For the $\text{mergeSort}$ D&C algorithm, we obtained the following recurrence relation:

$$T_r(\hat{n}) = \begin{cases} 
1 & n = 1 \\
2T_r\left(\frac{\hat{n}}{2}\right) + \hat{n} + 1 & n > 1 
\end{cases}$$

we also, obtained:

$$T_c(\hat{n}) = \hat{n} (\lg \hat{n}) + 2\hat{n} - 1$$

where $\hat{n} = 2^k$

- **Example 65.** Prove $T_r(\hat{n}) = T_c(\hat{n})$.

- **Example 66.** In a few recent proofs, we used the fact that $\lg n$ is monotonically increasing. Let’s prove it here. Prove $\lg(n - 1) < \lg n$  $\forall n > 1 \in \mathbb{N}$.

- **Example 67.** Demonstrate how to obtain the time complexity of $\text{binSearch}$ using the approach we used in class for $\text{mergeSort}$.

- **Example 68.** Demonstrate how to obtain the time complexity of $\text{mergeSort}$ using the approach we used in class for $\text{binarySearch}$.

- **Example 69a.** In this week’s lecture, we proved that $T_{\text{mergeSort}}(n) \in \Omega(n \lg n)$ without using induction. As yet another practice on induction, prove $T_{\text{mergeSort}}(n) \in \Omega(n \lg n)$ by induction.

- **Example 69b.** In this week’s lecture, there is a non-inductive proof for $T_{\text{mergeSort}}(n) \in O(n \lg n)$. As yet another practice on induction, prove $T_{\text{mergeSort}}(n) \in O(n \lg n)$ by induction.

- **Example 70.**

  **Part A** Develop a divide and conquer algorithm similar to $\text{mergeSort}$, but let’s call it $\text{merge3Sorted}$, in which a large list splits to 3 sublists and each sublist splits to 3 sublists until there is only one element in the sublist (note that a sublist with one element is considered sorted). Then, sorted sublists merge together to make a bigger sorted list.

  **Part B** Provide a recurrence relation for the time complexity of your $\text{merge3Sorted}$.

  **Part C** Using either the approach that we used for $\text{binSearch}$ or the one we used for $\text{mergeSort}$, demonstrate what the time complexity of $\text{merge3Sorted}$ is.

  **Note 1.** You can find a fair comparison of these approaches [here](#).

  **Note 2.** In the closed form you find in Part B, the base of $\log$ is perhaps 3.

We do not intend to publish solutions (or solutions outline) for any of the questions of the course notes, or extra practices. You are more than welcome to discuss your solutions with us.