Agenda:

Target Sum

Min Jumps

N digit number

2.5 hrs → 5 Ques

Contest → 8th Feb Thursday

Discussion → 9th Feb

Pass → ≥ 60% (3 Ques)

R1	R2	R3
2 Days	9 Days	20 Days

5 Ques → 1 DP, 1 Graphs

Placement

1. Mock Interview
2. Contest

] 1 month

Revision ← PSP ≥ 70%.

5 classes → Feb

1. Given an integer array A of size N and B. Find whether their exist a subset in A whose sum equal B.

A = [3, 34, 4, 12, 5, 2]

B = 9 (T) <4, 5> <3, 4, 2>

B = 30 (F)

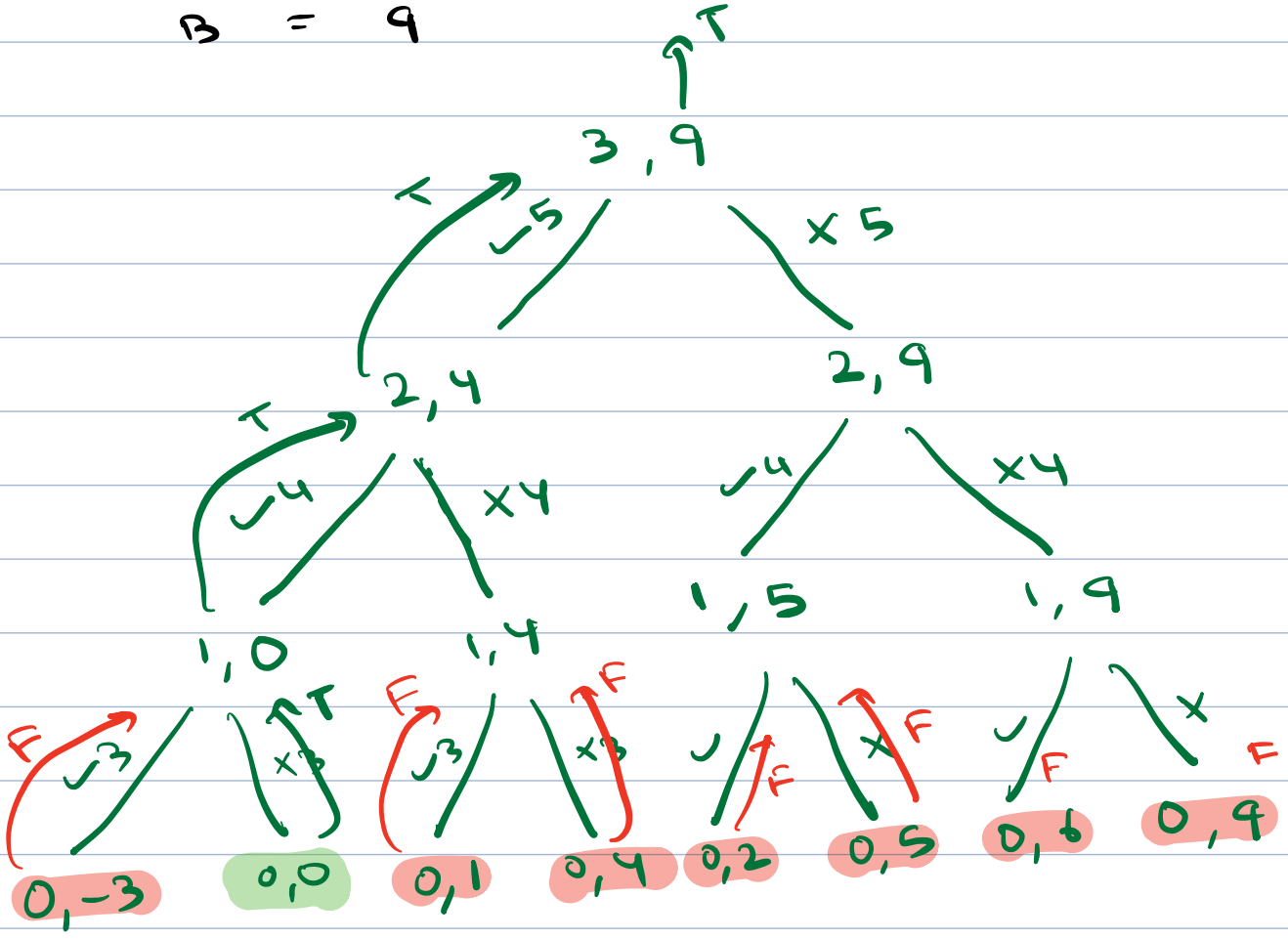
BF : Go to all subsets

$n \rightarrow 2^n$

$3 \rightarrow 2^3$

A = [3 4 5]
    0 1 2

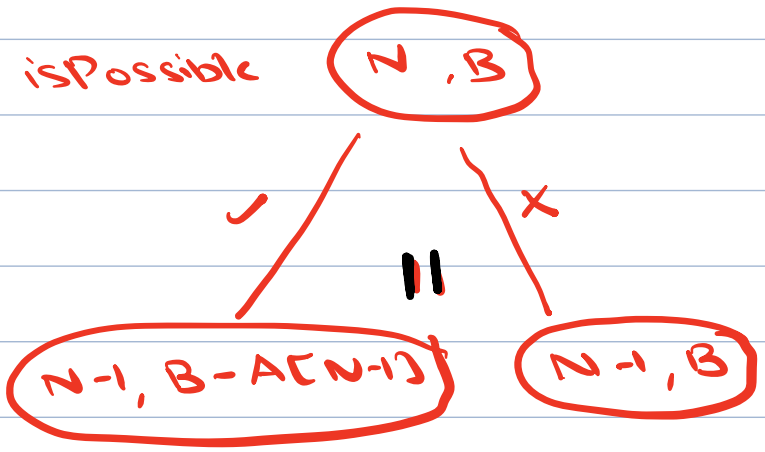
B = 9



TC: $O(2^N)$

A : [0 1 . . . . N-1]

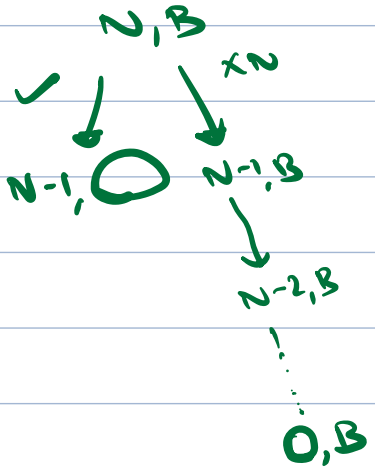
