Complexity of Algorithms, Spring 2026

Lower Bounds

So far in this class:

- Algorithms are analyzed via upper bounds

- Hardness results are the opposite:
  If I could solve problem X,
  then I could use it to solve
  SAT (or another NP-Hard problem)

Big question: How can we know we have the best algorithm?

Example: Sorting

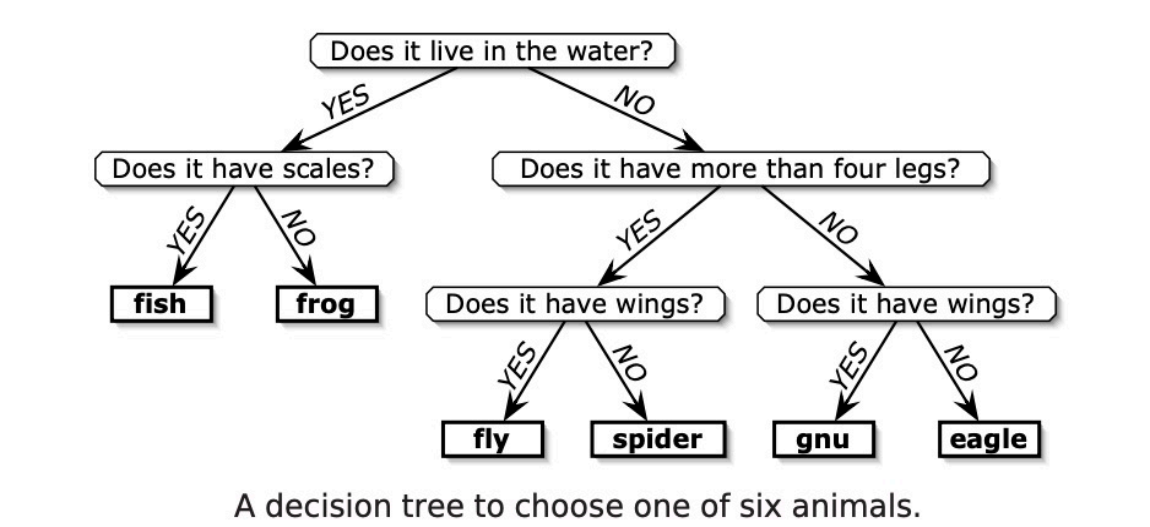
Algorithms:

Lower bound:

Need to prove that any possible algorithm would use that run time.

Decision tree: Representation of a query-based algorithm

"Run time":



A decision tree to choose one of six animals.

What questions are allowed?

↳ Fixes the model of computation

For sorting:

Leaves of tree:

Posets:

Binary relations between sets

A + B is a subset $R \subseteq A \times B$

Binary:

Also note:

Example: $\leq$ and $\mathbb{R} \times \mathbb{R}$

Another: Most functions, ie $f(x) = 2x+5$

Another: the power set $P(X)$ and $\subseteq$

Why?

- Relations are useful for modeling in areas like databases

Here, partial orders $\prec$ are used:

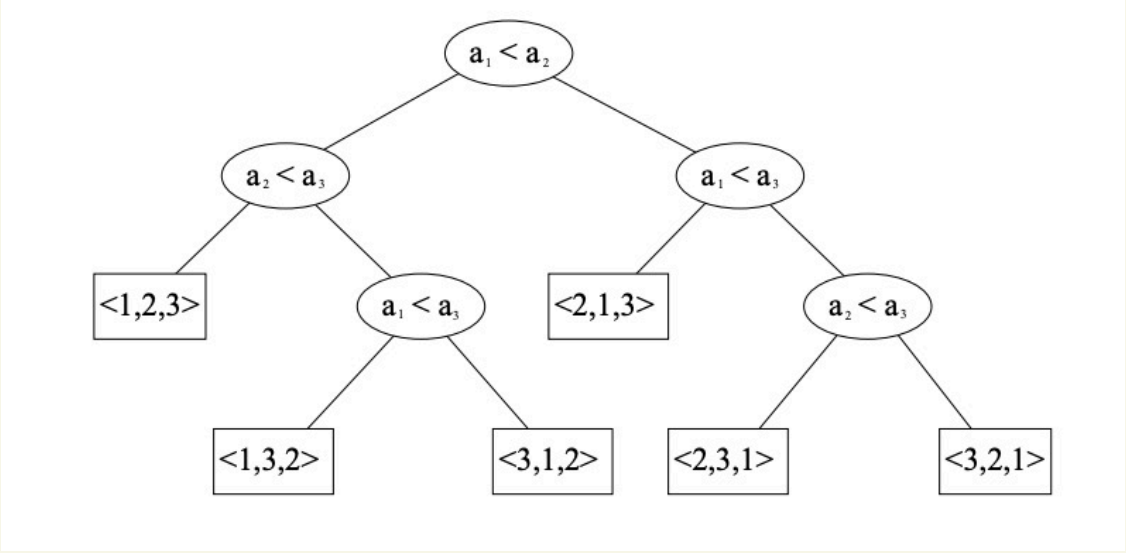
- asymmetric: if $x \prec y$, then $y \not\prec x$.
- transitive: if $x \prec y$ and $y \prec z$, then $x \prec z$
- irreflexive: $x \not\prec x$

Now, back to sorting:

Initally, no idea of order.

