CS5800 – ALGORITHMS

MODULE 4. DIVIDE & CONQUER - II

Lesson 2: Lower Bound

Ravi Sundaram

Topics

- Why lower bounds?
- Comparison-based algorithms
- A special case
- Decision Tree
- Orderings and height
- Summary

Lower bound – why?

- Lower bounds put a limit on what we can achieve
- They give us a target to shoot for
- Once we reach the limit, we can stop

Comparison-based algorithms

- With Merge Sort we have achieved an O(nlogn) algorithm
- Is O(nlogn) the best we can do?
- Yes, there is an $\Omega(\text{nlogn})$ lower bound on sorting algorithms
- Any comparison-based algorithm that sorts n elements must take at least cnlogn time in the worst case
- Key points:
  - comparison-based
  - No special assumptions on input

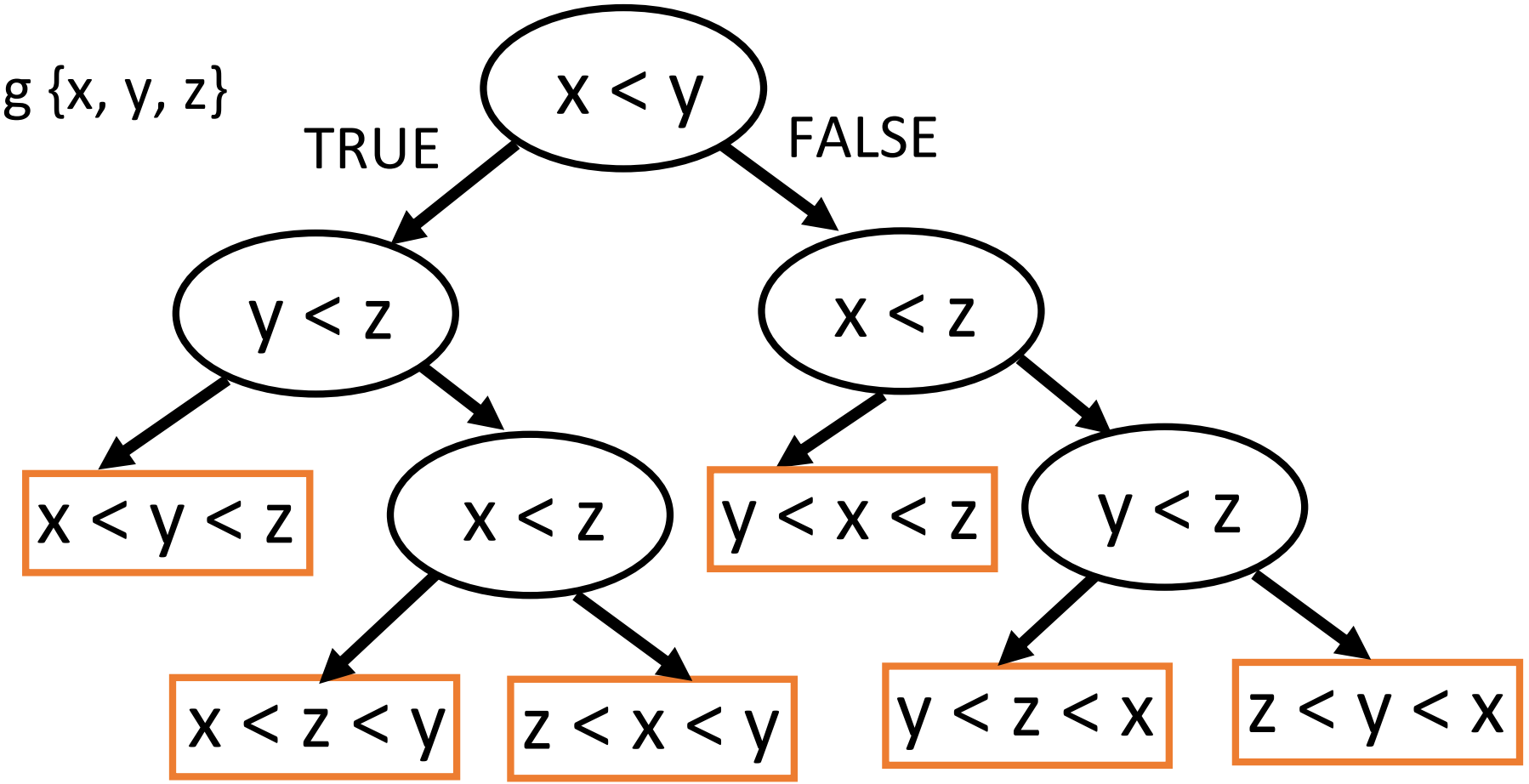
Special Case

- Suppose the input has only 0s and 1s

- Count the 0s, say there are k of them
- Output k 0s followed by n-k 1s
- O(n) – linear time
- Counting beats comparing but comparisons are general purpose

