PRINCIPLES OF DATA ORGANISATION

Multidimensional Indexing : Introduction

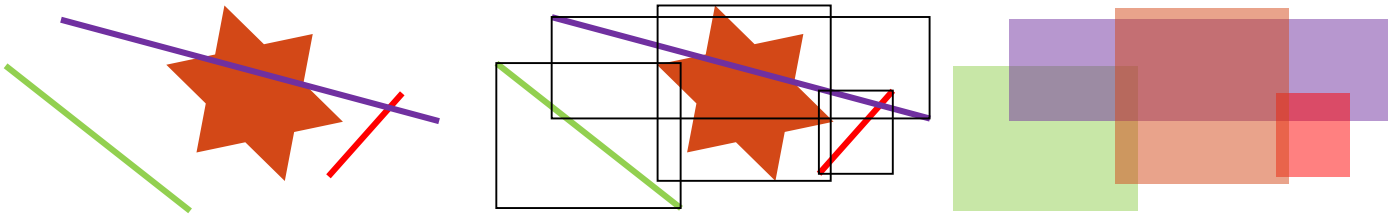
MOTIVATION

- How to search effectively in more than one dimension?
  - We know 1D
- How to represent spatial object in the database?
- Single Dimension-Based Indexing has an issue with locality

- Multidimensional indexes focus on storing spatial objects in such a way that objects close to each other in the space are also close in the structure and on the disk, i.e., maintain locality
  - We will now focus on basics not the secondary memory as before

OBJECTS APPROXIMATIONS

- General spatial objects are more complex than simple points
- To easily represent a possibly complex spatial object, we use approximation expressed by (Minimum) Bounding Rectangle/cube/box/object (MBR)
  - 2D: rectangle
- Comparison of objects is reduced to the comparison of their MBRs
  - Pros and cons

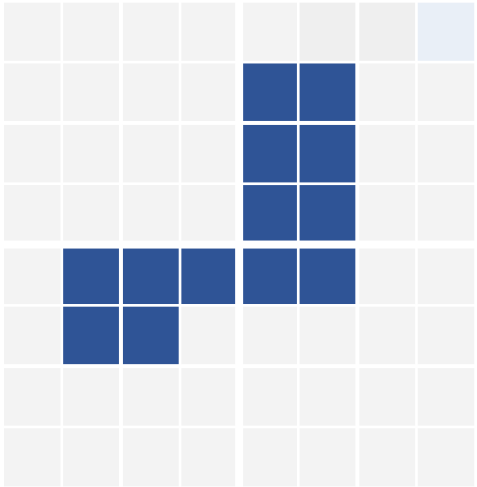


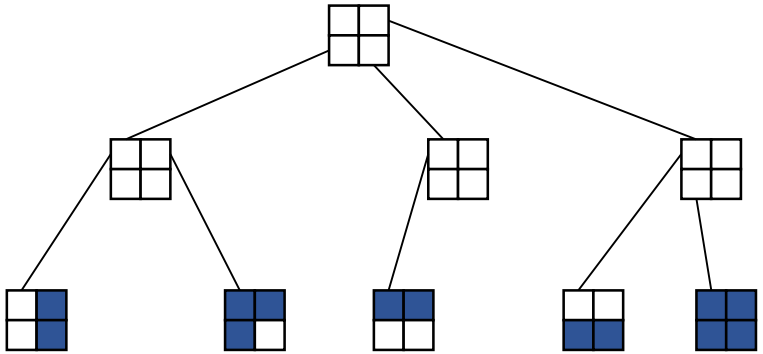
GRID-BASED INDEXING

- ***N***-dimensional grid covers the space and is not dependent on the data distribution in any way
  - The grid is formed in advance
  - We anchor the data
- Every point object can be addressed by the grid address
- Objects distribution in the grid does not have to be uniform → retrieval times for different grid cells may differ substantially for different parts of the space

