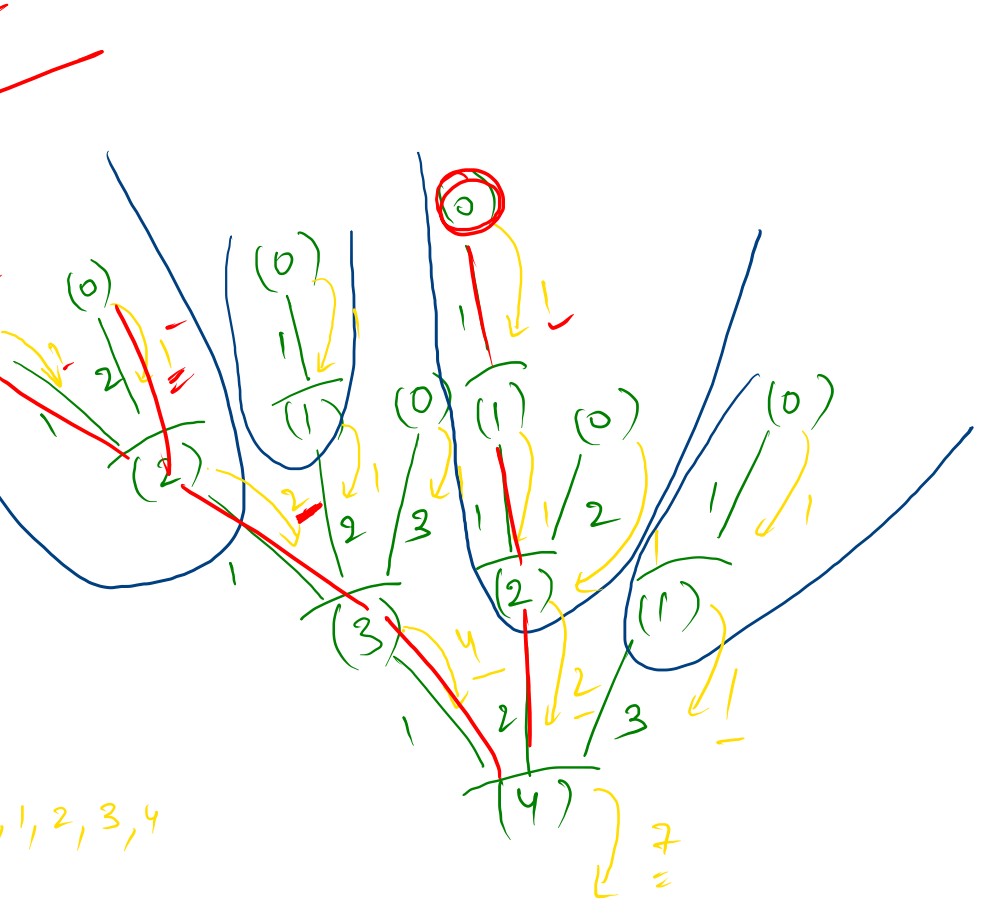
Climb Stairs

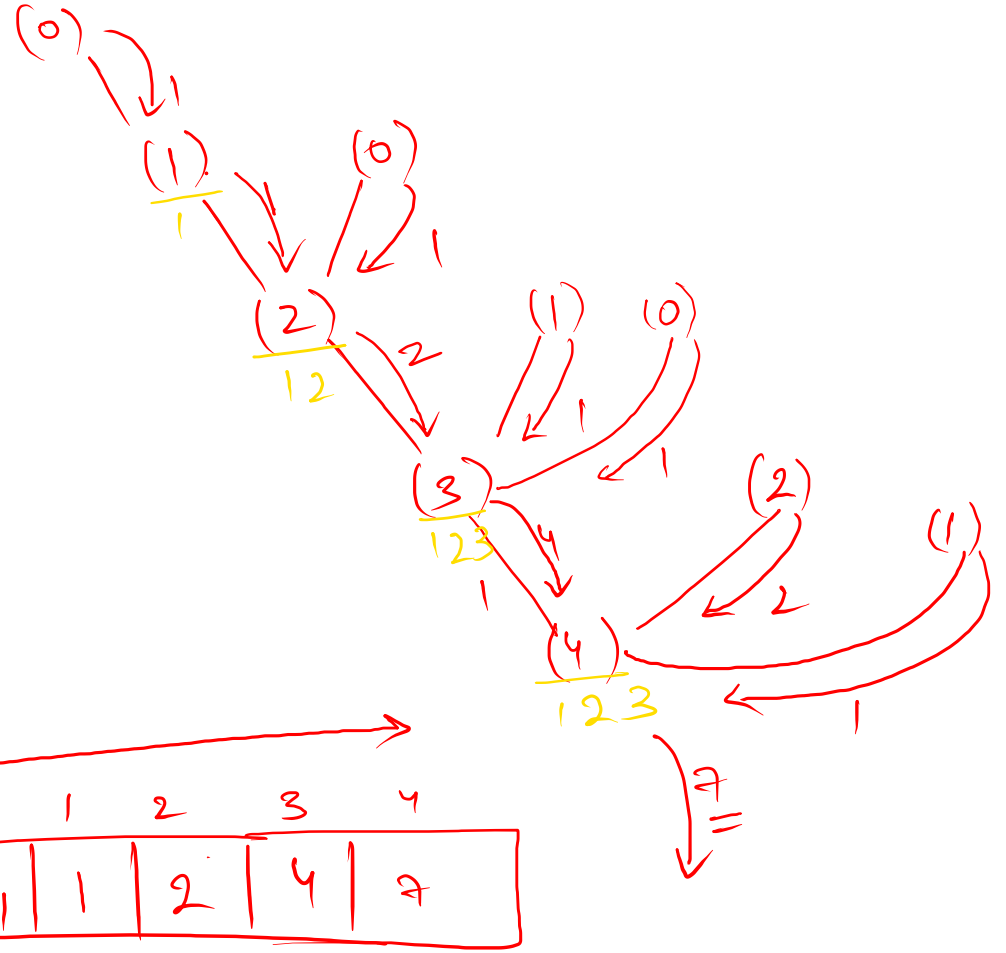


