Q4

According to this question, we should find the subset of jobs that maximises profit. So we first sort all jobs in decreasing order of profit and denoted as array p. This step runs in $O(nlogn)$ by using MERGE SORT.

Assume we have an empty list R that can be split into n's slots that can store jobs sequence.

Then, we use a single for loop and iterate on jobs in decreasing order of profit. And for each job i, we try to find an empty slot I in R that satisfy that $d_i > I$ and I is the greatest slot in range 1 to d_i. And put this job i in I's slot in list R and mark this slot is not empty. If there is no empty space for job i, which means no empty slot in range 1 to d_i, we will ignore this job and not consider this job.

Imagine the worst case, for n jobs, the first job we found slot in one times, second job we found in twice ..., nth job found slot in n times, so the time complexity is $O((n+1)n/2)$ which is $O(n^2)$. So the total time complexity of this algorithm is $O(n^2)$.

Optimality:

Assume there are two jobs i and j, $p_i < p_j$, but we consider job i first:

If list R have enough space for jobs, there is no difference with our greedy solution as we can simply swap these two jobs and obtain our greedy solution. Otherwise if there is only one room for a job, clearly we should consider job i first so that we can make more profit.

Assume there are two jobs i and j, $p_i = p_j$, and $d_i < d_j$, but we consider job j first:

If list R have enough space for jobs, there is no difference with our greedy solution as we can simply swap these two jobs and obtain our greedy solution. Otherwise if only job j in list, and in a place which is in range 1 to d_i and there will be no space for job i, so it's trivial that if we put job j more closer to its deadline, it may give place for i, so that our greedy solution will be obtained and increase the total profit.

Thus, our greedy method solution is optimal.