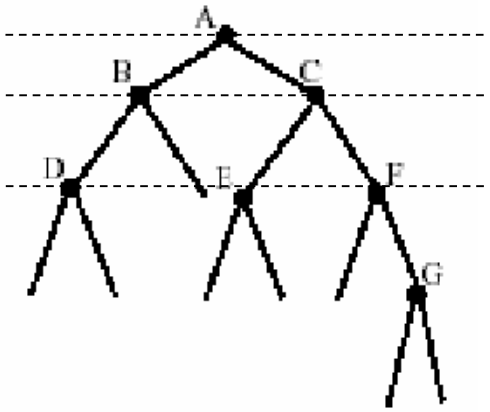
Agenda

- Spatial access methods
  - Computational model
  - General method
- R tree

Grid File or K-d tree?







Problem: Point Access Methods can index points. What about regions?

How to index the approximation directly?

How to handle higher dimensionality?

Methods

- General strategies
  - Organize the embedding space
  - **Organize the specific set of data**
- Methods
  - Use the transformation technique
  - Z-ordering and quadtrees
  - Spatial Access Methods (SAMs)
    - R-tree and variations

Agenda

- Spatial access methods
- R tree

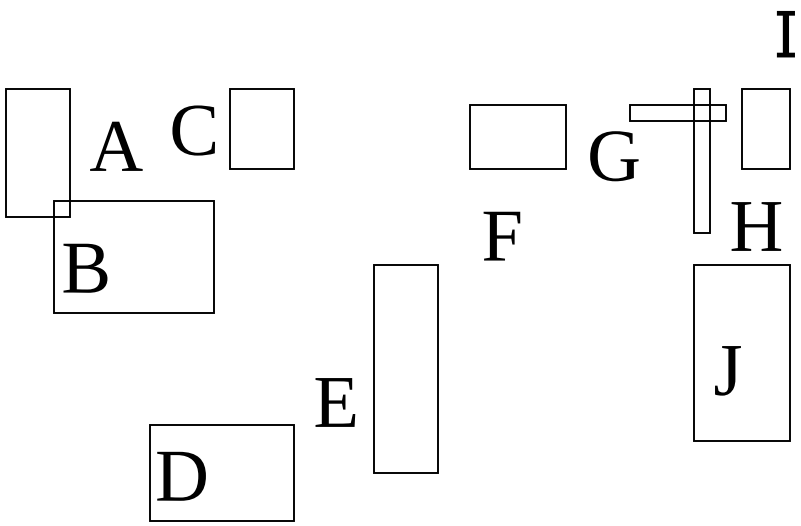
R-trees

- Organize the specific set of data
- Expanded version of **B+ tree**
  - Index nodes and data (leaf) nodes
  - All leaf nodes appear on the same level
  - Every node contains between *m* and *M* entries
  - The root node has at least 2 entries (children)
- A external memory tree

- R-tree can index both objects and points

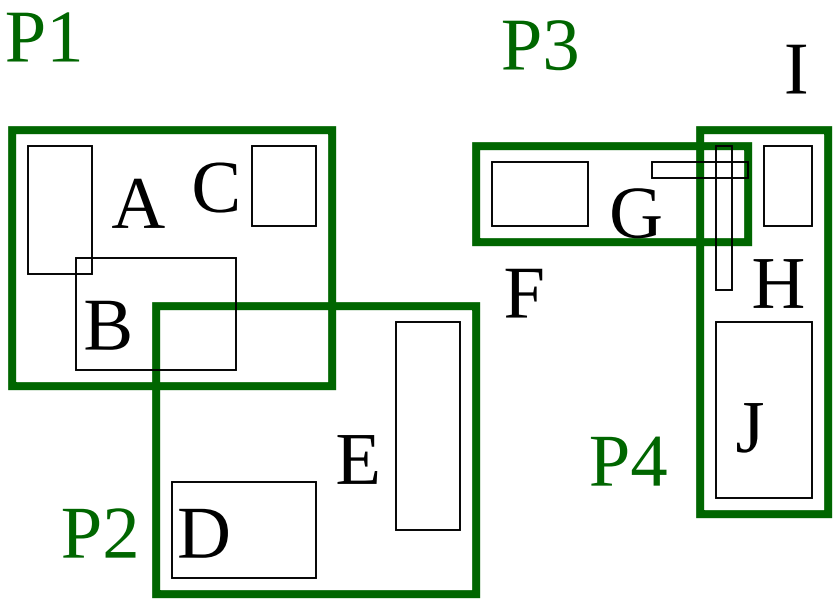
Example

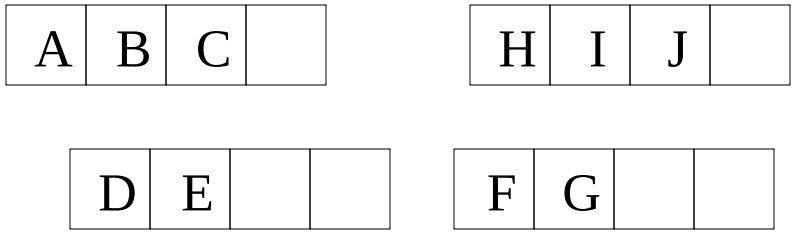
- Fanout 4 (i.e., M=4)
- Group nearby rectangles to parent minimal bounding rectangles (MBR);
- Each group → disk block



