Spatial Indexing I

Point Access Methods

PAMs

- Point Access Methods
  - Multidimensional Hashing: Grid File
    - Exponential growth of the directory
  - Hierarchical methods: kd-tree based
  - Space Filling Curves: Z-ordering
The problem
- Given a point set and a rectangular query, find the points enclosed in the query
- We allow insertions/deletions on line

Tree-based PAMs
- Most of tb-PAMs are based on kd-tree
- kd-tree is a main memory binary tree for indexing k-dimensional points
  - Needs to be adapted for the disk model
- Levels rotate among the dimensions, partitioning the space based on a value for that dimension
- kd-tree is not necessarily balanced
**kd-tree**

- At each level we use a different dimension

![Kd-tree diagram]

**Kd-tree properties**

- Height of the tree $O(\log_2 n)$
- Search time for exact match: $O(\log_2 n)$
- Search time for range query: $O(n^{1/2} + k)$
kd-tree example

External memory kd-trees

- Similar to B-tree, tree nodes split many ways instead of two ways
  - insertion becomes quite complex and expensive.
  - No storage utilization guarantee since when a higher level node splits, the split has to be propagated all the way to leaf level resulting in many empty blocks.
- Pack many interior nodes (forming a subtree) into a block.
  - it may not be feasible to group nodes at lower level into a block productively.
  - Many interesting papers on how to optimally pack nodes into blocks recently published.
LSD-tree

- Local Split Decision – tree
- Use kd-tree to partition the space. Each partition contains up to B points. The kd-tree is stored in main-memory.
- If the kd-tree (directory) is large, we store a sub-tree on disk
- Goal: the structure must remain balanced: external balancing property

Example: LSD-tree
LSD-tree: main points

- Split strategies:
  - Data dependent
  - Distribution dependent
- Paging algorithm
- Two types of splits: bucket splits and internal node splits