CS 580: Algorithm Design and Analysis

Upper and Lower Bounds for Sorting Algorithms

Merge Sort: O(n * log(n))

Q: Can we do better?

Answer: It depends on the model of computation.

Comparisons counted as the expensive operation

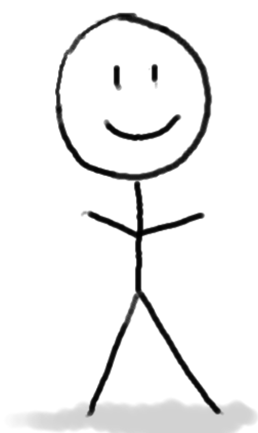
Which is which?  

Credit to Mary Wooters for lower bound slides

Comparison-based sorting

Want to sort these items.
There's some ordering on them, but we don't know what it is.

Is  bigger than  ?



Algorithm

YES

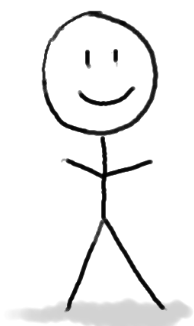
There is a genie who knows what the right order is.

The genie can answer YES/NO questions of the form:
is [this] bigger than [that]?

The algorithm's job is to output a correctly sorted list of all the objects.

Merge Sort and many other sorting algorithms work like this.

7	6	3	5	1	4	2



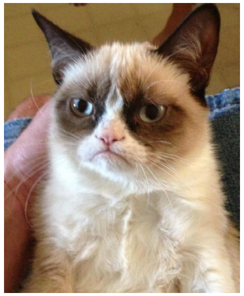
Is 7 bigger than 5 ? **YES**

Is 6 bigger than 5 ? **YES**

Is 3 bigger than 5 ? **NO**

5

etc.



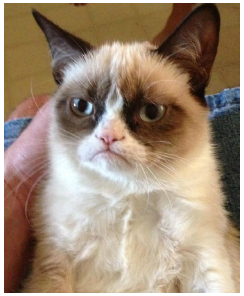
Lower bound of $\Omega(n \log(n))$.

- Theorem:
  - Any deterministic comparison-based sorting algorithm must take $\Omega(n \log(n))$ steps.

This covers all the sorting algorithms we know!!!

- How might we prove this?

1. Consider all comparison-based algorithms, one-by-one, and analyze them.



Lower bound of $\Omega(n \log(n))$.

- Theorem:
  - Any deterministic comparison-based sorting algorithm must take $\Omega(n \log(n))$ steps.

This covers all the sorting algorithms we know!!!

- How might we prove this?