Example

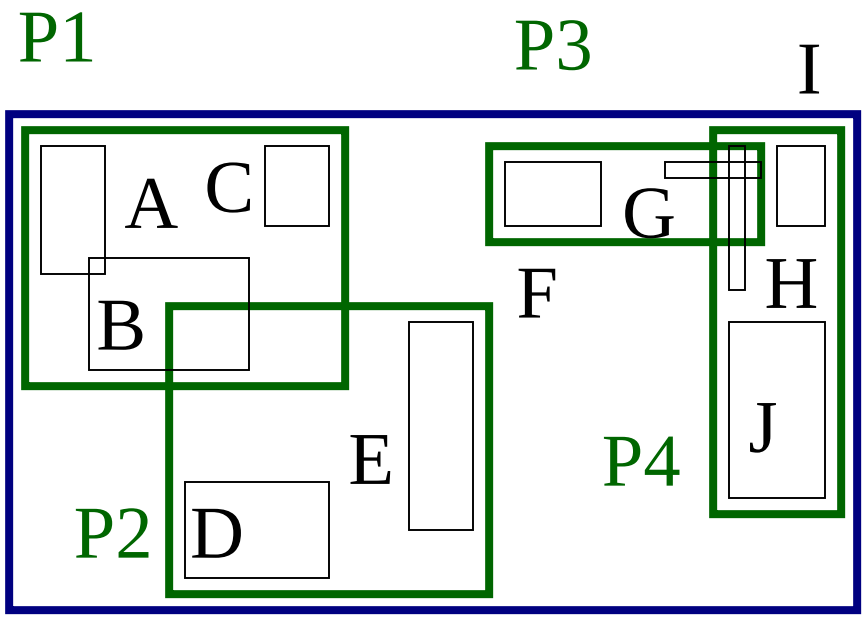
- Fanout 4

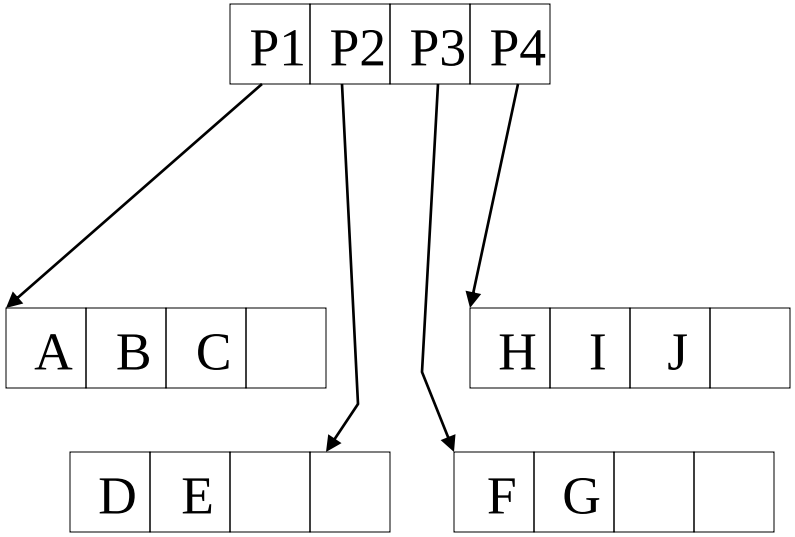




Example

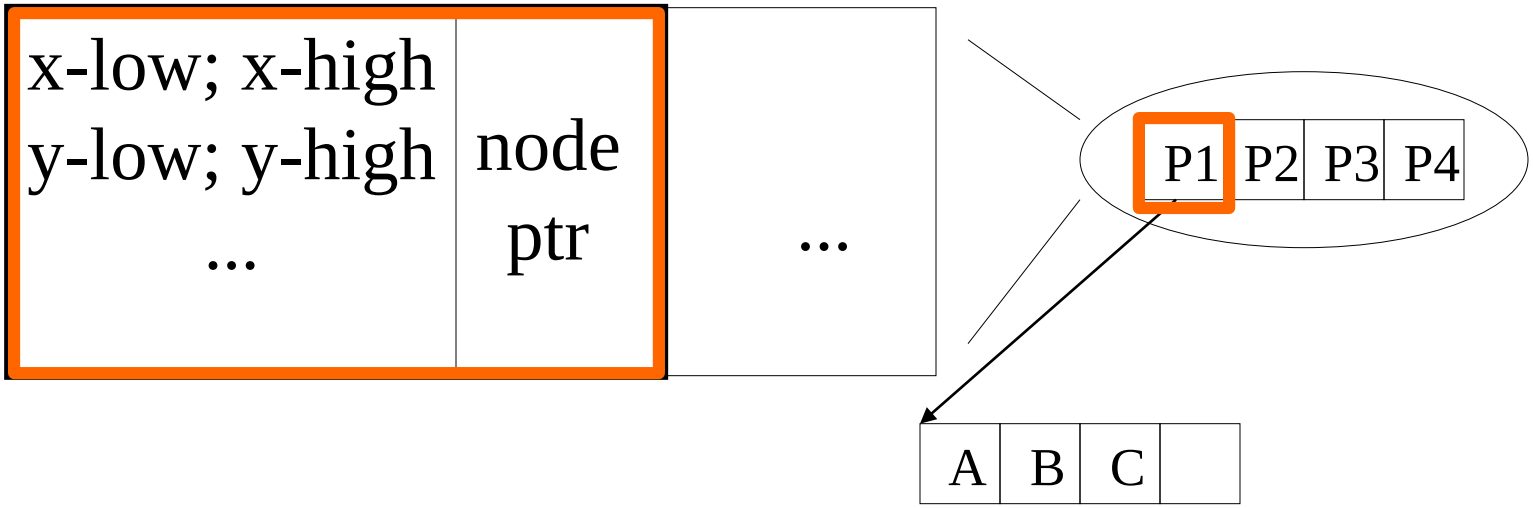
- Fanout 4





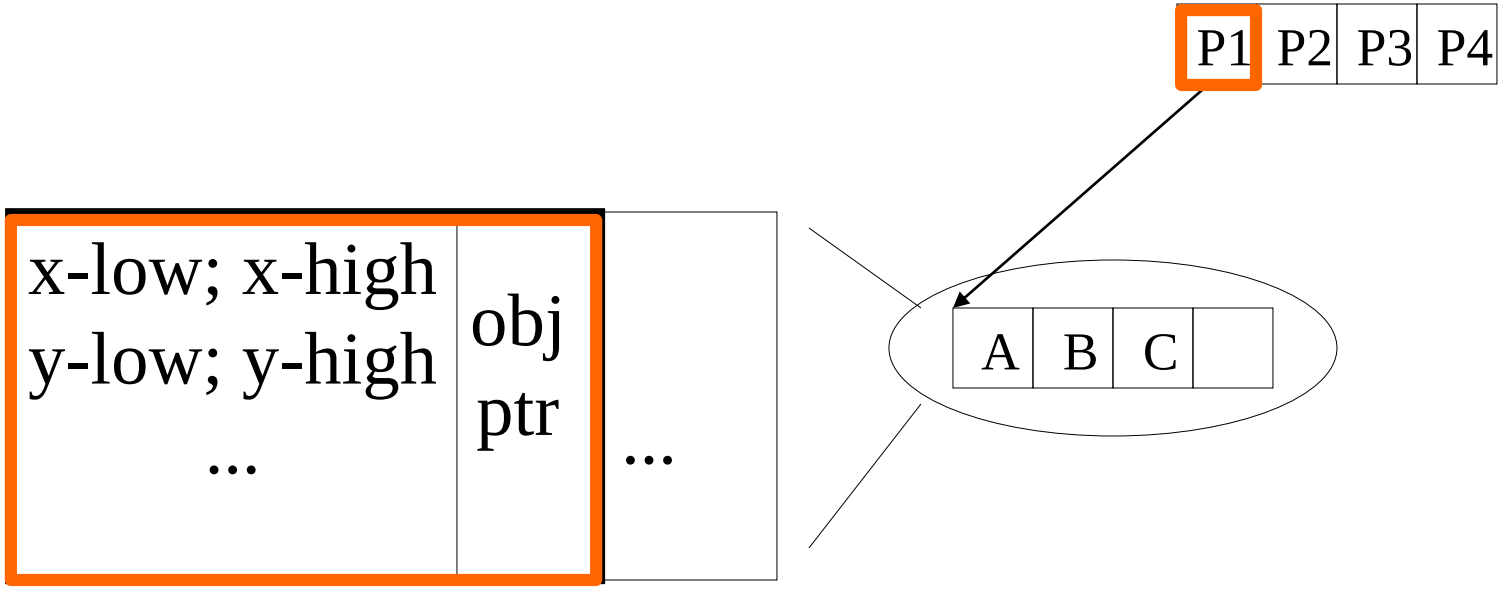
R-trees - Non-leaf nodes

- {(MBR; node_ptr)}



R-trees - Leaf nodes

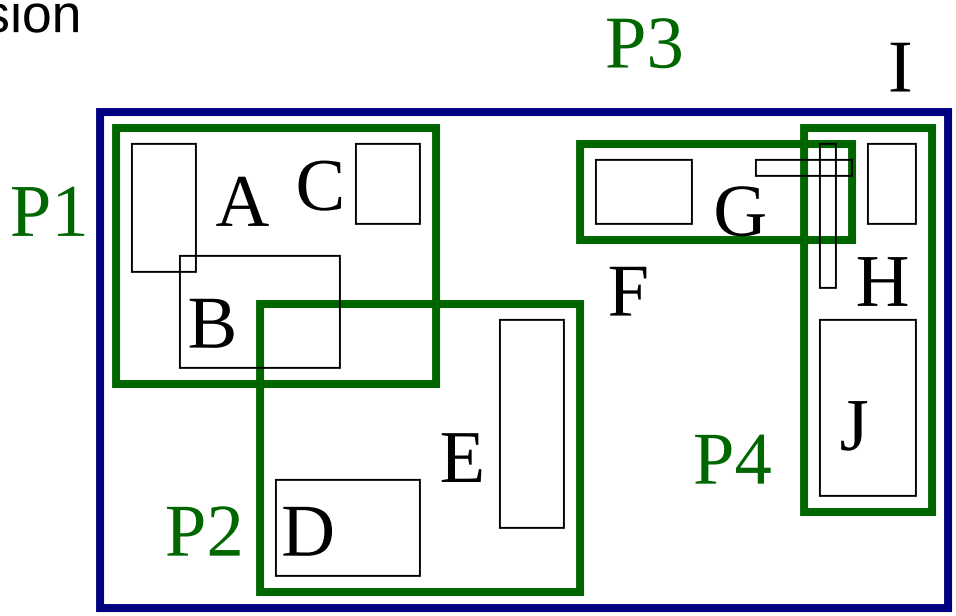
