COMPARISON-BASED ALGORITHMS represented as DECISION TREES
COMPARISON-BASED ALGORITHMS represented as DECISION TREES

example: an algorithm to sort 3 numbers: $a_1,a_2,a_3$ (no duplicates)
COMPARISON-BASED ALGORITHMS represented as DECISION TREES

example: an algorithm to sort 3 numbers: \( a_1, a_2, a_3 \) (no duplicates)

\[ a_1 : a_2 \] compare 2 numbers
COMPARISON-BASED ALGORITHMS represented as DECISION TREES

example: an algorithm to sort 3 numbers: $a_1, a_2, a_3$ (no duplicates)
COMPARISON-BASED ALGORITHMS represented as DECISION TREES

element: an algorithm to sort 3 numbers: $a_1, a_2, a_3$ (no duplicates)
COMPARISON-BASED ALGORITHMS represented as DECISION TREES

Example: an algorithm to sort 3 numbers: $a_1, a_2, a_3$ (no duplicates)
COMPARISON-BASED ALGORITHMS represented as DECISION TREES

eexample: an algorithm to sort 3 numbers: $a_1, a_2, a_3$ (no duplicates)
Comparison-based algorithms represented as decision trees

Example: an algorithm to sort 3 numbers: $a_1, a_2, a_3$ (no duplicates)
COMPARISON-BASED ALGORITHMS represented as DECISION TREES

example: an algorithm to sort 3 numbers: $a_1, a_2, a_3$ (no duplicates)
COMPARISON-BASED ALGORITHMS represented as DECISION TREES

e.g.: an algorithm to sort 3 numbers: $a_1, a_2, a_3$ (no duplicates)
COMPARISON-BASED ALGORITHMS represented as DECISION TREES

e.example: an algorithm to sort 3 numbers: 9, 4, 6

\[ a_1, a_2, a_3 \]
Every comparison-based algorithm has a corresponding decision tree (not just sorting)
Every comparison-based algorithm has a corresponding decision tree (not just sorting).

Note: different number of inputs $\rightarrow$ different tree!
Every comparison-based algorithm has a corresponding decision tree (not just sorting) Note: different # inputs → different tree!

- Every possible output must be represented by at least one leaf
Every comparison-based algorithm has a corresponding decision tree (not just sorting) Note: different #inputs → different tree!

- Every possible output must be represented by at least one leaf
- Every path from root to leaf represents the execution of the algorithm for specific input
Every comparison-based algorithm has a corresponding decision tree (not just sorting). Note: different #inputs $\rightarrow$ different tree!

- Every possible output must be represented by at least one leaf.
- Every path from root to leaf represents the execution of the algorithm for specific input.

$$\begin{array}{cc}
\text{path length} & \uparrow \\
\text{time complexity} & \downarrow
\end{array}$$
Every comparison-based algorithm has a corresponding decision tree (not just sorting) Note: different # inputs $\rightarrow$ different tree!

- Every possible output must be represented by at least one leaf
- Every path from root to leaf represents the execution of the algorithm for specific input

Longest path $\rightarrow$ worst-case time complexity
the **SORTING LOWER BOUND**

a lower bound for the worst-case time complexity of all comparison-based sorting algorithms
Consider the decision tree corresponding to any algorithm that can sort n items.

\[\text{e.g., the best algo EVER.}\]
Consider the decision tree corresponding to any algorithm that can sort $n$ items.\[\leftarrow\text{e.g., the best algo EVER.}\]

Every possible output must be represented by at least one leaf.
Consider the decision tree corresponding to any algorithm that can sort \( n \) items. 

\[ \rightarrow \text{e.g., the best algo EVER.} \]

Every possible output must be represented by at least one leaf.

\[ \rightarrow \text{Every permutation of the input must be represented} \]
Consider the decision tree corresponding to any algorithm that can sort $n$ items. e.g., the best algo EVER.

Every possible output must be represented by at least one leaf. Every permutation of the input must be represented.

