knapsack -> Category.

Agenda :-

-> Rod cutting

-> coin change

-> 0-1 knapsack - 2 .

Agenda :- -> val -> wt

cap □

0-1 knapsack

↳ 1 item of 1 type

unbounded kp.

-> no limit on no of items .

n items

wt.

val

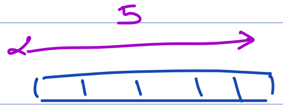
] cap .

Ques Rod cutting Problem

Given a rod of len n, and an array of len n, A[i] → Price of i len rod, find maxm val we can obtain by selling the rod.

N = 5,

$$A = \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \\ [0, & 1, & 4, & 2, & 5, & 6] \end{array}$$



2 + 3 → 6

4 + 1 → 6

2 + 2 + 1 → 9

3 + 1 + 1 → 4

1+1+1+1+1 → 5

5 → 6

Capacity → len of rod

wt → diff. lengths.

Price → —

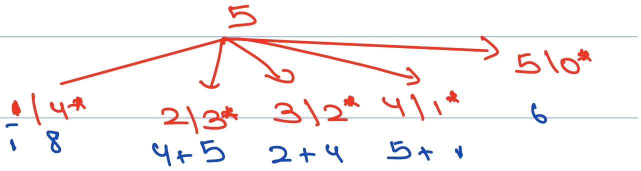
unbounded knapsack.

dp[i] → Max profit we can get by selling a rod of len i.

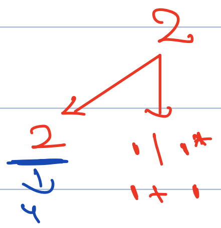
N = 5,

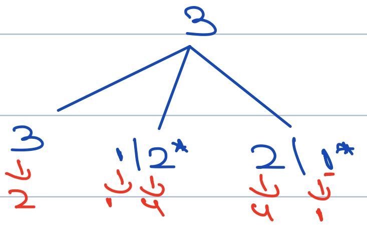
$$\begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \\ \end{array}$$

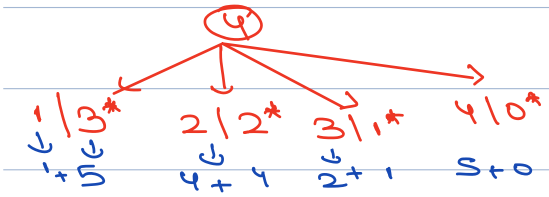
A = [0, 1, 4, 2, 5, 6]



	0	1	2	3	4	5
dp →	0	1	4	5	8	9
	-	1	2	1,2	2,2	1,2,2







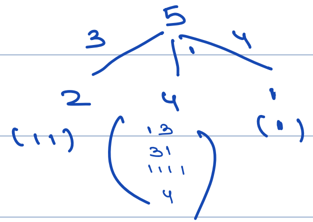
```
∀i   dp[i] = 0;                                T.C -> O(n^2)
                                                 S.C -> O(n)
            ⤻ 4
for   i -> 1 to  N
|                    ⤻ 1 to 4 -> 1,2 .
|        for  j -> 1 to i
|        |
|        |       dp[i] = max(dp[i], A[j]+dp[i-j]);
|        }                                  arr[1]+ dp[3]
}                                           arr[2]+ dp[2]
```

Ques Coin change Permutation $(x,y) \neq (y,x)$

k = 5 {1, 4} {4, 1} {1, 1, 1, 1, 1}

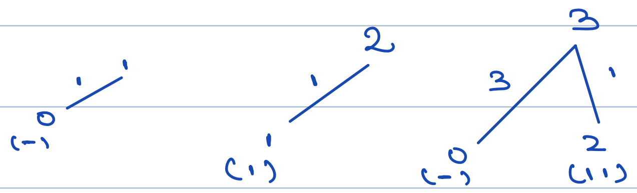
Ans = 6 {1, 1, 3} , {3, 1, 1} , {1, 3, 1}

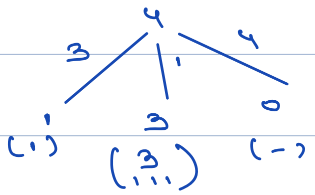
A = [3, 1, 4]



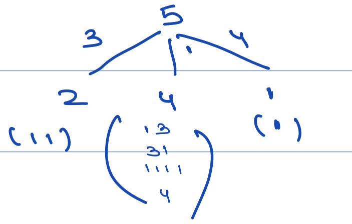
dp[i] → no. of permutations to pay i rupees.

