# Writeup

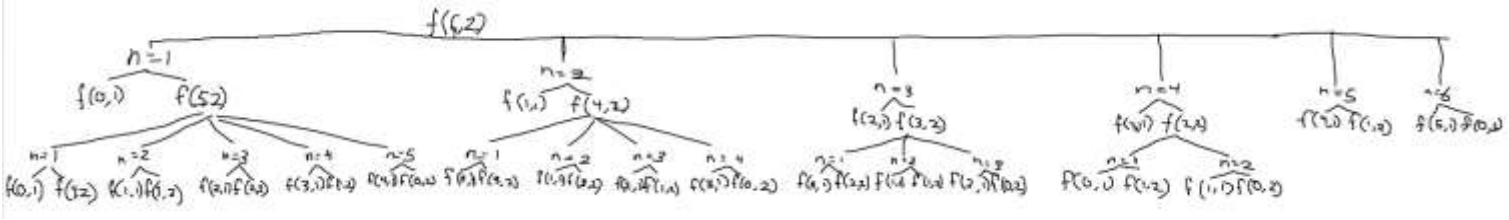
*Omar Mirza Fall 2019*

## 1. Running Trials and Tribulations

(a) We continually split the problems into two subproblems (safe trial or risky trial) in the code these problems are represented as

```
runTrialsRecur(i-1, days-1) // Risky trial
runTrialsRecur(possibleSpeeds-i, days) // Safe trial
```

(c)



(d) 6*2 = 12 subproblems

(e) N*M = NM subproblems

(f) We save our previous answers in a 2d array, then at the beginning of the next problem we check the 2darray if we already solved it.

## 2. Holiday Special - Putting Shifts Together

(a) The optimal solution to our main problem is composed of optimal solutions of smaller subproblems such as most consecutive chef for a smaller number of steps.

(b) Find the chef with the most consecutive chefs, choose them, repeat but for the remaining steps.

(d) O(nm) where n = num of chefs and m = num of steps

(e) Let ALG = {a1, a2, …, aj} be the solution generated by my algorithm and let OPT = {o1, o2, …, ok} be an optimal solution to this problem where k < j. Let i be the first switch where ai is not equal to oi. By design of our greedy algorithm, we can produce a list with the least number of switches between the group of chefs up till j switches. We can therefore use the cut and paste method to cut out oi and paste in ai. We can replace oi with ai all the way up to k, which would make the optimal solution {a1, a2, …, etc} So the optimal solution can be replaced with our greedy algorithm at every step which means our algorithm is optimal.

## 3. League of Patience

(a) For this problem I would adapt the Bellman-Ford Algorithm we discussed in class

(b) The time complexity for Bellman-Ford is O(VE)

(c) The genericShortest method is implementing dijkstra's algorithm

(d) We need to change the code such that the durations are updated dynamically. We also need to increment the time according to how long it takes to wait and reach each node.

(e) Dijkstra's algorithm is O(V lg V + E lg V). The optimal algorithm and data structure is Dijkstra's algo using fibonacci heap which is O(V lg V + E)