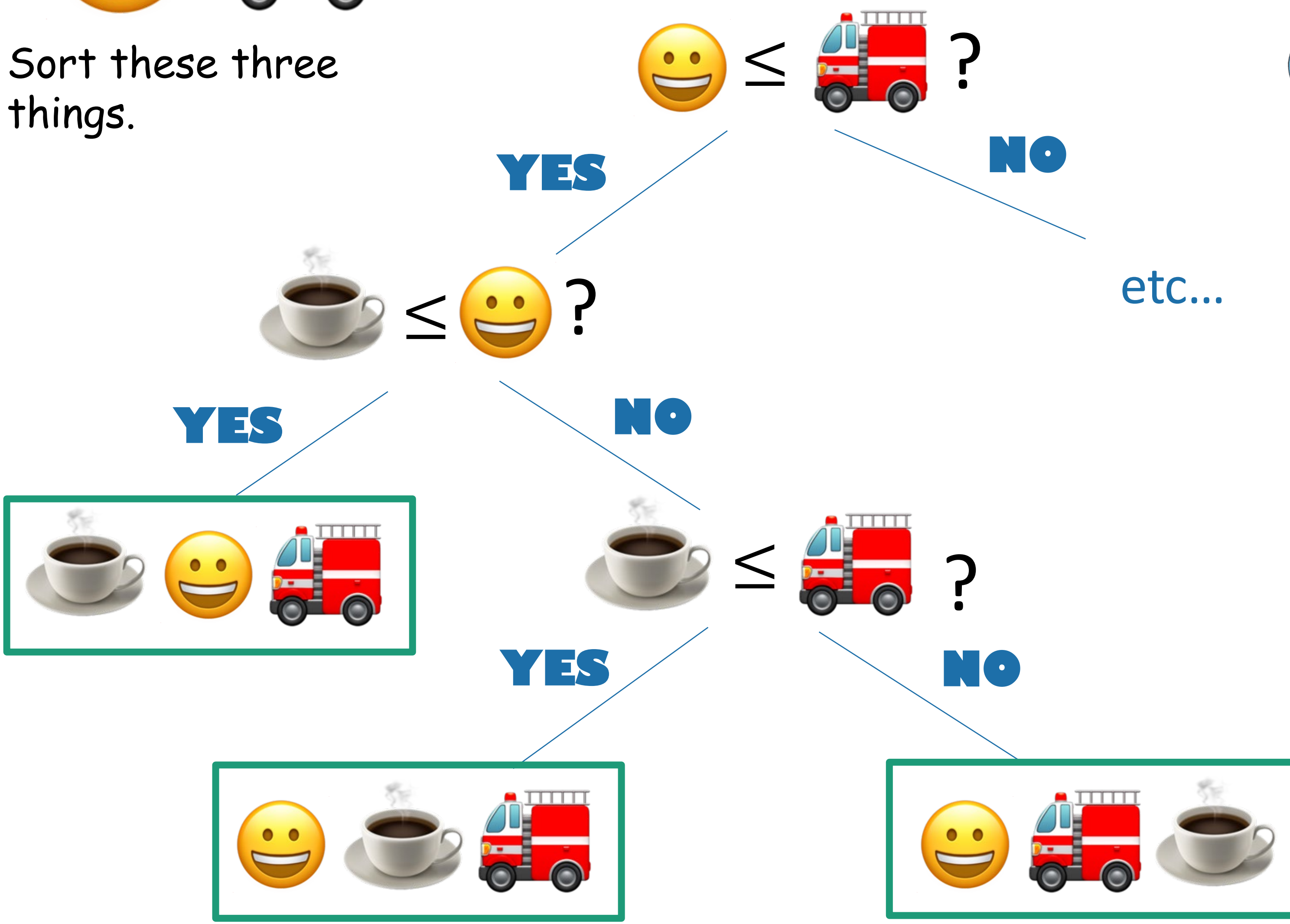
1. Consider all comparison-based algorithms, one-by-one, and analyze them.
2. Don't do that.

Instead, argue that all comparison-based sorting algorithms give rise to a **decision tree**.

Then analyze decision trees.