B :



int dp [N+1] [B+1] = {-1}

dp[N] [B]

int ar[]




```
bool isPossible ( int N, int B ) {
    if (B == 0)   return true
    if (B < 0)    return false
    if (N == 0)   return false

    if ( dp [N] [B] != -1 )   return dp[N][B]

    dp [N] [B] = ( isPossible ( N-1 , B- ar[N-1]) ||
                   isPossible (N-1 , B) )

    return dp [N] [B]
}
```

TC : O(N * B)

SC : O(N * B)

RS → min (N, B)

A = <3, 1, 4, 9, 7, 2>

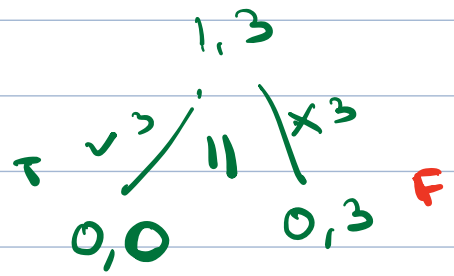
B = 6

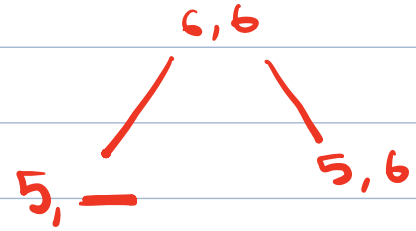
N = 6
B = 6

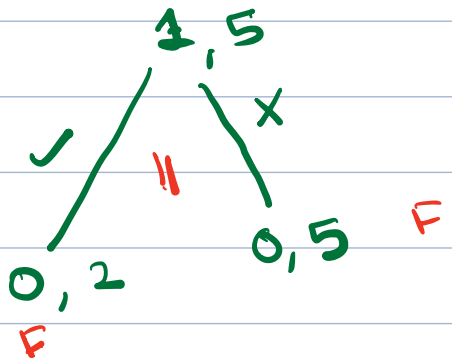
dp [7][7]

sum →

cnt		0	1	2	3	4	5	6
	0	T	F	F	F	F	F	F
3	1	T	F	F	T	F	* F	F
1	2	T	T	F	T	T	F	F
4	3	T			*		*	
9	4	T						
7	5	T						
2	6	T						*