- {(MBR; obj_ptr)} for leaf nodes



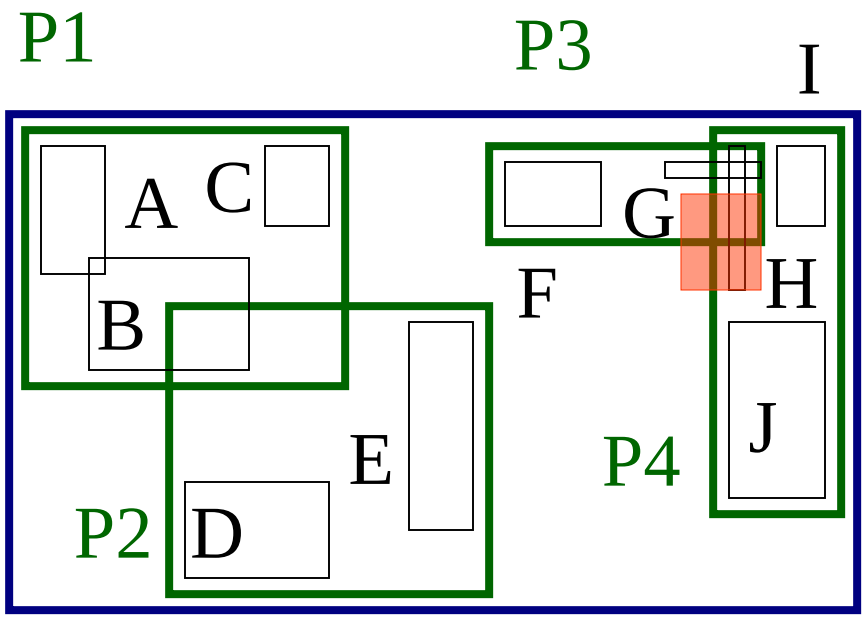
R-trees: Search

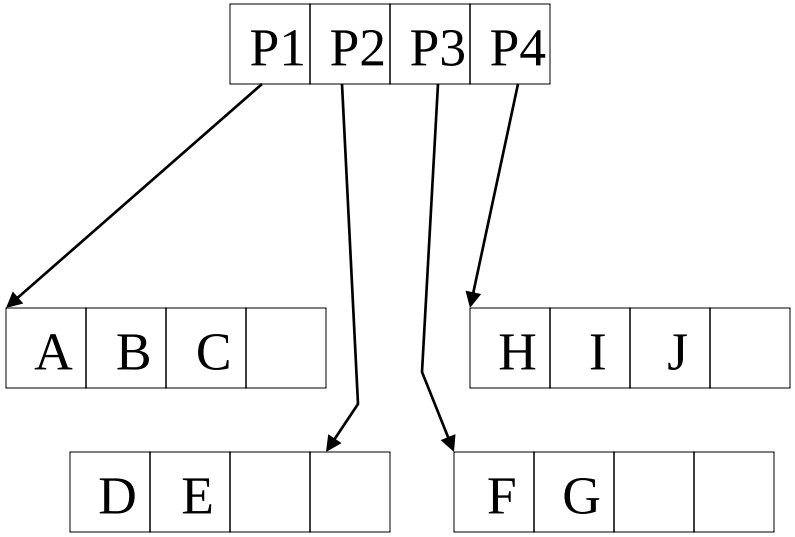
- Observations
  - Every parent node completely covers its 'children'
  - A child MBR may be covered by more than one parent - it is stored in ONLY ONE of them

