0/1 Knapsack !

Given N items. Each has a value & a weight associated with it. You also have a bag of capacity C. What is the maximum value of the items you can put in your bag.

The cumulative sum of weights in your bag cannot be greator than C.

You can either choose an item or ignore it.

Ex1 $N = 4$

W =	2	3	4	5	C = 7
V =	1	3	5	6	

⇒ ans = 8

Ex2

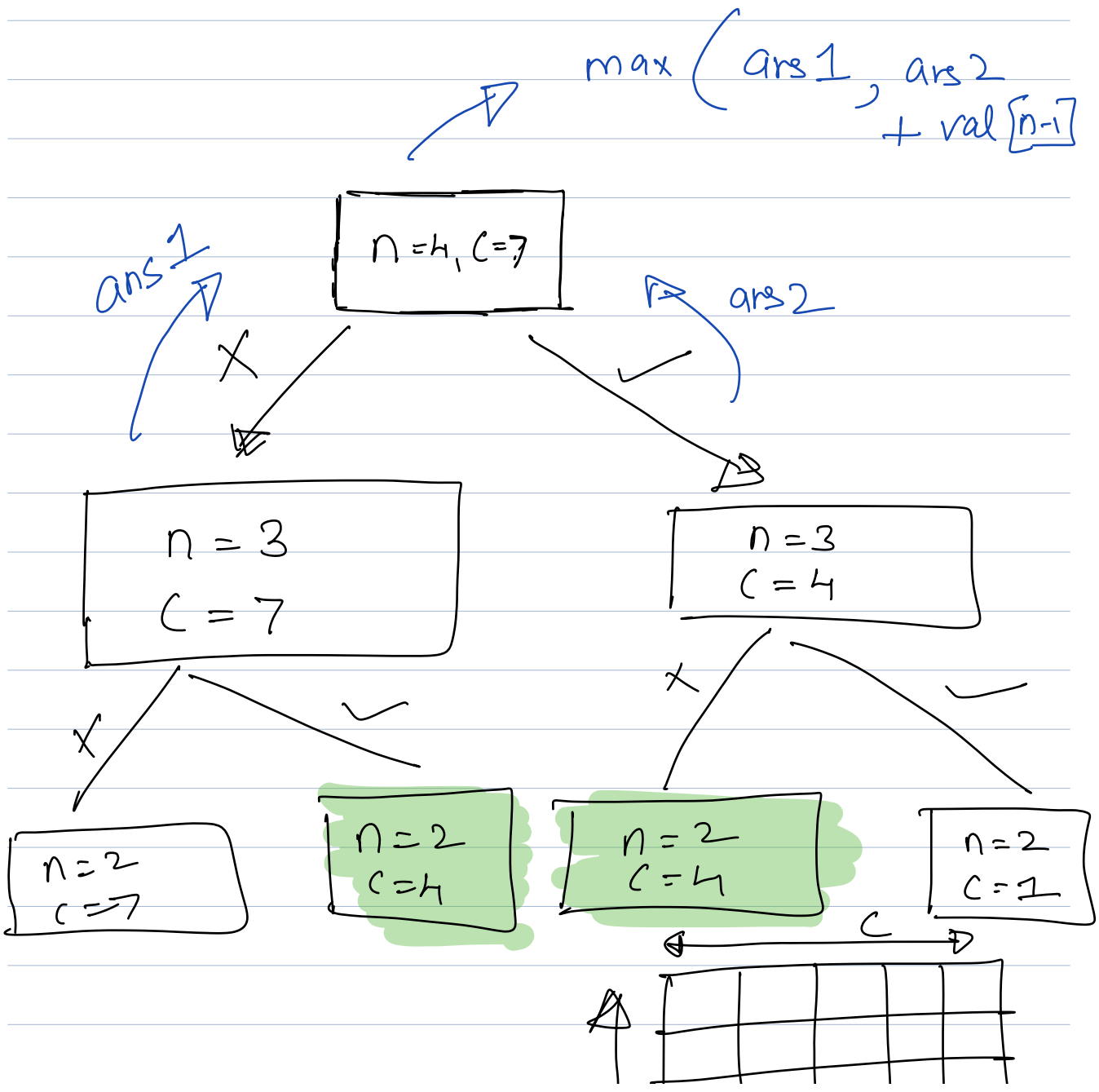
W =	6	5	5	C = 10
V =	19	10	10	
$\frac{V}{W}$ =	3.16	2	2	

