Q1

As for the program, the most important part is to understand queue. By the hint, we should enqueue when we encounter a deletion case, so when the dequeue? Note that we should recheck whether the new element should be deleted, then we should dequeue whenever a loop is over. In order to promote time efficiency, the loop should be the loop of queue. And that is how queue is appropriate. We have two lists, N list and A list, by the process, N list should be checked in the first round, then to avoid rechecking, we should look at the A list. We change element deletion in N by changing it to 0, and if the original value is still in the N list, then the respective value in N list should be deleted as well, that make N list switch to A list to promote time efficiency. By using dictionary, we can improve time efficiency in the first round (rather then simply “if N[i] in A”), while making it easier to change value in A, by reducing the time count of each element rather than changing the list itself.

In the beginning I fail to make use of the queue, instead, I use queue as a list which was a critical mistake. Then, by thinking how the program works, I know how to use queue eventually. Sometime you should not be trapped in the hint, just try to figure it out by yourself and you will know the hint actually makes sense.

Also, we can not scan the list again and again, even it is not a loop, it will take us much time.

Q2

As for the question, the most important part is to find the time complexity. We should be grateful that the question is quite simple since no strange input exist. We check time complexity of each F before N: (tc = time complexity, initial tc = 0, final tc is O(n^tc))

Digit N → tc + 1

Digit Digit → tc + 0

N N → tc + 0

Then, we check the loop. The simplify version is as follows.

F F E E→ 2 F makes a loop

F F F E E E F F E E → 2 loops, the bigger one will be counted.

F F E F F F E E E E → There are nesting. Nest1 F F E…E, Nest2 F…F F F E E E E, the bigger tc will be counted.

Minded that the third case will be easily ignored.

Finally, we output the maximum tc in each loop, and the question can be easily solved.

When I was solving the problem, I make a TLE mistake that I check the maximum of the list whenever it is updated. In fact, it

should be checked only once, otherwise, we will encounter a time complexity of O(n).