More Than a Data Structure

Binary Search as a Tree

Comparison Sorting Algorithms

All Possible Executions of a Algorithm as a Tree
Comparison Algorithms

- Each node is a **comparison**
- Particular input gives a path
- Ignore all the swaps
  
  We want a lower bound, anyway…

---

Insertion Sort

```cpp
void InsertionSort (
    int *array ,
    int n)
{
    for ( i = 1... n)
        x = array[i];
    for ( j = i ... 1 && array[j] > x)
        array[j] = array[j-1];
    array[j] = x;
}
```

- 3 Item Array
- One leaf for each possible permutation

---

General Sorting Algorithms

- In order to correctly sort, must have one leaf for each permutation of n items
- Smallest **height** when tree is **perfect**
What is log \( n! \) anyway?

Get it from the top

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

So \( \log n! \in O(n \log n) \)

The Bound Underneath

\[
\log n! = \log (n \cdot (n - 1) \cdot (n - 2) \cdots 2 \cdot 1) \\
= \log n + \log (n - 1) + \cdots + \log 2 + \log 1 \\
\geq \log \frac{n}{2} + \log \frac{n}{2} + \cdots + \log \left(\frac{n}{2} - 1\right) + \log \left(\frac{n}{2} - 2\right) + \cdots + \log 1 \\
= \frac{n}{2} \log \frac{n}{2} + \log \left(\frac{n}{2} - 1\right) + \log \left(\frac{n}{2} - 2\right) + \cdots + \log 2 + \log 1 \\
\geq \frac{n}{2} \log \frac{n}{2} \in \Omega(n \log n)
\]

\[
\log n! \in O(n \log n) \cap \Omega(n \log n) = \Theta(n \log n)
\]

Summing Up

Any Comparison Sorting Algorithm

\[
\leq \log n! \in \Theta(n \log n)
\]

Must Take at Least \( n \log n \) Time