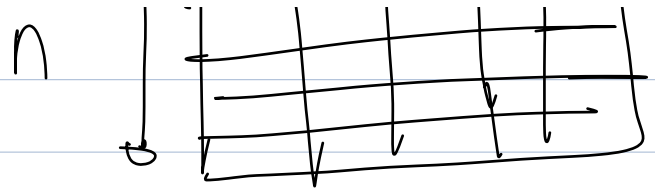
⇒ ans = 20

1 2 3 4

	0	1	2	3
$N=4$, $C=7$	$w =$	2	3	3	3
	v	1	3	5	6

Return max value using 1 to n items for capacity C.





func (int i, int j) ⇒ returns max value using (1 – i) items & j capacity.

func(int i, int j) max
- did not pick ith item → func (i-1, j)
- pick ith item → func (i-1, j - wt[i-1]) + val[i-1]

if (j ≥ wt[i-1])

Base Case ! if (i=0 || j=0) return 0

Time Complexity

No of States = $n \times C$

TC of a State = $O(1)$

TC $\Rightarrow$ No of State x TC of a State

TC $\Rightarrow O(n \times C)$

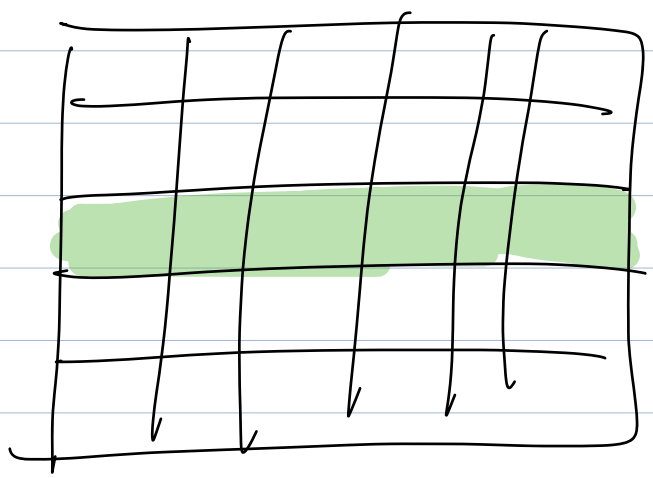
SC $\Rightarrow$ Stack Space + Storing of states

$O(n)$ $O(nC)$

$SC \Rightarrow O(nC)$

$SC = \max(n, C)$

$\Rightarrow O(C)$



```
int dp[n+1][c+1] = {-1};    int wt[]
                            int val[]

int maxValue (int i, int j) {
    if (i==0 || j==0)
        return 0

    if (dp[i][j] != -1) {
        return dp[i][j];
    }

    if (j >= wt[i-1]) {
        dp[i][j] = max (maxValue (i-1, j),
                        maxVal (i-1, j-wt[i-1])
                            + val[i-1]);
    } else {
        dp[i][j] = maxValue (i-1, j)
    }
    return dp[i][j];
}
```

0-N Knapsack / Unbounded Knapsack.

→ Same as last Question but a single item can be choosen multiple times.

