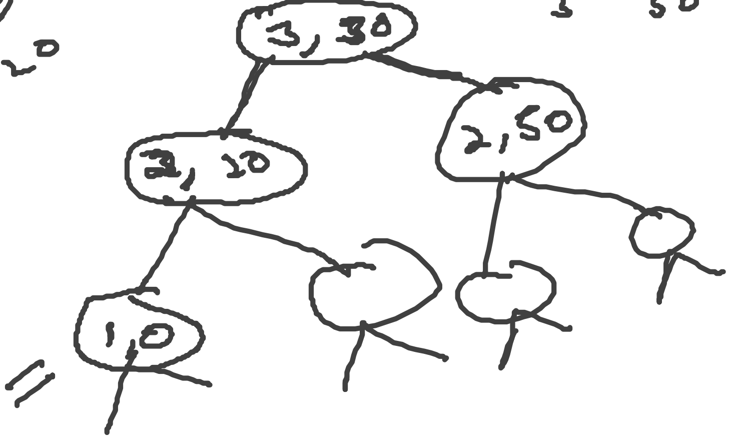
Day 12

12 January 2021 19:01

0/1 Knapsack

```
         0    1    2
Profit[] = {60, 100, 120}   P=0
                            120
wt[] : {  10, 20, (30)}
          0    1    2

W = 50

(50-10 = 40)

50-30=20

Max the profit = 220
wt[], val[], n, W
              3  50
```



Recur:

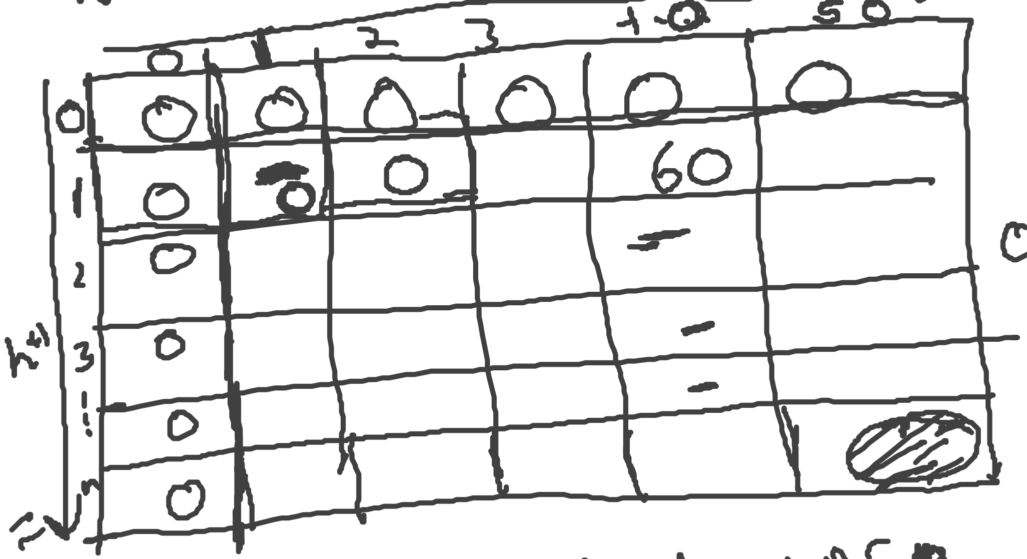
```
int Knapsack(int wt[], int val[], int W, n)
{

    if(n==0 || W==0)                         O(2^n)
        return 0;
      30    2        → 50
    if( Wt[n-1] <= W)
    {
        return max(val[n-1] + knap(wt, val,
                                W-wt[n-1], n-1,
                   knaps(wt, val, W, n-1));
    }
    else
    {
        return knaps(wt, val, W, n-1);
    }
int temp[n][W] = {-1};
int knaps(wt[], val[], W, W)
{

    if(n==0 || W==0)
        return 0
```

val[] = {60, 100, 120}

wt[] = {10, 20, 30}

W = 50

W+1 t[n][w]



60 + temp[

```
if(temp[n][w] != -1)
    return temp[n][w];
if(wt[n-1] <= w)
{
return temp[n][w] = ___ ;
}
else
  { return temp[n][w] = __ ;
```

O(n^2) 3

$t[1][1] = t[0][1]$

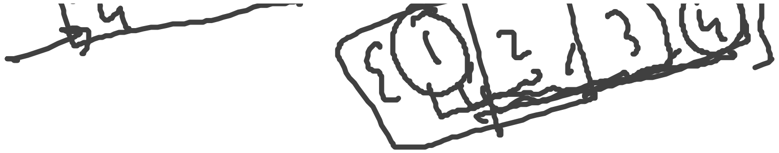
$t[1][10] = max(t[0][10], 60+t[0][0]$

max(val[n-1] + Knap(wt, val, n-1, w-wt[n-1]), Knap(wt, val, n-1, w));

wt = 2

wt[] = {1}

val[] = {10}



v d c 7 =

Values[]	weight[[]	index	0.	1	2	3	4	5	6	7. [Capaicty]
1	1	0	0	1	1	1	1	1	1	1
6	2	1	0	1	6	7	7	7	7	7
10 17	3	2	0	1	6.	10	11	16.	17	
16	5	3.	0	1	6	10	11	16.	17	22

Capacity = 3 and Index = 1 Math.Max(dp[0][3] , profit + dp[0][1])

Capcity = 4 , Index = 2 , Math.max(dp[1][4]. , profit[2] + dp[1][1]

Profit[i] + dp[i-1] [c-weights[i]]