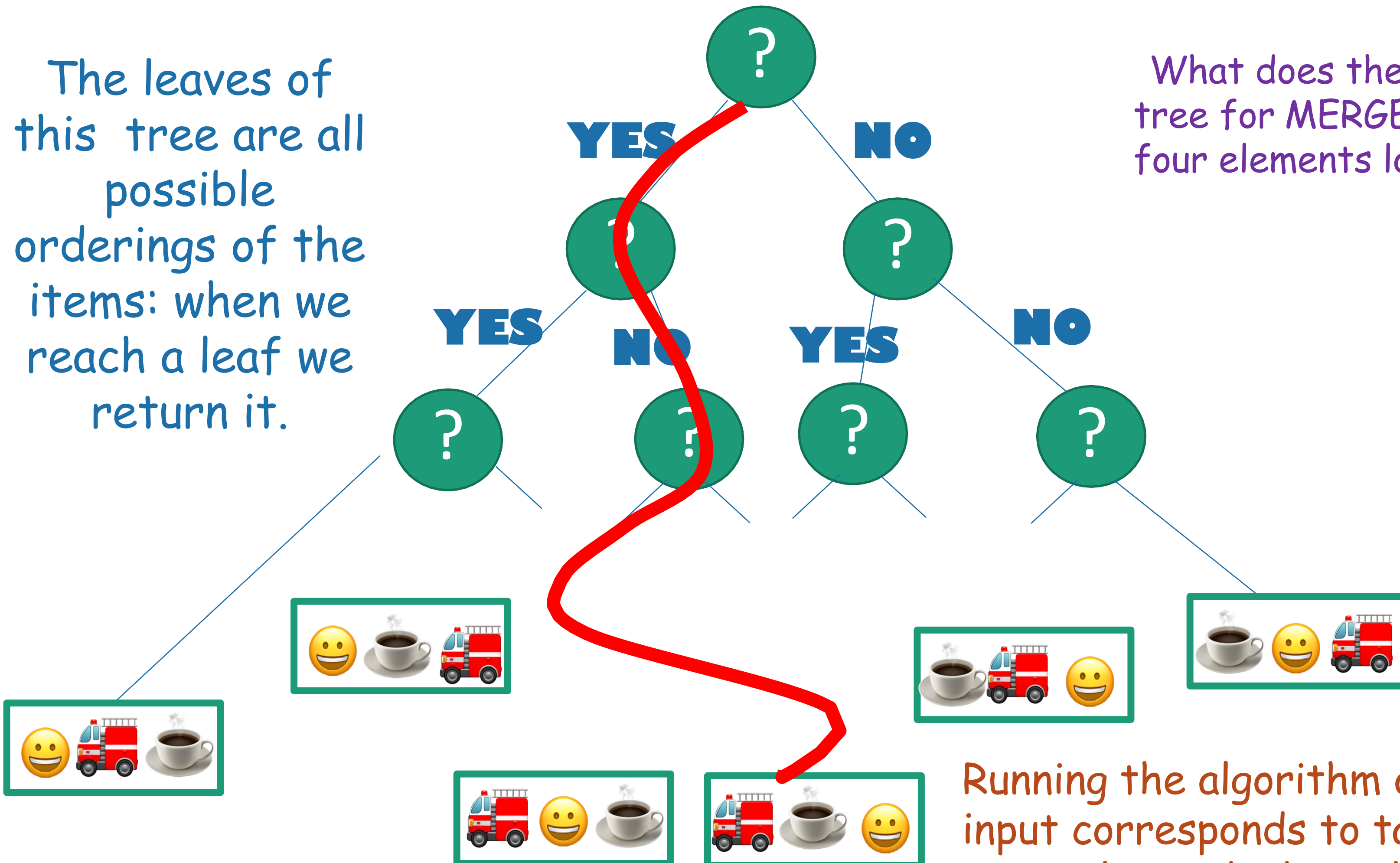
Decision trees

Sort these three things.



All comparison-based algorithms have an associated decision tree.

The leaves of this tree are all possible orderings of the items: when we reach a leaf we return it.