Ex) $C = 12$ $N = 3$

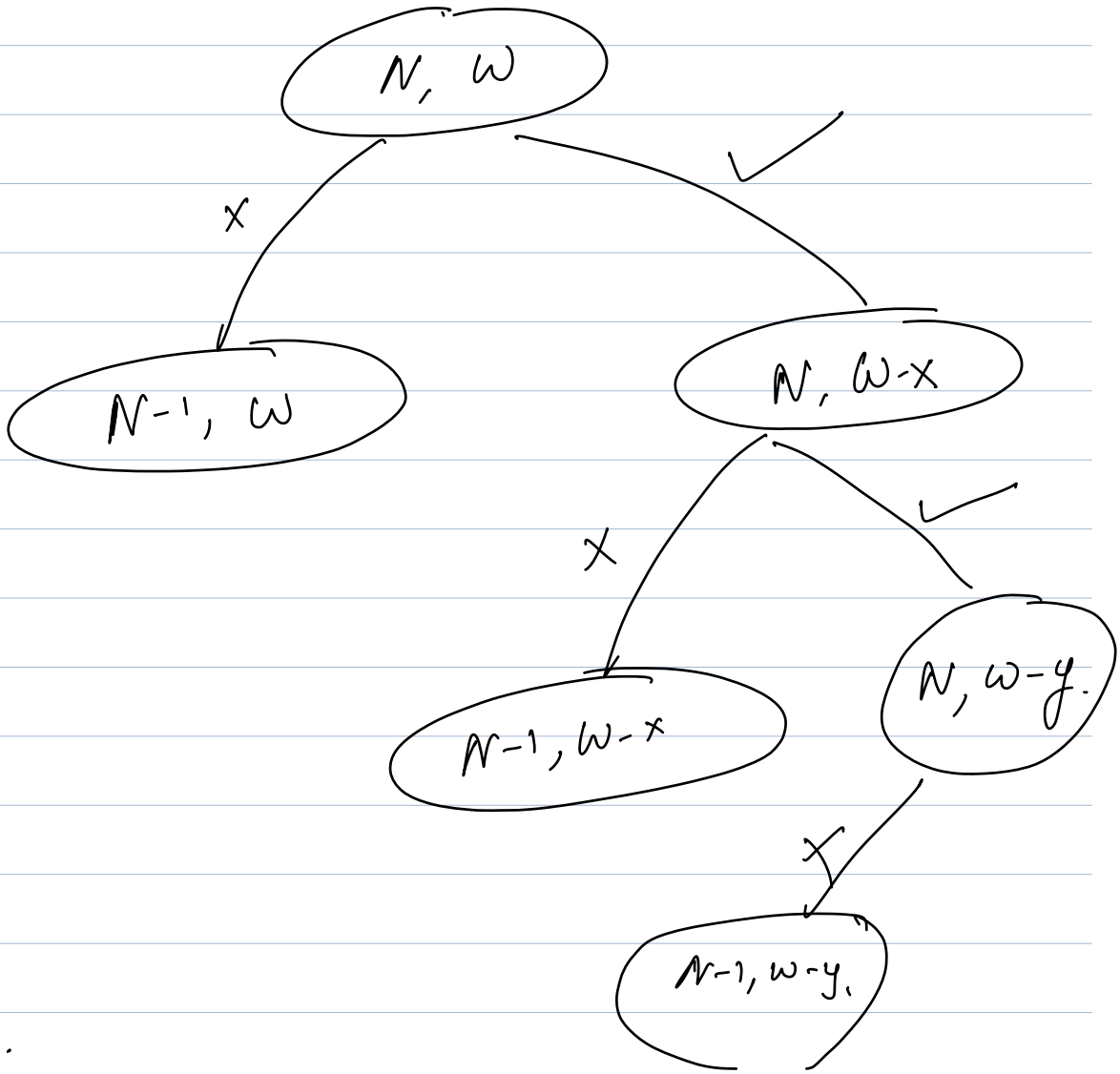
W	=	4	3	10
V	=	5	2	10

↳ take 3 times

ans ⇒ 15

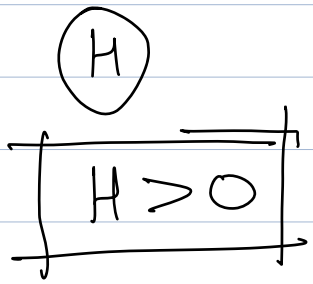
func(int i, int j) → max

- did not pick ith item → func(i-1, j)
- pick ith item → func(i, j - wt[i-1]) + val[i-1]