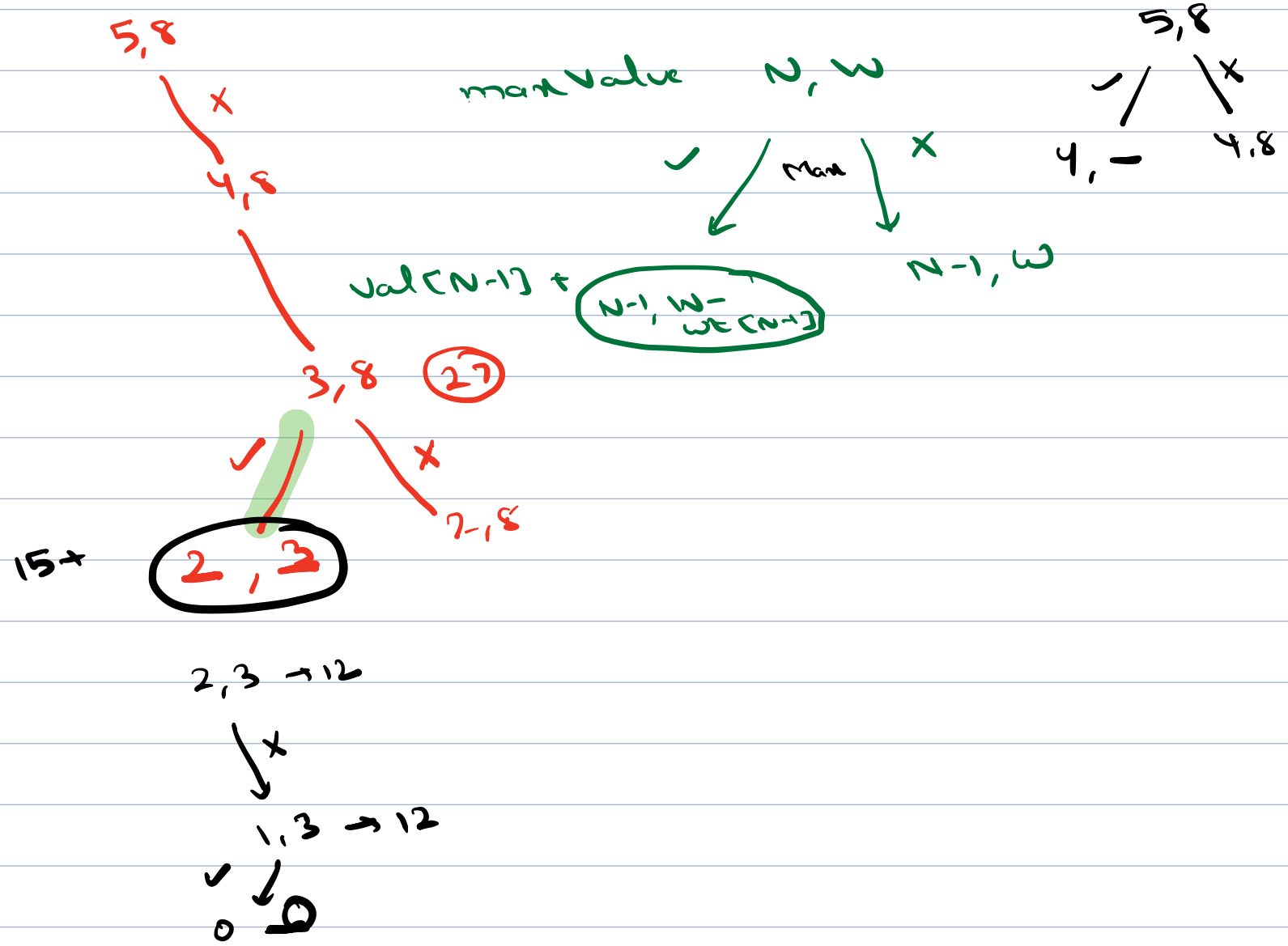




	0	1	2	3	4
N = 5	Wt	3	6	5	2	4
W = 8	V	12	20	15	6	10

capacity / W →

val	wt	cnt ↓	0	1	2	3	4	5	6	7	8
		0	0	0	0	0	0	0	0	0	0
12	3	1	0	0	0	12	12	12	12	12	12
20	6	2	0	0	0	12	12	12	20	20	20
15	5	3	0	0	0	12	12	15	20	20	27
6	2	4	0	0	6	12	12	18	20	21	27
10	4	5	0	0	6	12	12	18	20	22	27



dp[N+1][W+1]

