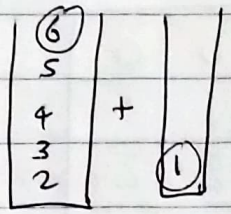
1335: Minimum Difficulty of a Job

Job Difficulty = [6, 5, 4, 3, 2, 1] , d = 2

- Arrange jobs in the "d" days, in such a way that total difficulty is minimum.
- Each day should have atleast one job.
- total difficulty in a day would be the hardest job on that day. (And job order should be preserved) 1 cannot be done before 6



⇒ Total difficulty = 1 + 6 = 7

Typical Question :- arrange n things at k places with some constraint

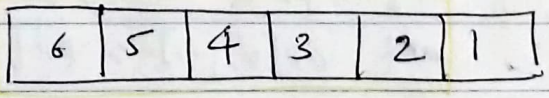
Brute-force approach

→ Start from beginning

0-index

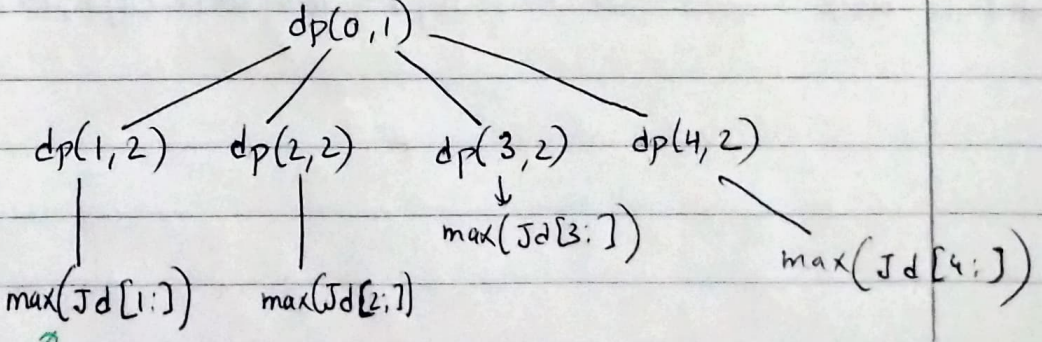
dp(0, 1) ← first day

dp(i, day)



d = 2

maximum (upto 5) can be put on day 1 since last day would require 1 job.

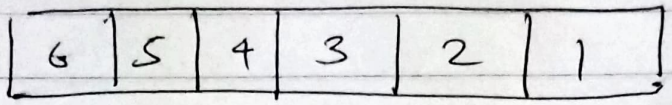


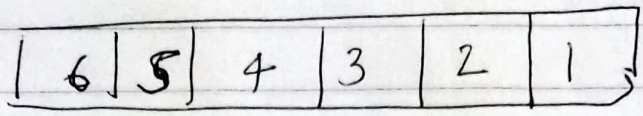
Now day = 2 (== d)

that means all elements (jobs) after that should be done on this day

dp(i, day) → starting from dp(0, 1)

→ Base case : (day == d)





starting from reverse direction ←
get the maximum at each position. [max(jD[pos:]

So that if case like dp(2, 2)
∴ That means we cannot go ahead since d == day, return maxim job difficulty from that position to end.

→ Recurrence relation:

$$dp(i, day) = \min\left(hardest + dp(j+1, day+1),\ dp(i, day)\right)$$

$$hardest = \max\left(jobDifficulty[j],\ hardest\right)$$

where $\boxed{i \le j \le length - (d - day)}$,

Using this technique with memorization would solve the problem.

- Another approach would be start from the end (day = d) and go upto day = 1.
(Since converting this one to bottom-up approach would be less difficult)