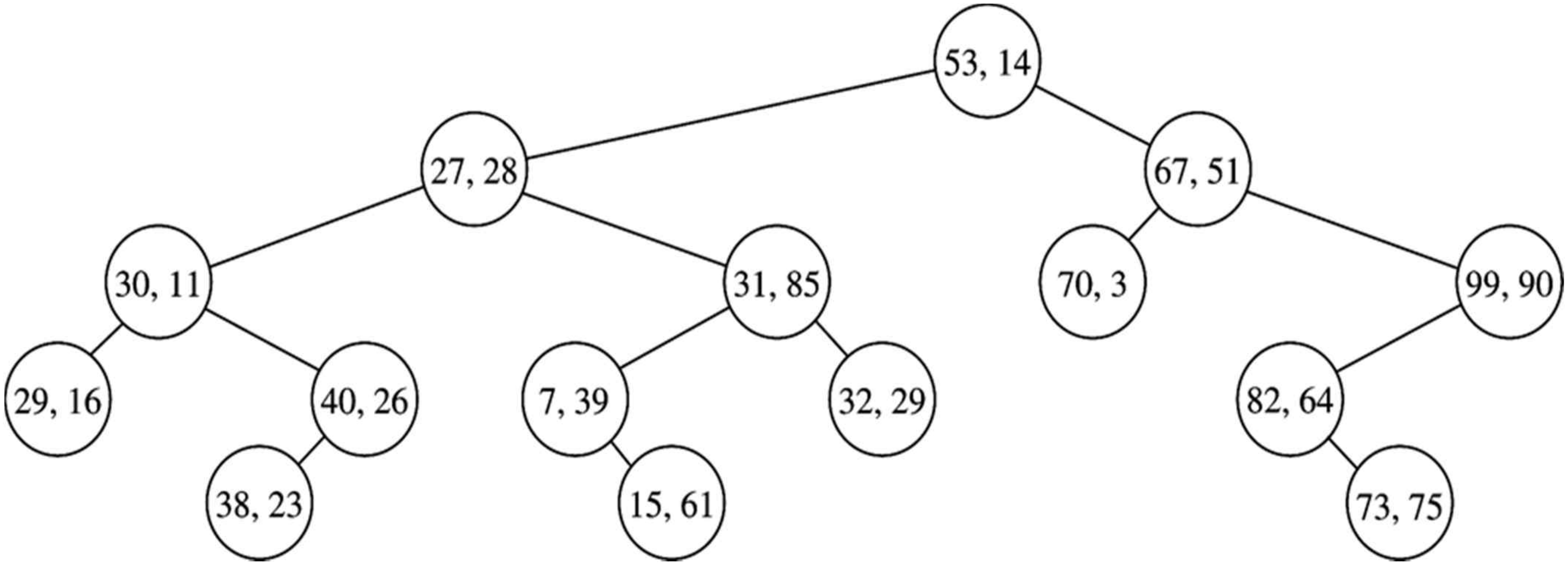
KD Tree

- Multiple dimensional data
  - Targeting: **Range queries** in databases of multiple keys:
    - Find all persons that meet

      35 ≤ *age* ≤ 40 and $50000 ≤ *yearly income* ≤ $60000
    - geographic information system

  - Extending BST from one dimensional to k-dimensional
    - It is a binary tree
    - Organized by levels (root is at level 0, its children level 1, etc.)
    - Tree branching at level 0 according to the first key, at level 1 according to the second key, etc.

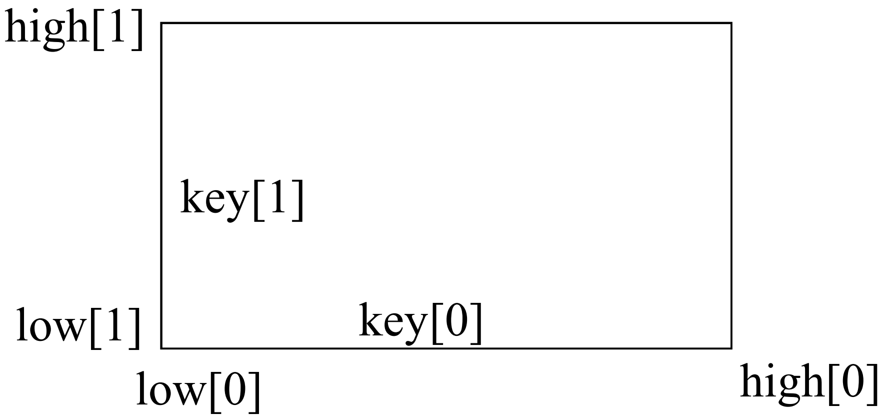
2D Tree


53, 14
27, 28
67, 51
30, 11
31, 85
70, 3
99, 90
29, 16
40, 26
7, 39
32, 29
82, 64
38, 23
15, 61
73, 75

