CSEG601 & CSE5601:
Spatial Data Management & Application

- Kd-trees -

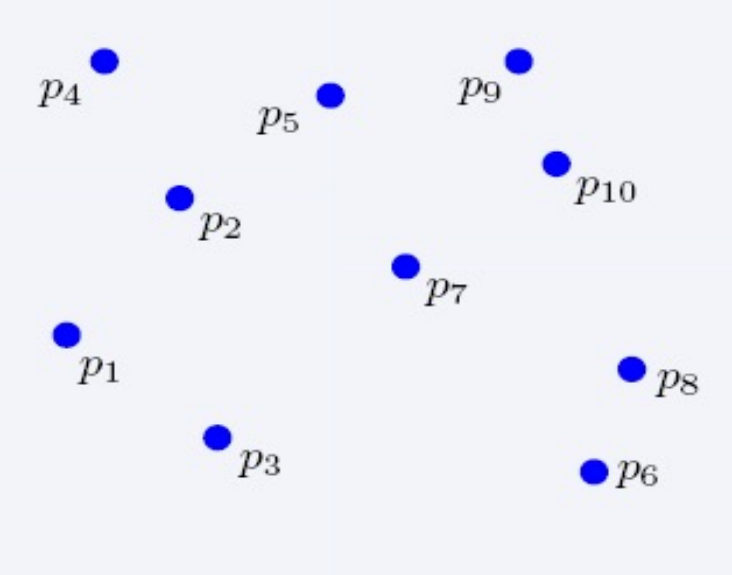
Sungwon Jung

Big Data Processing & DB Lab.
Dept. of Computer Science and Engineering
Sogang University
Seoul, Korea
Tel: +82-2-705-8930
Email : jungsung@sogang.ac.kr

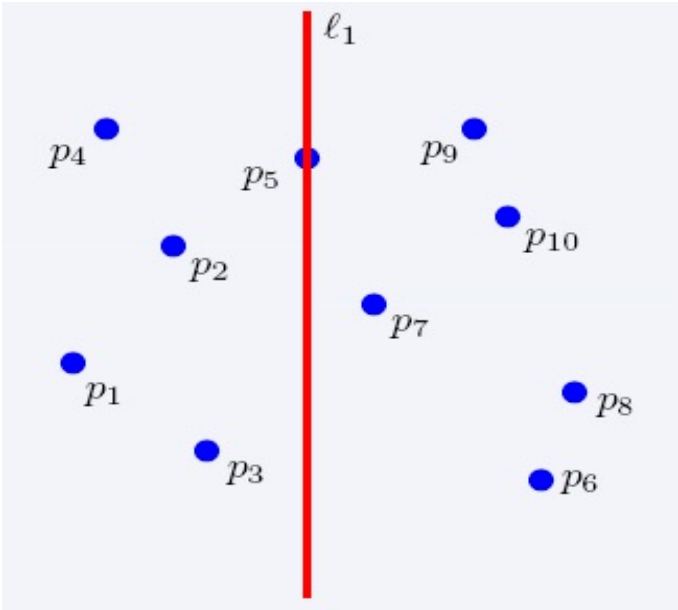
2-dimensional kd-trees

- Algorithm:
  - Choose **x** or **y** coordinate (alternate)
  - Choose the median of the coordinate; this defines a horizontal or vertical line
  - Recurse on both sides