    There may be overlaps → a point query may follow multiple branches.
  - Everything works for **any** dimension



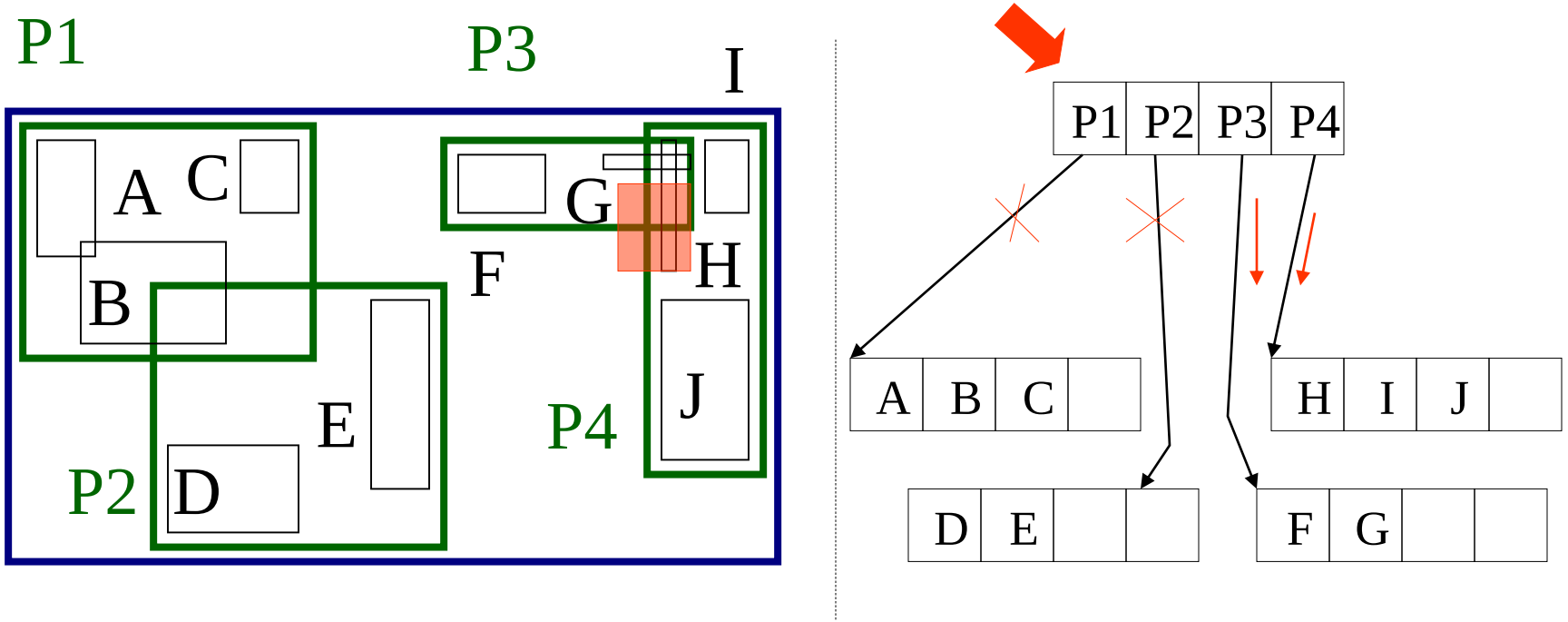
R-trees: Search



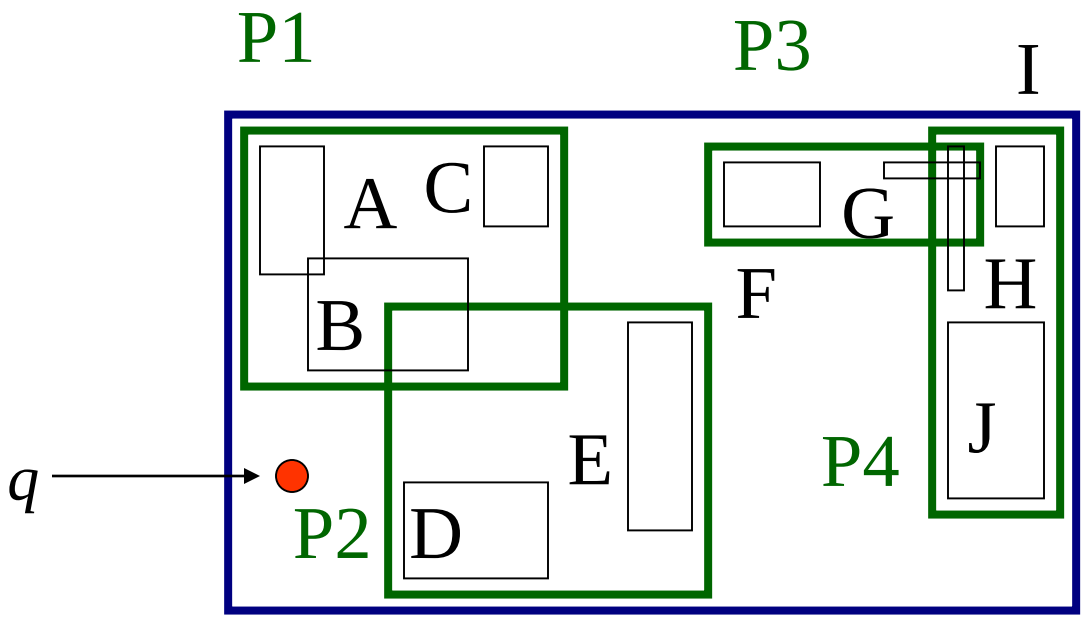


R-trees: Search

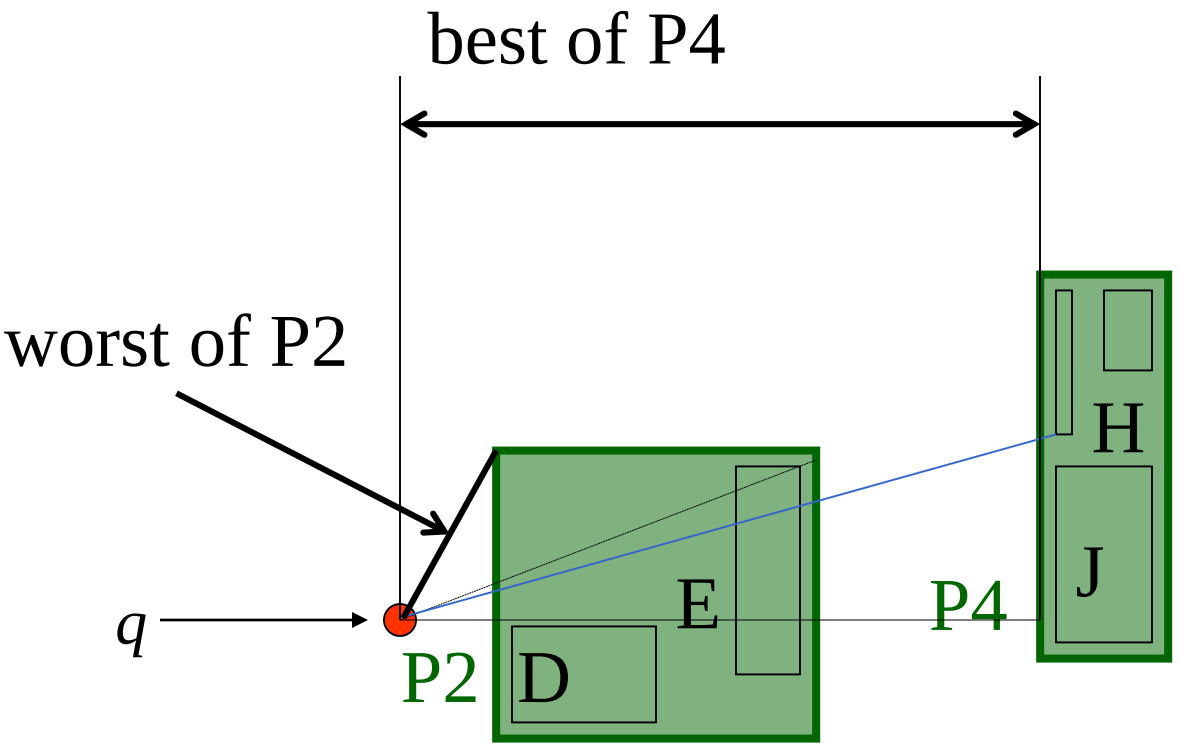
for each branch,
    if its MBR intersects the query region
        call range-query (or print out, if this is a leaf)



