PRINCIPLES OF DATA ORGANISATION

R+Tree & R*Tree

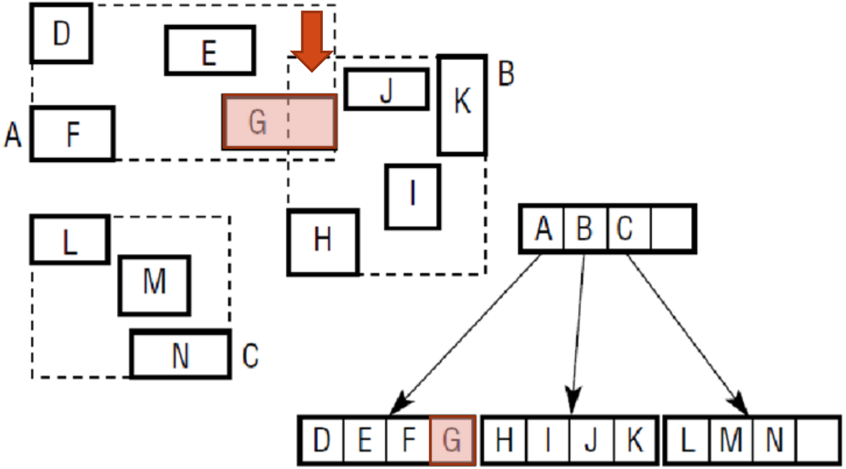
MOTIVATION

- How to search effectively in more than one dimension?
- B-tree for multidimensional data ~ R-tree
  - Theoretical problems with R-trees

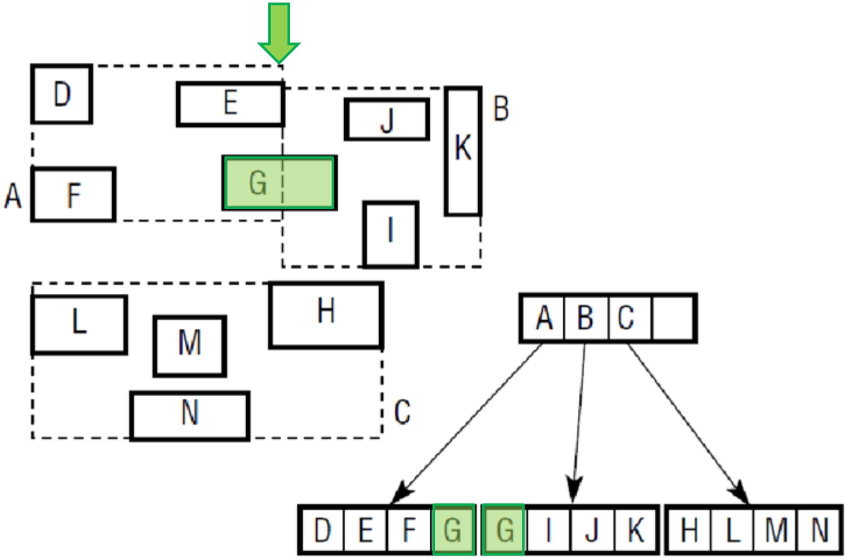
R+TREE

- Sellis et al 1987
- MBRs of R+-tree have zero overlap while allowing underfilled nodes and duplication of MBRs in the nodes
  - No minimum number of entries
- Achieved by splitting an object and placing it into multiple leaves if necessary
- Takes into account not only coverage (total area of a covering rectangle) but also overlap (area existing in one or more rectangles)
- Fewer paths are explored when searching, only one for point query
  - But insert requires cutting etc.

R+TREE : EXAMPLE



Branching factor M=4



Left R-Tree, right R+Tree, We can see that G is in two nodes.

