1@ Optimal Substructure: We depend on whether or not an athlete gets injure on a particular day at a particular speed.

if we look at the last step meaning

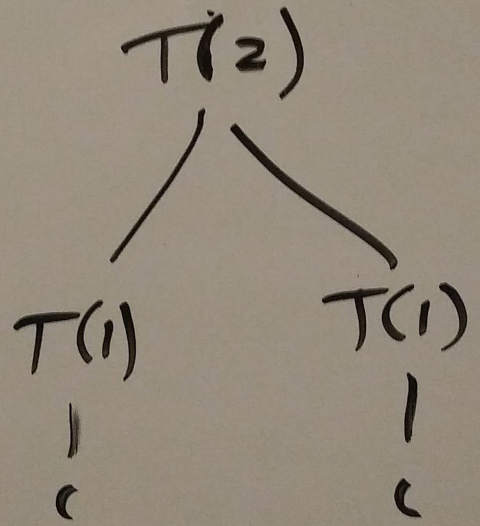
M = days left = 1 and N = speeds try so far = N - a

where $1 \leq a \leq N$

in other words: if the athlete make it to last day he/she most have try and pass atleast 1 speed test.

1 © # speeds = 6 # days = 2

Recurrence tree



where

c : The speeds try on that day.

1 (d) $6 \times 2 = 12$ sub problems

1 (2) $N \times M$ subproblems

1 (f) to memoize this problem we must store the solution to previous subproblems in a directory and only compute if solution not previously store

ex. store if an athlete can be sped up to $n \in \mathbb{N}$ without injure.

2. Thinking about the last step. 2 things can be true.

- the cook we looking at can do the final step in the recipe
- we need to switch to a different cook for last step.

* Also if a cook have already done a particular step (for a particular recipe) no further cook should do the same step.

20 (b)

Step 1. pick the cook who can do the most steps in the given recipe & assign him/her those steps.

Step 2. look at the missing steps to be done to complete the recipe and assign them to the next cook who can do the most out of the missing steps.

Step 3. repeat until no steps left for that recipe.

2(d) $O(n^2)$

Since in my version I have to look up/write to every cell in the 2D array.