N, W
x
N-1, W
N, W-x
x
N-1, W-x
N, W-y
x
N-1, W-y

H=4

-3	4	-6
-10	-3	1



Pour →

↓

H ⇒ 6

-3	4	-6
-10	-3	1

-3	4	-6
-10	-3	1

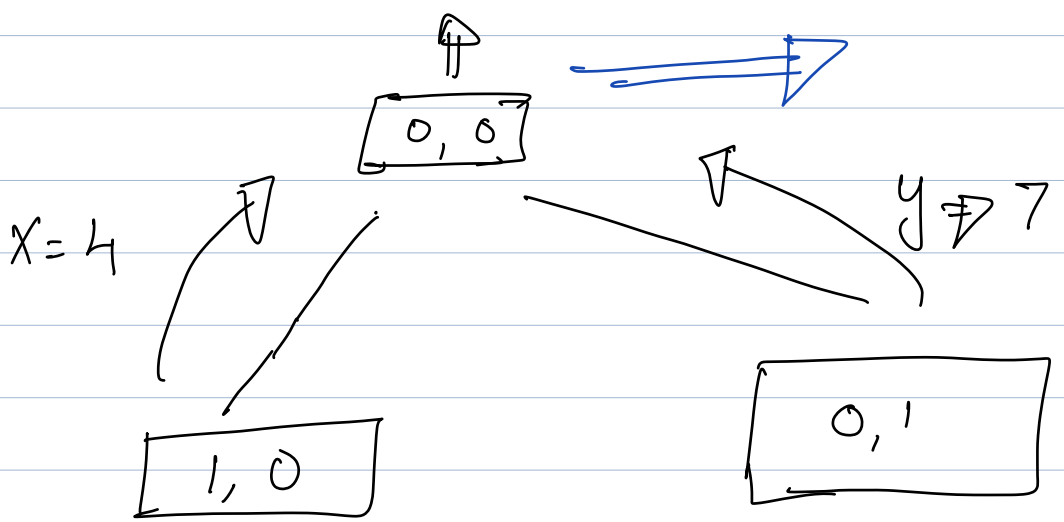
H ⇒ 17

[i,j] ⇒
return minimum
health to reach
princess from i,j.