Each comparison node is of the form "is $a_i < a_j$?"

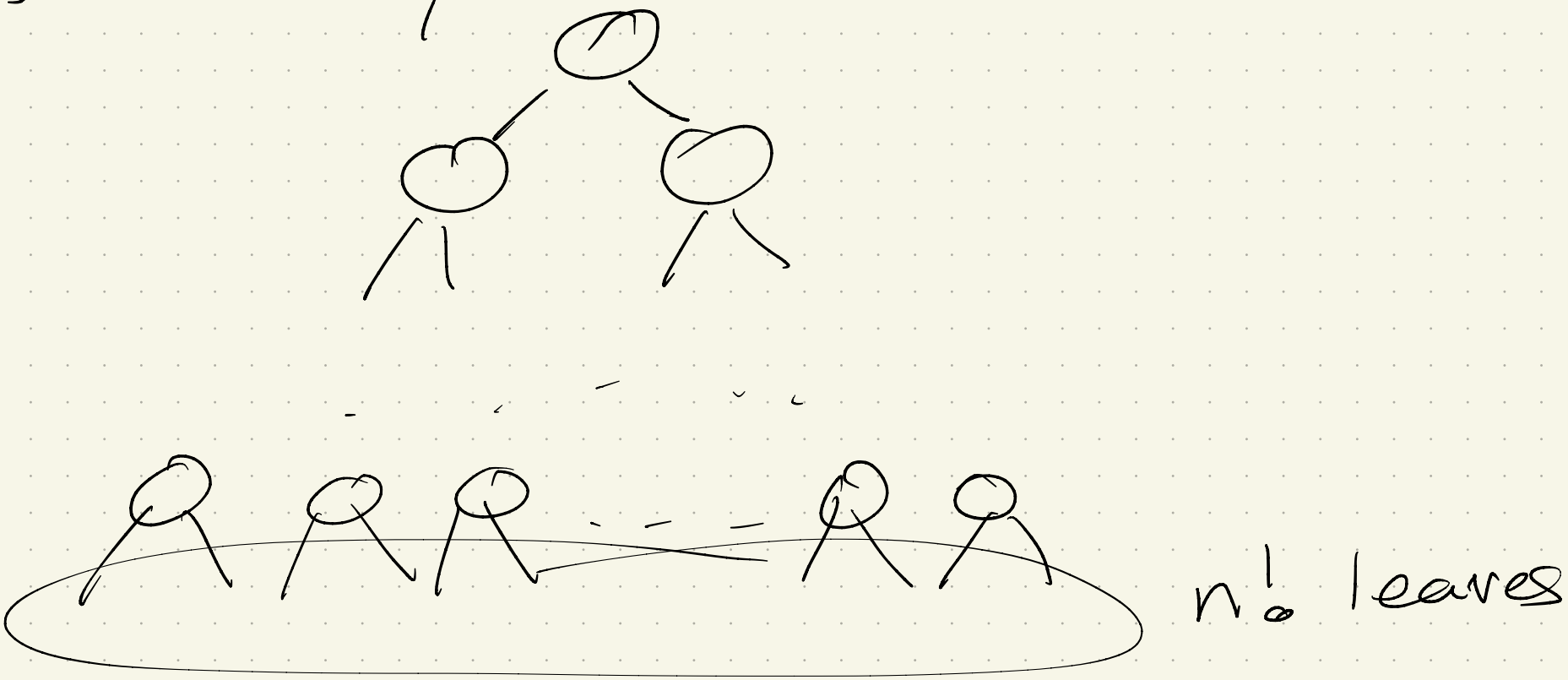
Input: a_1, a_2, a_3



Essentially, at each internal node:

At the leaves:

So, a binary tree:



How deep must it be?

We know a tree of height h has $\leq 2^h$ leaves.

So: $2^h \geq n!$

$\Rightarrow$

Actually a bit stronger:

Not just comparisons!

Any binary decision tree that sorts must have this height.

Does this always work?

No! Consider element uniqueness:

Given $x_1, \ldots, x_n$, is $x_i = x_j$ for any i, j?

How many outputs?

But note: some algorithms do not do comparisons!

Counting sort:

Runtime:

Radix sort:

for each digit (right → left)

perform Counting sort

Ex: 170, 45, 75, 90, 2, 802, 66

Runtime:

Real-RAM model:
Allows bitwise operations on words.

Journal of Algorithms **42,** 205–230 (2002)
doi:10.1006/jagm.2002.1211, available online at http://www.idealibrary.com on IDEAL®

Randomized Sorting in $O(n \log\log n)$ Time and Linear Space Using Addition, Shift, and Bit-wise Boolean Operations [1, 2]

Mikkel Thorup

AT&T Labs—Research, Shannon Laboratory, 180 Park Avenue, Florham Park, New Jersey 07932
E-mail: mthorup@research.att.com

Received June 11, 1997

Start with n "words".

Shorten them, & do radix sort.

How?

Reading for today: Searching

If unsorted we initally have no idea about any relationships among $a_1, \ldots, a_n$ (or if = target x).

If we compare two elements:

Create poset

m comparisons $\Rightarrow$

To find X :

What about sorted lists?

Back to decision tree!

How many leaves?

Some philosophy here:

- How tight do we want bounds?
- What about practical considerations?
- Other models?
- When is this even worth it??

Next time:

Adversarial bounds