// dp array

i = N , j = W

while (i > 0 && j > 0) {

dp (i, j)

x

dp(i-1, j)

if (dp[i][j] == dp[i-1][j]) {

i--

}

else { // itm elem selected → idx i-1

print (i-1)

j -= wt[i-1]

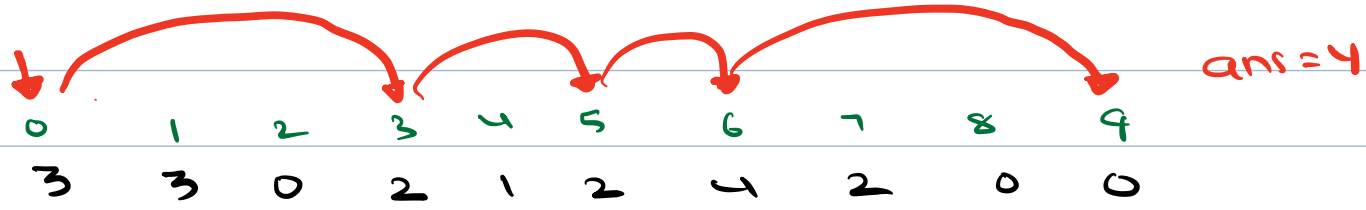
i--

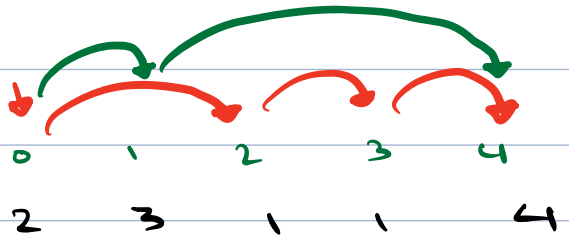
}

}

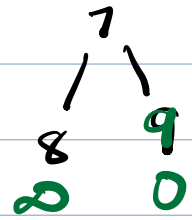
10:26

2. Given an integer array nums of length N. Initially, we're at 0, return min. no. of jumps to reach n-1.



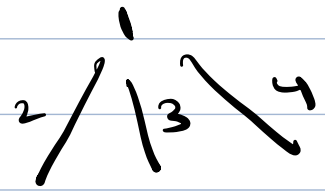


ans = 2

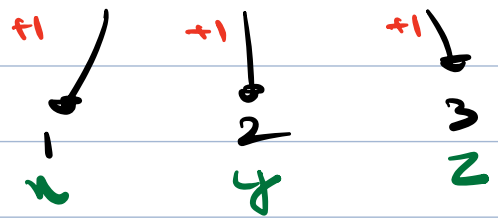


	0	1	2	3	4	5	6	7	8	9
	3	3	0	2	1	2	4	2	0	0
dp[]	4	4	∞	3	3	2	1	1	∞	0

dp[N]



minJumps(0) → min(x, y, z)



dp[i] → Min jumps from ith idx to reach the end

i i+1 i+2 i+3

3

```
int dp[n]
dp[n-1] = 0

for (i = n-2 ; i ≥ 0 ; i--) {

    int minsteps = INT_MAX

    for (j = i+1 ; j ≤ min(N-1, i + A[i]); j++) {

        minsteps = min(minsteps, dp[j])

    }

    if (minsteps != INT_MAX)
        dp[i] = minsteps + 1
    else
        dp[i] = minsteps

}

return dp[0]
```