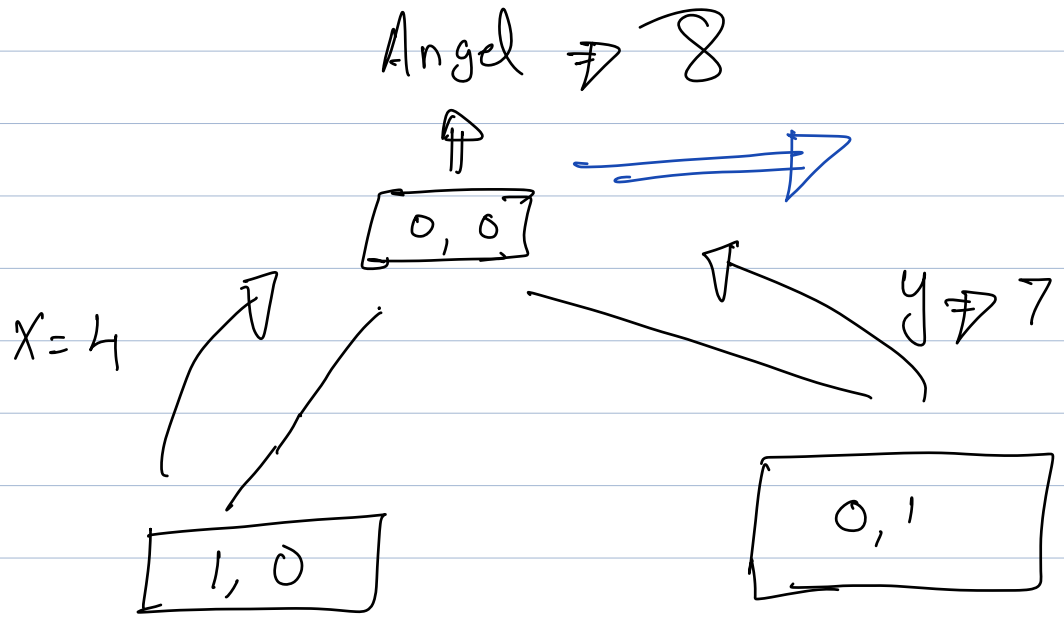
-3	4	-6
-10	-3	1

⇒ 8

Devil ⇒ -4.




```
if (arr[i][j] < 0) {
dp[i][j] = min(x, y) + abs(arr[i][j])
}
```



if (aa[i][j] > 0) {

int h ⇒ min(x, y) — aa[i][j]

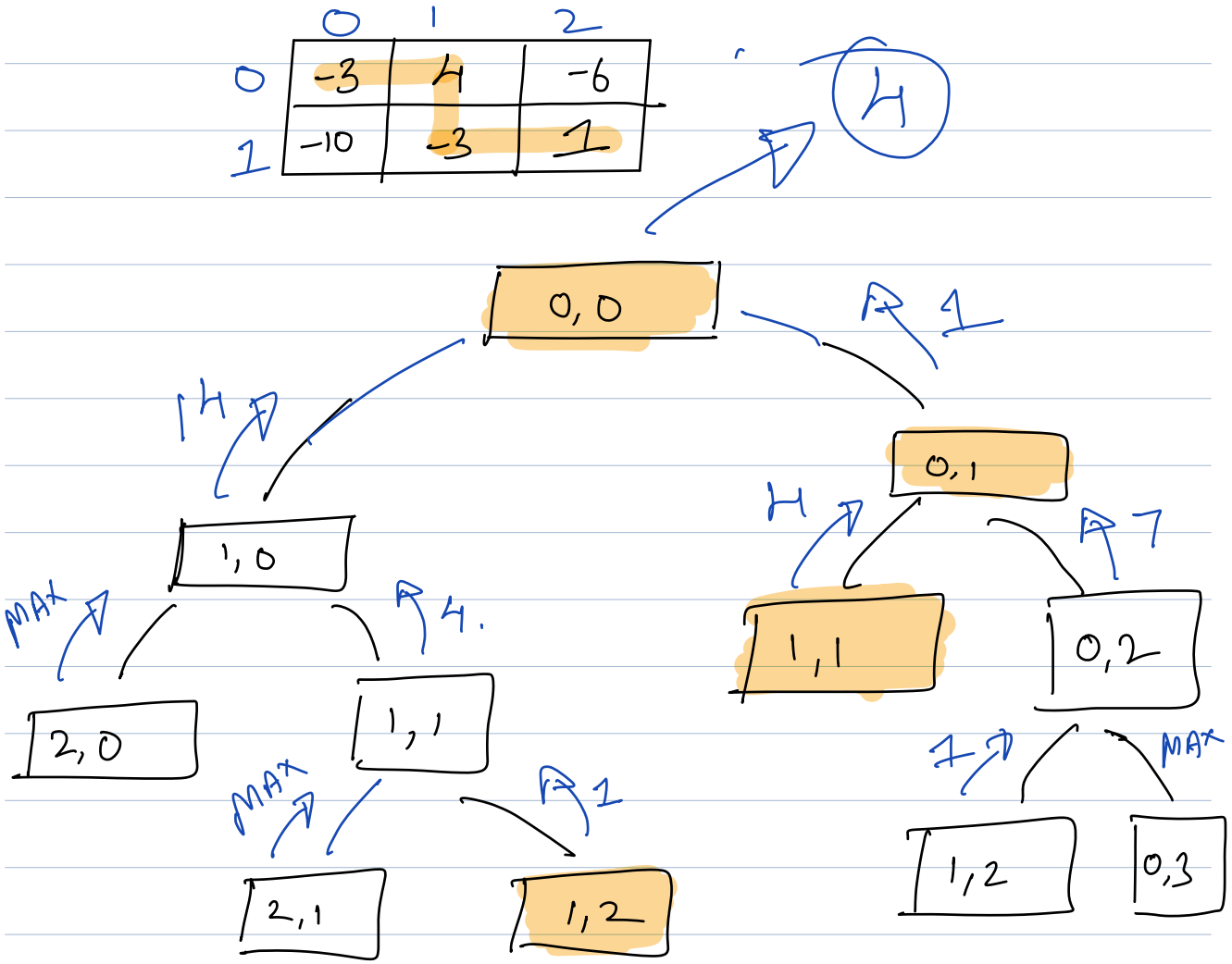
if (h ≤ 0)