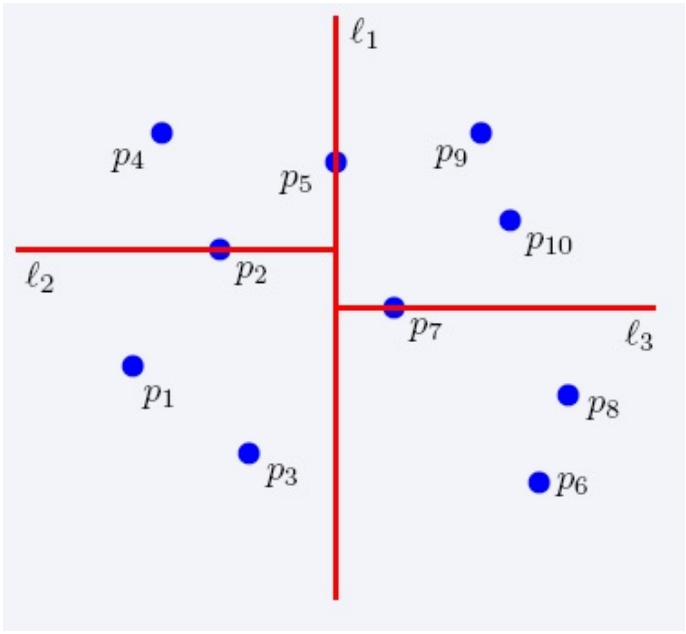
Construction of kd-trees



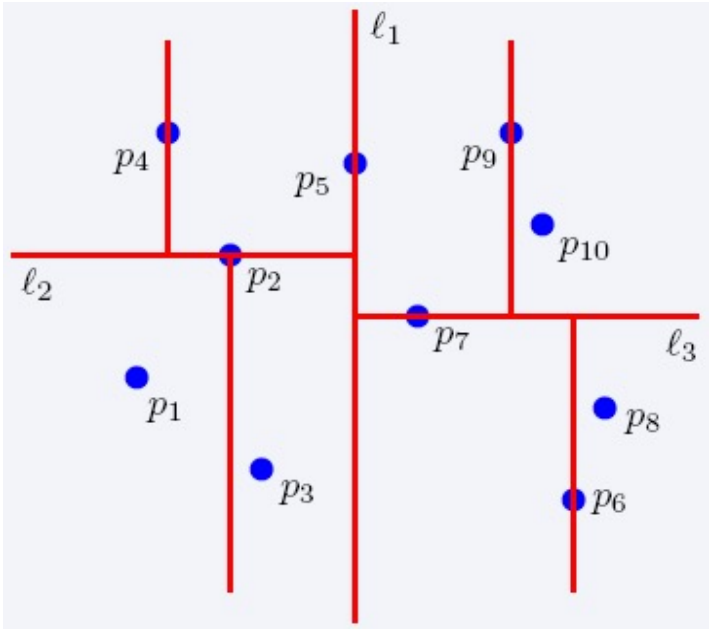
Construction of kd-trees



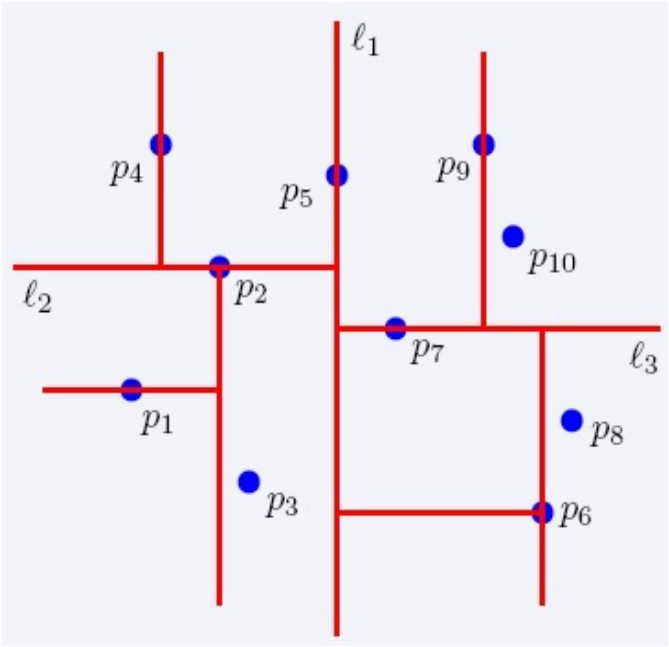
Construction of kd-trees



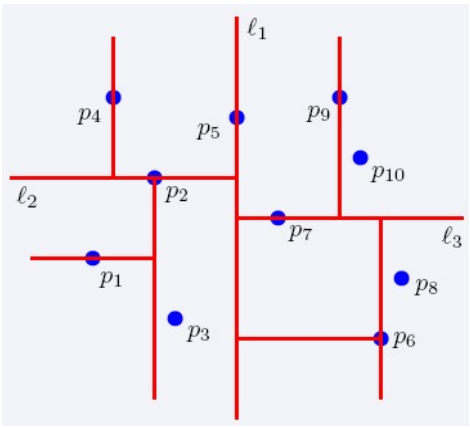
Construction of kd-trees

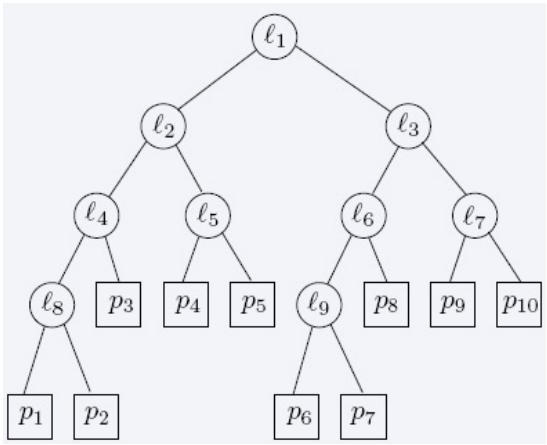


Construction of kd-trees

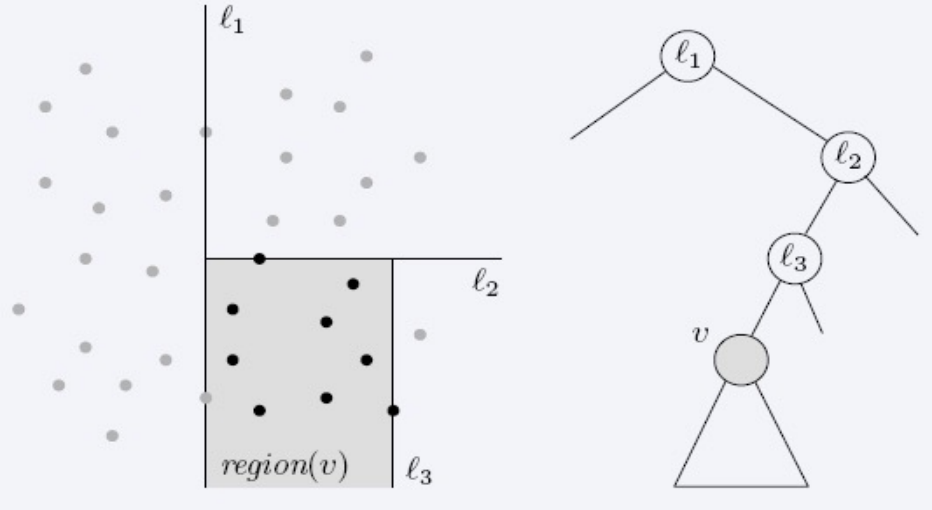


The complete kd-tree





Region of node **v**



Region(v) : the subtree rooted at **v** stores the points in black dots

Searching in kd-trees

- Range-searching in **2-d**
  - Given a set of **n** points, build a data structure that for any query rectangle **R** reports all point in **R**

kd-tree: range queries

- Recursive procedure starting from **v = root**
- **Search (v,R)**
  - If **v** is a leaf, then report the point stored in **v** if it lies in **R**
  - Otherwise, if **Reg(v)** is contained in **R**, report all points in the **subtree(v)**
  - Otherwise:
    - If **Reg(left(v))** intersects **R**, then **Search(left(v),R)**
    - If **Reg(right(v))** intersects **R**, then **Search(right(v),R)**