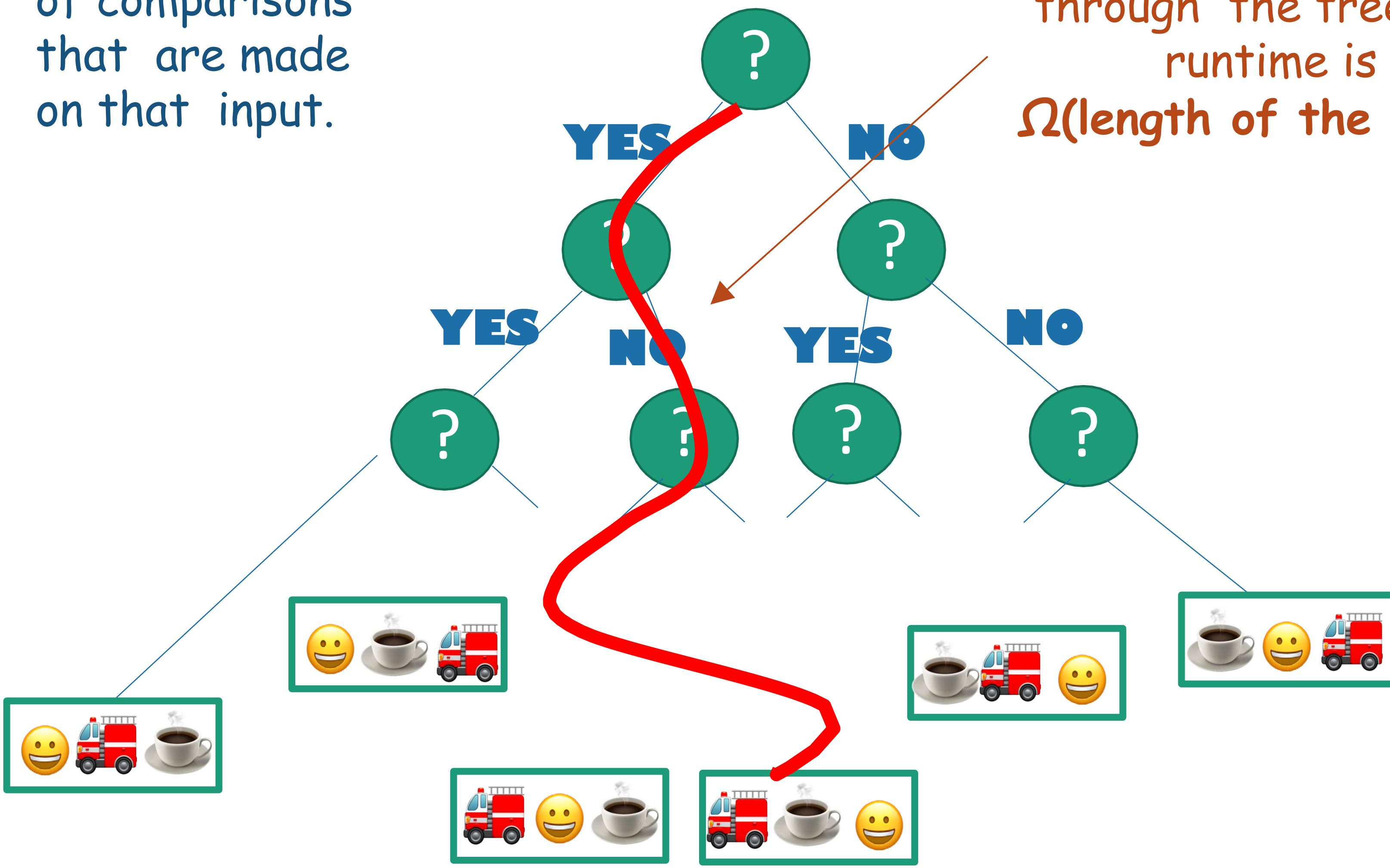
What does the decision tree for MERGESORTING four elements look like?

Running the algorithm on a given input corresponds to taking a particular path through the tree.

What's the runtime on a particular input?