	0	1	2	3	4	5
dp →	1	1	1	2	4	6
	-	1	11	3 111	13 31 1111 4	113 131 311 11111 41 14





A = [3, 1, 4]



```
int [] dp = new int [k+1];

    dp[0]=1;
                          i=s
    for (i=1; i<=k; i++) {
                           coins, j=0     3 Rupee
        for (j=0; j<n; j++) {
            if( i-arr[j]>=0) {
                dp[i]+= dp[i-arr[j]]
            }
        }
    }
```

T.C -> O(N*k)

S.C -> O(k)

Break   7:59 Am - 8:09 Am

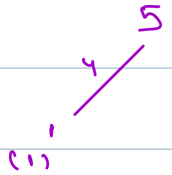
Ques

Ques Coin change Combination (x,y) = (y,x)

k = 5 {1,4} {~~4,1~~} {1,1,1,1,1}

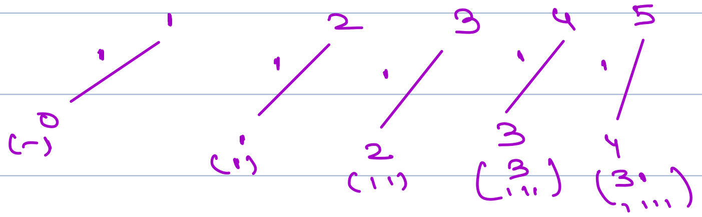
Ans = 3. {1,1,3} , {~~3,1,1~~} , {~~1,3,1~~}

A = [3,1,4]



dp[i] -> no of combinations to pay i rupees.

	0	1	2	3	4	5
dp ->	1	1	1	~~1~~ 2	~~0~~ 3	~~2~~ 3
	-	1	11	3 111	31 1111 4	311 11111 14



```
int [] dp = new int [k+1];
    dp[0]=1;
    for (i=1; i<=k; i++) {        order of these two.
        for (j=0; j<n; j++) {
            if( i-arr[j]>=0) {
                dp[i]+= dp[i-arr[j]]
            }
        }
    }
```

T.C -> O(N*k)

S.C -> O(k)

Ques 0-1 knapsack-2.

Given n items, with a wt & val. find max val which can be obtained by picking items s.t, total weight of all items $<= k$.

Note1 :- Every item can be picked only 1 time.

Note2 :- we can't pick partially.

$1 <= N <= 500$

$1 <= val[i] <= 50$

$1 <= wt[i] <= 10^9$

$1 <= cap <= 10^9$

T.C

↓

$O(n * cap)$

↓

500×10^9

$\boxed{5 \times 10^{11}}$

dp[i][j] → Max Profit we can get in a bag of cap j s.t we choose 0-i items.

↓

dp[n][k]

