Multidimensional Search Structures
Spatial Databases

• Spatial Objects
  – *Points*: location (x,y)
  – *Lines*: pairs of points (roads, coastal lines)
  – *Polygons*: list of points (states, countries)

• Data Types
  – *Point*: a data type with no extension (area)
  – *Region*: has location and boundary that defines the extension

• Spatial Queries
  – *Range queries*
    • “Find all Italian restaurants within 20 miles from UTA”
  – *Nearest neighbor queries*
    • “Find the 10 Italian restaurants that are nearest to UTA”
    • “Find the nearest fire station to Neddermann Hall”
  – *Spatial join queries*
    • “Find pairs of cities within 20 miles of each other”
    • “Find restaurants that are adjacent to university campuses”
Applications

• Geographical Information Systems (GIS)
  – Map systems
  – Resource management systems

• Computer-aided design and manufacturing (CAD/CAM)
  – VLSI design (avoiding overlaps, routing wires)

• Multimedia databases
  – Various dimensions (color, shape, texture)
Representation of Spatial Objects

- It is very expensive to work on real boundary lines
- *Minimum Bounding Rectangle* (MBR)

Testing for intersection:

- Test if MBRs intersect
- If they do, test if boundary lines intersect
Grid-Tree (G-Tree)

- Mainly for point data of any dimension
- Based on hypercubes
G-Tree Organization

- Rotate vertical/horizontal split
- When split, add 0/1 at the end

- G-Trees are organized in B⁺-trees
  - Key is the binary string
Searching G-Trees

• Search: find point \( P = (x_1, x_2, \ldots, x_n) \)
  – Let \( m \) be the number of bits of the largest bitstring
  – Find the \( m \)-bit region that contains \( P \):
    • Start with \( s_i = 0 \) and \( t_i = 1 \) for all \( 1 \leq i \leq n \)
    • For \( k = 0 \) to \( m-1 \):
      Slice over \( j = (k \mod n) + 1 \) dimension
      The \( k \)th bit of the bitstring is 0 iff \( x_j < (s_j + t_j)/2 \); then set \( t_j = (s_j + t_j)/2 \)
      Otherwise, it is 1 and set \( s_j = (s_j + t_j)/2 \)
  • For \( n = 2 \), \( m = 5 \), and \( P = (0, 1, 0.7) \):
    0.1 < ½     bit0 is 0
    0.7 > ½     bit1 is 1
    0.1 < ¼     bit2 is 0
    0.7 < ½+¼   bit3 is 0
    0.1 < 1/8   bit4 is 0
    Bitstring is 00010
  – Use the bitstring as the key for the G-Tree
Point Quadtrees

- Divide the space into four quadrants
- Represented as an unbalanced tree where each node has 4 children
  - Internal node: point
  - Leaf: empty
- Node: \((x, y, \text{NE}, \text{NW}, \text{SW}, \text{SE})\)
Spatial Operations on Quadtrees

- Search for a point \( P=(x,y) \)
  - if current node \( T \) in quadtree is leaf, then not found
  - if \( T=P \), then found
  - else find quadrant of \( T \) that includes \( P \) and search recursively:
    - e.g., if \( x>T.X \) and \( y>T.Y \) then search SE quadrant
- Given a rectangle \((x_1,y_1,x_2,y_2)\) find all points in the rectangle
  - Need to keep a set of points
  - The rectangle may intersect more than one quadrants
- Insert a point \( P=(x,y) \)
  - If current node is leaf, then replace leaf with \((x,y,nil,nil,nil,nil)\)
  - Else determine the quadrant and apply the algorithm recursively
R*-Trees

- It is like a B⁺-tree with key the MBR, but
  - There is no order in the rectangles in each node
  - Sibling rectangles may overlap

- Restrictions on R*-tree nodes:
  - The root has at least two children (unless it is the only node)
  - Every non-root node has \( m \) children where \( M/4 \leq m \leq M \)
  - The tree is balanced
Spatial Operations on R*-Trees

Find all objects that intersect rectangle 9:

Insert rectangle 9:
Insert a New Object Rectangle S

- At each node (starting from root) choose only one bounding rectangle R to insert the rectangle S so that the insertion of S into R has the least *overlap enlargement*.
  
  (Overlap enlargement of a rectangle is the total overlapping area of the rectangle with the other rectangles in the tree.)

- If there are many with the same overlap enlargement, choose the one with the smallest area.

- If overflow, split the node into two sets of nodes using a split axis so that there is no underflow and the perimeters of the two bounding boxes are minimized.