R*TREE

- Beckmann et al. 1990
- R*-tree tries to minimize coverage (area) and overlap by adding another criterion - margin
- Utilisation ~ 70%
- Modification of insert procedure

INSERT: CHOOSELEAF

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

N ← T;
WHILE N ≠ leaf DO
    IF following level contains leaves THEN
        choose F from N minimizing overlap (**F ∪ E**) and solve ties by picking F whose F.I needs minimal extension or having minimal area;
    ELSE { **no change** }
        choose F from N where F.I needs **minimal extension** to I' while E.I ⊂ F.I' and area(F.I') is minimal;
    N:=F.p;
L:= N;

SPLITTING IN R*TREE

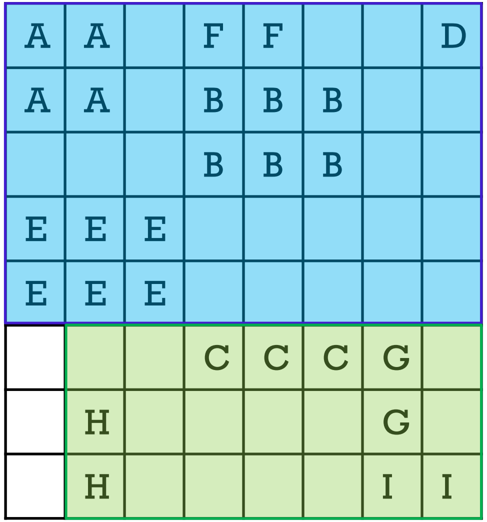
- Exhaustive algorithm where entries are sorted based on available axes.
- For each axis, $\boldsymbol{M}-\boldsymbol{2m}+\boldsymbol{2}$ distributions of $\boldsymbol{M}+\boldsymbol{1}$ entries into 2 groups are determined.
- For each distribution following so-called goodness values are computed (G_i denotes i-th group)
  - area : $area(MBR(G_1)) + area(MBR(G_2))$
  - margin : $margin(MBR(G_1)) + margin(MBR(G_2))$
  - overlap : $area(MBR(G_1) \cap MBR(G_2))$

INSERT : SPLITNODE

Split_RS(P,PP,E)
ChooseSplitAxis();
Distribute();

ChooseSplitAxis
FOREACH axis DO
    Sort the entries along given axis;
    S ← sum of all **margin-values** of all different distributions;
Choose the axis with the minimum S as split axis;

Distribute
Along the split axis, choose the distribution with minimum **overlap-value**. Resolve ties by choosing the distribution with minimum **area-value**;



M = 8, m = 3

ChooseSplitAxis:

X: AEH, FBCGID 22 + 26 … margin
AEHF, BCGID 26 + 26
AEHFB, CGID 28 + 26
AEHFBC, GID 28 + 20
sum = 202

Y: DFA, BECGHI 20 + 30
DFAB, ECGHI 22 + 26
DFABE, CGHI 26 + 22
DFABEC, GHI 28 + 24
sum = **188** … pick the minimum = split axis

Distribute:

DFA, BECGHI 8 … overlap
DFAB, ECGHI 0
DFABE, CGHI 0
DFABEC, GHI 7

DFAB, ECGHI 64 … area
DFABE, CGHI 61

FORCED REINSERT

- When inserting into rectangles created long in the past, it can happen that these rectangles cannot guarantee good retrieval performance in the current situation
- Standard split causes only local reorganization of the rectangles
- To achieve dynamic reorganizations R*-tree forces entries to be reinserted during the insertion routine

Insert : SplitNode

OverflowTreatment
IF the level is not the root level AND this is the first call of **OverflowTreatment** within this Insert THEN
    Reinsert();
ELSE
    Split();

Reinsert
FOREACH M + 1 entries of a node N DO
    Compute the distance between the centers of their rectangles and the center of the bounding rectangle of N;
Sort the entries in decreasing order of their distances;
P := first **p** entries from N; { **p** is a parameter which can differ for leaf and non-leaf node }
FOREACH E ∈ P DO remove E from N; { Includes shrink of the bounding rectangle }
FOREACH E ∈ P DO Insert(E);