$\#\text{leaves} \geq n!$
Consider the decision tree corresponding to any algorithm that can sort n items.

\[ \text{e.g., the best algo EVER.} \]

Every possible output must be represented by at least one leaf.

\[ \text{Every permutation of the input must be represented} \]

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

To have n! leaves, the tree needs a height of at least \( \log_2(n!) \).
Consider the decision tree corresponding to any algorithm that can sort \( n \) items.  
\[
\Rightarrow \text{e.g., the best algo EVER.}
\]

Every possible output must be represented by at least one leaf.  
\[
\Rightarrow \text{Every permutation of the input must be represented}
\]
\[
\Rightarrow \text{#leaves} \geq n!
\]

To have \( n! \) leaves, the tree needs a height of at least \( \log_2(n!) \).  

Conclusion: For every comparison-sort algorithm,  
\[
\text{worst-case time complexity} \geq \log_2 n!
\]
Approximating $\log_2 n!$
\[ \log(n!) = O(?) \]
\log(n!) = O(?)
\log(n!) \leq \log(n^n)
\[ \log(n!) = O(?) \]
\[ \log(n!) \leq \log(n^n) = n\log n \]
\log(n!) = O(n \log n)
\log(n!) \leq \log(n^n) = n \log n
\log(n!) = \Omega(?)
\[ \log(n!) = O(n \log n) \]
\[ \log(n!) \leq \log(n^n) = n \log n \]

\[ \log(n!) = \Omega(?) \]
\[ \log(n!) = \log(n \cdot (n-1) \cdot (n-2) \cdot (n-3) \cdots 3 \cdot 2 \cdot 1) \]
\[ \log(n!) = O(n \log n) \]
\[ \log(n!) \leq \log(n^n) = n \log n \]

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

\[ \log(n!) = \log(n \cdot (n-1) \cdot (n-2) \cdot (n-3) \cdot \ldots \cdot 3 \cdot 2 \cdot 1) \]
\[ = \log(n \cdot 1 \cdot (n-1) \cdot 2 \cdot (n-2) \cdot 3 \cdot (n-3) \cdot 4 \cdot \ldots \cdot n \cdot (n-\frac{n}{2}) \cdot (n-\frac{n}{2})) \]

\( \Rightarrow \text{exactly if } n: \text{even} \)
\[ \log(n!) = O(n \log n) \]
\[ \log(n!) \leq \log(n^n) = n \log n \]

\[ \log(n!) = \Omega(?) \]
\[ \log(n!) = \log(n \cdot (n-1) \cdot (n-2) \cdot (n-3) \cdots \cdots 3 \cdot 2 \cdot 1) \]
\[ = \log(n \cdot 1 \cdot (n-1) \cdot 2 \cdot (n-2) \cdot 3 \cdot (n-3) \cdot 4 \cdots \cdots \cdot (n-\frac{n}{2}) \cdot (n-\frac{n}{2})) \]
\[ \geq \log(n \cdot n \cdot n \cdot n \cdot n \cdot n \cdot \cdots \cdot n) \]

\( (\sim n/2 \text{ terms}) \)
\[
\log(n!) = O(n \log n)
\]
\[
\log(n!) \leq \log(n^n) = n \log n
\]

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

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

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

\[
\log(n!) \leq \log(n^n) = n \log n
\]

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

\[
\log(n!) = \log(n \cdot (n-1) \cdot (n-2) \cdot (n-3) \cdots \cdots 3 \cdot 2 \cdot 1)
\]

\[
= \log(n \cdot 1 \cdot (n-1) \cdot 2 \cdot (n-2) \cdot 3 \cdot (n-3) \cdot 4 \cdots \cdots n \cdot \frac{n}{2} \cdot (n - \frac{n}{2})
\]

\[
\geq \log\left(n \cdot n \cdot n \cdot n \cdot n \cdot n \cdot \ldots \cdot n\right)
\]

\[
= \log(n^{n/2}) \quad \text{(assume } n \text{ even)} \quad \Rightarrow \quad \log(n!) \geq \frac{n}{2} \log n
\]

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