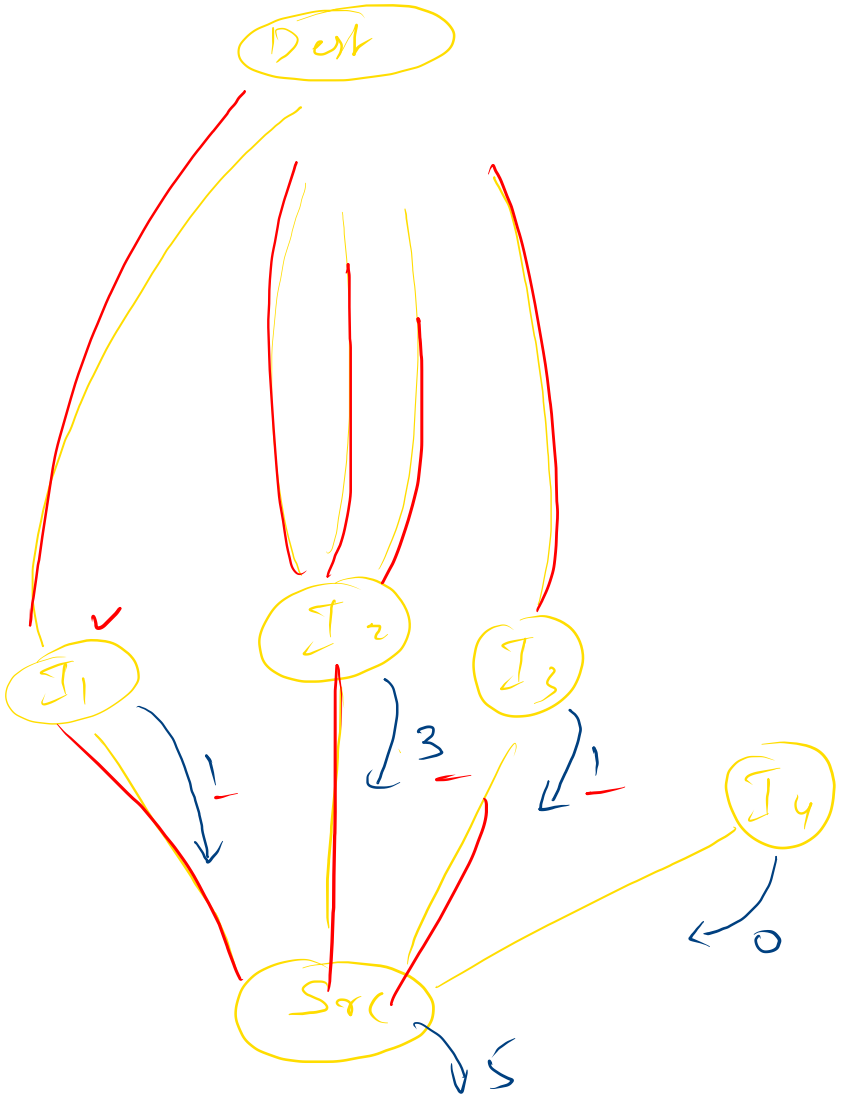
no. of ways to climb n stairs?