QUAD-TREE

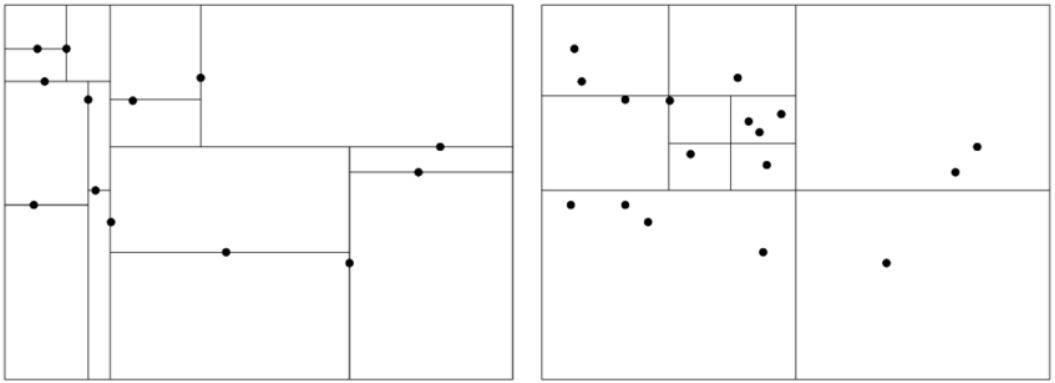
- Finkel, Bentley; 1974
- Tree structure representing recursive splitting of a space into quadrants
  - Quad = quadrant (4 regions)
- Each node has from zero to four children
- Typically the regions are squares
  - Any arbitrary shape is possible



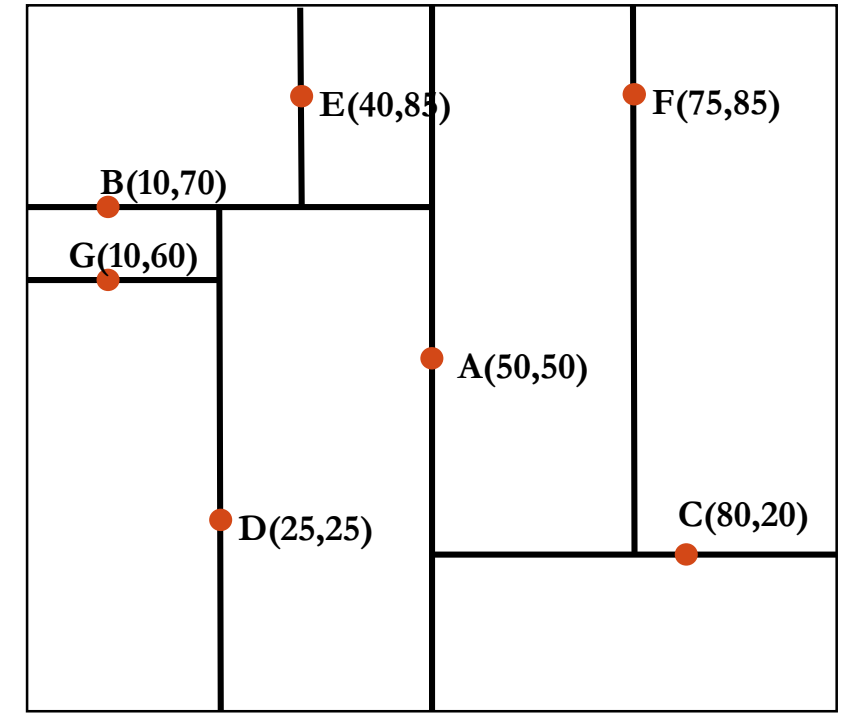


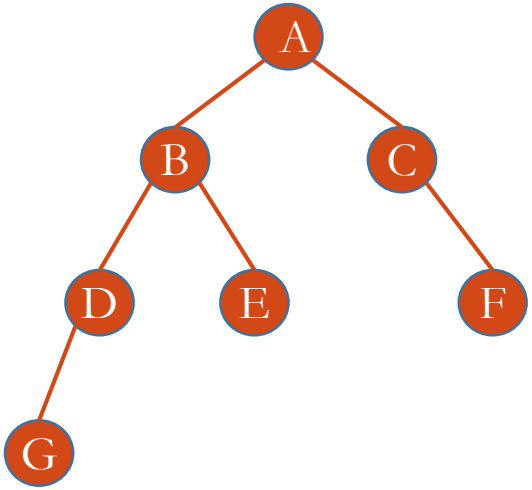
K-D-TREE

- Bentley; 1975
- Problem: quad tree can be unbalanced
- Objects in ***k***-dimensional space
- Binary search tree where inner nodes consist of a point, an axis identification (hyperplane in ***d***-dimensions), and two pointers
- Inner nodes correspond to hyperplanes splitting space into two parts where the location of the hyper plane is defined by the point