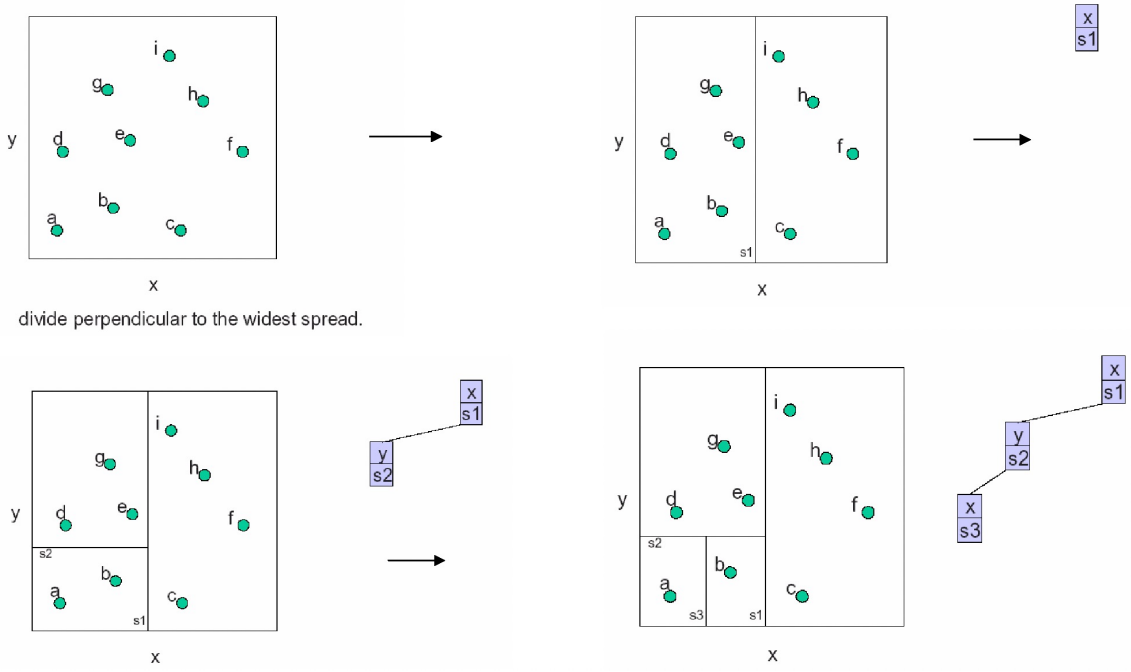
Construction of the **d**-dimensional kd-trees

- The construction algorithm is similar as in **2-d**
- At the root we split the set of points into two subsets of same size by a hyperplane vertical to x_1-axis
- At the children of the root, the partition is based on the second coordinate: x_2-coordinate
- At depth **d**, we start all over again by partitioning on the first coordinate
- The recursion stops until there is only one point left, which is stored as a leaf

