PRINCIPLES OF DATA ORGANISATION

R-Tree

MOTIVATION

- How to search effectively in more than one dimension?
  - Objects in 2D we want to query
    - E.g., find all natural museums in a certain distance from your home
- How to represent a spatial object in the database?
  - Single Dimension-Based Indexing has an issue with locality
    - Space filling curve = reduce 2D to 1D + e.g., B-tree
  - Usage of full spatial information: Quad-tree, K-D-tree, K-D-B-tree
    - Not optimal for storing data in the secondary memory
- We want a B-tree for multidimensional data: R-tree

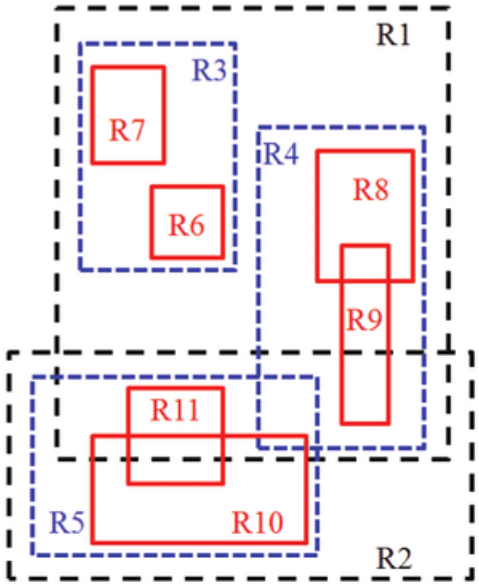
R-TREE : INTRODUCTION

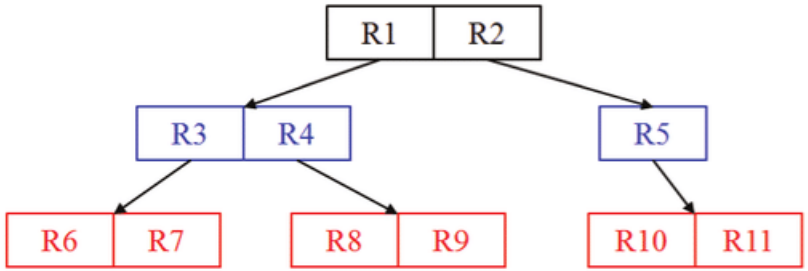
- Guttman 1984
- Height-balanced tree
  - Extension of redundant B-tree for spatial data
    - Pointers to data only from the leaf level
- Nodes correspond to disk pages
- Each node contains ***n***-dimensional bounding box ***I***
  - MBRs (minimum bounding rectangle)
- The leaf level contains pointers to the spatial objects
- Inner levels contain MBRs
  - MBR of a node is MBR of all children

DEFINITION

- Every leaf contains between m and M index records
  - M = the maximum number of entries in a node
  - $m \leq M/2$
    - Not fixed to 50%! (We can choose.)
- Every inner node other than root contains between m and M entries
- The root has at least 2 children unless it is a leaf
- Each node contains a set of entries E consisting of
  - $E.p$ – pointer to the child node (inner node) or spatial object identifier (leaves)
  - $E.I$ – n-dimensional bounding box $I = (I_0, I_1, \ldots, I_{n-1})$, where I_j corresponds to the extent of the object I along j-th dimension
    - For each record E, $E.I$ is the minimum bounding rectangle
- All leaves appear at the same level
  - The tree is balanced

