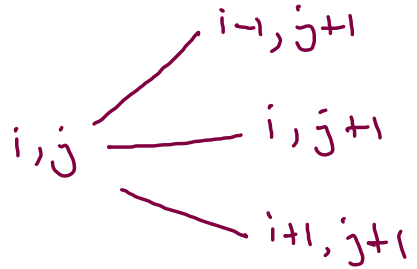
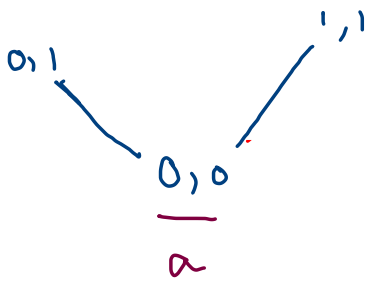
goldmine

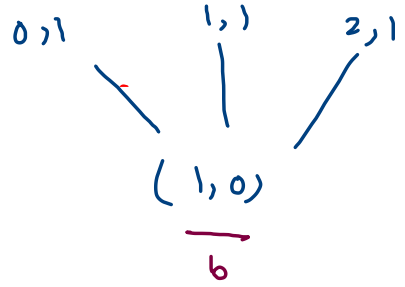
	0	1	2	3	4
0	10	5	8	19	7
1	6	4	20	8	13
2	3	20	30	10	5
3	6	12	26	14	6

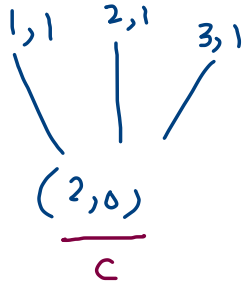
st -> 0^{th} cols

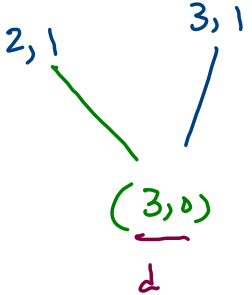


ans -> max (a,b,c,d)









	0	1	2	3	4
0	10	5	8	19	7
1	6	4	20	8	13
2	3	20	30	10	5
3	6	12	26	14	6

	0	1	2	3	4
0	67	57	40	32	7
1	79	57	52	21	13
2	76	73	53	23	5
3	79	65	49	20	6

dp[i][j] -> Max(dp[i][j+1], —
dp[i-1][j+1], /
dp[i+1][j+1]) \

\+ gold[i][j]

dp[i][j] -> max gold collected if we start digging from (i,j)