K-d tree construction method 2

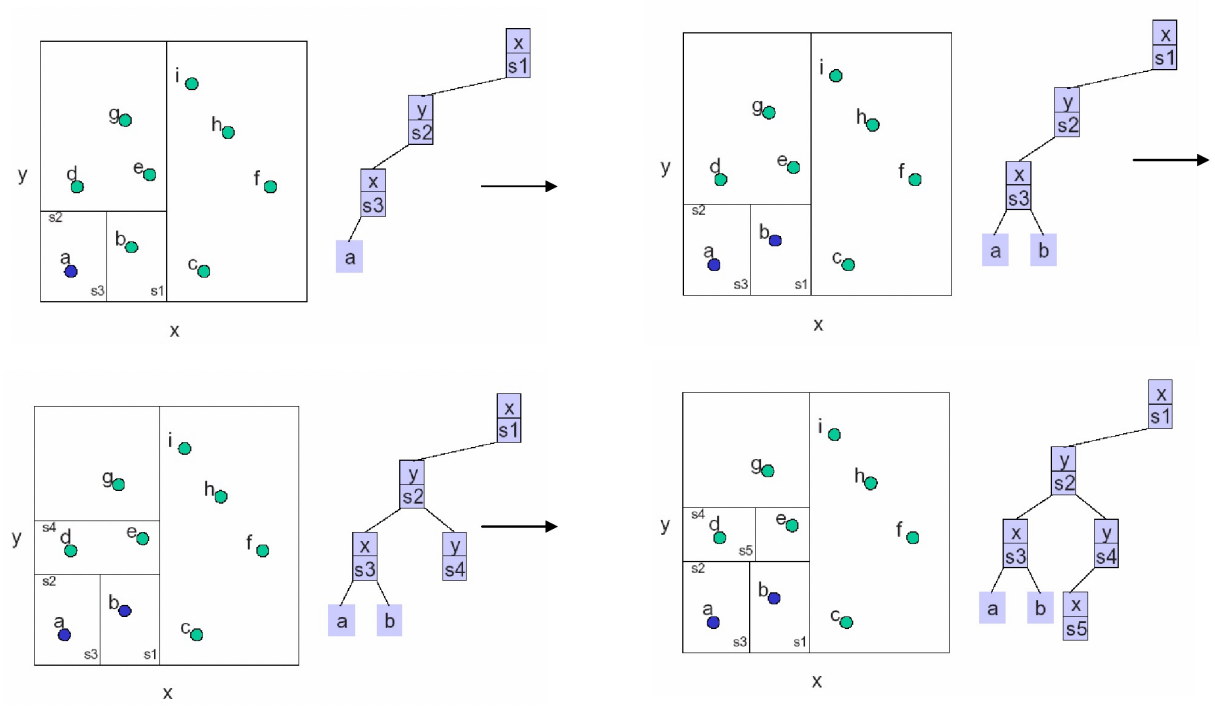
- If there is just one point, form a leaf with that point.
- Otherwise, divide the points in half by a line perpendicular to one of the axes.
- Recursively construct k-d trees for the two sets of points.
- Division strategies:
  - divide points perpendicular to the axis with widest spread.
  - divide in a round-robin fashion.