dp[i][j] = 1

else

dp[i][j] = h

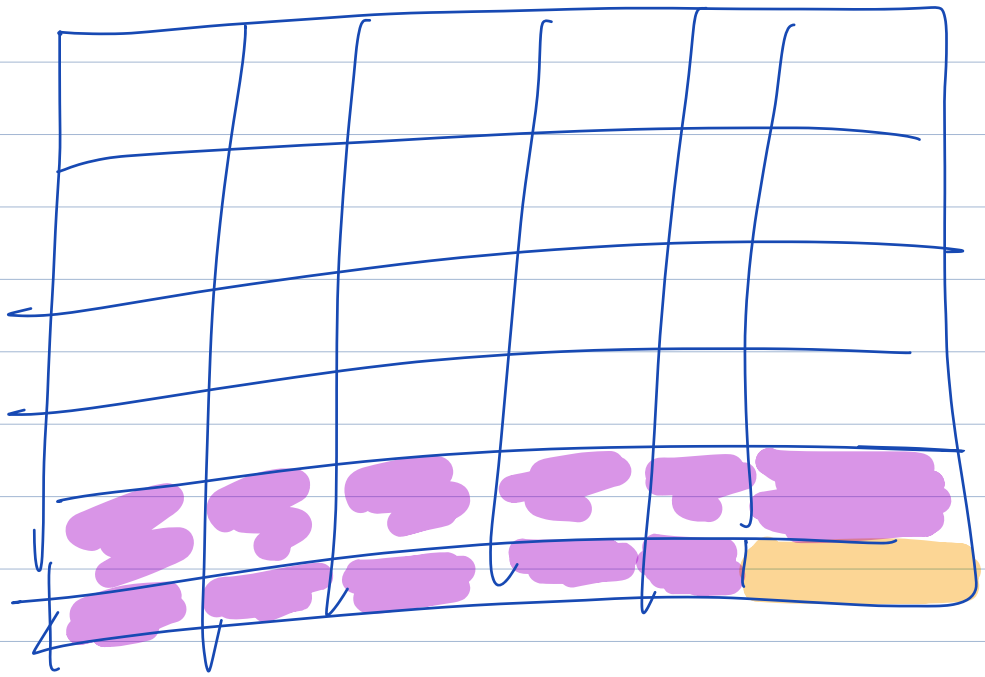
}

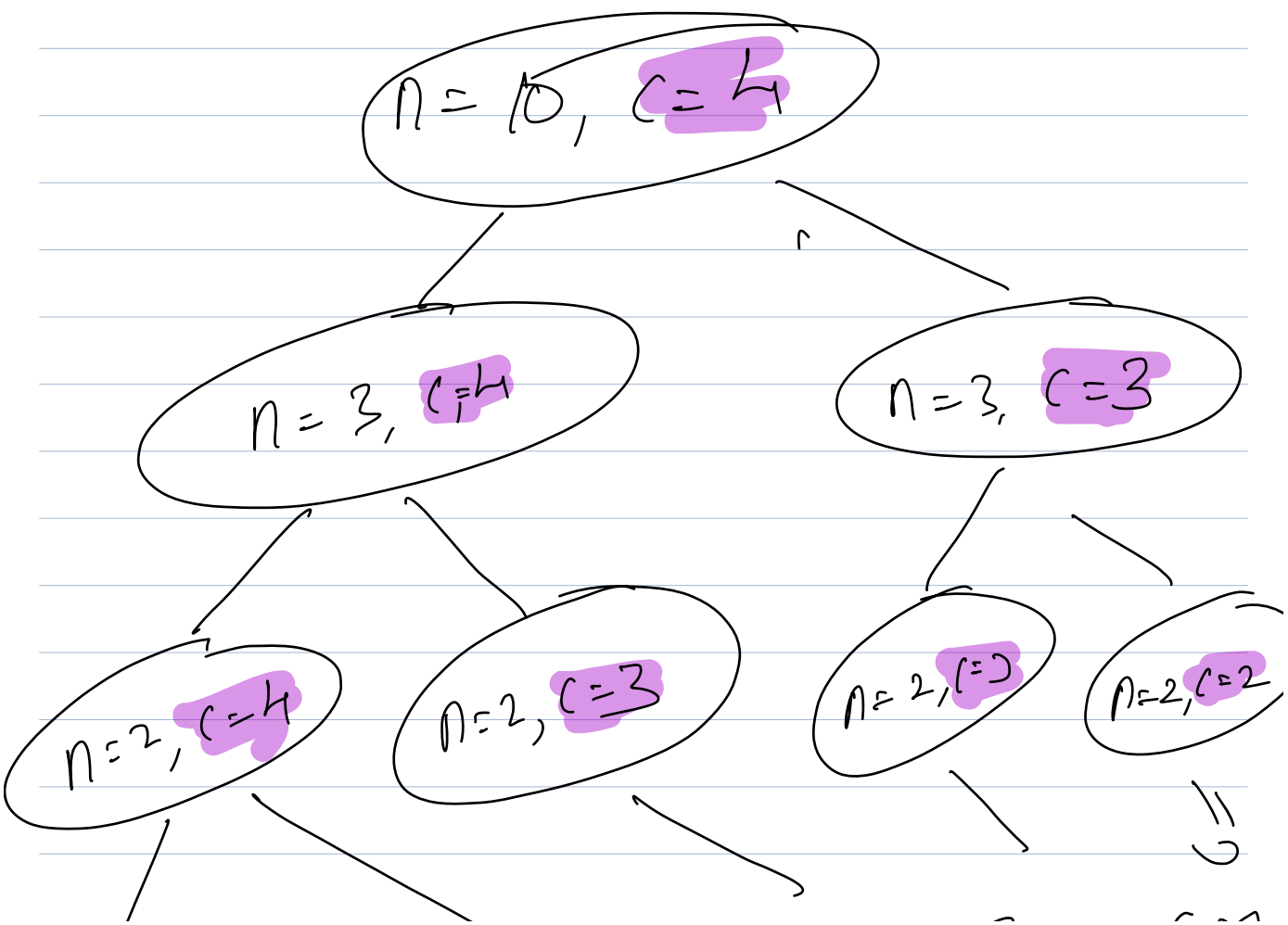


