(Q4) The problem is an extension of weighted interval scheduling dynamic programming problem.

Heart of the algorithm (for all the jobs on a particular day)

① $S[i][j]$ = max income generated when brother has access to 1 to i jobs & sister has access to 1 to j jobs (or vice-versa).

② $S[i][j] =$

① $S[i-1][j-1]$, when $i == j$ & jobs[i].valid $== 0$

② $\max(\text{jobs}[i].\text{pay} + S[p[i]][p[j]], S[i-1][j-1])$, where $i == j$ & childrenCount $>= 4$ & jobs[i].valid $== 1$

③ $\max(\text{jobs}[i].\text{pay} + S[p[i]][j-1], S[i-1][j-1])$, where $i == j$ & childrenCount < 4 & jobs[i].valid $== 1$

④ $S[i][j-1]$, when $j > i$ & jobs[i].valid $== 0$

(or) when $j > i$ & jobs[i].valid $== 1$ & jobs[i].childCount $>= 4$

⑤ $\max(\text{jobs}[j].\text{pay} + S[i][p[j]], S[i][j-1])$, when $j > i$ & jobs[i].valid $== 1$ & jobs[j].childCount < 4

"jobs[i].valid" means whether this job is b/w valid timings, not after 2300 & not before 0600

(3) Solution : S[n][n]

(Every days max-income will be at S[n][n], but the actual answer will be cumulative sum of all S[n][n] for every day there is some work).

(Q4) Complexity analysis

For ~~every day~~ jobs on every day, we compute a 2D-array to keep track of max income when both brother & sister are working & hence the complexity is $O(n^2)$.

Note: Repeating the same dynamic programming algorithm for each day we have a job won't increase the complexity of the algorithm.

The algorithm's complexity would only change by a constant factor = no. of days, which is a constant.

∴, overall complexity $\Rightarrow O(\text{constant} \times n^2)$

$\Rightarrow O(n^2)$