Algorithms represented as Decision Trees

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

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

Example: sort $a_1, a_2, a_3$

Verify on $9, 4, 6$
- $a_1, a_2, a_3$

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

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. 

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

\[\text{for a good algo}\]

\[\text{for sorting}\]
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.

$\Rightarrow$ so max path length $= \binom{n}{2}$

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

$\Rightarrow$ It's huge and repetitive.
It really lists every possible execution of algo.
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 = \( h \quad \Rightarrow \quad \#\text{leaves} \leq 2^h \)  

[binary tree; every node has 2 children]

so, \( n! \leq \#\text{leaves} \leq 2^h \)  

\[ \log n! \leq \log 2^h \Rightarrow \underbrace{h}_{\geq \log(n!)} \]

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

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

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

extra analysis of \( \log n! \) follows
\[ \log(n!) = O(n \log n) \]
\[ \log(n!) \leq \log(n^n) = n \log n \iff \log(n!) \leq 1 \cdot n \log n \quad \text{for } n \geq 1 \]

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

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

so \[ \frac{1}{2} n \log n \leq \log(n!) \leq n \log n \]

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