Lower Bound

- Trivial Lower Bounds: Based on sizes. For example, searching needs to take $\Omega(n)$ time.
- Adversarial Method: Based on any program trying to solve the problem, use an adversarial input to make sure that it takes lot of time.
- Information Theoretic Methods: Based on counting, argue that the computation cannot get enough information in less steps to solve the problem.

Decision Tree

- Consider an algorithm being represented by a tree:
  - each node represents the comparison that is made
  - the two child sub-trees represent the continuation of the algorithm based on whether the answer to the comparsion is true of false.
  - At the leaves, an answer is given.
  - Thus, the total number of leaves must be at least the total number of possible anwers

Lower Bound for Sorting Problem

- Assumption: Sorting algorithm works only on the basis of comparisons. That is, it cannot look at the "value" of an item, and only compare different items using $\leq$.
- $\Omega(n\lg n)$ lower bound for comparison based sorting.

- Let T be the decision tree that represents the algorithm for input of size n (that is n elements in the input). In this decision tree, each node represents a comparison, and the left/right sub-tree denotes what the algorithm does based on the result of comparison being true or false.
- Note that the tree must have at least $n!$ leaves, as it must have at least one leaf for each possible order of the inputs.
- Let h denote the height of the comparison tree.
- Thus we have that $2^h \geq n! \sim \sqrt{2\pi n}(n/e)^n$.
- Since $\log(n!) = \Theta(n \log n)$, we have that $h \in \Omega(n \log n)$

Can we do better?

- Note that the lower bound holds, whatever the value inside the arrays may be, as long as one can only do comparisons for the array elements.
- Can we do better if we use something other than comparison among array elements?
- Linear time, indexing ...