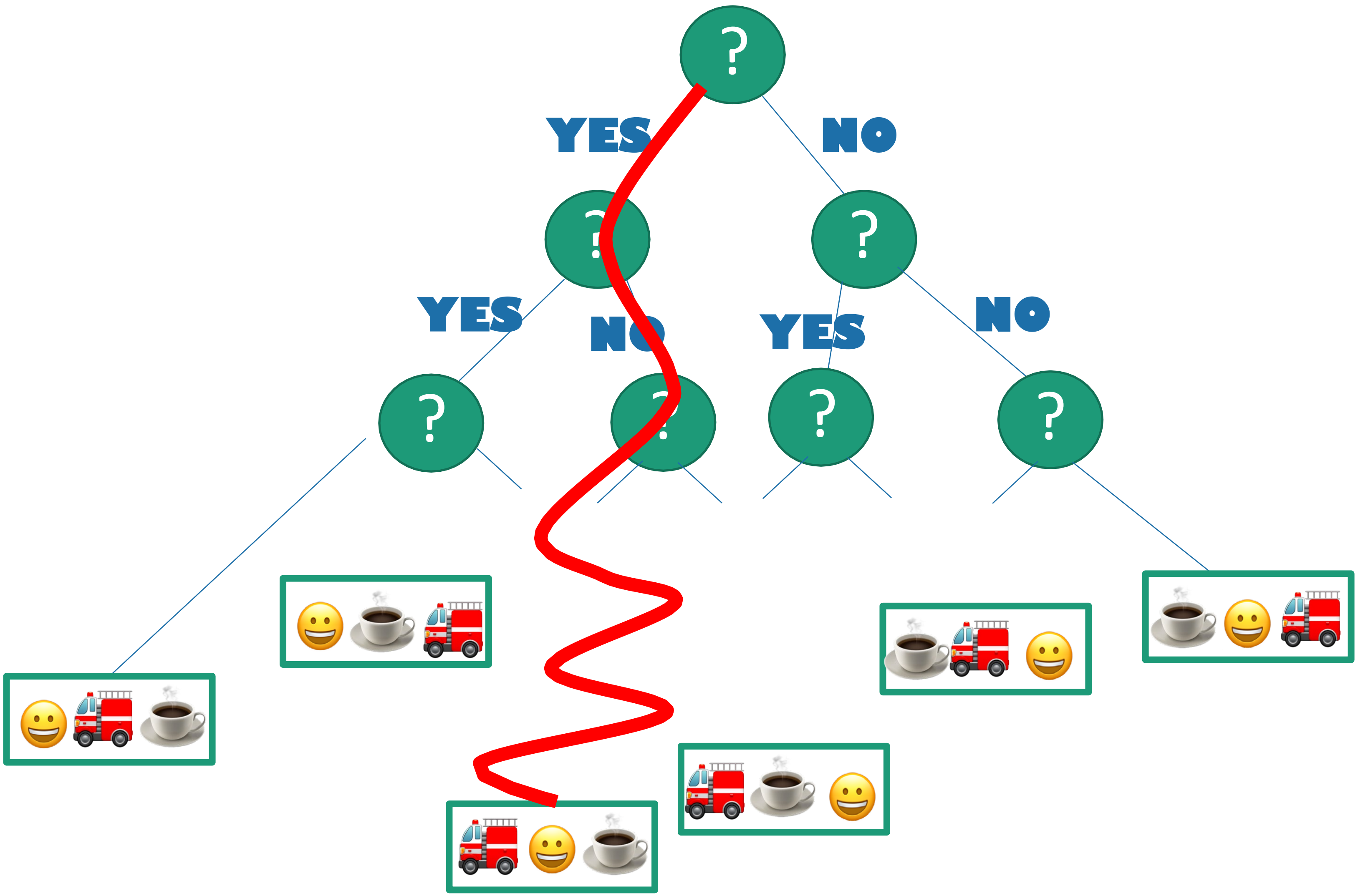
At least the number of comparisons that are made on that input.

If we take this path through the tree, the runtime is Ω(length of the path).