n = 4 → 7

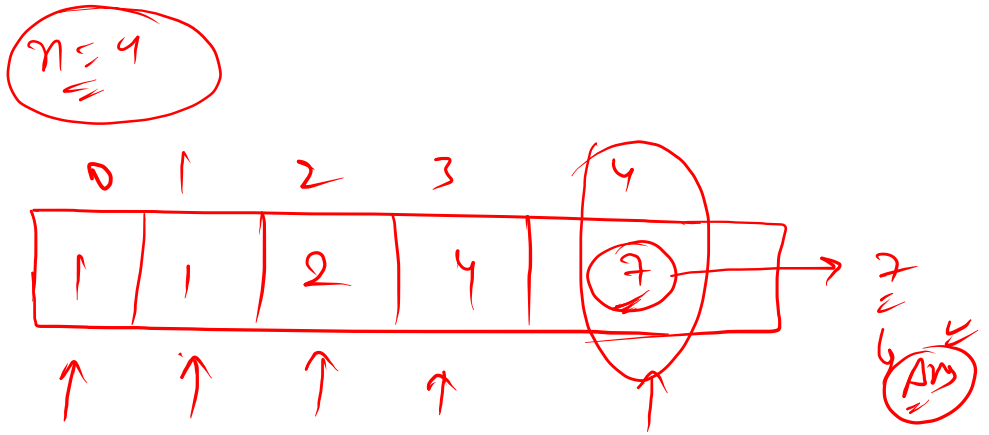
n = 4

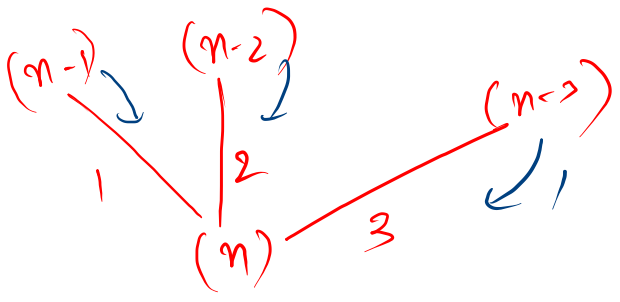
```java
public static int climbStairs(int n,int dp[]){
    if(n == 0){
        return dp[n] = 1;
    }
    if(dp[n] != 0){
        return dp[n];
    }
    int tways = 0;
    if(n-1 >= 0){
        tways += climbStairs(n-1,dp);// 1 len
    }
    if(n-2 >= 0){
        tways += climbStairs(n-2,dp);// 2 len
    }
    if(n-3 >= 0){
        tways += climbStairs(n-3,dp);// 3 len
    }

    return dp[n] = tways;
}
```