k-d tree construction example

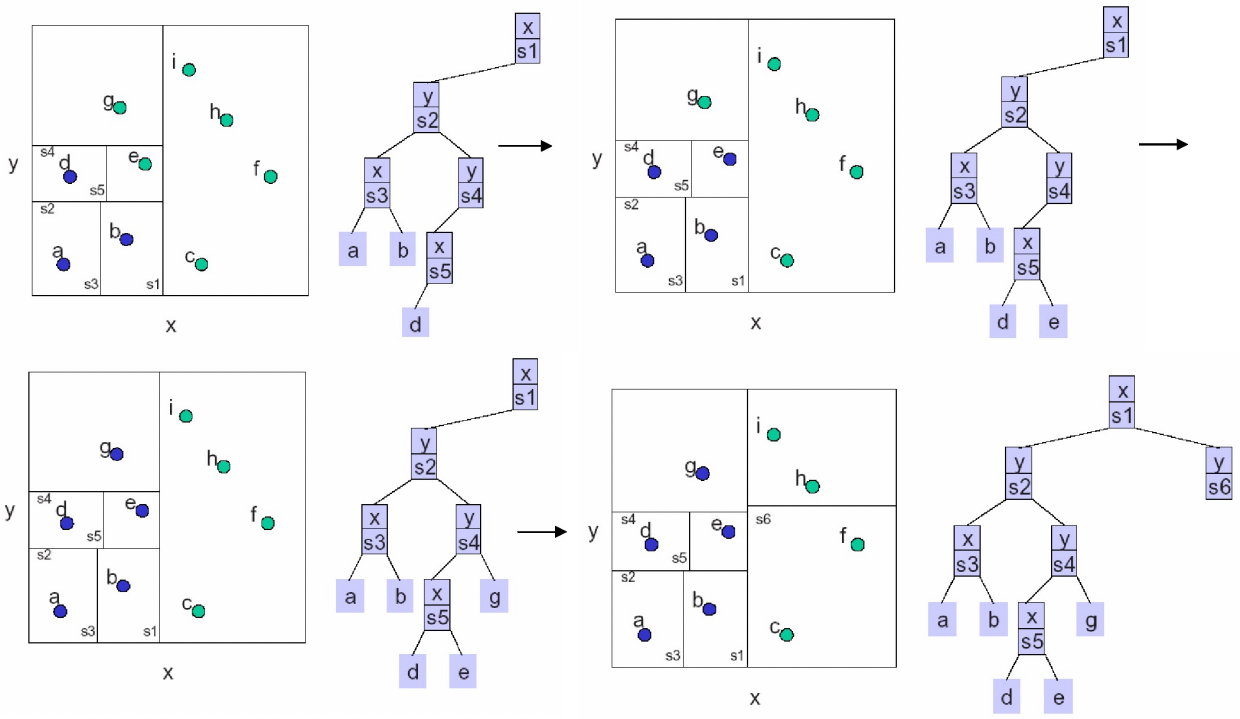


divide perpendicular to the widest spread.