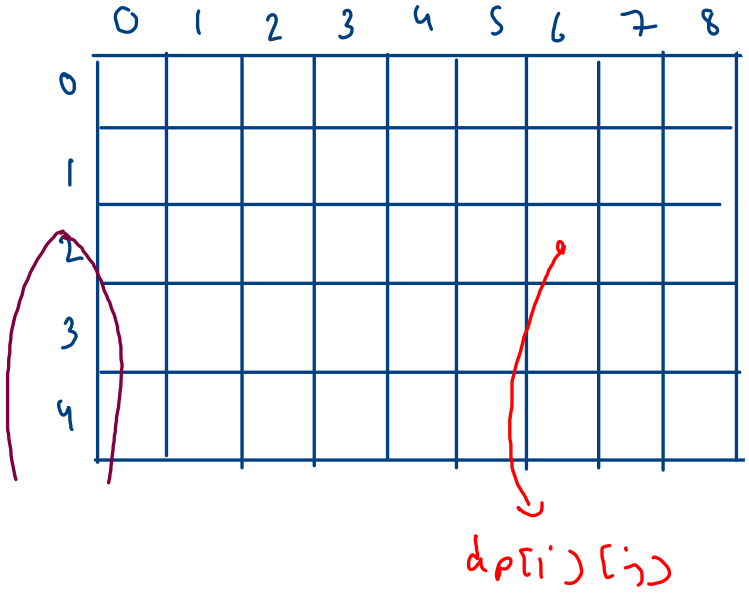
target sum subset

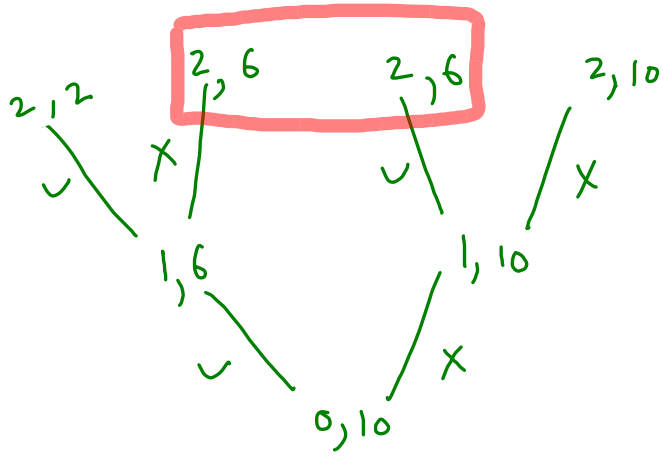
4	4	7	1	3
0	1	2	3	4

```
5
4
2
7
1
3
10
```

sb

target -> 8





2d

i -> 0 to 4

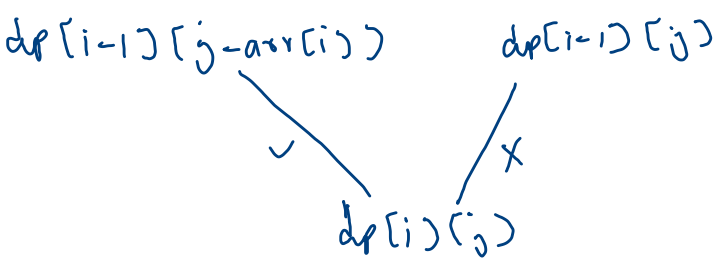
rem_tar -> 0 to tar

4	2	7	1	3
0	1	2	3	4

target = 11

	0	1	2	3	4	5	6	7	8	9	10	11
4 - 0	✓	X	X	X	✓	X	X	X	X	X	X	X
2 - 1	✓	X	✓	X	✓	X	✓	X	X	X	X	X
7 - 2	✓	X	✓	X	✓	X	✓	✓	X	✓	X	✓
1 - 3	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
3 - 4	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

dp[i][j] -> ij elements [0 to i] can create a subset of target 'j'