$1 <= N <= 500$

$1 <= val[i] <= 50$

$1 <= wt[i] <= 10^9$

$1 <= cap <= 10^9$

max profit

↓

$500 \times 50 \Rightarrow 25 \times 10^3$

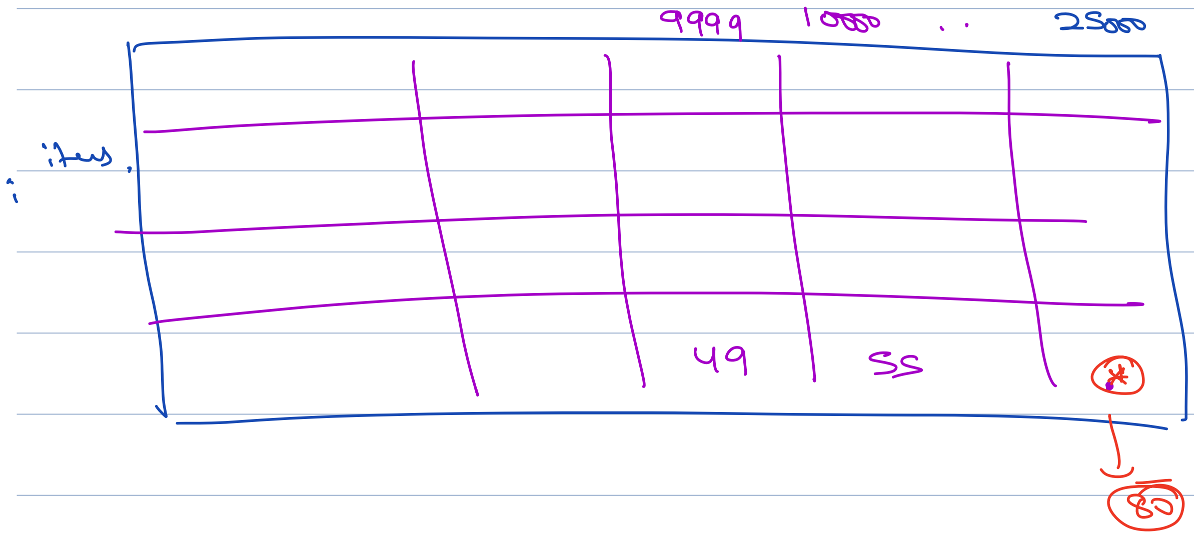
$\Rightarrow 25000$.

cap = 50.

dp[i][J] → min wt. Req. to get J profit

if we choose first i items.

J → profit.



wt	10,	20,	30
val	1000	500	50

dp[i][J] → min wt. Req. to get J profit

if we choose first i items.

J → profit.

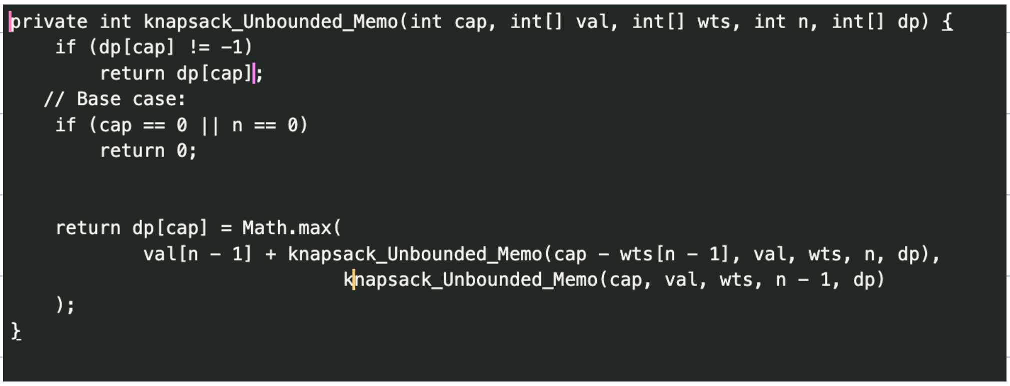
wt[] → 4 1 5 4 3 7 4

val[] → 2 1 2 1 2 1 2

Profit

cap = 6 .

val	wt	0	1	2	3	4	5
–	–	0	∞	∞	∞	∞	∞
2	4	0	∞	4	∞	∞	∞
1	1	0	1	4	5	∞	∞
2	5	0	1	4	5	9	10



```
private int knapsack_Unbounded_Memo(int cap, int[] val, int[] wts, int n, int[] dp) {
    if (dp[cap] != -1)
        return dp[cap];
   // Base case:
    if (cap == 0 || n == 0)
        return 0;


    return dp[cap] = Math.max(
            val[n - 1] + knapsack_Unbounded_Memo(cap - wts[n - 1], val, wts, n, dp),
                              knapsack_Unbounded_Memo(cap, val, wts, n - 1, dp)
    );
}
```