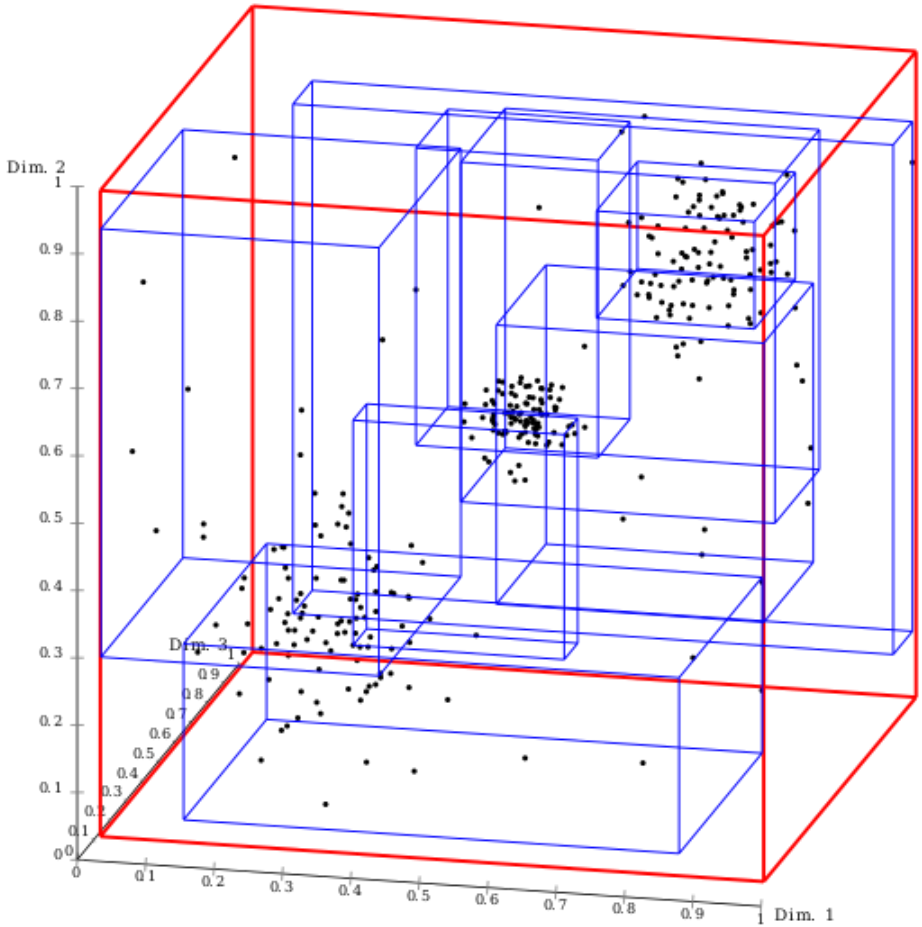
- Height of an R-tree with n records $\leq \log_m n$





Search query: all objects that intersect with a query object (a point or a bounding box)

https://www.researchgate.net/figure/R-Tree-indexing-example-2D-visualization-a-hierarchical-dependencies-b_fig4_225542188



