1. **Lower bound analysis for comparison-based sorting**

A comparison-based sorting algorithm sorts by comparing two items at a time. One way to think about this is suppose we access information about the items only by using calls of the form `thisItem.compareTo(otherItem)`. Track the number of such calls.

**Theorem 1.** Comparison-based sorting requires $\Omega(n \log n)$ comparisons to sort $n$ items (in the worst case).

**Proof.** Making $k$ comparisons yields $2^k$ possible outcomes. We have $n!$ possible permutations of the $n$ items, which must be distinguishable by the outcomes of the $k$ comparisons. Therefore, $2^k \geq n!$. (Think of this as playing “20 questions” where I am secretly thinking of a permutation of $n$ items, and you must determine that permutation by asking $k$ yes/no questions of the type “is item 23 before item 201?”) Now $n! \geq (n/2)^{n/2}$, so $k \geq \frac{n}{2} \log_2 \frac{2n}{2} = \Omega(n \log n)$. □

2. **Average analysis for quick sort**

**Theorem 2.** Sorting $n$ items with quick sort uses $O(n \log n)$ comparisons on average.

For simplicity, suppose we are sorting unique elements. Let $x = (x_1, \ldots, x_n)$ be a permutation of $\{1, 2, \ldots, n\}$ chosen uniformly at random. Let $T(n)$ be the expected (average) number of comparisons when sorting $x$.

**Proof.** Make $n - 1$ comparisons of the pivot with everything else. There are $i$ things smaller than the pivot and $n - i - 1$ things bigger. Note that $i$ is uniformly distributed. So we have

$$T(n) = (n - 1) + \frac{1}{n} \sum_{i=0}^{n-1} T(i) + T(n - i - 1)$$

$$= n - 1 + \frac{2}{n} \sum T(i)$$

$$nT(n) = n(n - 1) + 2 \sum T(i)$$

Subtract (*) with (*)

$$nT(n) - (n - 1)T(n - 1) = n(n - 1) - (n - 1)(n - 2) + 2T(n - 1)$$

$$nT(n) = (n + 1)T(n - 1) + 2(n - 1)$$

$$\frac{T(n)}{n+1} = \frac{T(n-1)}{n} + \frac{2(n-1)}{n(n+1)}$$

$$\leq \frac{T(n-1)}{n} + \frac{2}{n+1}$$

$$\leq \frac{T(n-2)}{n-1} + \frac{2}{n} + \frac{2}{n+1}$$

$$\cdots$$

$$\leq \frac{T(1)}{2} + 2\left(\frac{1}{3} + \frac{1}{4} + \cdots + \frac{1}{n+1}\right) = O(\log n)$$

So $T(n) = O(n \log n)$, as desired. □

If we use a binary search tree (BST), we can get a one-line proof by using linearity of expectation:
Proof. The number of comparisons of quick sort on $x$ is the number of comparisons when building a BST when we add items of $x$ in order. For $i < j$, let $X_{ij}$ be the indicator that $x_j$ is compared to $x_i$ (when adding $x_j$). Let $X = \sum_{i<j} X_{ij}$. Note that $(x_1, \ldots, x_i, x_j)$ is a permutation of those $i + 1$ elements chosen uniformly at random. So $x_j$ is next to $x_i$ with probability $2/(i + 1)$. So, by linearity of expectation, we have

$$
E[X] = \sum_{i<j} E[X_{ij}] = \sum_{i<j} \frac{2}{i+1} \overset{*}{=} O\left(\sum_j \log(j)\right) = O(n \log n),
$$

where $*$ is because $\int_0^{j-1} \frac{1}{x+1} dx = \log j$. \hfill \square