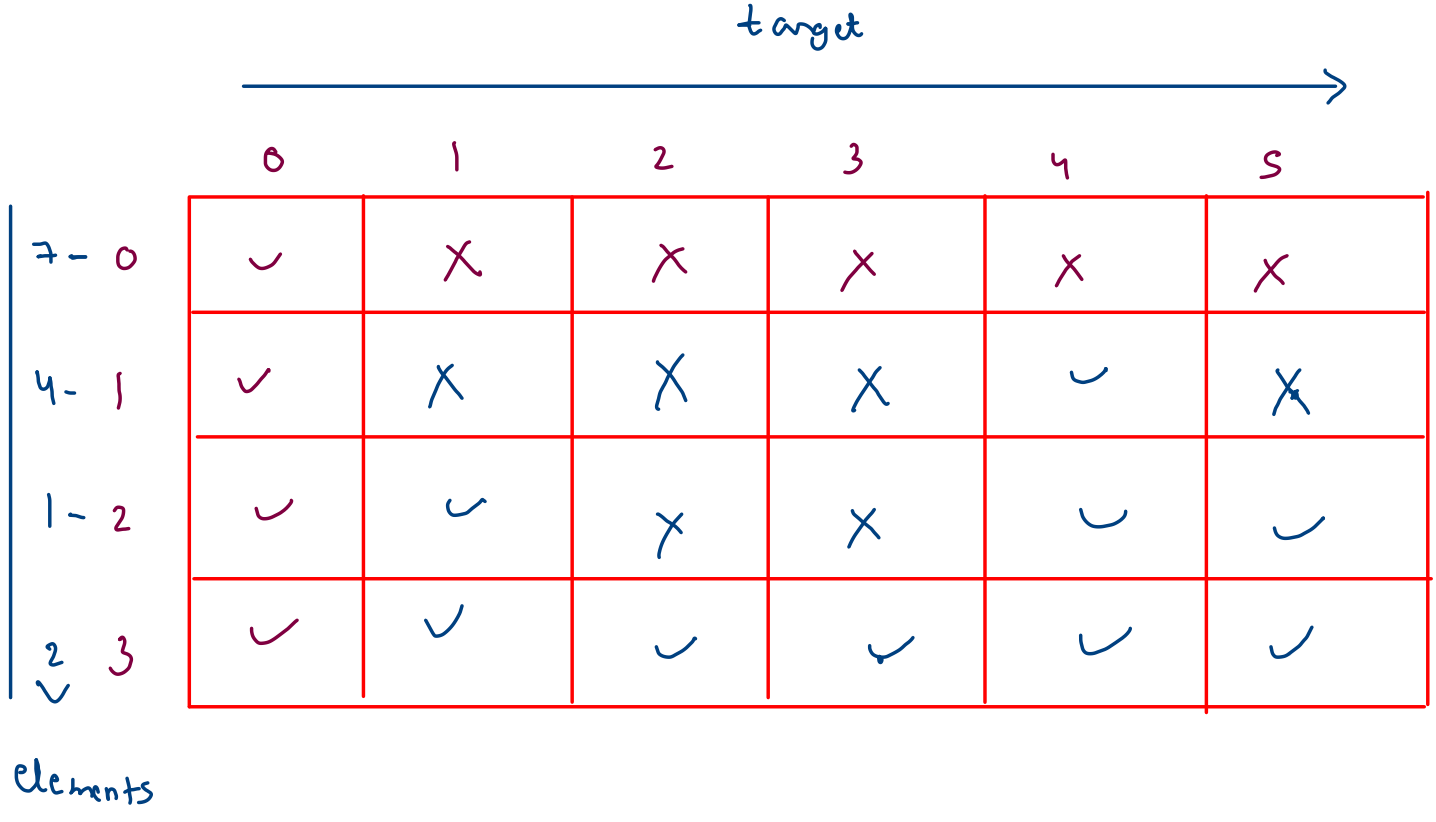
sum_th = j - arr[i] ;