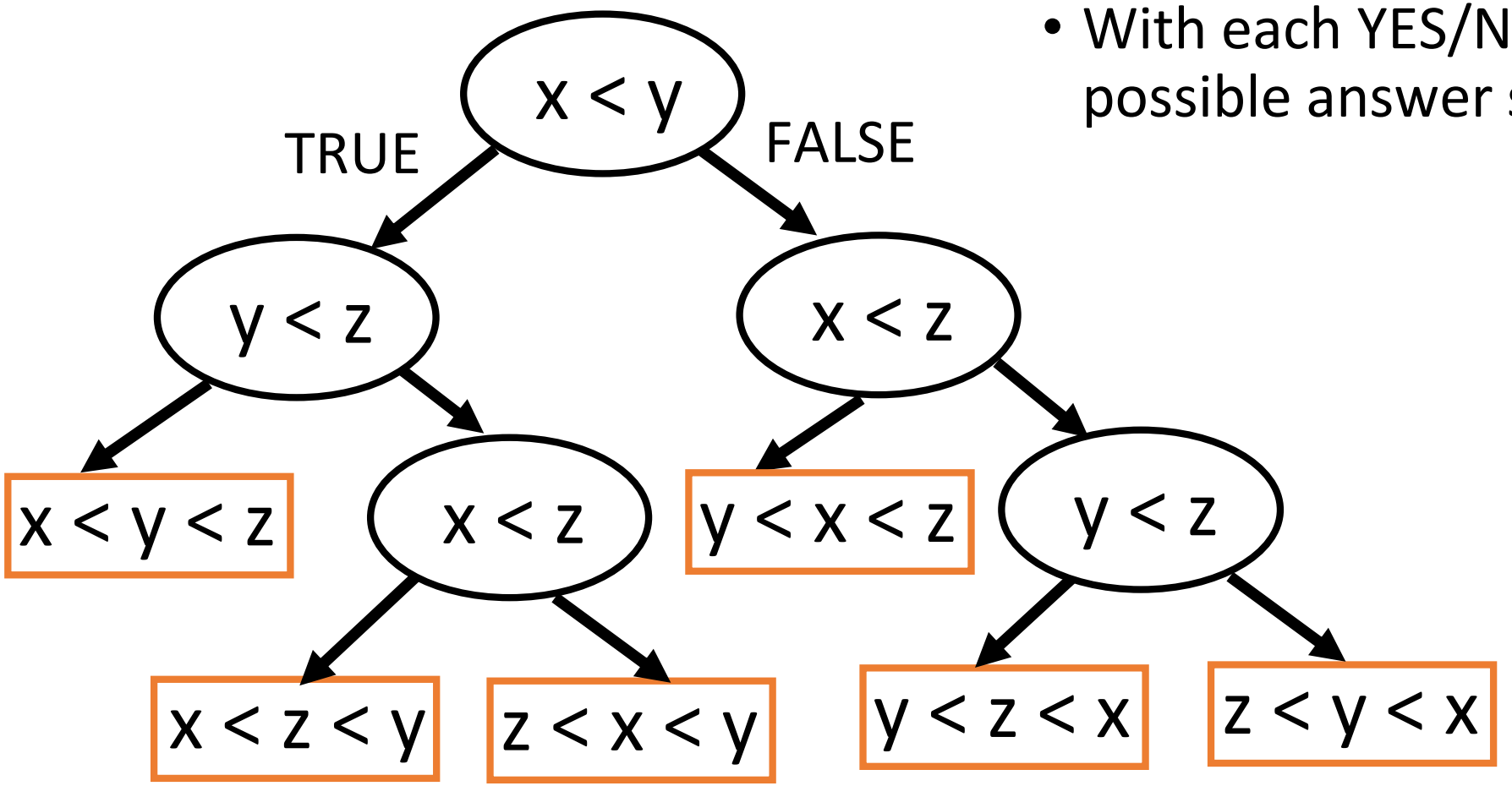
Decision tree

- Consider sorting {x, y, z}



Decision tree

- Alternate way of thinking:
- Initially, start with all possible orders
- With each YES/NO answer shrink possible answer set until one order left



Decision tree

- Any comparison-based sorting algorithm is a decision tree – a binary tree
- Each internal node is a comparison between two elements
- Each leaf node is the sorted ordering of the elements
- The height h (= length of longest path from root to leaf) of the tree is the worst-case complexity of the algorithm

Decision tree

- How many possible orders are there of n elements?
- n!
- Hence number of leaves must be greater than n!

Decision tree

- How many leaves can a binary tree of height h have?
- 2^h
- Hence $2^h \geq n!$
- or, $h = \Omega(\log n!)$

Decision tree

- But,

$$\log n! = \log 1 + \log 2 + \ldots \log n$$
$$\geq \log n/2 + \log n/2 + \ldots \log n/2 \quad \text{(n/2+1 of them)}$$
$$\geq (n/2)\log n/2$$
$$= \Omega(n\log n)$$

- Thus,

$h = \Omega(n\log n)$ and the lower bound is proven

Summary

- Lower bounds are important
- They tell us what to try for
- And when to stop trying
- Main Takeaway: the complexity of comparison-based sorting is $\Theta(n \log n)$, i.e. Merge Sort is an optimal sorting algorithm