TC : $O(N^2)$

SC : $O(N)$

↓

TC : $O(N)$

SC : $O(1)$

3. Given N and S, find N digit numbers with sum S (sum of N digits)

N = 2 S = 4

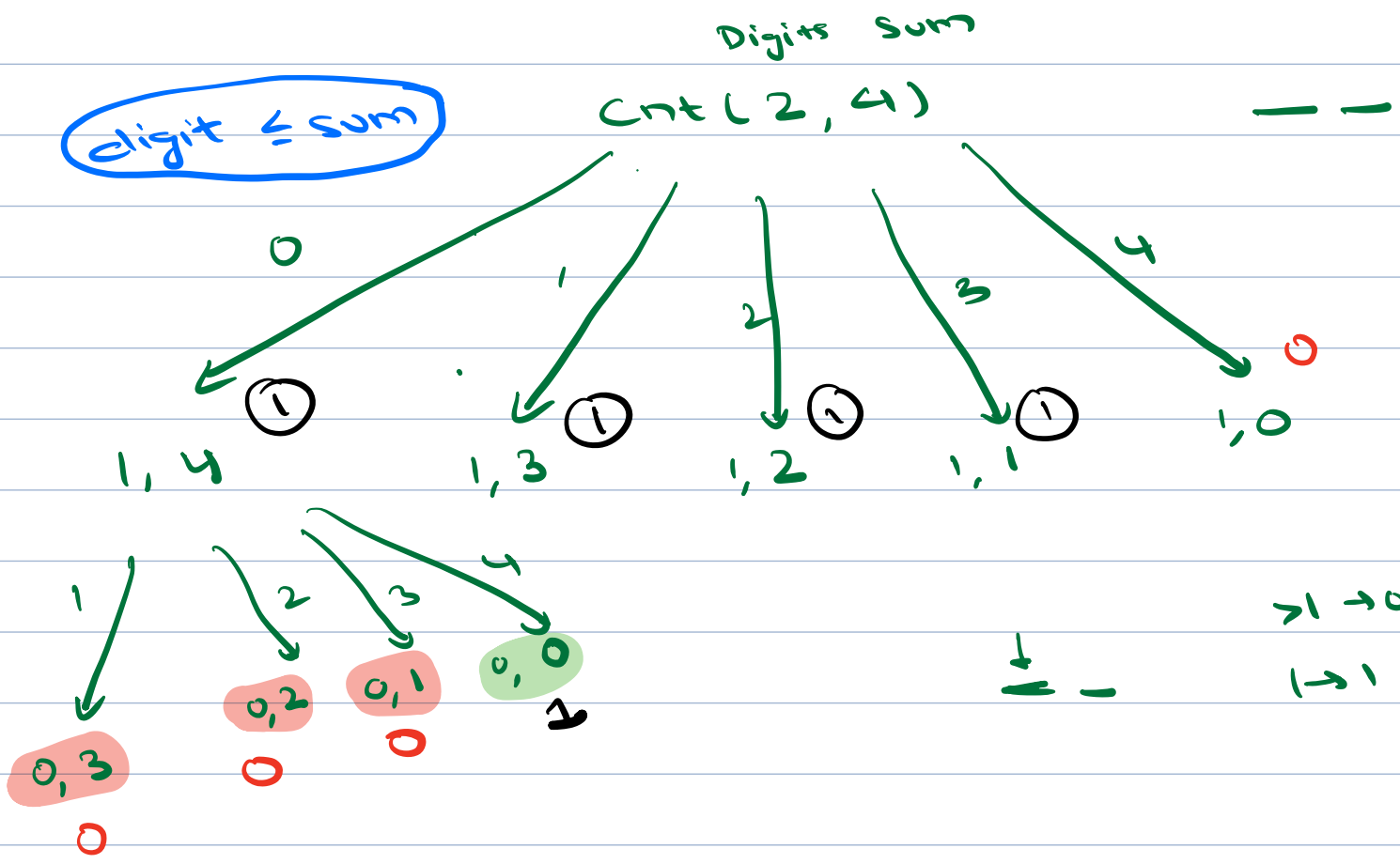
13, 31, 22, 40

ans = 4

N = 1 S = 3

3

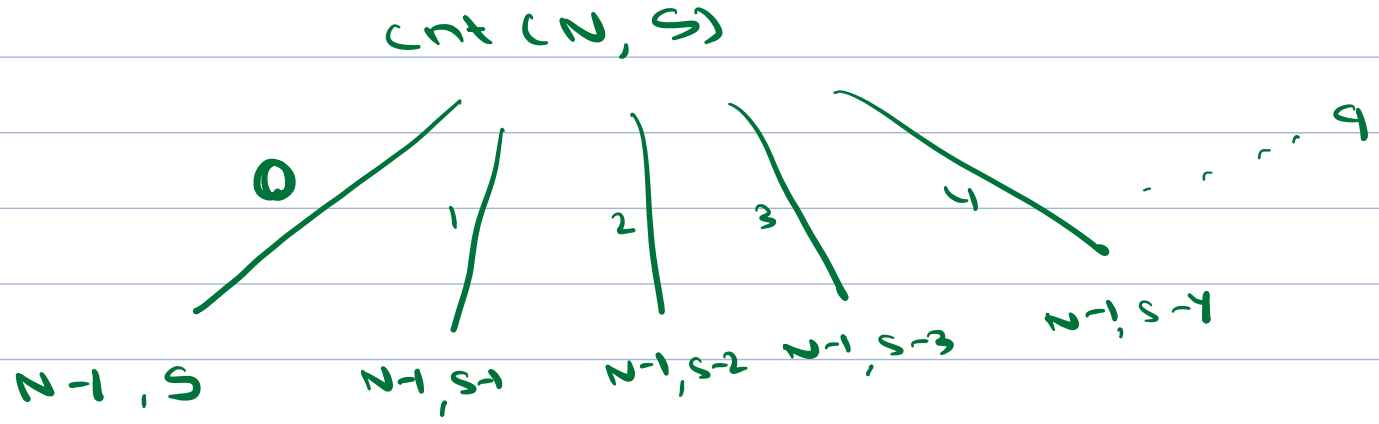
ans = 1

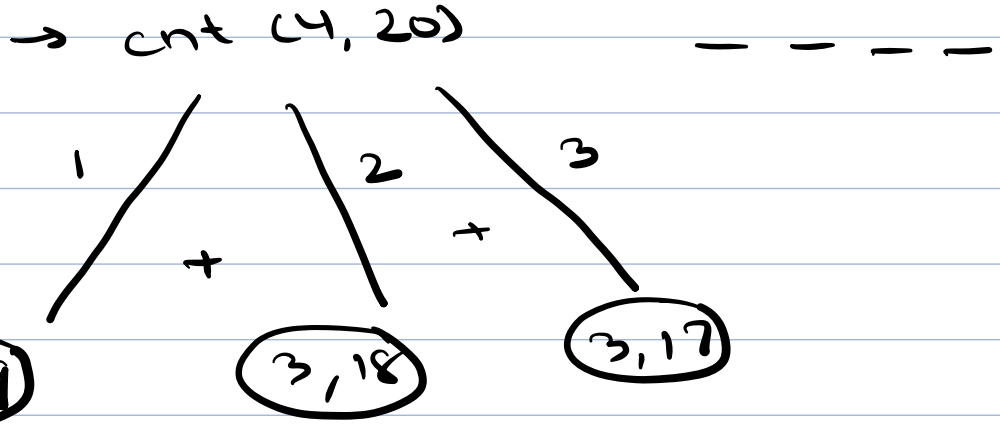


sum = 0 → digit = 0 True

→ digit = 0 False

digit = 0 → sum > 0 False





19 ← ---

⑤

점화식 → dp[N][S]

```
int dp[N+1][S+1] = {-1}

int cnt(int N, int S) {
    if (N == 0 && S == 0)   return 1
    if (N == 0 || S == 0)   return 0

    if (dp[N][S] != -1)   return dp[N][S]
    int ans = 0
    if (N == 1) {
        for (d = 1 ; d <= 9 ; d++) {
            if (d <= S)
                ans += cnt(N-1, S-d)
        }
    }
    else {
        for (d = 0 ; d <= 9 ; d++) {
            if (d <= S)
                ans += cnt(N-1, S-d)
        }
    }

    dp[N][S] = ans
    return ans
}
```

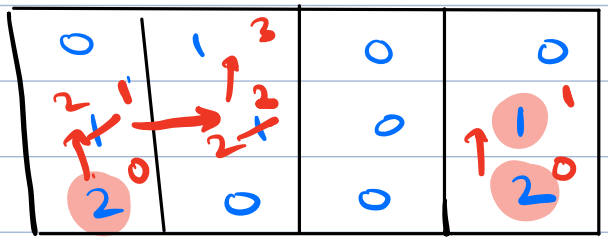
TC : O(N*S)

SC : O(N*S)