R-trees - NN search



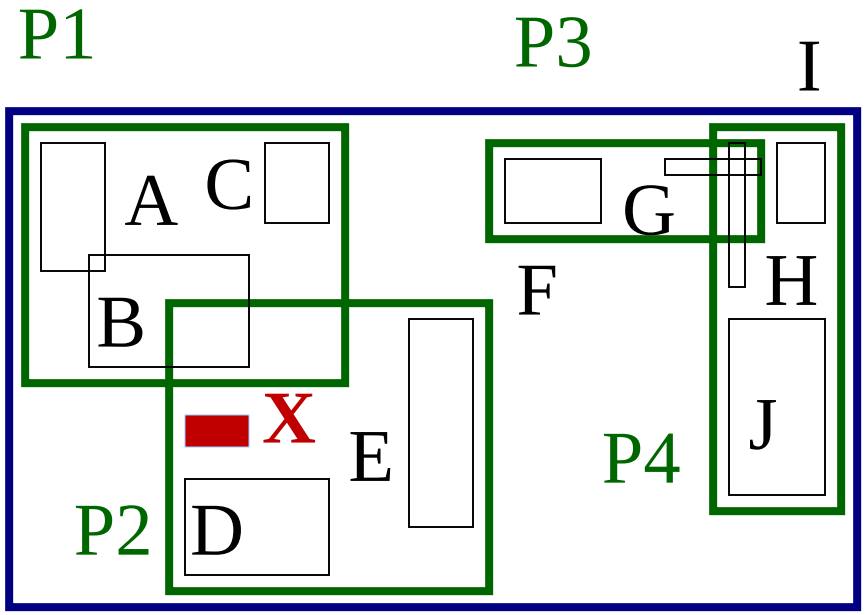
R-trees - NN search

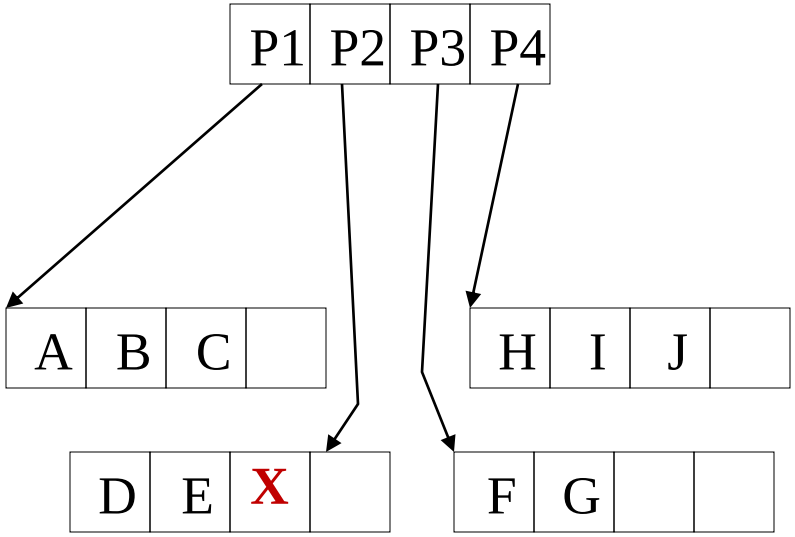


→ P4 is useless for nn search

R-trees: Insertion

Insert X

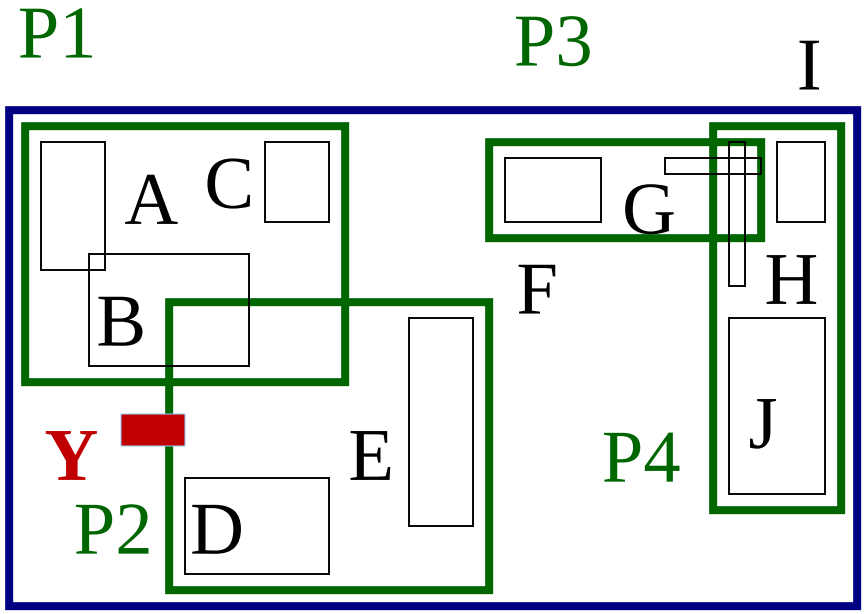


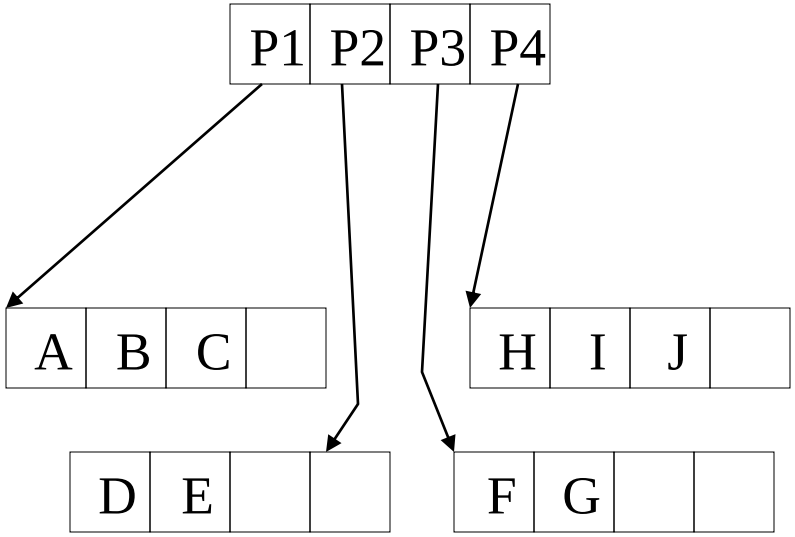


R-trees: Insertion

Insert Y

1) Find the node to insert the new object, insert and adjust MBR