https://en.wikipedia.org/wiki/R-tree

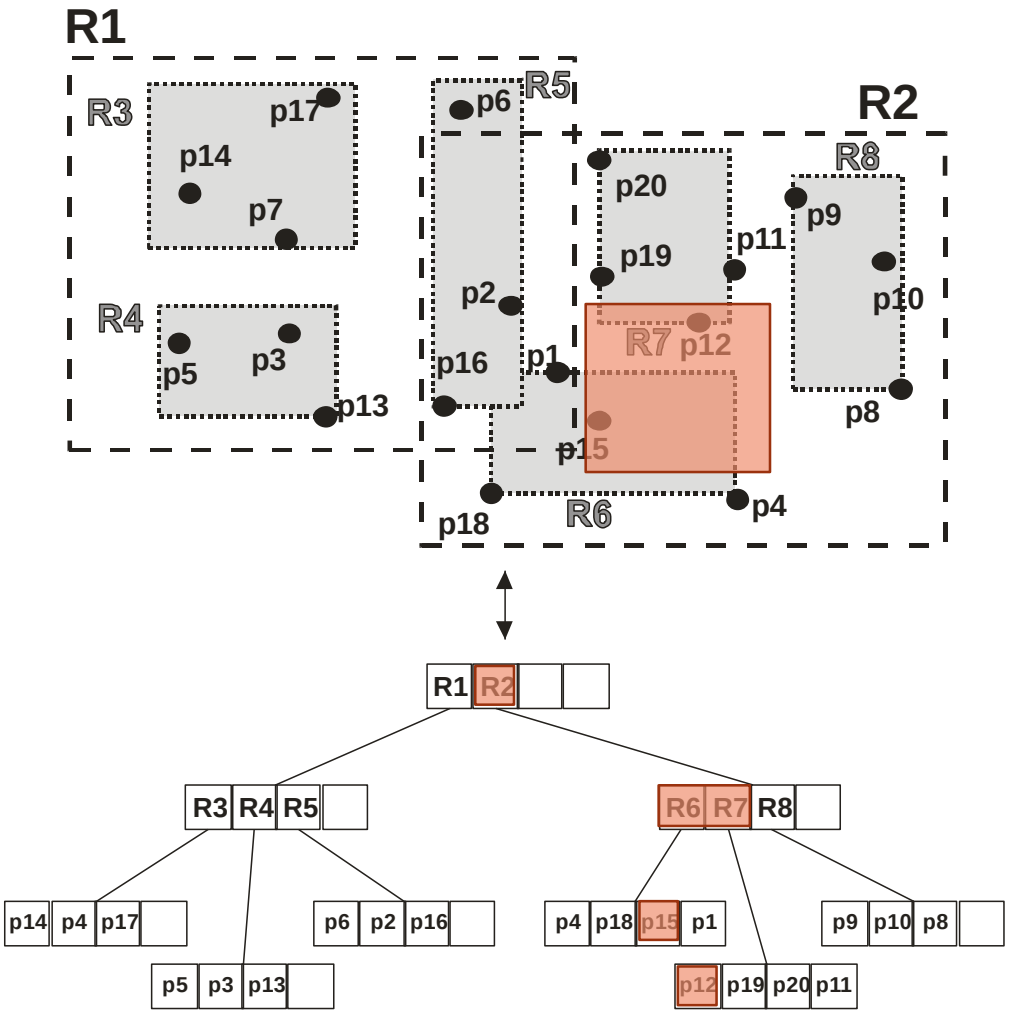
SEARCH

- Result of a search is a set of objects intersecting the query object
  - Search key is represented by the bounding box of a query object
- Search may need to follow multiple root-to-leaf paths ~ no guaranteed worst-case performance
  - The more MBRs intersect the worse the performance
  - Update algorithms force the bounding rectangles to be as much separated as possible (small, minimal ovelapping) allowing efficient filtering while searching

```
Search_R(T,S)
Input: R-tree with a root T, rectangle S
Output: identifiers of objects overlapping S

IF T != leaf THEN
   FOR EACH E ∈ T DO
      IF E.I ∩ S THEN Search_R(E.p,S);
ELSE
   FOR EACH E ∈ T DO
      IF E.I ∩ S THEN Output(E.p,S);
```

SEARCH EXAMPLE



INSERT

Insert_R(T,E)
Input: R-tree with a root T, index record E
Output: updated R-tree

ChooseLeaf(T,L,E); {chooses leaf L for E}
IF E fits in L THEN
  Insert(L,E);
  LL ← NIL;
ELSE
  SplitNode(L,LL,E);
AdjustTree(L,LL,T); {propagates changes upwards}
IF T was split THEN
  install a new root;