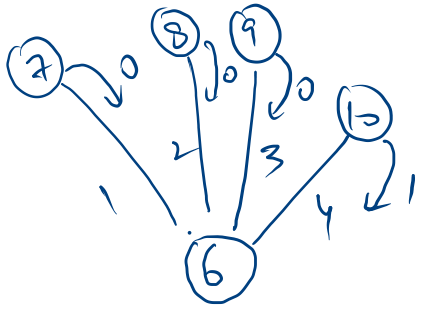

Dest
J1
J2
J3
J4
Src
1
3
1
0
5

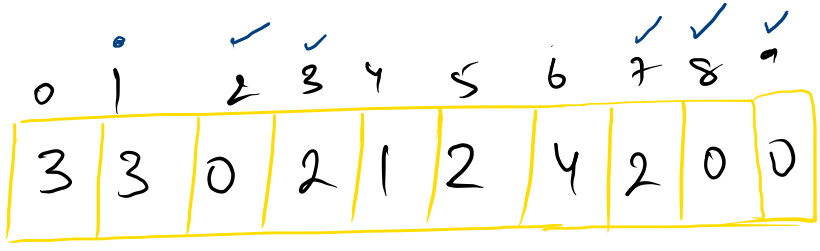
```java
public static int climbStairsT(int n){
    int dp[] = new int[n+1];

    for(int i = 0 ; i <= n ; i++){
        if(i == 0){
            dp[0] = 1;
        }else if(i == 1){
            dp[i] += dp[i-1];
        }else if(i == 2){
            dp[i] = dp[i-1] + dp[i-2];
        }else{
            dp[i] = dp[i-1] + dp[i-2] + dp[i-3];
        }
    }

    return dp[n];
}
```

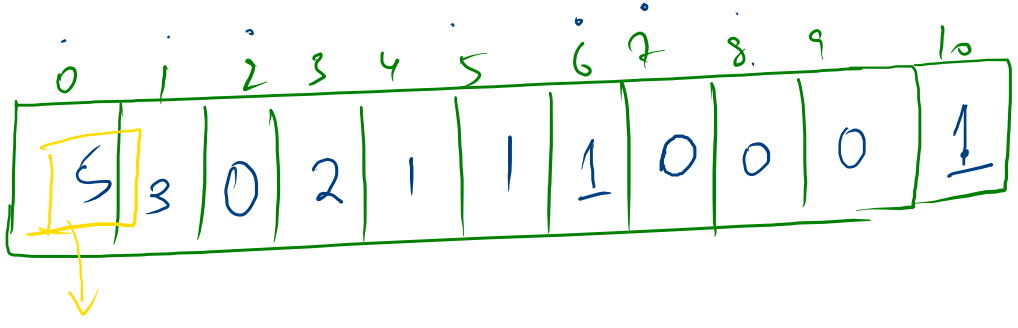




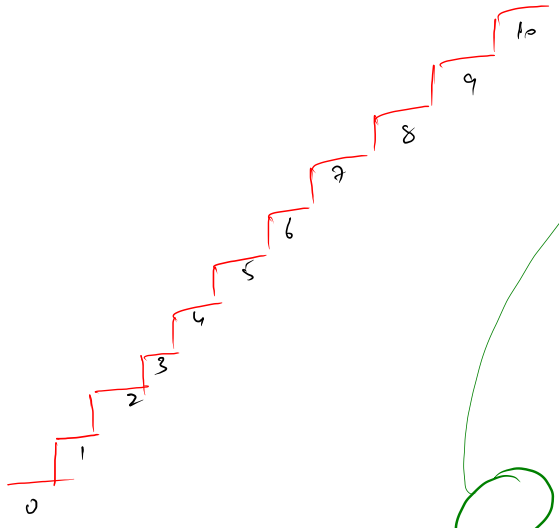
n (Stairs)

