a) Optimal substructure.

Given $M$ days / $N$ paces.

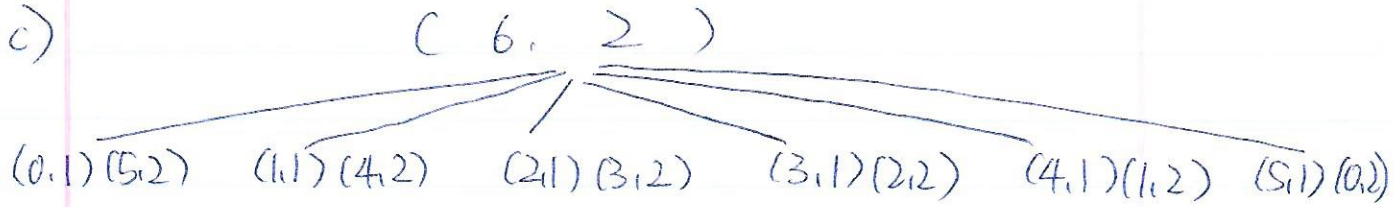
The trials have 2 different results.

1. The athelete is injured, then the athelete is unable to continue this day & unable to run any speed above this one.

   if the athelete is testing the $i^{th}$ speed & is injured, then this athelete has $(M-1)$ days left & $(i-1)$ speed to run (train).

2. The athelete is not injured, then this athelete can continue to run that day which means $(N-i)$ speed left to run (test).

c)




d) 12

e) $M * N$.

f). 1. Create a 2D array to store the result.
2. Base case.
3. Do the recursive call. if we get a call from the same state, we return it from the memory.

a). Find the cook who can do or sign up the most consecutive steps, then find the next cook who can do the most consecutive steps for the rest.

b). Find the cook who can do most amount of consecutive steps and move to the next cook who can do the most consecutive for the rest steps until no steps is available to schedule.

d). # cook $n$
# step $m$.
$O(m \cdot n \cdot m) = O(nm^2)$.

e). Prove optimal: suppose there exists an OPT has $l$ switch, and our algorithm has $k$ switch, $l < k$.
ALG : $A_1, A_2, A_3 \cdots A_k$.
OPT : $O_1, O_2, O_3 \cdots A_l$.
Let $i$ be the place where ALG & OPT first different.
$A_1 \cdots A_{i-1} = O_1 \cdots O_{i-1}$.
By design, we pick the cook who can do the most number of consecutive steps.
If we swap, we proceed for $A_k$ & $A_l$ the same way.
So there is a contradiction that the OPT solution could do better.

a). The Dijsktra's shortest Path algorithm can be applied to this problem. Find the shortest distance between the start Sth location to the destination Tth location. Since it is to find the shortest path between S & T, so when u == T, the loop ends. We also need to apply the wait time that we need to substract the Dates of arrive time and start time. of the next guest.

b) $O(V^2)$

c) Dijsktra's shortest Path algorithm.

d). I used all code from the genericShortest method, we are given the start location S (no change), and destination Tth location, so we need to stop when Tth location is reached instead of going through all vertex. Also the wait time ~~to~~ can be calculated from the existing method: getNextQuestTi & minutesBetween. that we need to add the wait time to the result, and also need to update the arrive time (start time the player intend to play) on the ~~next~~ next vertex.

e). $O(V^2)$.

According to Geeksfor Geeks, the Dijkstra's shortest path algorithm | Greedy Algo 7, If the input graph is representented using adjacency list, it can be reduced to $O(E \log V)$ with the help of binary heap.