ChooseLeaf(T,L,E)
Input: R-tree with a root T, index record E
Output: leaf L

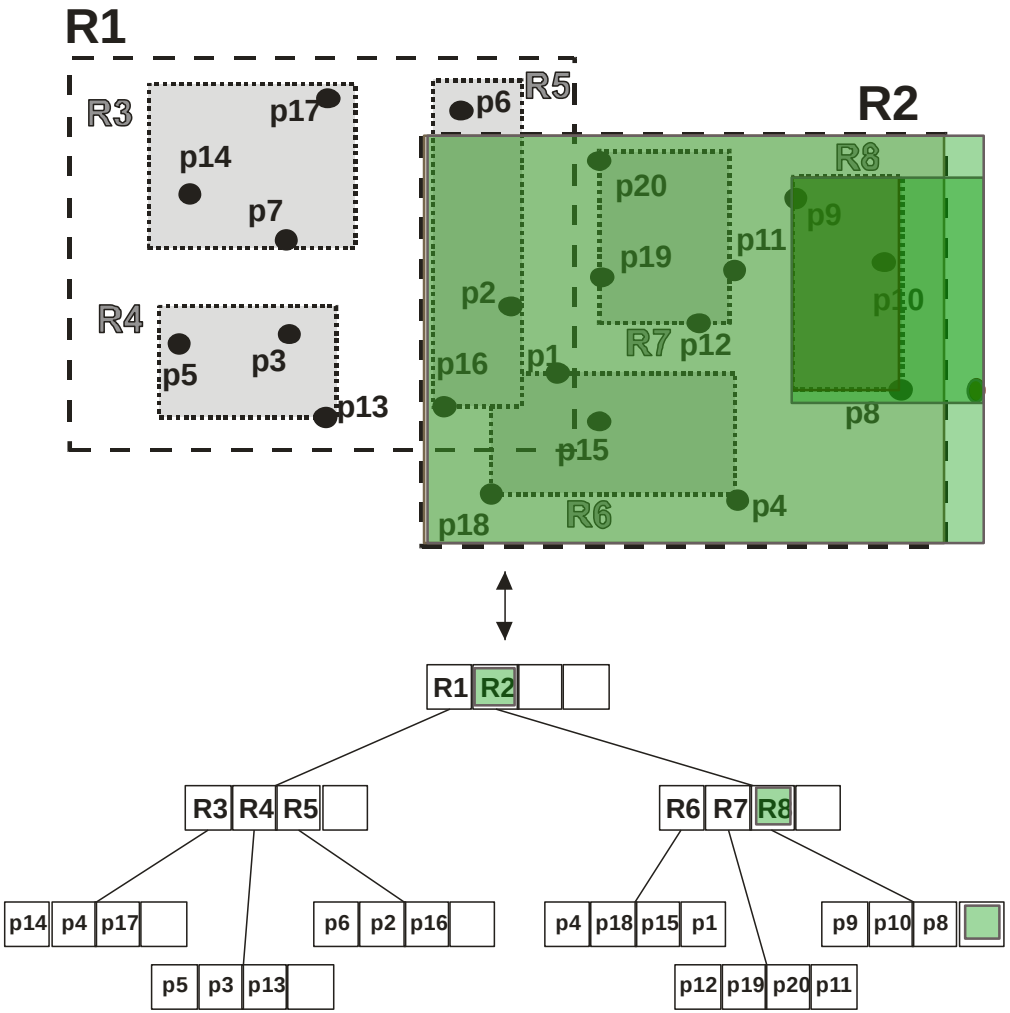
N ← T;
WHILE N ≠ leaf DO
  chose such entry F from N whose F.I needs least enlargement to include E.I, in case of tie choose F.I with smallest area;
  N ← F.p;
L ← N;

INSERT

AdjustTree(L,LL,T)
Input: R-tree with a root T, leafs L and LL
Output: updated R-tree

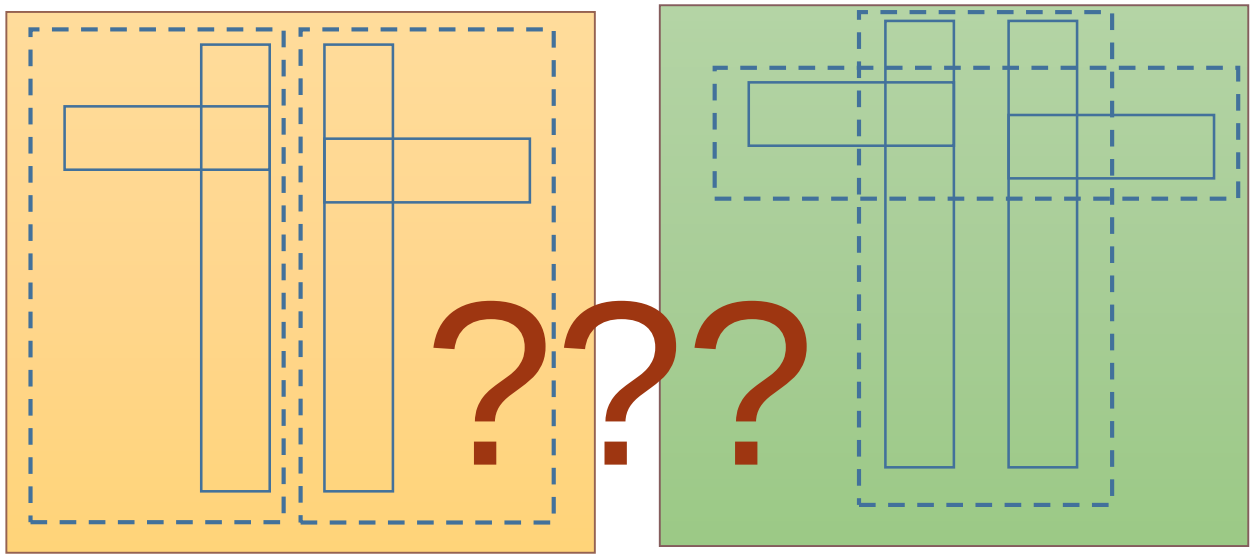
N ← L;
NN ← LL;
WHILE N ≠ T DO
  P ← Parent(N);
  PP ← NIL;
  modify EN.I in P so that it contains all rectangles in N;
  IF NN != NIL THEN
    create E_{NN}, where E_{NN}.p = NN and E_{NN}.I covers all rectangles in NN;
    IF E_{NN} fits in P THEN
      Insert(P, E_{NN});
      PP ← NIL;
    ELSE
      SplitNode(P,PP, E_{NN});
  N ← P;
  NN ← PP
LL ← NN;

