# HW6: Proving Upper and Lower Bounds

**Q1: Show that the merge sort algorithm is in $\Theta(n \log n)$. Hint: You need to prove both the upper bound and lower bound in order to show the tight bound.**

**A1:**

**(Showing the upper bound)** Assuming that the original set contains $n$ elements. Since the partition splits the set into half, there are at most $\log_2 n$ splits before we reach sets with only one element inside. Note that the merge process takes place at each split would need to have $n$ comparisons. The upper-bound of the merge-sort algorithm is thus $O(n \log_2 n)$.

**(Showing the lower bound)** There could be many different inputs to the algorithm, i.e. different orders of the numbers. Precisely, n! different inputs. Imagine that we are reordering the elements in the array based on the answers. In this case, sorting the list of numbers is equivalent to identifying an input (a specific permutation of the elements) that can be converted to a sorted list based on our answer-driven reordering. For each obtained answer, the inputs will be partitioned into two groups, a group is consistent with the answer and the other is not. In the worst case, the answer is given by a demon who wishes the algorithm to run as slow as possible. So, the demon will always choose the answer that is consistent with the larger group. In this case, the size of the valid inputs is at most reduced to half for each question we ask. In this case, we have to ask at least $\log_2$ n! questions to eventually reduce the number of valid inputs to 1. Using Stirling's approximation, we have $log_2 n! \approx n\, log_2 n$ and therefore all comparison-based algorithms is in $\Omega(n\, log_2 n)$.

**(Showing the tight bound)** Since the algorithm is in both $O(n \log_2 n)$ and $\Omega(n\, log_2 n)$, it is in $\Theta(n \log n)$.