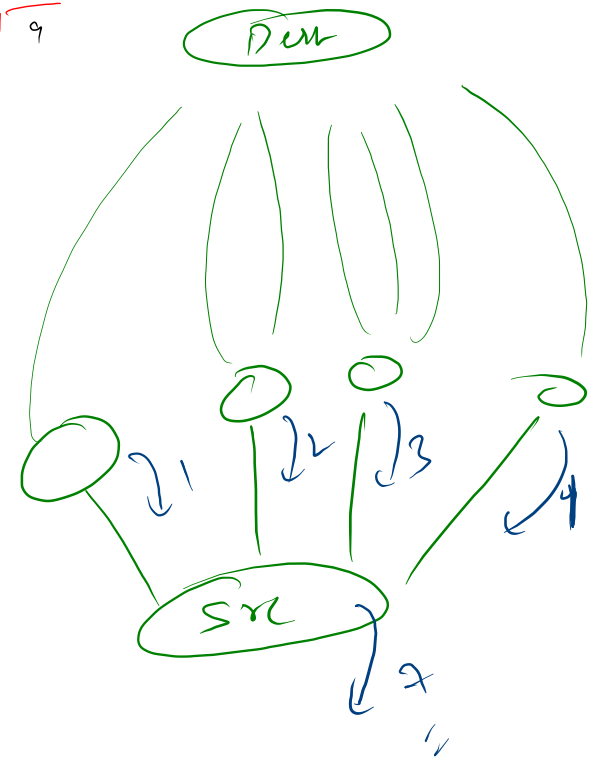






dp[i] => no. ways to climb from i^{th} → dest (nth)

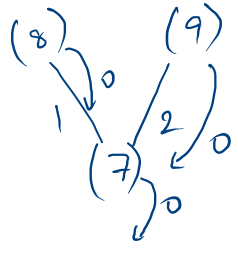


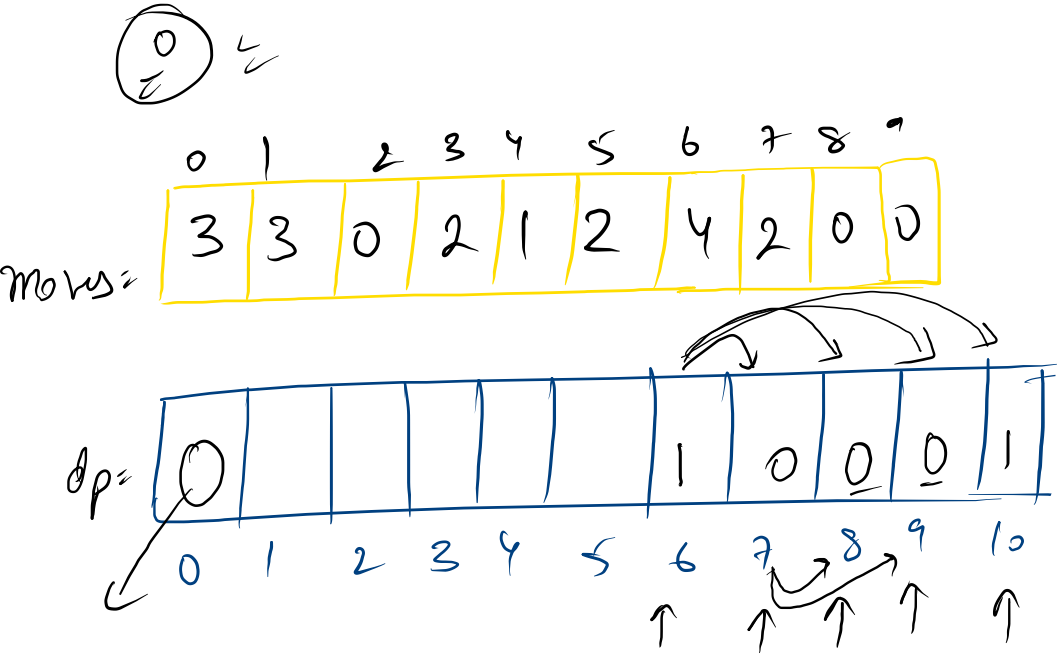


Tabulation

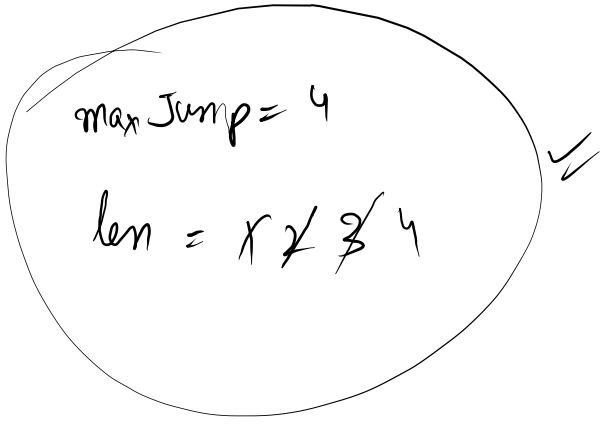
1. Memory
2. Memory
3. Traversal & sol^{n}