2) (Maybe) **Extend the parent MBR**
3) (Maybe) Split node: partition the MBRs into two groups

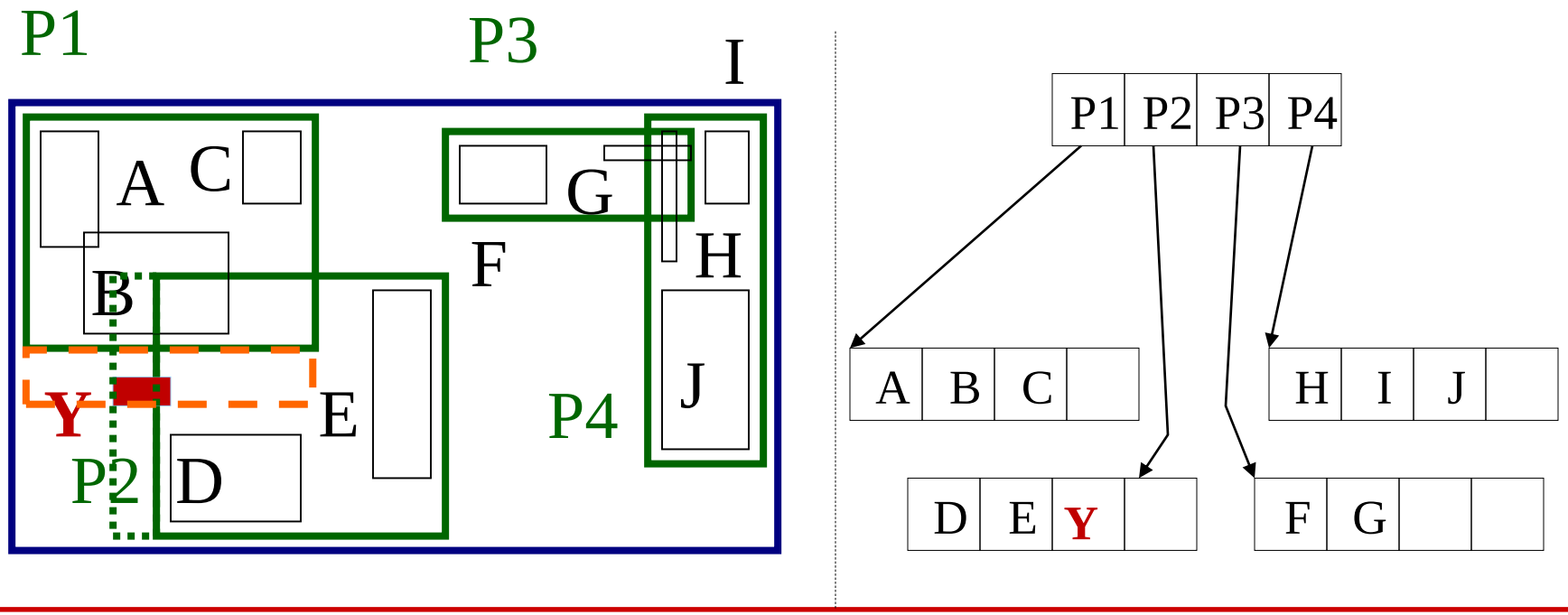




R-trees: Insertion

How to find the node to insert the new object?

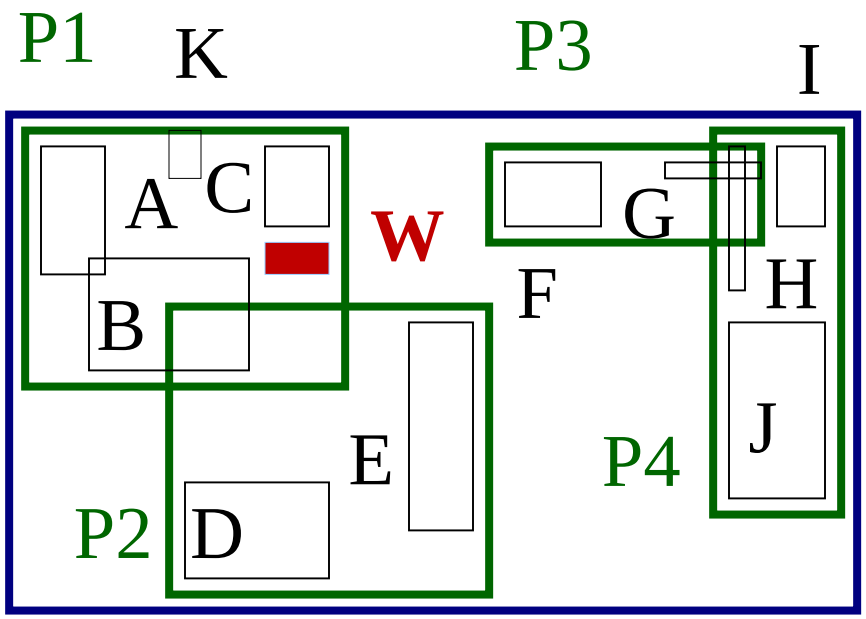
Find the entry that needs the least enlargement to include Y

