

CS61B Lecture #20

- **Administrative:**

- Test Review Session this Sunday 4-6 in 306 Soda.
- Sample tests online.

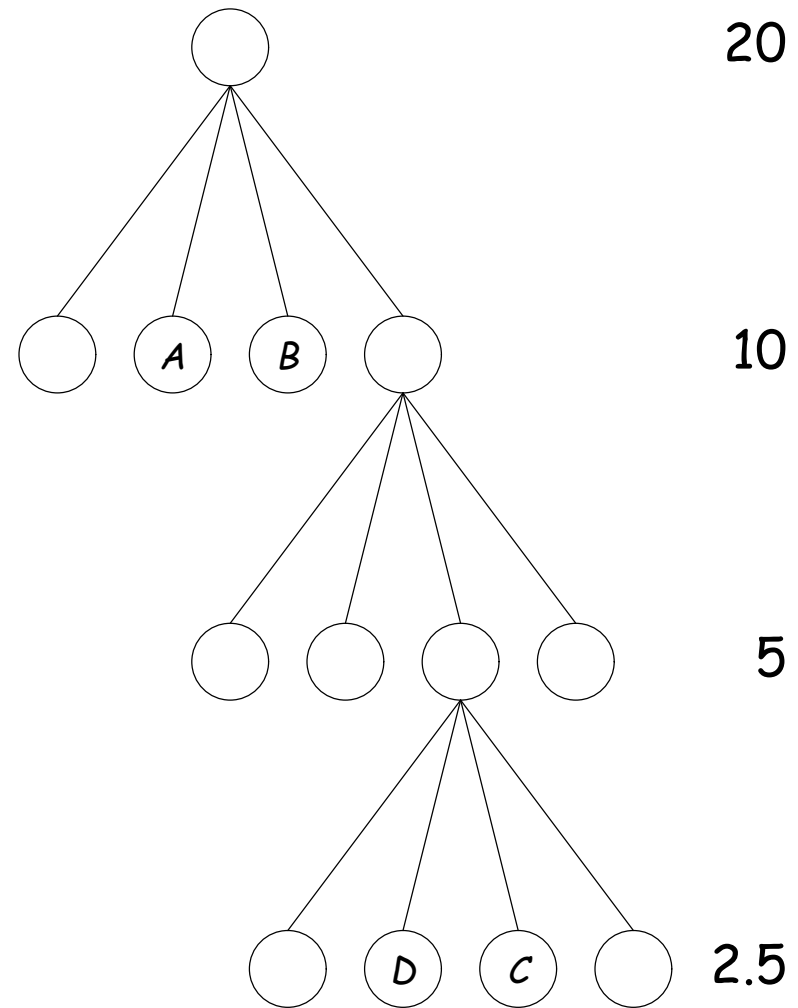
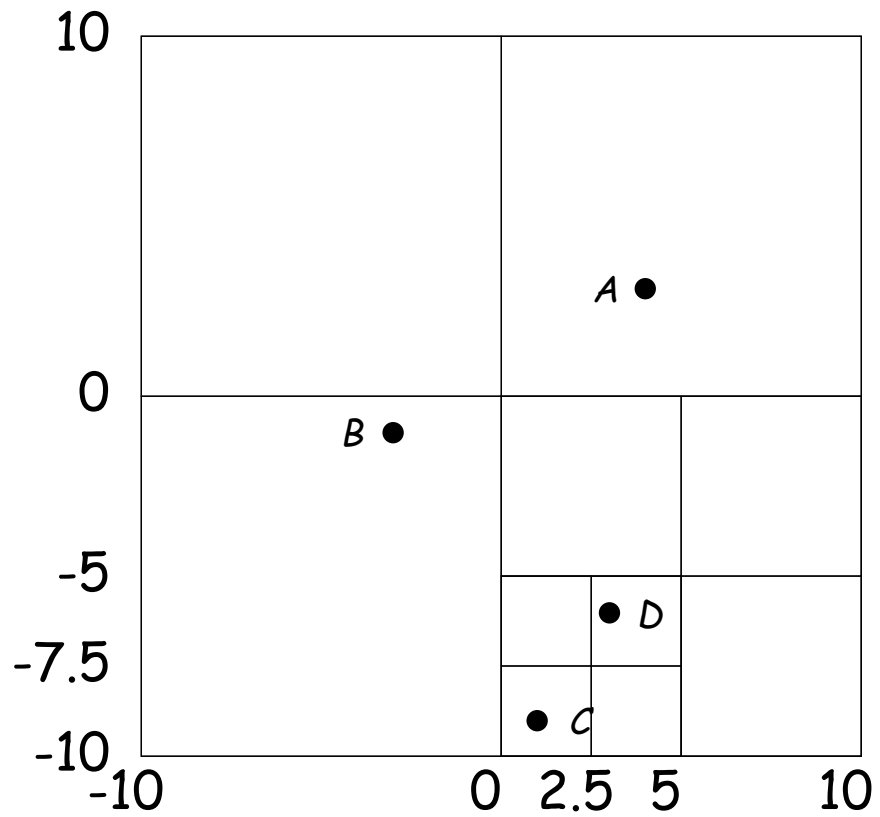
- Today: Projects 1 and 2.

- **Readings for Next Topic:** *DS(IJ)*, Chapter 5; Goodrich and Tamassia, Chapter 6.

A Leap Ahead: Quad Trees

- Want to *index* information about locations so that items can be retrieved by position.
- Quadtrees do so using standard data-structuring trick: *Divide and Conquer*.
- Idea: divide (2D) space into four *quadrants*, and store items in the appropriate quadrant. Repeat this recursively with each quadrant that contains more than one item.
- So, a quadtree consists of a bounding rectangle, B and either
 - Nothing (an empty quadtree), or
 - An item that lies in that rectangle, or
 - Four quadtrees whose bounding rectangles are the four quadrants of B .

Representation



Navigating a Quadtree

- To find an item at (x, y) in quadtree T ,
 - If (x, y) is outside the bounding rectangle of T , or T is empty, then (x, y) is not in T .
 - Otherwise, if T contains a single item, compare it to (x, y) .
 - Otherwise, T consists of four quadtrees. Recursively look for (x, y) in the appropriate one.
- Similar procedure works when looking for all items within some rectangle, R :
 - If R does not intersect the bounding rectangle of T , or T is empty, then there are no items in R .
 - Otherwise, if T contains a single item, return it if it is in R , and otherwise nothing.
 - Otherwise, T consists of four quadtrees. Recursively look for points in R in each of them.