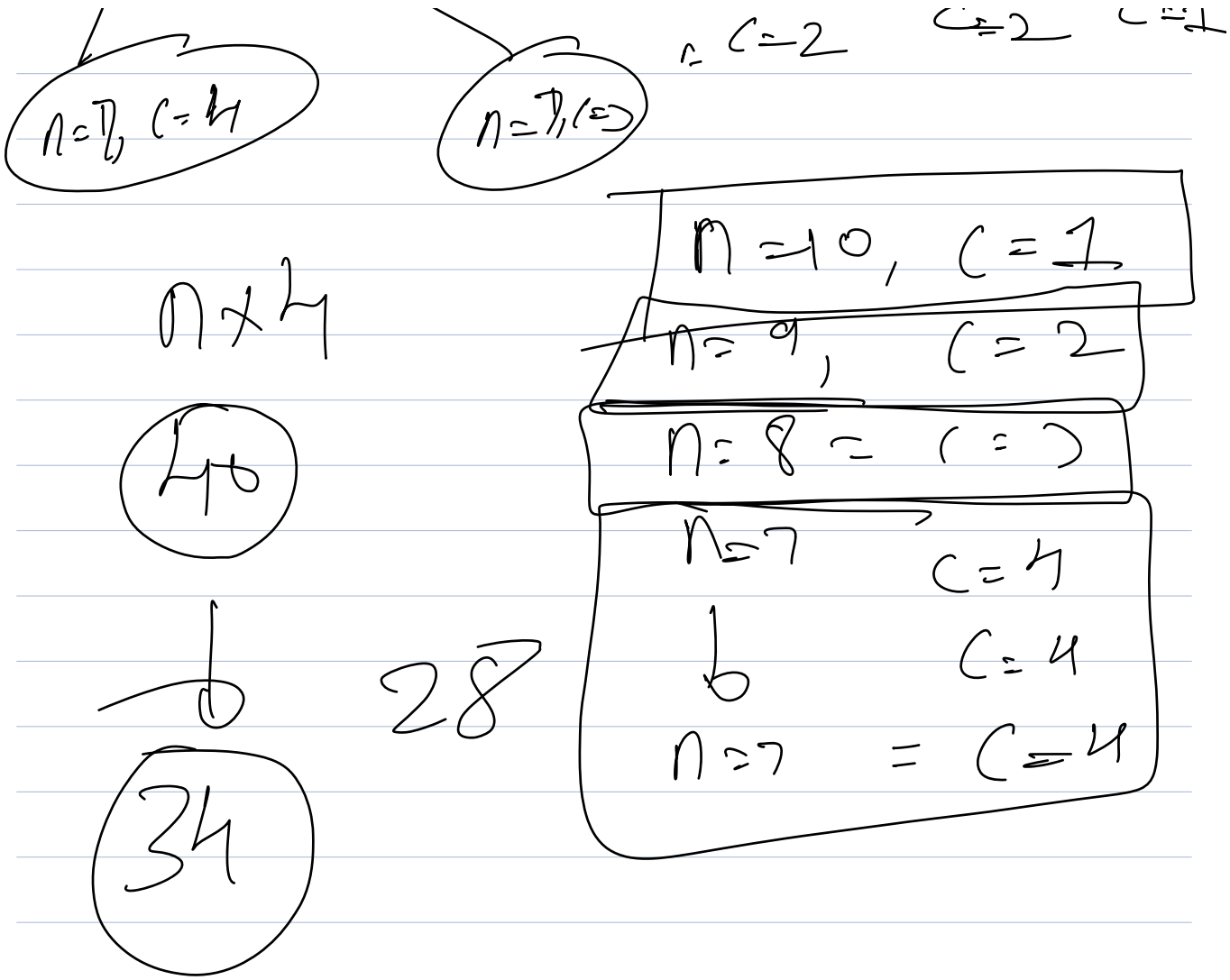
dp [m] [n]

Time Complexity = O(mn)

Space Complexity => O(mn)




n = 10, c = 4
n = 3, c = 4
n = 3, c = 3
n = 2, c = 4
n = 2, c = 3
n = 2, c = 3
n = 2, c = 2


n=7, c=4
n=7, c=3
∴ c=2
c=2
c=1
n=10, c=1
n=9, c=2
n=8 = (=)
n=7 c=4
c=4
n=7 = c=4
n×4
40
28
34