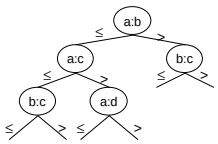
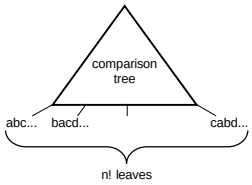
Lower Bound on Sorting

- When sorting n items, how many different answers are possible?
  - n!
- How is an answer (typically) determined?
  - Use comparisons
- To sort a,b,c,...
  - Note: these represent numbers in the original input order



Comparison Trees

- Every sorting algorithm has a corresponding *comparison tree*
  - Note that other stuff happens during the sorting algorithm, we just aren't showing it in the tree
- The comparison tree must have n! (or more) leaves because a valid sorting algorithm must be able to get any of n! possible answers
- Comparison tree for sorting n items:



Time vs. Height

- The worst-case time for a sorting method must be ≥ the height of its comparison tree
  - The height corresponds to the worst-case number of comparisons
  - Each comparison takes Θ(1) time
  - The algorithm is doing more than just comparisons
- What is the minimum possible height for a binary tree with n! leaves?

  Height ≥ log(n!) = Θ(n log n)
- This implies that <u>any</u> comparison-based sorting algorithm <u>must</u> have a worst-case time of Ω(n log n)
  - Note: this is a lower bound; thus, the use of big-Omega instead of big-O

Using the Lower Bound on Sorting

<u>Claim</u>: I have a PQ
- Insert time: O(1)
- GetMax time: O(1)

- True or false?

False (for general sets) because if such a PQ existed, it could be used to sort in time O(n)

True for items in a small range (Note: such a set can be sorted in O(n) time)

<u>Claim</u>: I have a PQ
- Insert time: O(loglog n)
- GetMax time: O(loglog n)

- True or false?

False (for general sets) because it could be used to sort in time O(n loglog n)

True for items with priorities in range 1..n [van Emde Boas] (Note: such a set can be sorted in O(n) time)

Sorting in Linear Time

There are several sorting methods that take linear time (see Chapter 9 in CLR)

- Counting Sort
  - sorts integers from a small range: [0..k] where k = O(n)
- Radix Sort
  - the method used by the old card-sorters
  - sorting time O(dn) where d is the number of "digits"
- How do these methods get around the Ω(n log n) lower bound?
  - They don't use comparisons
- What sorting method works best?
  - QuickSort is best general-purpose sort
  - Counting Sort or Radix Sort can be best for *some* kinds of data

Aside: An Open Question on Lower Bounds for Sorting

How long does it take to a sort an n-by-n table of numbers?

- $O(n^2 \log n)$ because there are n^2 numbers in the table

What if it's an addition table?

+	1	3	5	8
2	3	5	7	10
10	11	13	15	28
12	13	15	18	20
14	15	17	19	22

- Shouldn't it be easier to sort than an arbitrary table?