Question 2

Zeal Liang

Z5325156

First, we use an array A to store the deadlines of each task.

Then we sort the array A incrementally using a merge sort algorithm.

Suppose Alice leaves all her tasks to be completed by the deadline, so that her anger is 0. But it may happen that she needs to write more than one task on a given day. So, Alice needs to schedule her time from back to front.

Let's start on the last day, and if there are k tasks to be written on day i, then move k-1 of them forward to day i-1 and add k-1 to the rage value.

Then continue planning for day i-1, and so on. Note that if there are no tasks to do on a particular day, simply find the first deadline day that is less than i and continue planning straight ahead from that day until all tasks are completed and the rage value is returned.

But if there is more than one task left on the first day, it means not all tasks can be completed by their deadlines.

The algorithm has a time complexity of $O(n \log n)$ for the merge-sort part of the first step. The worst case remaining is to perform the task move on each day, which is linear time, so the overall time complexity of this algorithm is still $O(n \log n)$.