INSERT EXAMPLE



SPLITTING IN R-TREE

A good split should minimize the probability of searching in too much nodes → the total covering area should be minimized ~ dead space (space not covering any objects) should be minimized, but overlaps are problematic as well



SPLITTING IN R-TREE

- Exhaustive algorithm
  - Exhaustive generation of all possible divisions
  - 2^{M-1}
- Quadratic cost algorithm
  - The best seed (= a pair of MBRs) is picked
    - Two entries that would consume the most space if put together
  - The remaining MBRs are added one by one
    - Pick the one that would make the biggest difference in the area when put into one of the two groups
    - Add it to the one with the least difference
- Linear-cost algorithm
  - Seed picking is based on finding rectangles with the greatest normalised separation along any dimension

SPLITNODE (GUTTMAN)

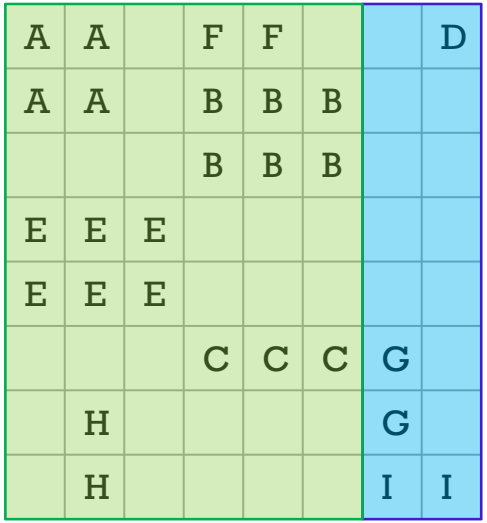
SplitNode(P,PP,E)
Input: node P, new node PP, m original entries, new entry E
Output: modified P, PP

PickSeeds(); { chooses first E_i and E_j for P and PP }
WHILE not assigned entry exists DO
  IF remaining entries need to be assigned to P or PP in order to have the minimum number of entries m THEN
    assign them;
  ELSE
    $E_i \leftarrow$ **PickNext**() {choose where to assign next entry}
    Add E_i into group that will have to be enlarged least to accommodate it. Resolve ties by adding the entry to the group with smaller area, then to the one with fewer entries;

Quadratic complexity

PickSeeds()
FOREACH E_i, E_j ($i \neq j$) DO
  $d_{ij} \leftarrow$ area(J) − area(E_i.I) − area(E_j.I);
    { J is the MBR covering E_i and E_j }
  pick E_i and E_j with maximal d_{ij};

PickNext()
FOREACH remaining E_i DO
  $d_1 \leftarrow$ area increase required for MBR of P and E_i.I;
  $d_2 \leftarrow$ area increase required for MBR of PP and E_i.I;
pick E_i with maximal $|d_1 - d_2|$;



M = 8, m = 3

PickSeeds:
A and B: (3x6)-(4+6) = 8
…
A and I: (8x8)-(4+2) = **58** … maximum death area
…

