Algorithms represented as Decision Trees

Internal nodes represent comparison of two elements $i, j$
Branches represent outcome of comparison
Left: $i < j$
Right: $i > j$
Algorithms represented as decision trees

Internal nodes represent comparison of two elements $i, j$
Branches represent outcome of comparison
Left: $i < j$
Right: $i > j$

Example: sort $a_1, a_2, a_3$

Each leaf is a possible output
Each root→leaf path represents an execution of algo.

Verify on $9, 4, 6$
$a, a_2, a_3$

Any decision-based algorithm can be encoded as a decision tree.
If you are designing a decision tree, it’s up to you to avoid comparing the same elements many times.

The worst-case run-time is precisely the longest root-leaf path. You shouldn’t compare \( a_i : a_j \) twice on one path.

\[ \Rightarrow \text{so max path length} = \binom{n}{2} \]

For a good algo for sorting
If you are designing a decision tree, it's up to you to avoid comparing the same elements many times. The worst-case run-time is precisely the longest root-leaf path. You shouldn't compare $a_i:a_j$ twice on one path.

\[ \text{so max path length } = (n) \]

Why not write all algorithms this way? (so much prettier than pseudocode)

\[ \text{It's huge and repetitive.} \]
\[ \text{It really lists every possible execution of algo.} \]
\[ \text{You actually might need a different tree for each } n. \]

What is the shortest possible tree for comparison-sort?
A correct decision tree for sorting must have every possible output represented at a leaf node.

#leaves > ?
A correct decision tree for sorting must have every permutation of the input represented at a leaf node.

\[ \#\text{leaves} \geq n! \]

Height of tree = worst case time = \( h \) \( \implies \#\text{leaves} \leq 2^h \) [binary tree; every node has 2 children]

So, \( n! \leq \#\text{leaves} \leq 2^h \) \( \implies \log n! \leq \log 2^h \implies h \geq \log n! \)

Stirling's formula: \( n! \geq (\frac{n}{e})^n \)

extra analysis of \( \log n! \) follows

\[ h \geq \log \left(\frac{n^n}{e^n}\right) = n \cdot \log \frac{n}{e} = n \log n - n \log e - \Theta(n) \]

\[ h = \Omega(n \log n) \]
\[
\log(n!) = O(n \log n) \\
\log(n!) \leq \log(n^n) = n \log n \Rightarrow \log(n!) \leq c \cdot n \log n \quad \text{for } n \gg 1
\]

\[
\log(n!) = \Omega(n \log n)
\]

\[
\log(n!) = \log(n \cdot (n-1) \cdot (n-2) \cdot (n-3) \cdots \cdots 3 \cdot 2 \cdot 1) \\
= \log \left( n \cdot 1 \cdot (n-1) \cdot 2 \cdot (n-2) \cdot 3 \cdot (n-3) \cdot 4 \cdots \cdots n \cdot \frac{n}{2} \cdot \frac{n}{2} \right) \\
\geq \log \left( n \cdot n \cdot n \cdot n \cdot \ldots \quad n \right) \\
= \log \left( n^{\frac{n}{2}} \right) \quad \text{(assume } n \text{ even)} \Rightarrow \log(n!) \geq \frac{n}{2} \log n
\]

So \( \frac{1}{2} n \log n \leq \log(n!) \leq n \log n \)

In fact, Stirling's approximation: \( \ln(n!) = n \cdot \ln(n) - n + O(\ln(n)) \)