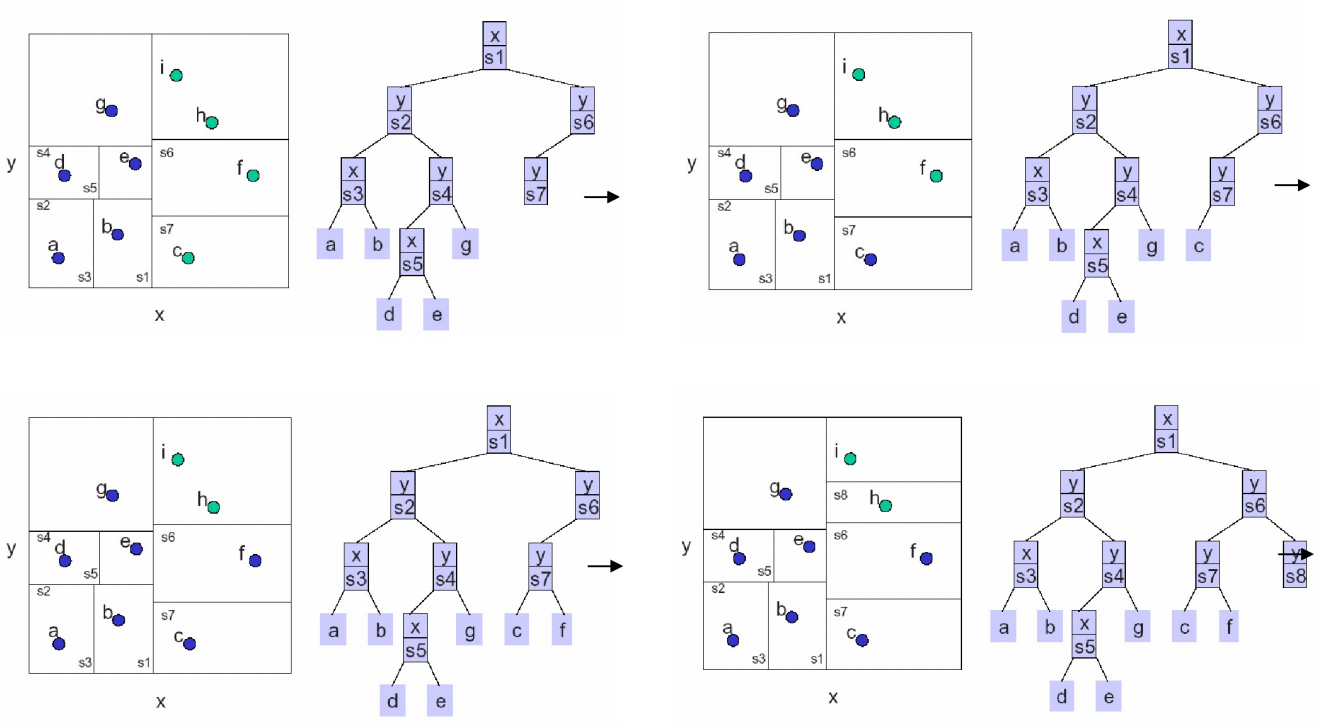
k-d tree construction example



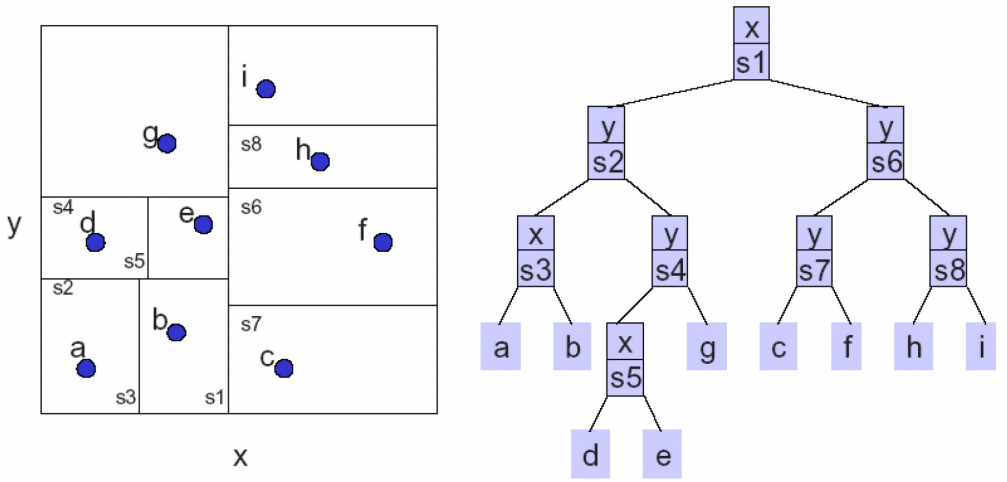
k-d tree construction example



k-d tree construction example



k-d tree construction example



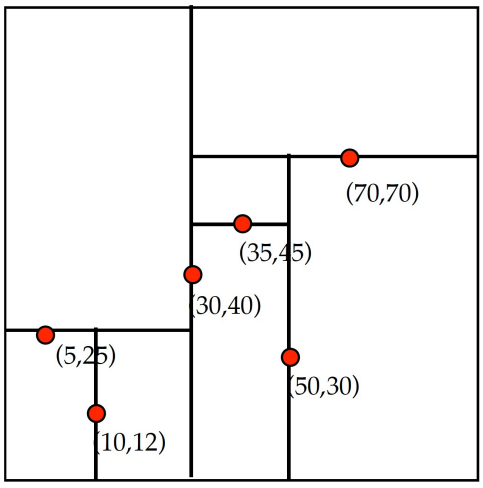
k-d tree Construction Complexity

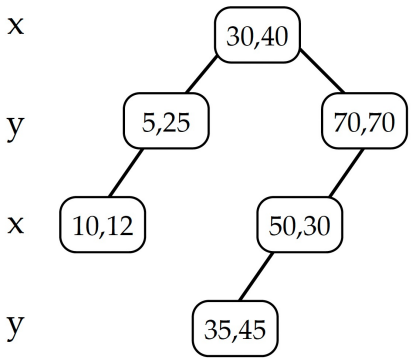
- First sort the points in each dimension:
  - O(*dn* log *n*) time and *dn* storage.
  - These are stored in A[1..*d*,1..*n*]
- Finding the widest spread and equally dividing into two subsets can be done in O(*dn*) time.
- Constructing the k-d tree can be done in O(*dn* log *n*) and *dn* storage

K-d tree construction method 3

kd-tree example

insert: (30,40), (5,25), (10,12), (70,70), (50,30), (35,45)





kd-tree example

insert: (30,40), (5,25), (10,12), (70,70), (50,30), (35,45)