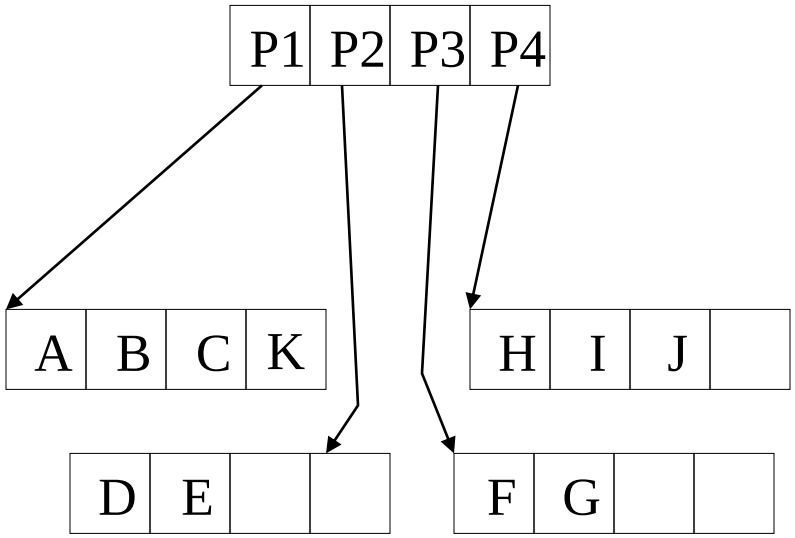


R-trees: Insertion - Split

Insert W

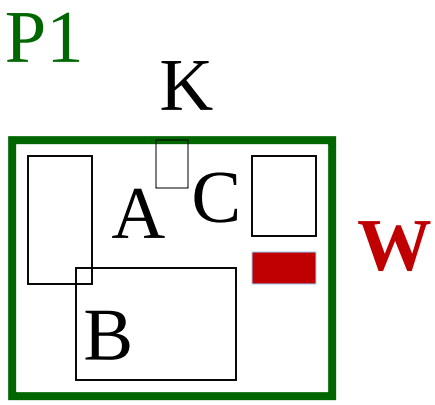
Split node: partition the MBRs into two groups. How?





R-trees: Insertion - Split

- Split node P1



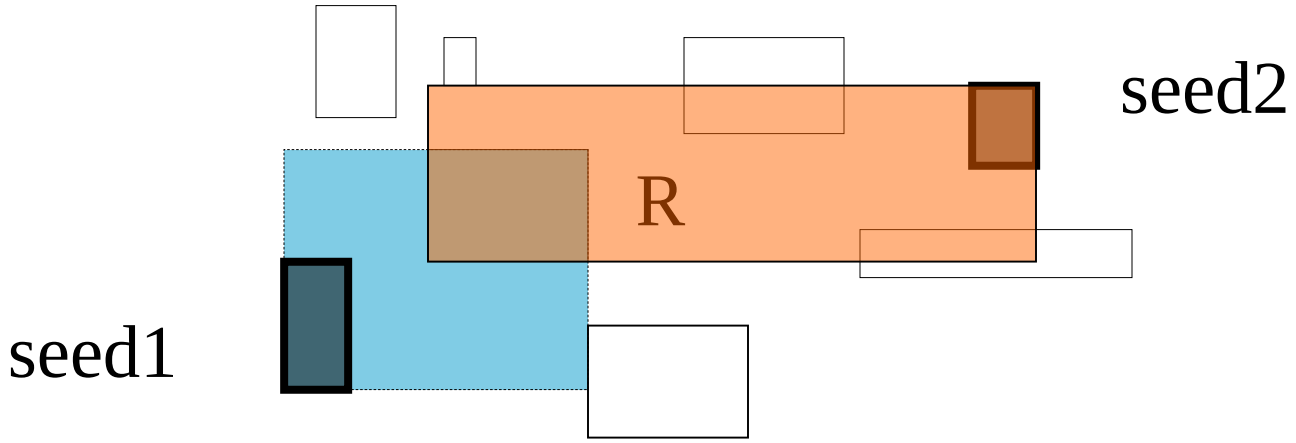
•Linear split

•Exponential split:

Approximately 2^M choices

R-trees: Insertion - Split

- Pick two rectangles as ‘seeds’
- Assign each rectangle ‘R’ to the ‘closest seed’
- ‘Closest’: the smallest increase in area