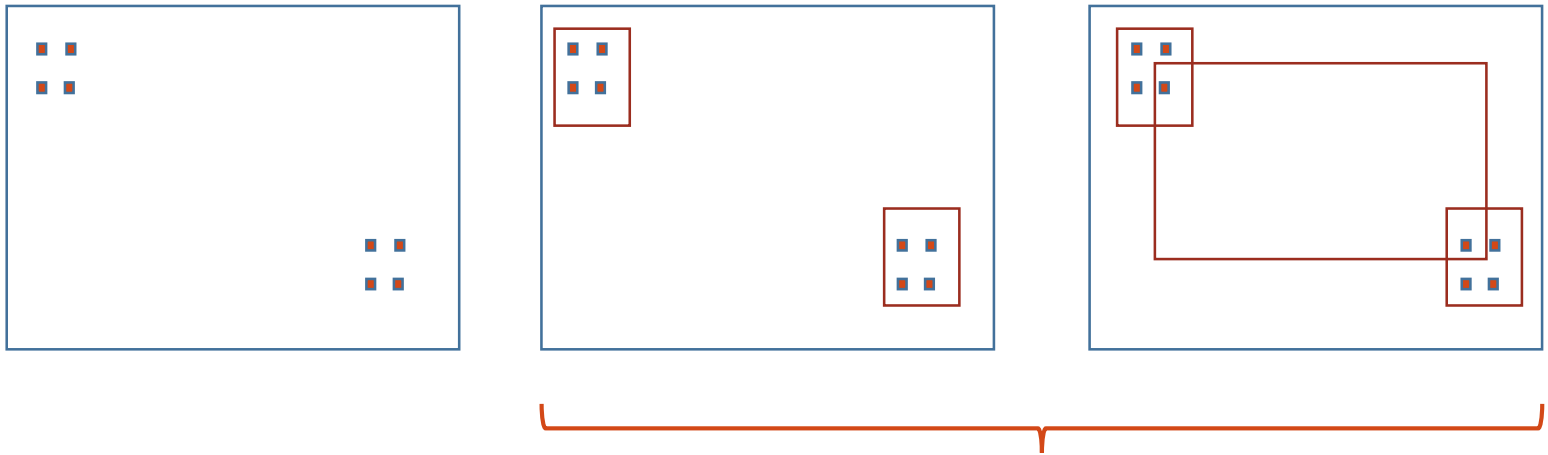
PickNext:

	A	**I**	**Δ**	
B	18-4 = 14	35-2 = 33	19	
C	32	13	19	
D	13	14	1	
E	11	38	27	
F	6	38	32	
G	45	4	**41**	… maximum difference
H	12	12	0	

Merge G and I and re-compute values for this new group (IG)
We get groups: AFBECH and IGD

Note: If m = 4, we get AFBEC and IGHD (and there will be a big overlap)

THEORETICAL PROBLEMS WITH R-TREES

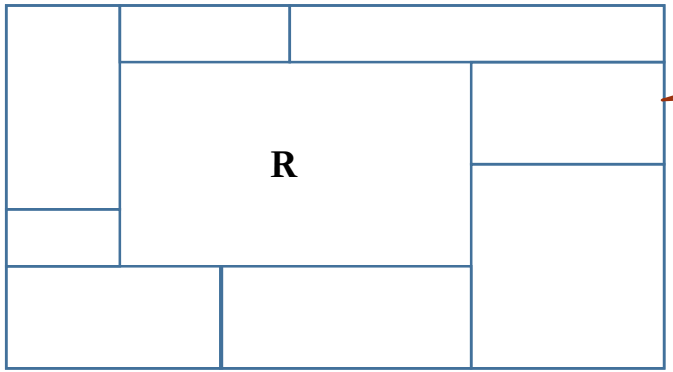


The insert procedure can lead to both arrangements.

THEORETICAL PROBLEMS WITH R-TREES

For any finite set S of disjoint regions in plane, there **does not** always exist such a set of MBRs where:

- Every region resides in exactly one MBR
- Every MBR bounds n regions where $1<n<m$
- The intersection of all the MBRs is empty



Cannot be split into two non-overlaping groups