target -> 5

7 4 1 2

```
boolean[][]dp = new boolean[arr.length][target+1];
```

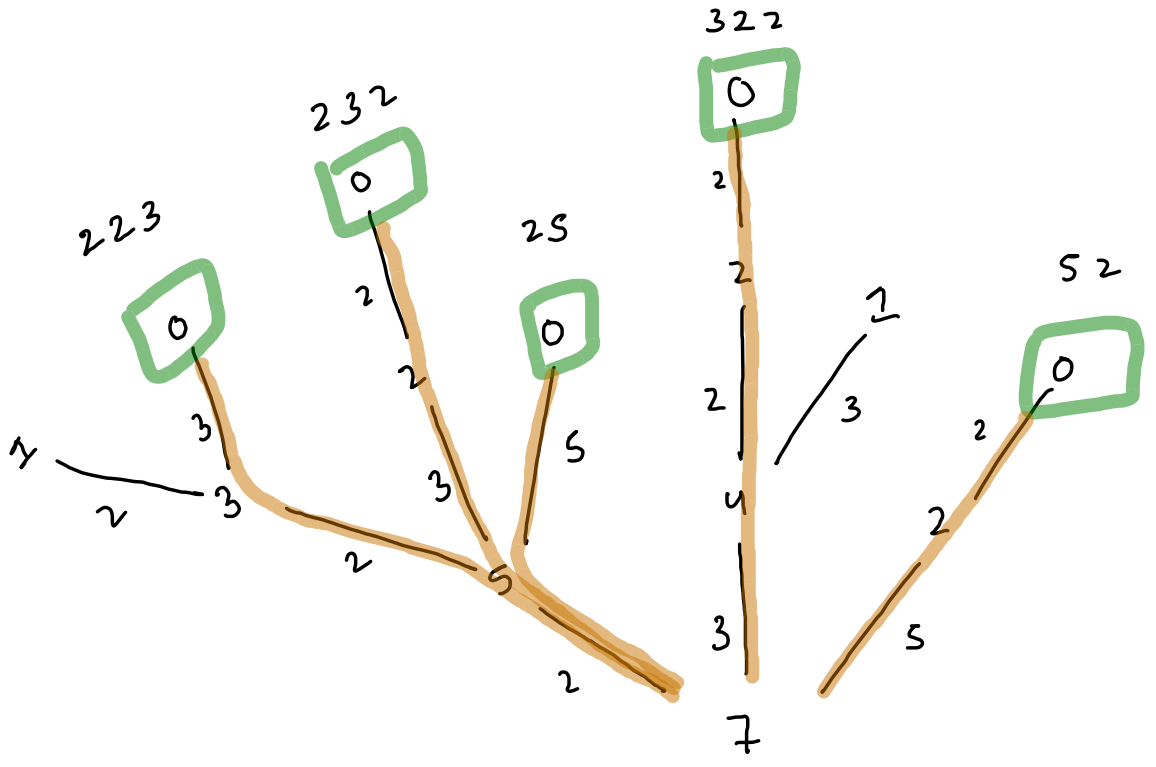
```
for(int i=0; i < dp.length;i++) {
    for(int j=0; j < dp[0].length;j++) {

        if(j == 0) {
            //target 0
            dp[i][j] = true;
        }
        else if(i == 0) {
            //single element
            if(arr[i] <= target) {
                dp[i][arr[i]] = true;
            }
        }
        else {
            boolean exc = dp[i-1][j];
            boolean inc = false;

            if(j-arr[i] >= 0) {
                inc = dp[i-1][j-arr[i]];
            }

            dp[i][j] = inc || exc;
        }
    }
}
```



Coin change permutations

Coins: [2, 3, 5]

amt = 7



ans:

2 2 3

2 3 2

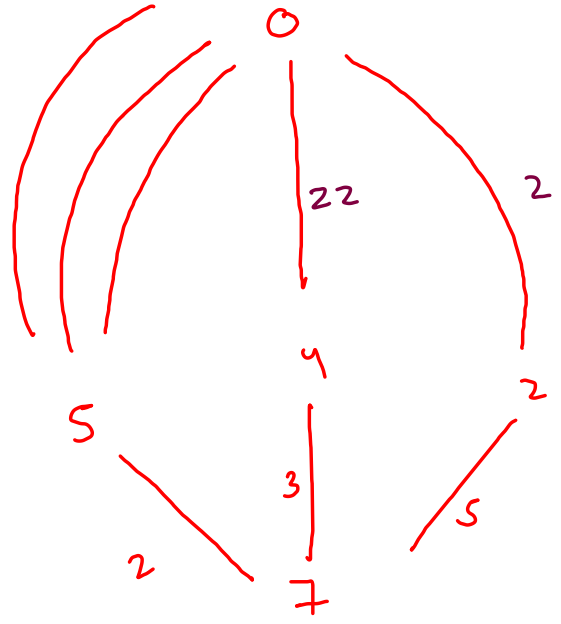
3 2 2

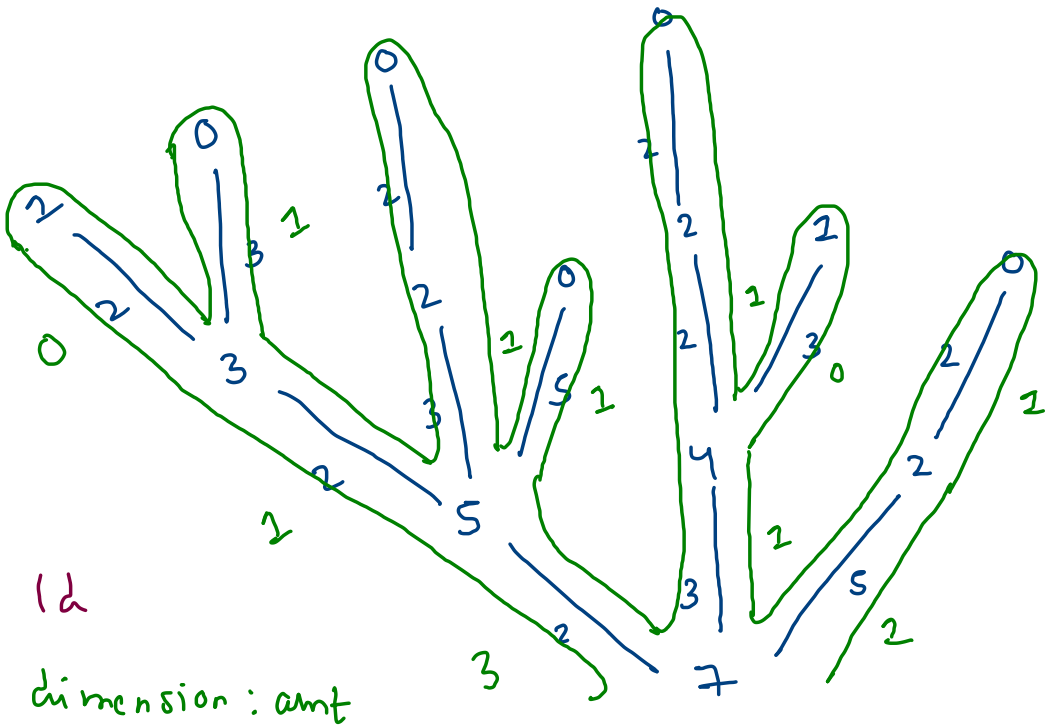
2 5

5 2

coins: [2, 3, 5]

23, 32, 5





1d

dimension: amt

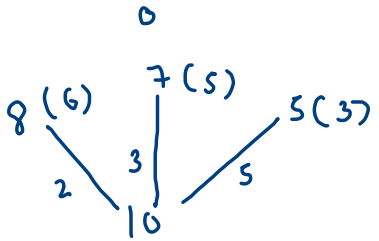
0 to 7

5 ways

coins: [2, 3, 5]

amt = 10

1	0	1	1	1	3	2	5	6	8	14
0	1	2	3	4	5	6	7	8	9	10
.		2.	3.	22.	32. 23.,5.	222. 33.	322. 223. 232. 25. 52.			

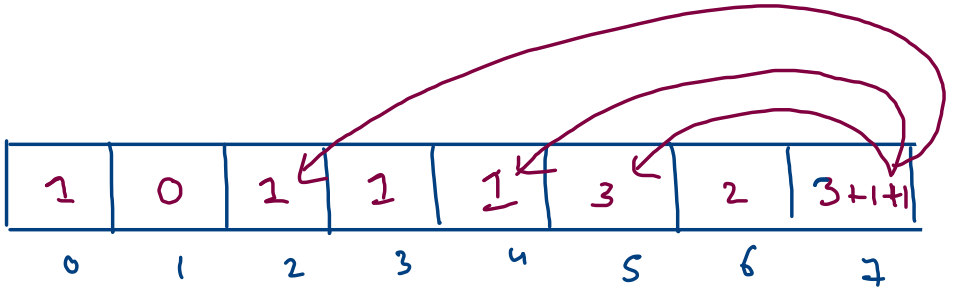


dp[i] -> no. of permutation to pay 'i' amount using all coins.