Insert, Range search

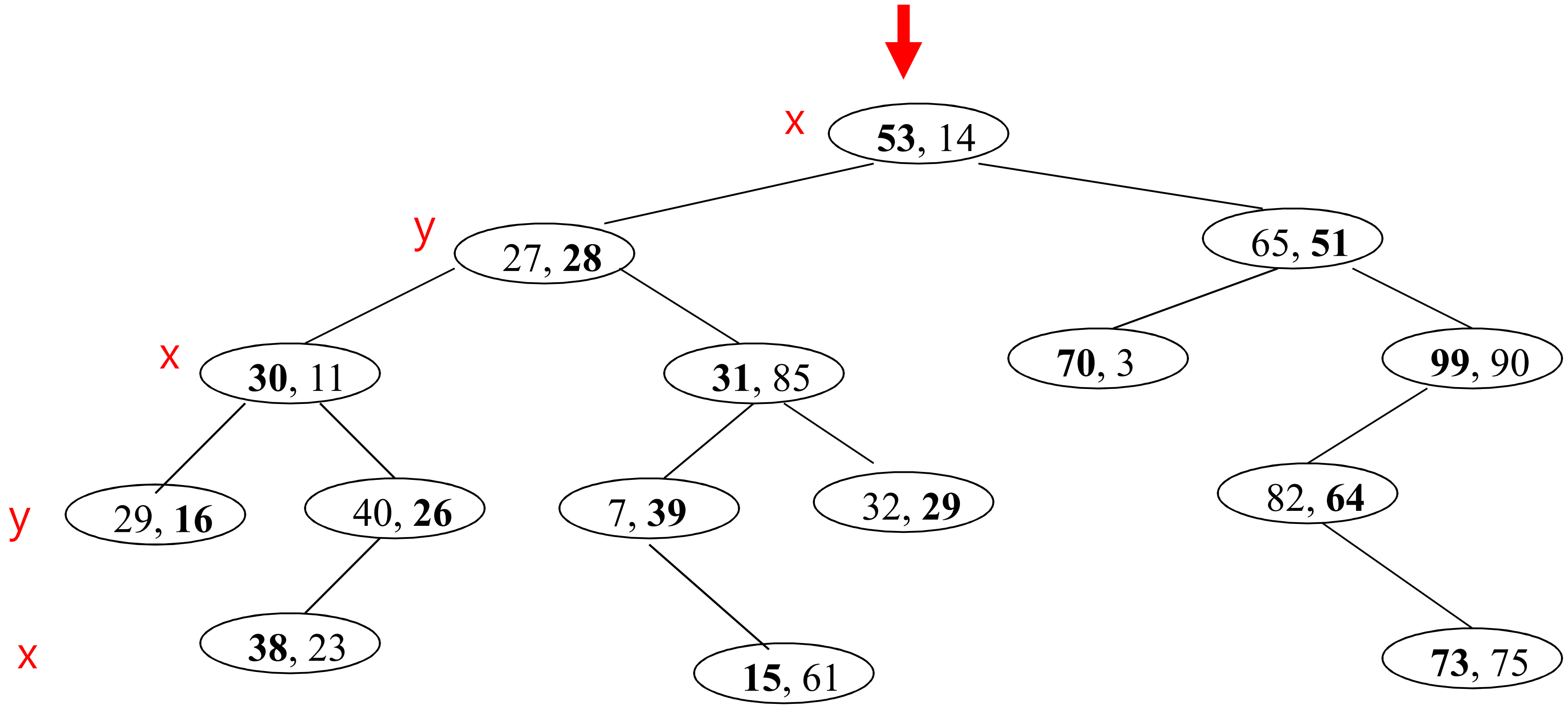
- Insert
  - A 2D item (vector of size 2 for the two keys) is inserted
  - New node is inserted as a leaf
  - Different keys are compared at different levels
- Range search

key vectors in rectangular range:



Range Search

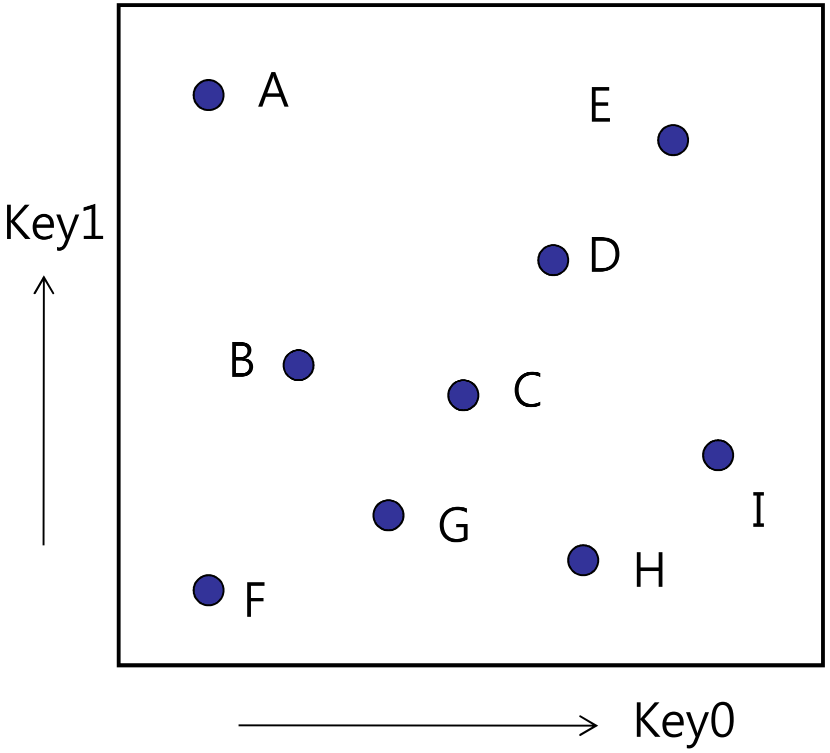
[35, 40] x [23, 30]



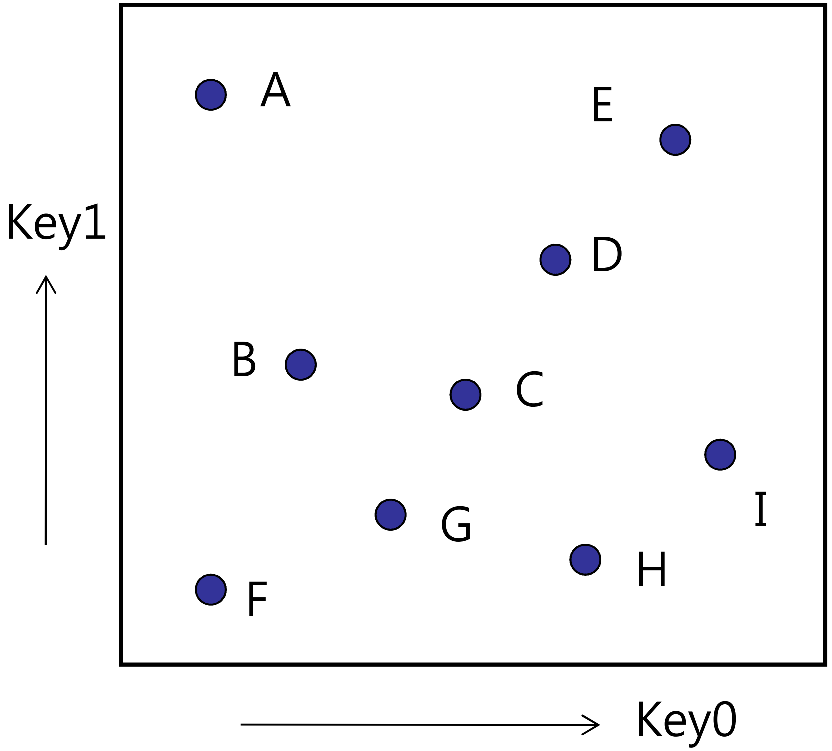
low[0] = 35, high[0] = 40;

low[1] = 23, high[1] = 30;

KD Tree Construction



Details



1. Sort points in each dimension

Key0 :									
Key1 :									

2. Split the points in half by the middle point in the selected dimension

3. Build the sorted lists for the other dimensions

KD Tree Performance

- Insert
  - Average and balanced trees: ___________
  - Worst case: ________
- Search with a square range query
  - for m matches.
    - Perfectly balanced tree:

      KD trees: O(m + ______)

      2D trees: O(m + _____)
- Construction
  - Balanced trees : ___________

Range query performance

- Range query in a perfectly balanced 2D tree:
  - Consider one boundary of the square (say, low[0])
  - Let T(n) be the number of nodes to be looked at with respect to low[0].

More …

- Remove
  - No good remove algorithm beyond lazy deletion
    (mark the node as removed)
- Balancing KD Tree
  - No known strategy to guarantee a balanced 2D tree
  - Periodic re-balance