R-trees: Insertion - Split

- How to pick Seeds:
  - **Linear:**

    Find the highest and lowest side in each dimension

    Choose the pair with the greatest separation

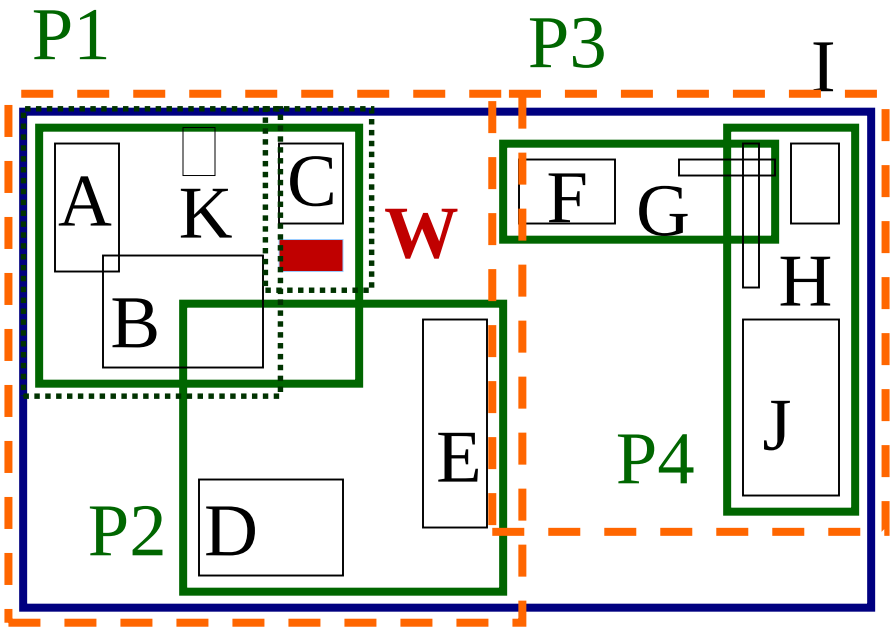
  - **Quadratic:**

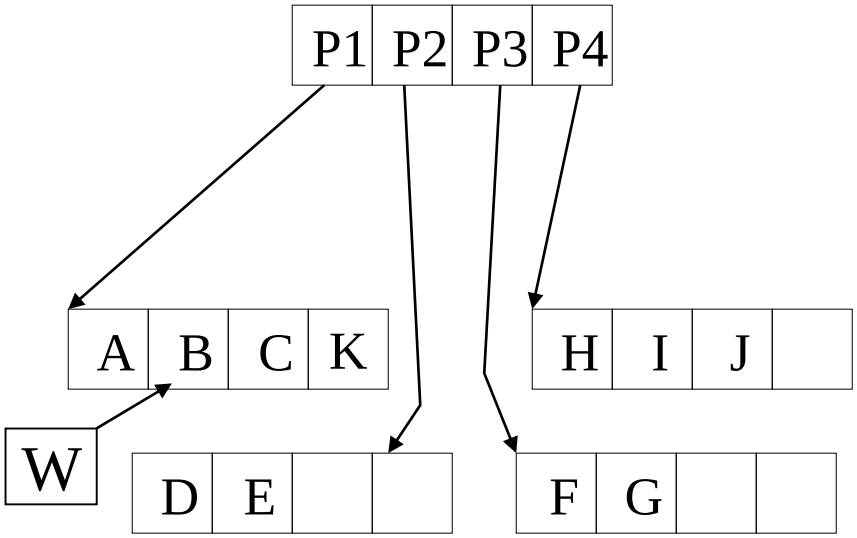
    For each pair E1 and E2, calculate the rectangle J=MBR(E1, E2) and d= J-E1-E2.

    Choose the pair with the largest d

R-trees: Insertion – Overall method

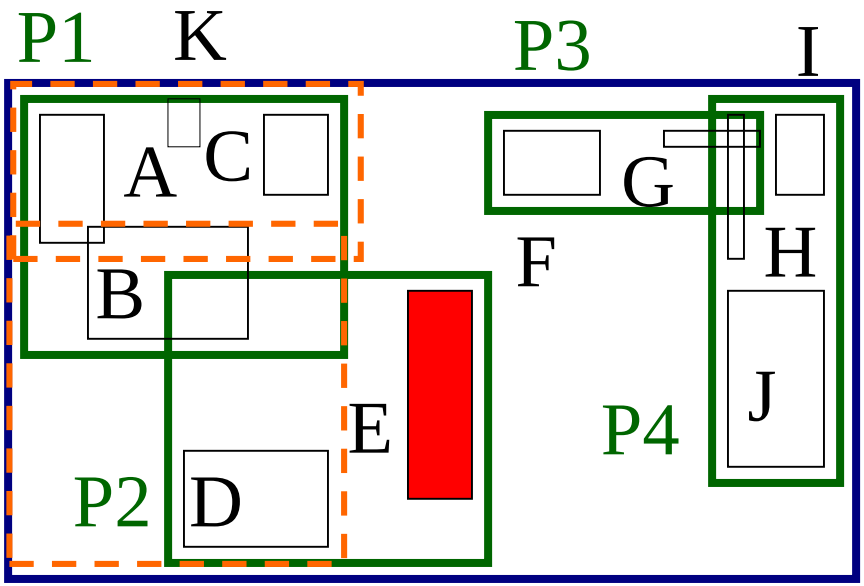
- Find the leaf node to insert an entry E
  (Find the entry that needs the least enlargement to include E)
- If leaf node is full, then **Split,** otherwise insert there
  - Propagate the split upwards, if necessary
- Adjust parent nodes

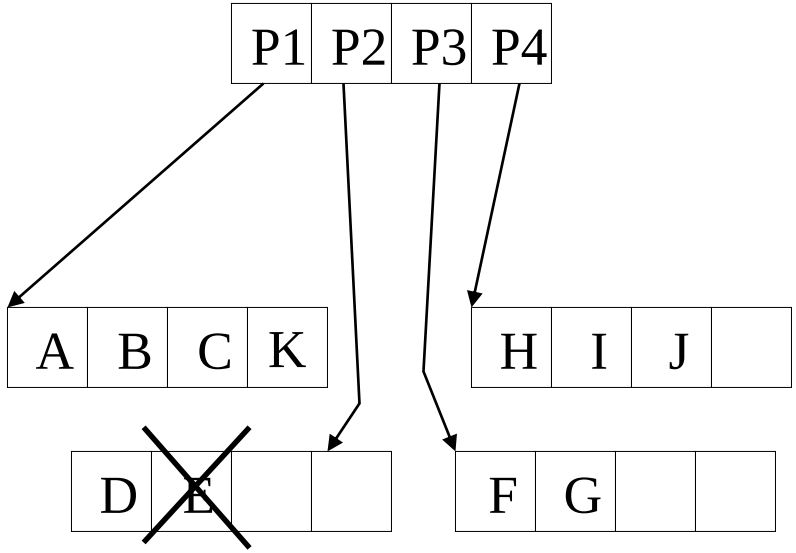




R-Trees: Deletion

- Find the leaf node that contains the entry E
- Remove E from this node
- If underflow:
  - Eliminate the node by removing the node entries and the parent entry
  - Reinsert the orphaned (other entries) into the tree using **Insert**
  - Propagate upward if necessary





Conclusion remarks

- R tree
  - Expanded version of B+ tree
  - A multi-way external memory tree for multi-dimensional points or objects
  - Index the approximation (MBR) directly, **without dividing the spatial objects**
  - Overlaps cause performance decrease
  - Handle higher dimensionality comparing with grid file or k-d tree
  - Most widely used spatial access method in both research and industrial fields
  - Many variants: SS-tree, SR-tree, …
- "Dimensionality curse":
  - All the multidimensional index structures suffer from high dimensionalities

Homework

- Read 5.2 and additional readings
- Given a 2D data set by yourself, construct the R tree

Thanks

Feedback welcome