Application areas

- geographic information systems (GIS)
  - area, line and point entities
  - huge databases ($10^4$ to $10^9$ objects)

- image analysis, recognition

- computer graphics, games
  - 2D & 3D algorithm speedup (ray casting, collisions)

- industry, CAD
  - VLSI design, component positioning, collisions
Elementary tasks I

- **point localization** in 2D net
  - looking for an area object which contains the point

- **nearest N points** from the given center
  - global variation: looking for the closest point pair

- **curve intersections** (polylines)
  - collisions in a set of curves (polylines)

- **point objects closest to the given curve** (route)
Elementary tasks II

- **interval queries** in 2D space (databases)
- **set operations** on map entities
  - areas, line objects, points
- **collision tests (interferences)**, minimal distances among planar objects (VLSI)
- **object processing in some geometric order**
  - increasing distance from a given center
“Region quadtree”

- representation of areal regions in 2D
- splitting exactly in the middle
- information is stored in leaf nodes only
“Region quadtree”
Pyramid

- representation of areal regions in 2D
- additional summary info in inner nodes
- pyramid nodes up to level: E
Pyramid

- pyramid nodes up to level: D
Pyramid

- pyramid nodes up to level: C
Pyramid

- pyramid nodes up to level: B
“PR quadtree” (Point-Region)

- **representation of point objects**
- **splitting exactly in the middle**
- **object information stored in leaf nodes (all levels, max. one object per node)**
“PR quadtree” (Orenstein)
Bucket methods

- node can store information about **more than** one object
  
  \[ 0 < \text{object}\# \leq \text{Max} \]

  - **Max** is constant value tuned for storage efficiency (record size..)

- memory & time efficiency
  
  - less overhead in pointer management

- **AB-tree** analogy
“Bucket PR quadtree”

- representation of point objects
- splitting exactly in the middle
- object information stored in leaf nodes (all levels, ≤ Max object per node)
"Bucket PR quadtree"

Max = 2
"K-D tree" (Bentley)

- representation of point objects
- adaptive splitting - one coordinate at a time (binary tree).
  regular alternation of the coordinates
- object information stored in all nodes
“K-D tree”
“Adaptive K-D tree” (Friedman)

- representation of point objects
- adaptive splitting - one coordinate at a time (binary tree).
  Exact object coordinates not used
- object information stored in leafs only

\[ \begin{align*}
  x_1 & \quad y_1 \\
  x_2 & \quad y_2 \\
  x_3 & \quad y_3 \\
  x_4 & \quad y_4 \\
  x_5 & \quad y_5 \\
  x_6 & \quad y_6 \\
\end{align*} \]
“Adaptive K-D tree”
“Range trees”

- Efficient implementation of range queries
  - \( \langle x_{\min}, x_{\max} \rangle \times \langle y_{\min}, y_{\max} \rangle \) in 2D
  - Complexity: \( O(\log_2 N + F) \) in 1D, \( O(\log_2^2 N + F) \) in 2D

- “1D range tree”
  - Balanced binary search tree, leaves are connected by a double-linked list

- “2D range tree”
  - Balanced binary search tree for the \( x \) coordinate
  - Every inner node contains “1D range tree” (\( y \) coord.) for all points in the relevant region
“2D range tree”

- balanced binary tree for x-coordinate
- object information in leaves only
- inner nodes: 1D range trees for y-coordinate
“2D range tree”

Query: \((3.5, A.5) \times (6, 4)\)
“R-tree” (Guttman)

- objects can be areal
- bounding boxes in inner nodes (the whole subtree must fit into the bounding box)
- object references in leaf nodes
“R-tree”
“Strip tree” (Ballard)

- planar curve (polyline)

- adaptive splitting induced by the curvature

- oriented bounding rectangle defined by: starting point \( (P_1) \), endpoint \( (P_2) \), widths \( (W_L, W_R) \)
Intersection of two curves

- "strip trees" were built for both curves
  - trivial cases
  - subdivision is necessary
Bounding system hierarchies

- "Sphere tree" (Palmer, Grimsdale, 1995)
  - simple tests & transform, approximation not so good

- "AABB tree" (Held, Klosowski, Mitchell, 1995)
  - simple test, more complicated transform

- "OBB tree" (Gottschalk, Lin, Manocha, 1996)
  - simple transform, more complicated test, good approximation

- "K-dop tree" (Klosowski, Held, Mitchell, 1998)
  - complicated transform & test, excellent approximation
Directional pass

- data pass using specific **directional order**
  - visibility (front-to-back or vice versa) in orthographic projection
  - ”plane sweep” pass (“sweeping”)

- pass from the **center point**
  - visibility (form-factors, ..) in perspective projection
  - kNN search (“k-Nearest Neighbors”)

Presumption

- **hierarchical** space decomposition
  - inclusion conditions on objects only, not needed for bounding boxes (e.g. "strip tree")

- effective computation of **minimal distance of each cell / box** ... \( d(B) \)
  - distance from sweep plane or center point of the pass
  - distance need not be an infimum (but has to be an lower bound)
Algorithm

- auxiliary data structure $H$ (heap)
  - efficient operations: $\text{Min}$, $\text{DeleteMin}$, $\text{Insert}$

1. put the root node into $H$
   - $d(B)$ is used for heap sorting

2. if $\text{Min}(H)$ is an object, process it and remove it

3. if $\text{Min}(H)$ is a hierarchy cell, remove it and put all its children back into the heap

4. repeat steps 2 and 3 until the heap $H$ is empty or until all required output objects are processed (kNN)
Example I

- \( \{R_0\} \rightarrow \{R_1, R_2\} \)
- \( \{R_1, R_2\} \rightarrow \{R_3, R_4, R_2\} \)
- \( \{R_3, R_4, R_2\} \rightarrow \{R_7, R_4, R_8, R_2\} \)
- \( \{R_7, R_4, R_8, R_2\} \rightarrow \{\text{Kladno}, R_4, \text{Kralupy}, R_8, R_2\} \)

1. Kladno
- \( \{R_4, \text{Kralupy}, R_8, R_2\} \rightarrow \{R_9, \text{Kralupy}, R_8, \text{Jílové}, R_2\} \)
Example II

- \{R_9, Kralupy, R_8, Jílové, R_2\} → \{Řevnice, Mníšek, Kralupy, Černošice, R_8, Jílové, R_2\}

\(\overset{2}{-}\overset{5}{\Rightarrow}\) Řevnice, Mníšek, Kralupy, Černošice

- \{R_8, Jílové, R_2\} → \{Roztoky, Praha, Jílové, R_2\}

\(\overset{6}{-}\overset{8}{\Rightarrow}\) Roztoky, Praha, Jílové
Example III

- \( \{R_2\} \rightarrow \{R_5, R_6\} \)
- \( \{R_5, R_6\} \rightarrow \{\text{Kostelec}, R_6, \text{Brandýs}\} \)

9 Kostelec

- \( \{R_6, \text{Brandýs}\} \rightarrow \{\text{Brandýs}, \text{Říčany}, \text{Úvaly}\} \)

10-12 Brandýs, Říčany, Úvaly
The End

More information: