Multi-Dimensional Data Sets

- Current data structures such as search trees can work only with 1-dimensional data.
- A big inherent problem with multi-dimensional data is that they are not comparable.
- In a 2-d setting, which is bigger? (10, 15) or (22, 8)?
- So need new data structures that can impose an order on multi-dimensional data.

Request control

The Case of 1-Dimension

- Identifying the relevant points in a 1-dimensional setting is possible.
- Think of a search tree T, that can locate all the values between x and y (both inclusive).
- Can be done in time O(k+log n) where k is the number of such elements.
- How?

17:15 Request control

AS
Advanced Data Structures ...

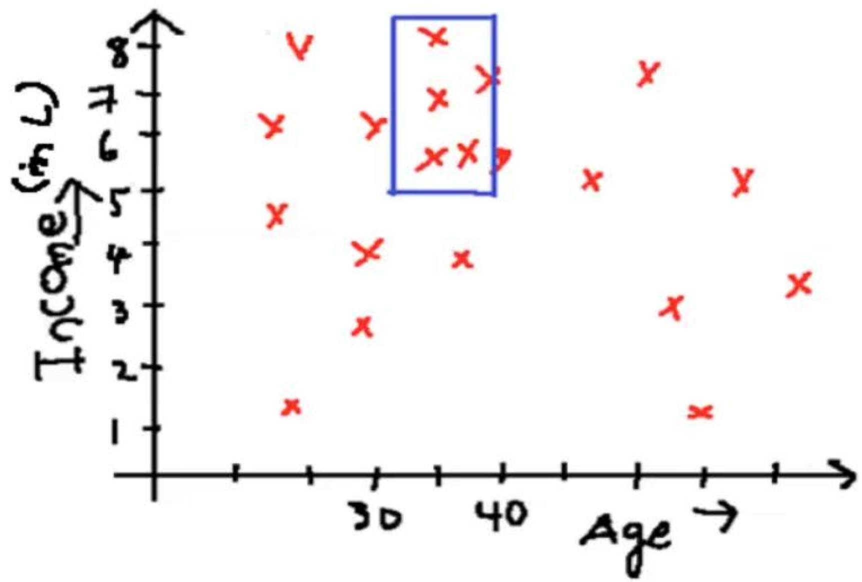
The Case of 1-Dimension

- Identifying the relevant points in a 1-dimensional setting is possible.
- Think of a search tree T, that can locate all the values between x and y (both inclusive).
- Can be done in time O(k+log n) where k is the number of such elements.
  - Need not filter the n points which takes O(n) time.
- How?

- Need a similar technique and a data structure for higher dimensional data sets.
- A solution that is faster than O(n).

Three Solutions

- We will study **three** different data structures in this context.
- We will also seek to solve a standard query:
- Given a rectangular region q = [a,b]x[c,d], find all the points of P that are inside q.
- This is called as a range query.



Some Solution Ideas

- In each case, suppose there are n points in a d-dimensional space.
- Can consider all these points and arrive at the result.
- Typically, however, the region of interest, or the query region, has far fewer points. Can find the result on these subset of points.
- But, identifying these points itself may take time.

A New Way for Data Structures

- Preprocess the points and create a suitable data structure.
- This data structure can then be used repeatedly for answering any query.
- Some parameters of efficiency:
  - Space used by the data structure
  - Time taken to build the data structure
  - Time to answer a query
- Query time is lower bounded by the size of the output. This is denoted k.
- So, we seek query time of O(k+polylog(n)) so that the result is practically fast.

23:18 Request control

A New Way for Data Structures

- Seemingly easy with one dimensional data.
- Build a balanced binary search tree of the n points.
- Exercise: Given two values x and y with $x < y$, can find the values in the input set that are between x and y.
- Can do this in time $O(k + \log n)$.
- Is it possible to do similar things in two and further dimensions?

25:18 Request control

AS

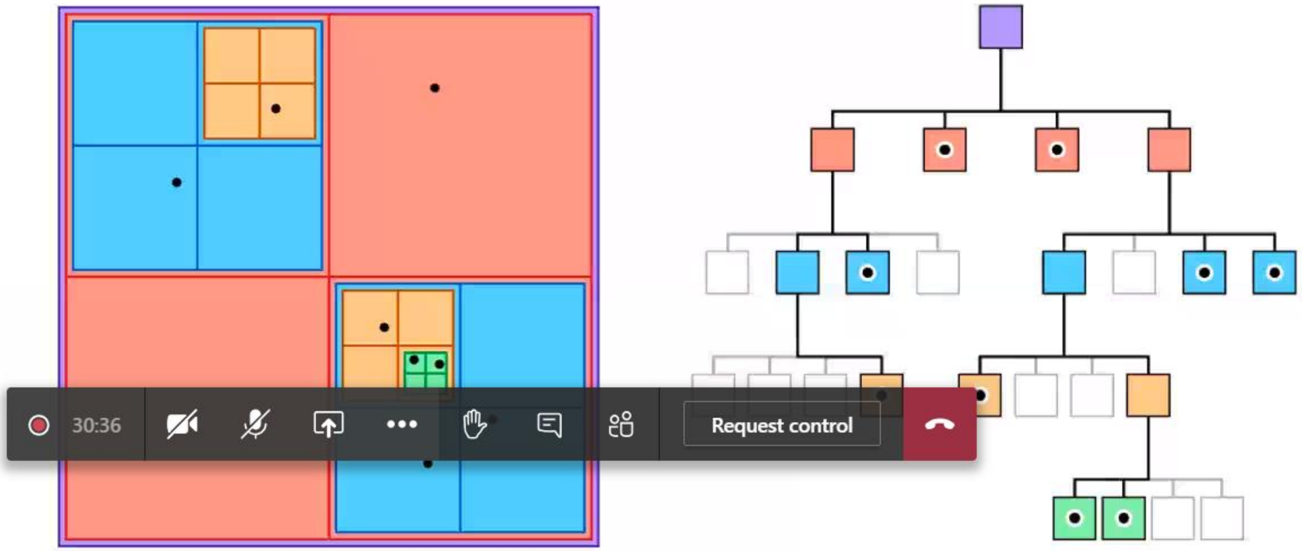
First Solution – A Quad Tree

- We will first generalize a binary search tree to be useful for 2-d points.
- A node now has four children, labeled NE, NW, SE, and SW.
- At a node u, the points in the subtree rooted at the NE child have x- and y- coordinates larger than that of at u.

First Solution – A Quad Tree

- We will first generalize a binary search tree to be useful for 2-d points.
- A node now has four children, labeled NE, NW, SE, and SW.
- At a node u, the points in the subtree rooted at the NE child have x- and y- coordinates larger than that of at u.

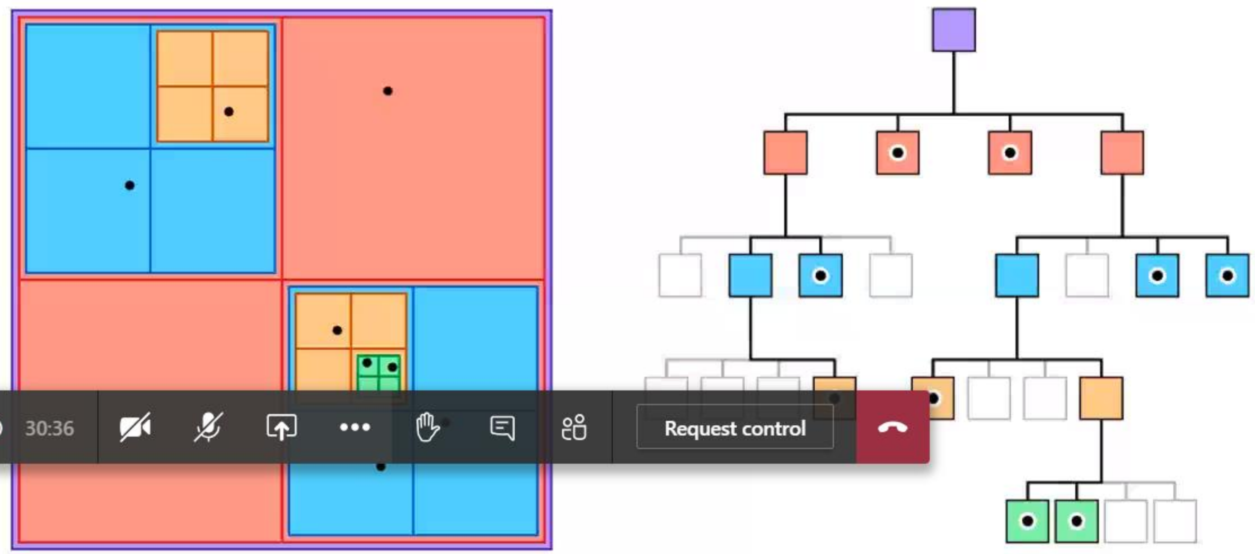
NW	NE
SW	SE



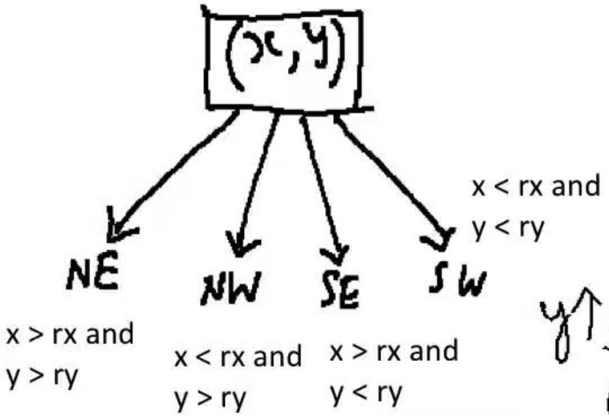
First Solution – A Quad Tree

- We will first generalize a binary search tree to be useful for 2-d points.
- A node now has four children, labeled NE, NW, SE, and SW.
- At a node u, the points in the subtree rooted at the NE child have x- and y- coordinates larger than that of at u.

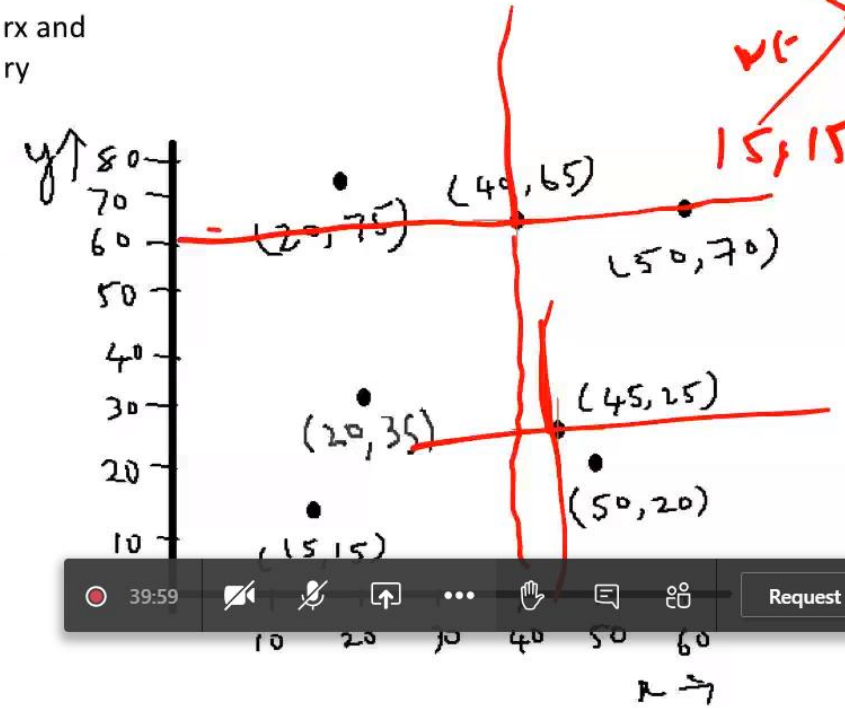
NW	NE
SW	SE



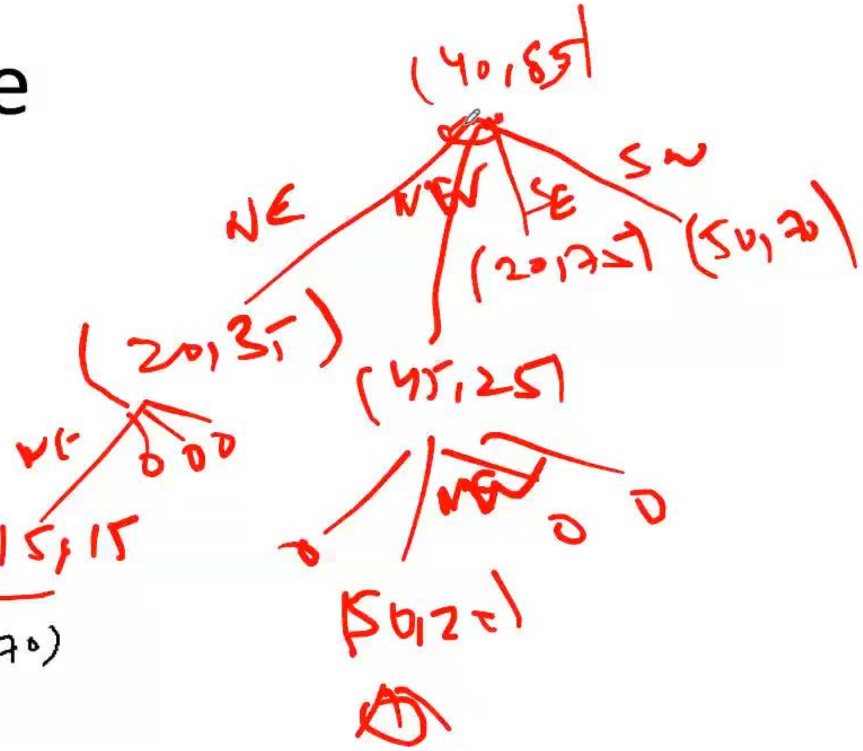
Quad Tree



(x,y)
NE
NW
SE
SW
x > rx and
y > ry
x < rx and
y > ry
x > rx and
y < ry
x < rx and
y < ry



(20,75)
(40,65)
(50,70)
(20,35)
(45,25)
(50,20)
(15,15)



(40,65)
NE
NW
SE
SW
(20,35)
(45,25)
(50,20)

Quad Tree

- Similar routines to that of a binary search tree.
- To Insert(p = (x,y)) into a quad tree Q, do the following:
  - Start from the root of the tree.
  - Let the point at the root be r = (rx, ry).
  - Four cases:
    - If x > rx and y > ry – Insert p in the NE child.
    - If x > rx and y < ry – Insert p in the SE child
    - If x < rx and y > ry – Insert p in the NW child.
    - If x < rx and y < ry – Insert p in the SW child.

Quad Tree

- Similar routines to that of a binary search tree.
- A Delete(p) routine can also be designed akin to the Delete routine of a binary search tree.

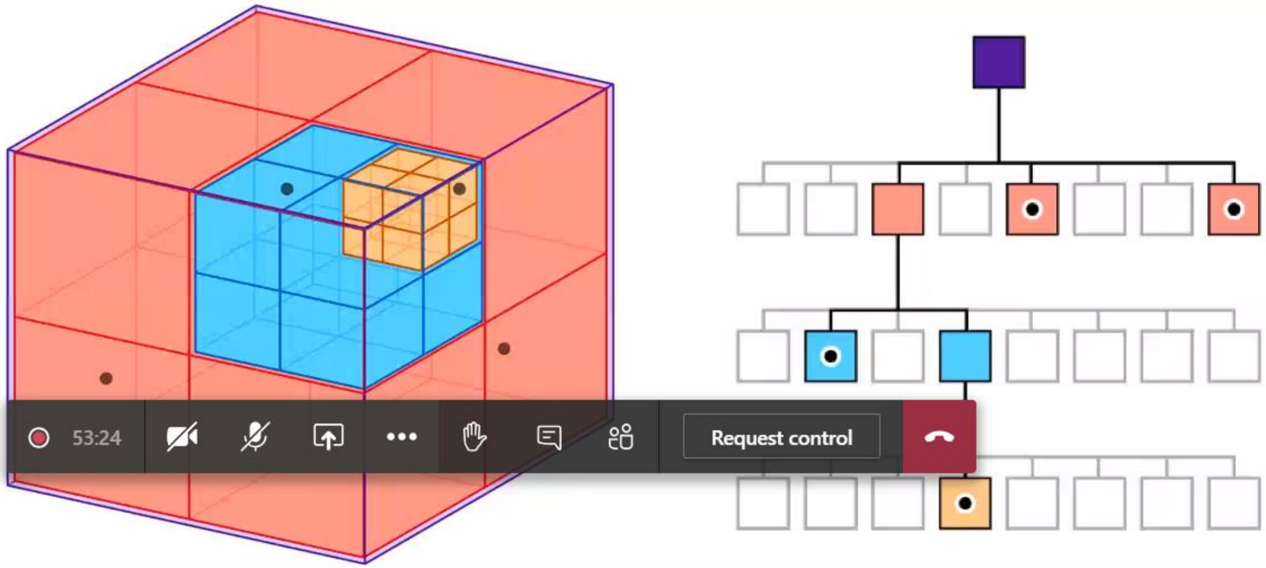
- A Find(p) routine is also similar. Start from the root, and depending on the x and y coordinates, search one of the four subtrees.

45:21 Request control

Quad Tree

- But there are several disadvantages:
- Height balance is difficult to achieve on insertions and deletions.
- Number of children grows exponentially with the number of dimensions.