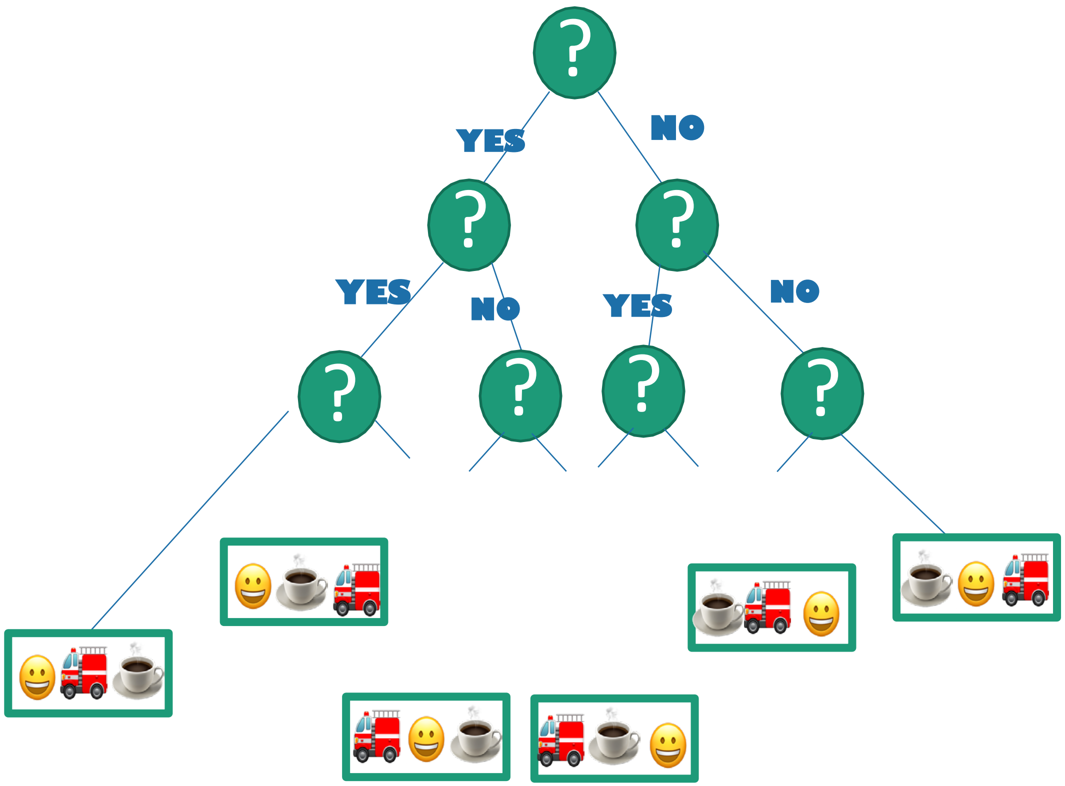
What's the worst-case runtime?

At least Ω(length of the longest path).



How long is the longest path?

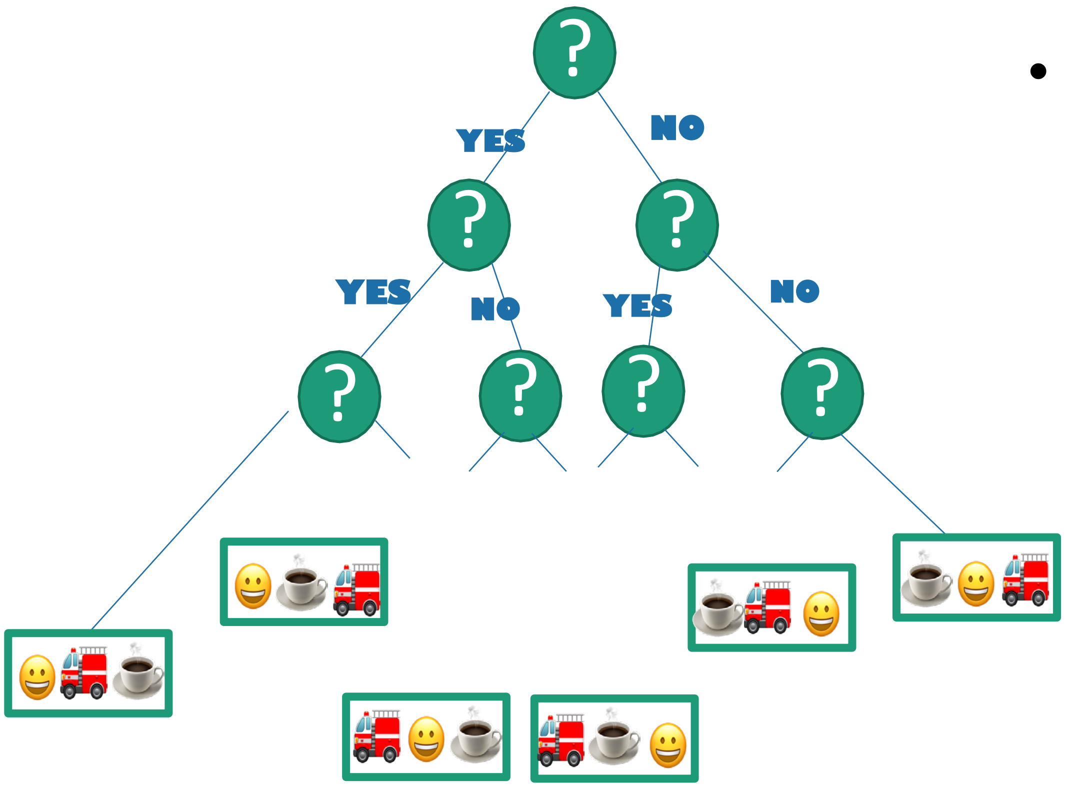
We want a statement: in all such trees, the longest path is at least ______