K-D-TREE : EXAMPLE





Discriminator (axis)

x

y

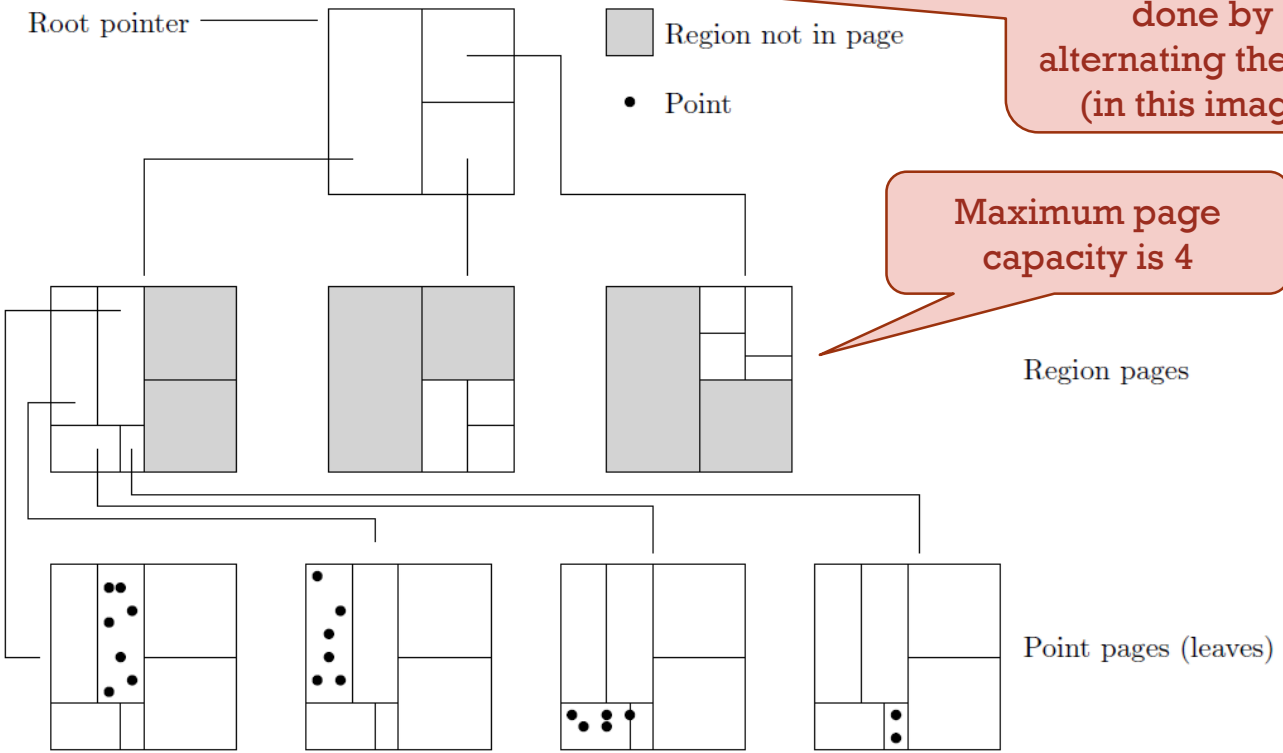
x

y

K-D-B-TREE

- Robinson; 1981
- Problem: k-d-tree is designed for main memory
  - What if it does not fit there?
- Combination of K-D-Tree and B-Tree
- Each tree node is stored as a page, but unlike B-trees 50% utilisation can not be guaranteed
- Each inner node contains multiple split axes to fill the node's capacity
- Leaf nodes contain indexed records (points)
  - Like in redundant B-trees
- Splitting and merging happens analogously to B-trees

K-D-B-TREE : EXAMPLE



Axis selection is done by alternating the axis (in this image)

Maximum page capacity is 4