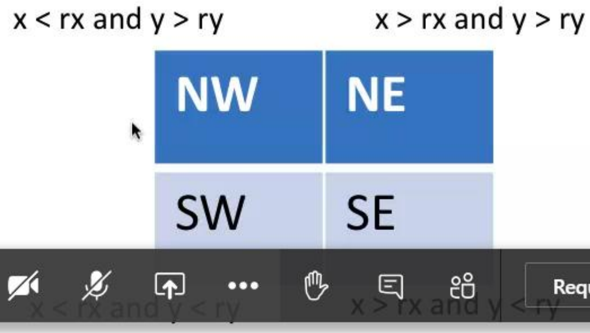
Octree
for 3 dimensions



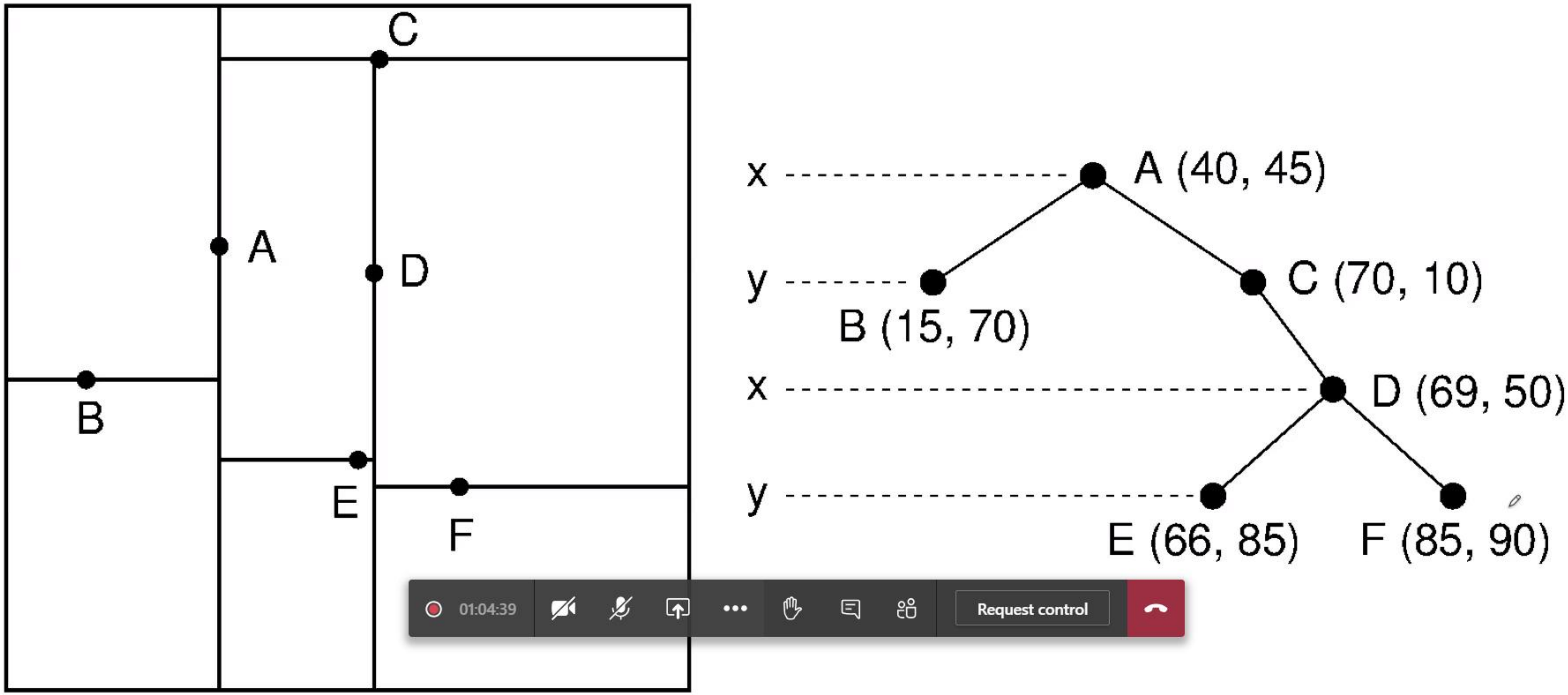
Quad Tree

- Build a quad tree for the following set of 2-dimensional points. (Insert points into an initially empty quad tree in the same order)

- (15, 35), (20, 40), (25, 25), (10, 20), (50,10), (30, 30), (40, 20), and (45, 55)



A Better Solution – A 2-d Tree



A Better Solution – A 2-d Tree

- Inserting into the 2-d tree is straight forward.
- Proceed from the root of the tree, at each level comparing the appropriate coordinate value.
- For higher dimensions, cycle over all the d dimensions over d levels.

Unmute

01:10:51 Request control

AS

Advanced Data Structures ...

A Better Solution – A 2-d Tree

- Insert the following points into an initially empty 2-d tree.

- (15, 35), (20, 40), (25, 25), (10, 20), (50,10), (30, 30), (40, 20), and (45, 55)

01:11:25 Request control