How long is the longest path?

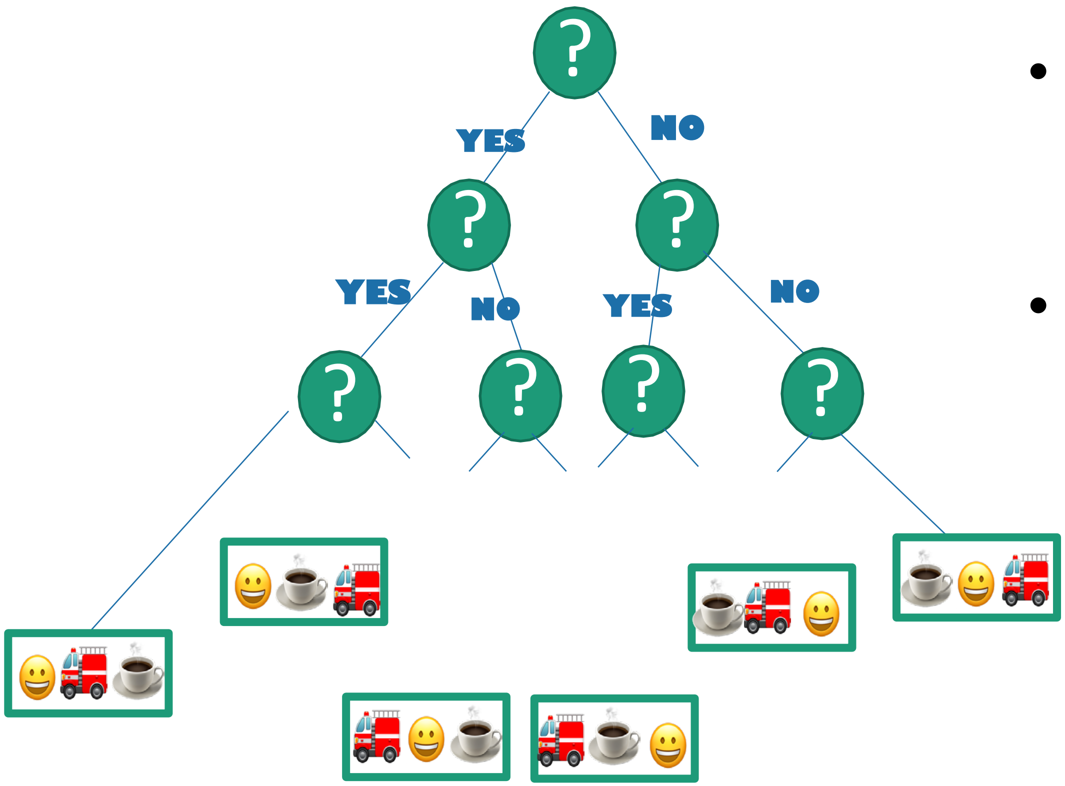
We want a statement: in all such trees, the longest path is at least ______

- This is a binary tree with at least______leaves.



How long is the longest path?

