Sorting Conclusions

David Kauchak
cs302
Spring 2013

Administrative

- Find a partner to bounce ideas off of
- Assignment 2
  - Proofs should be very concrete
    - Proving big-O, must satisfy definition
  - When comparing (i.e. ordering) functions you're not sure about, set them equal to eachother and do some manipulations
    - e.g. compare n and 3^{\sqrt{\log n}} or n^n and 3^{3^n}
  - Be careful of things that seem "too good to be true", e.g. a new O(n log n) sorting algorithm that you've never heard of ☺

Administrivia continued

- Assignment 3
  - Substitution method is a proof by induction. Follow the steps we talked about before.
  - Make sure to prove/justify any non-trivial steps (e.g. showing big-O, Ω and θ)
  - If you're stuck on an algorithm, brainstorm different possibilities
    - Even start with the naïve algorithm and think about where it's inefficient
- Latex/writeups
  - Use vertical space to help make your argument/steps more clear
  - Look at what is being generated and make sure it looks like you expect (e.g. n^2.5 vs. n^{2.5} or n^1+\epsilon vs n^{1 + \epsilon}

Sorting bounds

Mergsort is O(n log n)

Quicksort is O(n log n) on average

Can we do better?

Comparison-based sorting

Sorted order is determined based **only** on a comparison between input elements

- A[i] < A[j]
- A[i] > A[j]
- A[i] = A[j]
- A[i] ≤ A[j]
- A[i] ≥ A[j]

Do any of the sorting algorithms we've looked at use additional information?

- No
- All the algorithms we've seen are comparison-based sorting algorithms

Comparison-based sorting

Sorted order is determined based **only** on a comparison between input elements

- A[i] < A[j]
- A[i] > A[j]
- A[i] = A[j]
- A[i] ≤ A[j]
- A[i] ≥ A[j]

In Java (and many languages) for a class of objects to be sorted we define a comparator

What does it do?

Comparison-based sorting

Sorted order is determined based **only** on a comparison between input elements

- A[i] < A[j]
- A[i] > A[j]
- A[i] = A[j]
- A[i] ≤ A[j]
- A[i] ≥ A[j]

In Java (and many languages) for a class of objects to be sorted we define a comparator

What does it do?

- Just compares any two elements
- Useful for comparison-based sorting algorithms

Comparison-based sorting

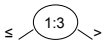
Sorted order is determined based **only** on a comparison between input elements

- A[i] < A[j]
- A[i] > A[j]
- A[i] = A[j]
- A[i] ≤ A[j]
- A[i] ≥ A[j]

Can we do better than O(n log n) for comparison based sorting approaches?

Decision-tree model

- *Full* binary tree representing the comparisons between elements by a sorting algorithm
- Internal nodes contain indices to be compared



- Leaves contain a complete permutation of the input

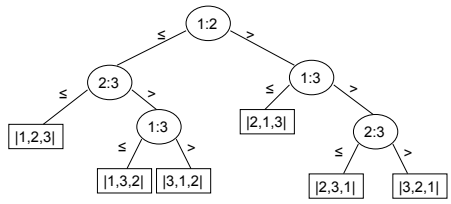
[3, 12, 7] ⇨ | 1,3,2 | ⇨ [3, 7, 12]

| 1,3,2 |

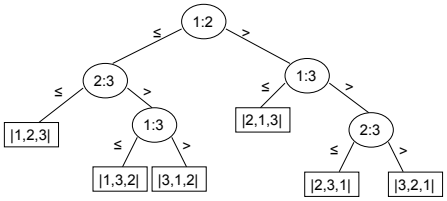
[7, 3, 12] ⇨ | 2,1,3 | ⇨ [3, 7, 12]

- Tracing a path from root to leave gives the correct reordering/permutation of the input for an input

A decision tree model

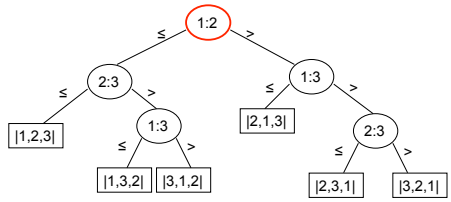


A decision tree model



[12, 7, 3]

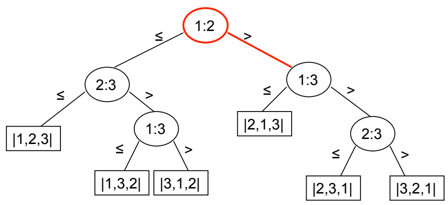
A decision tree model



[12, 7, 3]

Is 12 ≤ 7 or is 12 > 7?

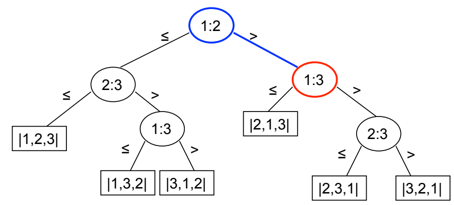
A decision tree model



[12, 7, 3]

Is 12 ≤ 7 or is 12 > 7?

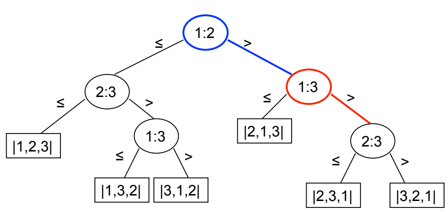
A decision tree model



[12, 7, 3]

Is 12 ≤ 3 or is 12 > 3?

