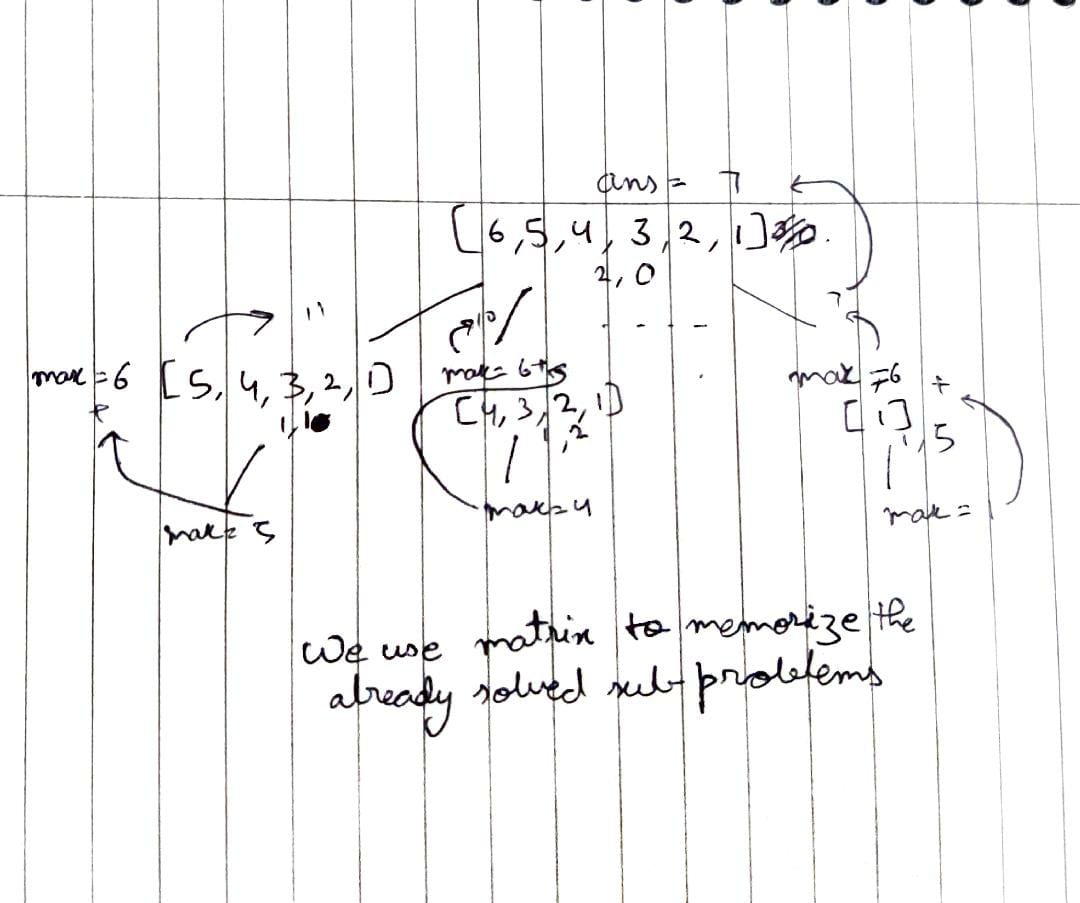
Question: https://leetcode.com/problems/minimum-difficulty-of-a-job-schedule/

The question seemed to be too difficult because it took me long to understand the question, so I have simplified the question;

1. Cut array into d subarrays with min of 1 element and calculate the sum of max of each subarray and return the minimum of such sum possible.

So the idea is to check all possible combinations.

We do it using depth first search where we calculate the minimum sum which can be generated from indx pos by making d cuts.



Code:  
class Solution {

public int minDifficulty(int[] jobDifficulty, int d) {

if(jobDifficulty.length<d) return -1;

int dp[][] = new int[d+1][jobDifficulty.length];

for(int[] p:dp) Arrays.fill(p, -1);

return dfs(jobDifficulty, dp, d, 0);

}

public int dfs(int[] jobDifficulty, int[][] dp, int d, int indx){

if(d==1){

int max = 0;

while(indx<jobDifficulty.length) max=Math.max(max, jobDifficulty[indx++]);

return max;

}

if(dp[d][indx]!=-1) return dp[d][indx];

int max=0, res=Integer.MAX\_VALUE;

for(int i=indx; i<jobDifficulty.length-d+1; i++){

max=Math.max(max, jobDifficulty[i]);

res=Math.min(res, max+dfs(jobDifficulty, dp, d-1, i+1));

}

return dp[d][indx]=res;

}

}

Github Link :<https://lnkd.in/ecwtJeaz>