4. Rotten Oranges

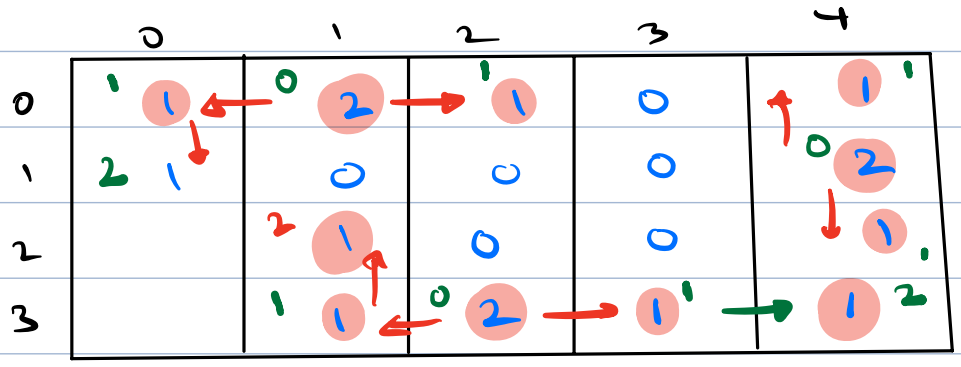
mat[N][M] → 0 empty
→ 1 fresh
→ 2 rotten

Every minute any fresh orange adjacent to a rotten orange becomes rotten, find min time when all oranges become rotten.
If not possible, return -1.

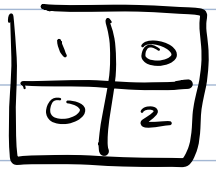


T = 3

3 × 4



T = 2



T = -1

	0	1	2	3	4
0	2	2	2	0	+2
1	2	0	0	0	2
2	0	+2	0	0	+2
3	0	+2	2	+2	+2

Multi source BFS

(0,1,0) (1,4,0) (3,2,0) (0,0,1) (0,2,1)

(0,4,1) (2,4,1) (3,1,1) (3,3,1)

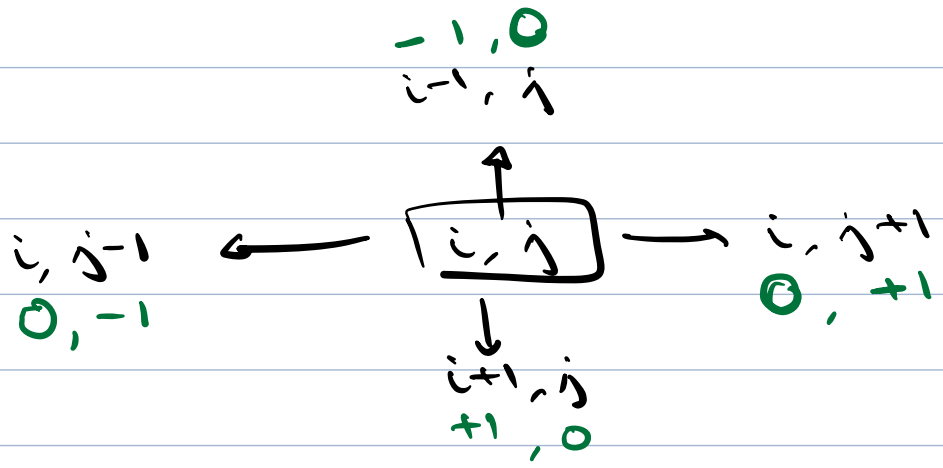
(1,0,2) (3,4,2) (2,1,2)

```
class triplet {
    int i, j, time
};

queue < triplet > q

for (i = 0 ; i < N ; i++) {
    for (j = 0 ; j < M ; j++) {
        if (mat[i][j] == 2)
            q.enqueue({i, j, 0})
    }
}
```

```
int row[4] = <-1, 0, 1, 0>
int col[4] = <0, 1, 0, -1>
int maxtime=0
while(!q.empty()) <
    Triplet t = q.front
        q.dequeue()
    maxtime = max(maxtime, t.time)
      for(int nbr=0 ; nbr<4 ; nbr++)
        | int nbri = t.i + row[nbr]
        | int nbrj = t.j + col[nbr]
        >
      if(nbri ≥0  && nbri<n
       && nbrj ≥0 && nbrj<m
       && mat[nbri][nbrj]==1){
        | q.enqueue(<nbri, nbrj, t.time+1>)
          mat[nbri][nbrj] == 2
        >
    >
```




```
int row[4] = <-1, 0, 1, 0>
int col[4] = <0, 1, 0, -1>
             U  D  L  R
```