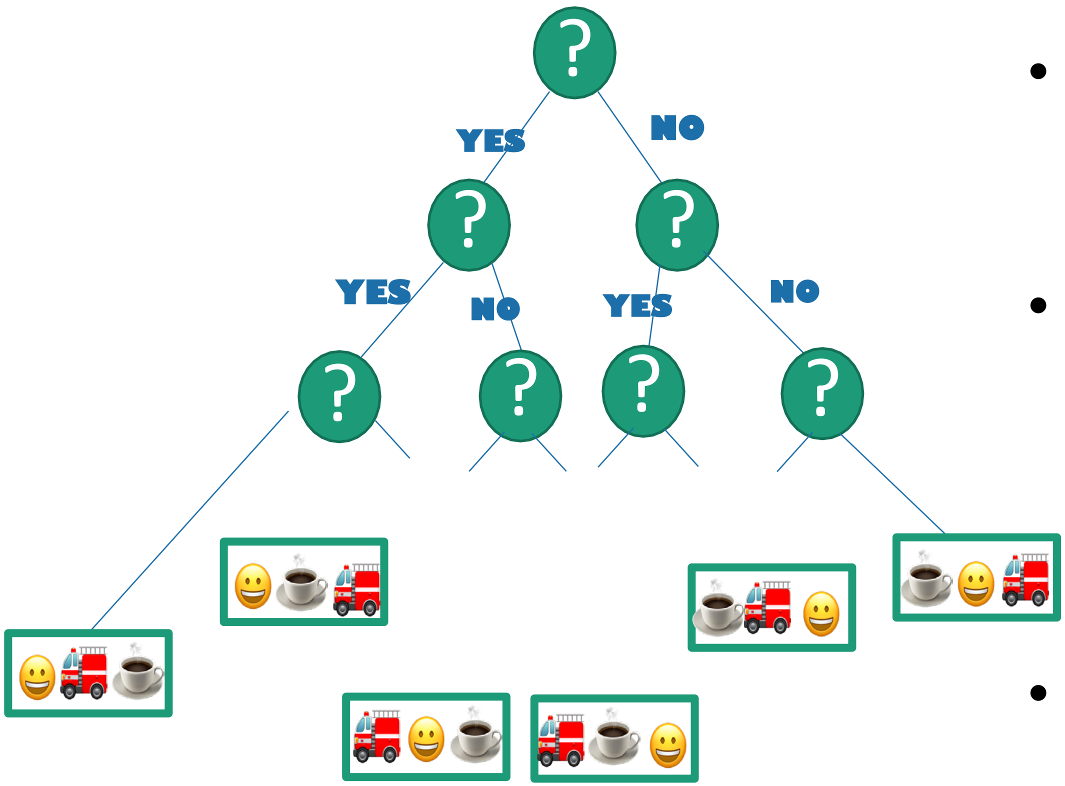
We want a statement: in all such trees, the longest path is at least ______



- This is a binary tree with at least___n!___leaves.
- The shallowest tree with n! leaves is the completely balanced one, which has depth ______

How long is the longest path?

We want a statement: in all such trees, the longest path is at least ______

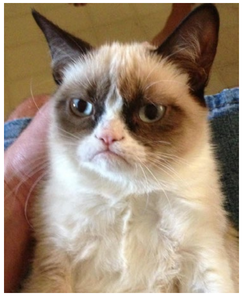


- This is a binary tree with at least__n!__leaves.
- The shallowest tree with n! leaves is the completely balanced one, which has depth __log(n!)__.
- So in all such trees, the longest path is at least log(n!).

- n! is about $(n/e)^n$ (Stirling's formula).
- log(n!) is about n log(n/e) = Ω(n log(n)).

Conclusion: the longest path has length at least Ω(n log(n)).

Lower bound of $\Omega(n \log(n))$.



- Theorem:
  - Any deterministic comparison-based sorting algorithm must take $\Omega(n \log(n))$ steps.
- Proof:
  - Any deterministic comparison-based algorithm can be represented as a decision tree with n! leaves.
  - The worst-case running time is at least the depth of the decision tree.
  - All decision trees with n! leaves have depth $\Omega(n \log(n))$.
  - So any comparison-based sorting algorithm must have worst-case running time at least $\Omega(n \log(n))$.