CS240, Spring 2022
Assignment 2: Question 2

Q2) We want to prove the following: there is no comparison-based algorithm that can merge m sorted arrays of length m into a unique sorted array of length m^2 doing $O(m^2)$ comparisons. We argue by contradiction, and we assume that it is possible, so that we have such an algorithm (which we call FastMerge).

Modify MergeSort in order to use FastMerge, and derive a contradiction. You may use the following property: if a function $T(n)$ satisfies $T(n) = \sqrt{n}T(\sqrt{n}) + O(n)$, then $T(n) = O(n \log(\log(n)))$. Do not worry about n being a perfect square or not.

We will prove using contradiction, that is we will assume there is a comparison-based algorithm (called FastMerge) that can merge m sorted arrays of length m into a unique sorted array of length m^2 doing $O(m^2)$ comparisons.

Moving on we will modify merge sort so that it will do the following steps:

1. Split the starting array into $\sqrt{n}$ new arrays of size $\sqrt{n}$ and sort theses new arrays.
2. Merged them back into a bigger arrays using FastMerge.

Step (1) will involve $\sqrt{n}$ recursive calls all of which have a time complexity of:

$$T(n) = \sqrt{n}T(\sqrt{n})$$

And since splitting happens in constant time, we wont need to factor it in.
Step (2) will take $O(n)$ time as we assumed when starting our proof. Thus we will have a total time complexity of:

$$T(n) = \sqrt{n}T(\sqrt{n}) + O(n)$$

From what we are given in the question we get the new equation:

$$T(n) = O(n \log(\log(n)))$$

However we know from the slides that merge sort has a time complexity of:

$$T(n) = O(n \log(n))$$

In order to compare these time complexities we will do a limit comparison test to get:

$$\begin{aligned} L &= \lim_{n->\infty} \frac{n \log(\log(n))}{n \log(n)} \\ &= \lim_{n->\infty} \frac{\log(\log(n))}{\log(n)} \end{aligned}$$

Before moving forward we will make a variable $x = \log(n)$ and then take the derivative in respect to x, note that as n approaches ∞ so will x.

$$
\begin{aligned}
L &= \lim_{x->\infty} \frac{\log(x)}{x} \\
&= \lim_{x->\infty} \frac{\frac{1}{x}x - ln(x)}{x^2} \\
&= \lim_{x->\infty} \frac{1 - ln(x)}{x^2} \\
&= \lim_{x->\infty} \frac{1}{x^2} - \frac{ln(x)}{x^2} \\
&= \lim_{x->\infty} \frac{1}{x^2} - \frac{\frac{ln(x)}{x}}{x} \\
&= \frac{1}{\infty} - \frac{0}{\infty} \\
&= 0
\end{aligned}
$$

Therefore by the limit test we get that:

$$n \log(\log(n)) \in o(n \log(n))$$

This means that $n \log(\log(n))$ has a growth rate less then $n \log(n)$, which is a contradiction because we know that merge sort has a time complexity of $O(n \log(n))$. Thus we have disproved by contradiction that there is no comparison-based algorithm that can merge m sorted arrays of length m into a unique sorted array of length m^2 doing $O(m^2)$ comparisons.