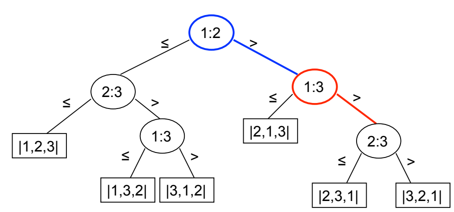
A decision tree model



[12, 7, 3]

Is 12 ≤ 3 or is 12 > 3?

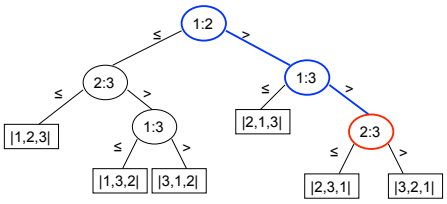
A decision tree model



[12, 7, 3]

Is 12 ≤ 3 or is 12 > 3?

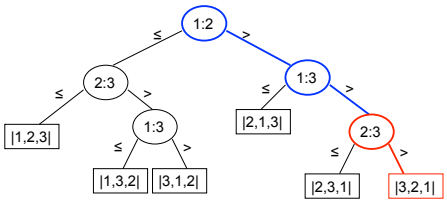
A decision tree model



[12, 7, 3]

Is 7 ≤ 3 or is 7 > 3?

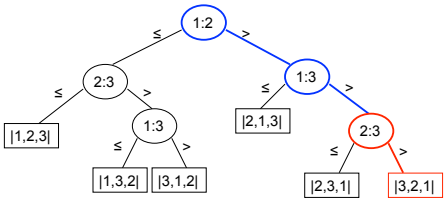
A decision tree model



[12, 7, 3]

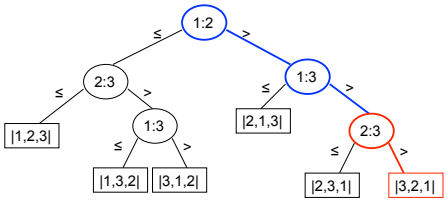
Is 7 ≤ 3 or is 7 > 3?

A decision tree model



[12, 7, 3] ⇨ 3, 2, 1

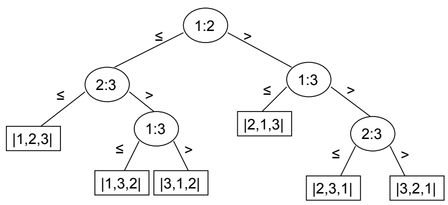
A decision tree model



[12, 7, 3] ⇨ [3, 7, 12]

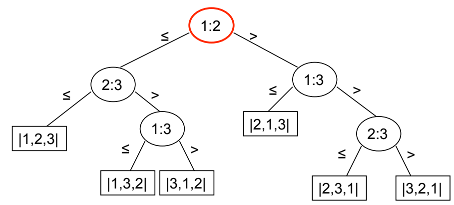
3, 2, 1

A decision tree model



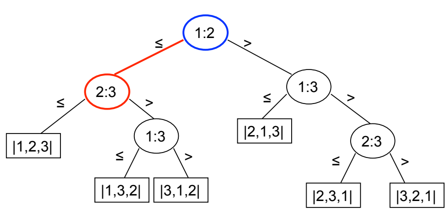
[7, 12, 3]

A decision tree model



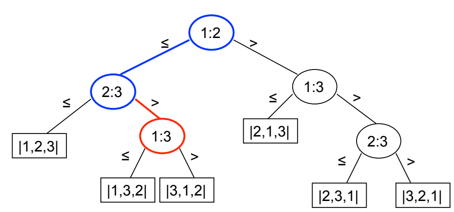
[7, 12, 3]

A decision tree model



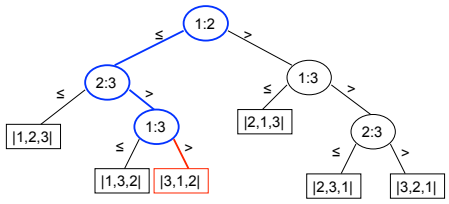
[7, 12, 3]

A decision tree model



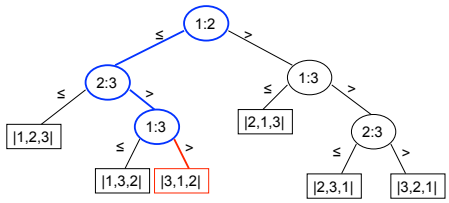
[7, 12, 3]

A decision tree model



[7, 12, 3]

A decision tree model



[7, 12, 3] ⇒ [3, 7, 12]

3, 1, 2

How many leaves are in a decision tree?

Leaves **must** have all possible permutations of the input

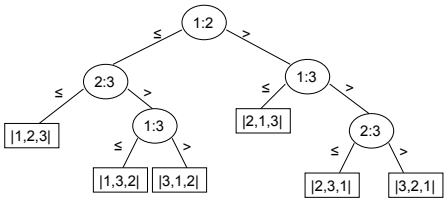
What if decision tree model didn't?

Some input would exist that didn't have a correct reordering

Input of size *n*, *n!* leaves

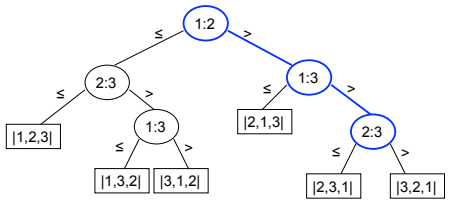
A lower bound

What is the worst-case number of comparisons for a tree?



A lower bound

The longest path in the tree, i.e. the height



A lower bound

What is the maximum number of leaves a binary tree of height *h* can have?

A complete binary tree has 2^h leaves

$$2^h \geq n!$$

$$h \geq \log n!$$

$$h = \Omega(n \log n)$$ from hw ☺

Can we do better?