Question 2 by Dan Nguyen (z5206032)

Let there be an array of n task deadlines, T, where task i has a deadline $T[i] = d_i$. Tasks can be done on the same day it is due and only a single task can be completed in a single day.

Let there be a variable, R, which sums the rage of doing a task earlier than its deadline i.e. if a task i is done on a day j then the R is increased by $d_i - j$.

Let there be a variable, S, which represents the day that the n tasks have been received.

Merge-sort T from latest task deadline to earliest task deadline such that:

$$T = (d_n, d_{n-1}, ..., d_1)$$

Use a greedy algorithm to set what day a task should be done on.

Iterating through T, set the task to be completed on the day it is due i.e. task n is done on day d_n. This accumulates zero rage. If there is another task that is due on the same day then set the other task's completion date as the next earliest day before the deadline with a task vacancy i.e. task $n-1$ is done on day $d_n - 1$ or if there is already a task on day $d_n - 1$ then the next earliest day is $d_n - 2$ and so on. This is done for each task with the same due date. This will accumulate some rage and so R is increased by $d_i - j$ for each task done before the due date.

If any tasks are set to be completed before S then it's impossible that all tasks can be completed by their deadlines. Otherwise, it is possible to complete all tasks by their deadlines and the minimum total rage R can be returned.

Iterating through T has an expected time complexity of $O(n)$, however the expected time complexity of the merge-sort algorithm is $O(nlog(n))$. Therefore, the final expected time complexity is $O(nlog(n))$ as required.