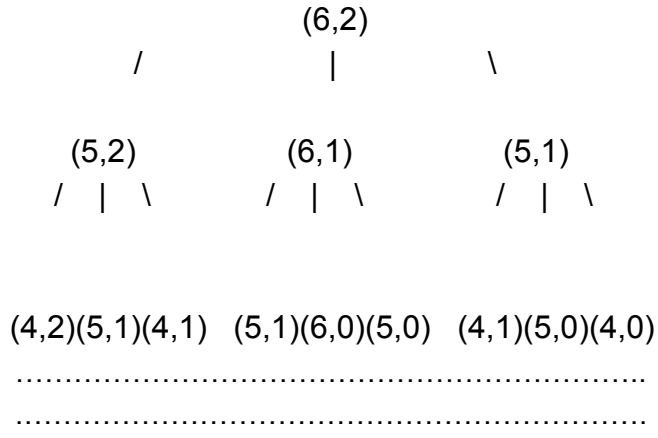
Running Trials & Tribulations

a) To solve for runTrialsRecur(N, M) we need to find the minimum number of trials able to run for runTrialsRecur(N-1, M), runTrialsRecur(N, M-1), and runTrialsRecur(N-1, M-1).

b)Coded

c)



They all reduced to all variations of (0,M),(N,0) where N=possibleSpeeds and M=days.

d) $6 \times 2 = 12$ *Subproblems*

e) $N \times M$ *Subproblems*

f) Create an array A[possibleSpeeds][days]. Set your boundary conditions.

- A[0,....,possibleSpeeds][0] to 0s. Column 0 has all zeros.
- A[0][0,...,days] to 0s. Row 0 has all zeros.
- A[1,2,....,possibleSpeeds][1] to 1s. Column 1 has all ones.
- A[1][1,2,...,days] to 1s. Row 1 has all ones.
- Implement the recursive solution within a 2D array and save the min inside the array.

g)Code commented out

Holiday Special - Putting Shifts Together

a) Give a person with the most amount of steps they can do. Somehow save a record of the steps they can do. Assign them those steps. When you know that one person in the group has reached the maximum amount of steps that they can do, move on to the next person, assign them the steps that do not overlap with the previous cook, make a record of the steps they can do and repeat.

b)For the i^{th} person in M give the most j^{th} steps to the capacity of the i^{th} person (C_i), if at capacity then move forward to the $i^{th}+1$ person in M to give the $j^{th}+1$ steps up to their capacity (C_{i+1}). Display the steps each cook has to do.

League Of Patience

a) Implement Dijkstra's Shortest Path algorithm in genericShortest() but call getNextQuestTime() to calculate the time before you reach another node (quest).

b) Time complexity should be the same as genericShortest(), $O(V + V^2)$ and the getNextQuestTime() appears to take constant time.

c) The genericShortest method is implementing Dijkstra's Shortest Path algorithm.
d) Create containers to hold the current time, the wait time to be able to move on.
e) The current time complexity of "genericShortest" given V vertices and E edges is $O(V + V^2)$.
According to Introduction To Algorithms the time complexity could be improved by changing the data structure used to build the min priority queue. So by building the implementing the min priority queue with a Fibonacci heap the runtime would be $O(V \log V + E)$.
f)Not coded
g)Not coded