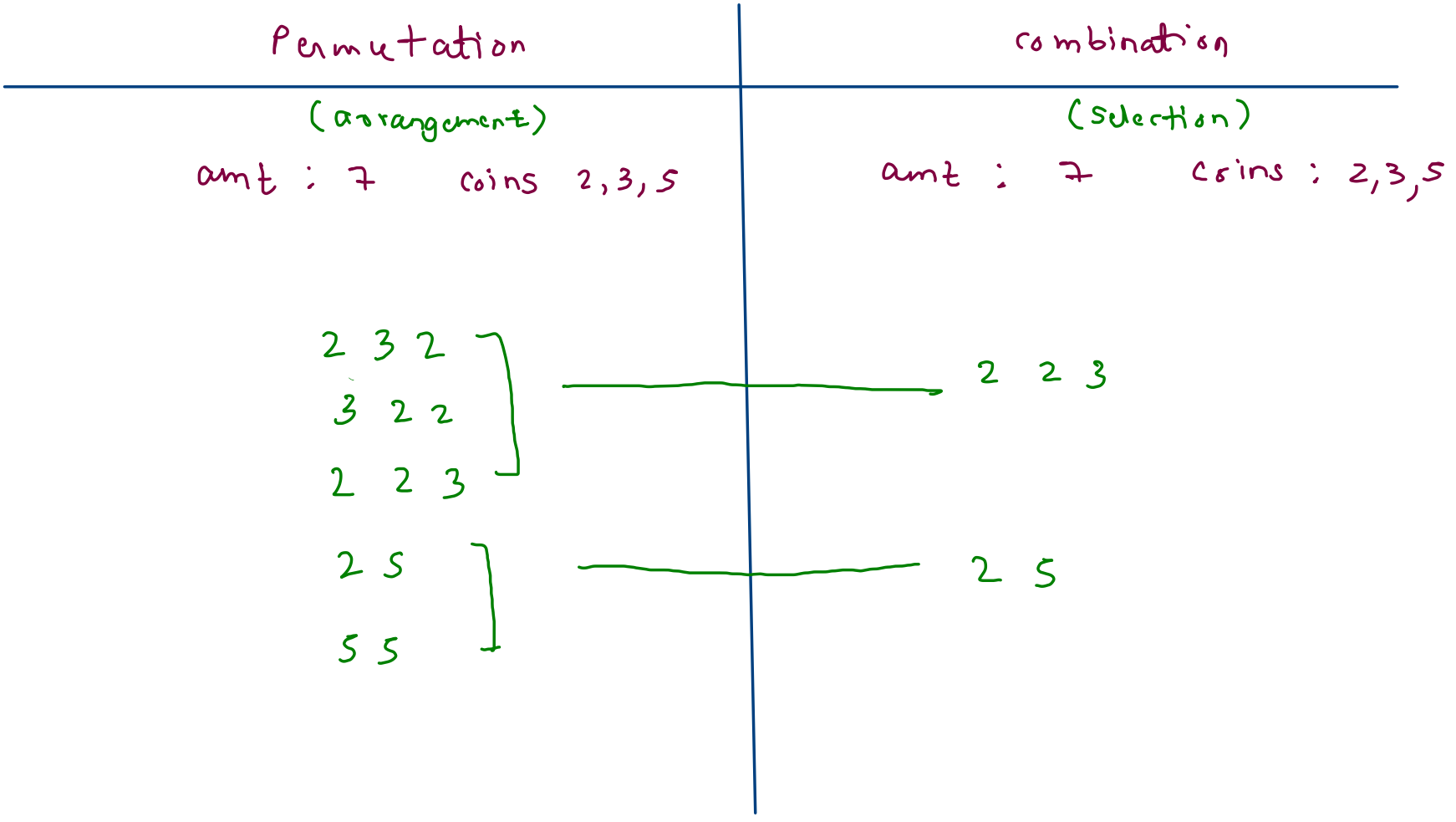
```java
public static int ccp(int[]coins,int amt) {
    int[]dp = new int[amt+1];

    //dp[i] -> ways to pay 'i' amount

    dp[0] = 1;

    for(int i = 1; i <= amt;i++) {
        for(int j = 0; j < coins.length;j++) {
            int rem_amt = i - coins[j];

            if(rem_amt >= 0) {
                dp[i] += dp[rem_amt];
            }
        }
    }

    return dp[amt];
}
```

amt = 7

coins = [2, 3, 5]



dp[7] = 5


Permutation
(arrangement)
amt : 7 coins 2,3,5
2 3 2
3 2 2
2 2 3
2 5
5 5
combination
(selection)
amt : 7 coins : 2,3,5
2 2 3
2 5