Many sorting algorithms are comparison based.
- They sort by making comparisons between pairs of objects
- Examples: bubble-sort, selection-sort, insertion-sort, heap-sort, merge-sort, quick-sort, ...
- Let us therefore derive a lower bound on the running time of any algorithm that uses comparisons to sort \( n \) elements, \( x_1, x_2, \ldots, x_n \).

Let us just count comparisons then.
- Each possible run of the algorithm corresponds to a root-to-leaf path in a decision tree.

The height of this decision tree is a lower bound on the running time.
- Every possible input permutation must lead to a separate leaf output.
- If not, some input \( \ldots 4 \ldots 5 \ldots \) would have same output ordering as \( \ldots 5 \ldots 4 \ldots \), which would be wrong.
- Since there are \( n! = 1 \times 2 \times \cdots \times n \) leaves, the height is at least \( \log (n!) \).

Any comparison-based sorting algorithm takes at least \( \log (n!) \) time.
- Therefore, any such algorithm takes time at least

\[
\log (n!) \geq \log \left( \frac{n}{2} \right)^{\frac{n}{2}} = \left( \frac{n}{2} \right) \log \left( \frac{n}{2} \right).
\]

That is, any comparison-based sorting algorithm must run in \( \Omega(n \log n) \) time.