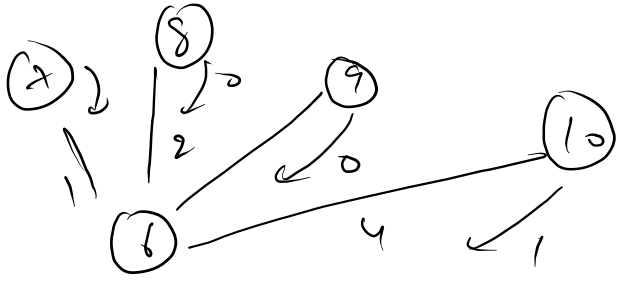
Easier → Complex






```java
int dp[] = new int[n+1];

for(int i = n ; i>= 0 ; i--){
    if(i == n){
        dp[i] = 1;
    }else{
        int maxJump = moves[i];
        for(int len = 1 ; len <= maxJump && len + i <= n; len++){
            dp[i] += dp[i+len];
        }
    }
}
```





=n

Src=0 dest=10

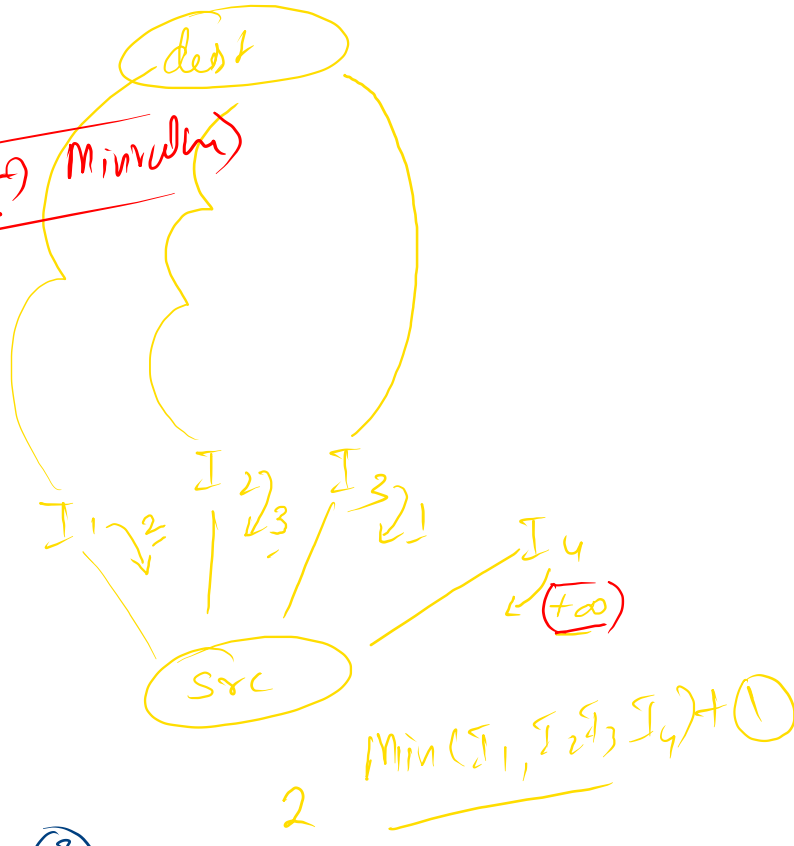
0	1	2	3	4	5	6	7	8	9
3	3	0	2	1	2	4	2	0	0

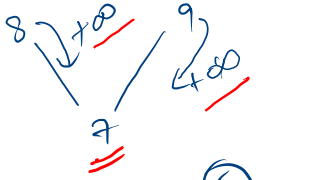
Moves

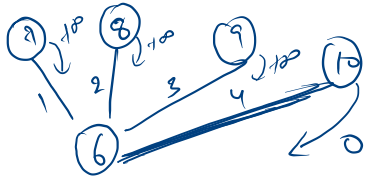
dp[]:

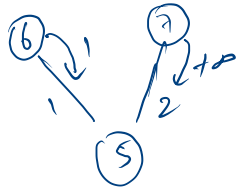
0	1	2	3	4	5	6	7	8	9	10
4	4	+∞	3	3	2	1	~~+∞~~	+∞	+∞	0

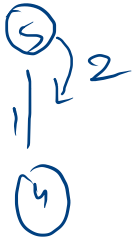
+∞ → Max_value + Min value

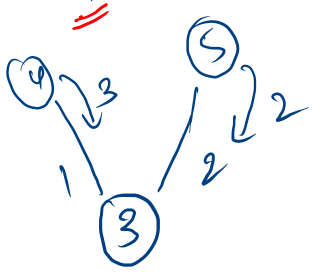


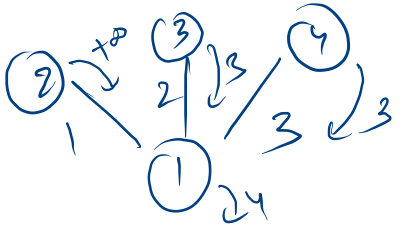


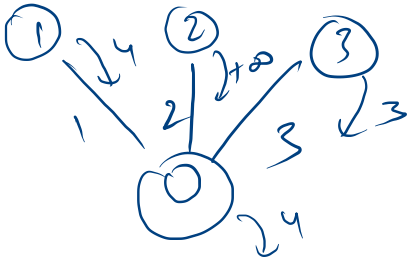










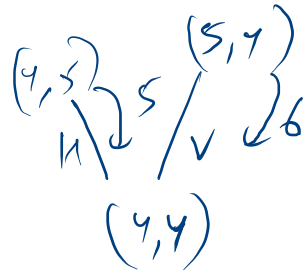


Cost =

	0	1	2	3	4	5
0	9	1	4	2	8	2
1	4	3	6	5	0	4
2	1	2	4	1	4	6
3	2	0	7	3	2	2
4	3	1	5	9	2	4
5	2	7	0	8	5	1

dp =

	0	1	2	3	4	5
0	23	23	24	20	21	19
1	24	22	23	18	13	17
2	20	19	17	13	13	13
3	21	19	19	12	7	7
4	23	20	19	16	7	5
5	23	21	14	14	6	1



Src ⇒ (0,0)

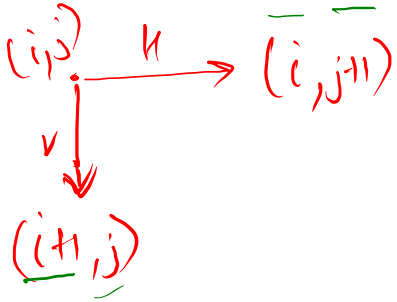
dest ⇒ (nr-1, nc-1)

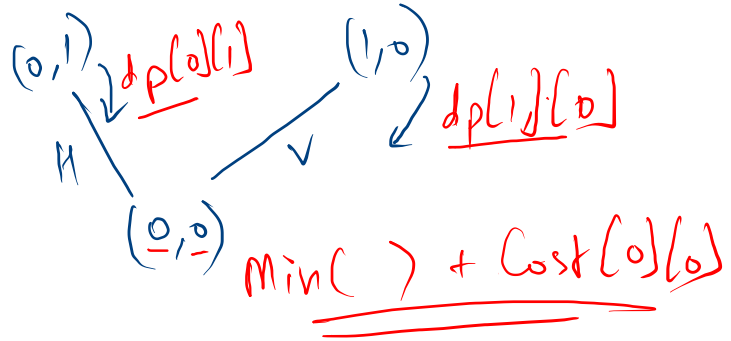
To find ⇒ min cost to travel from Src → dest

Given ⇒ dir

// M_1 ⇒ dp[i][j] ⇒ min cost to travel from (i,j) → dest

M_2 ⇒ dp[i][j] ⇒ min cost to travel from src → (i,j)





Cost:

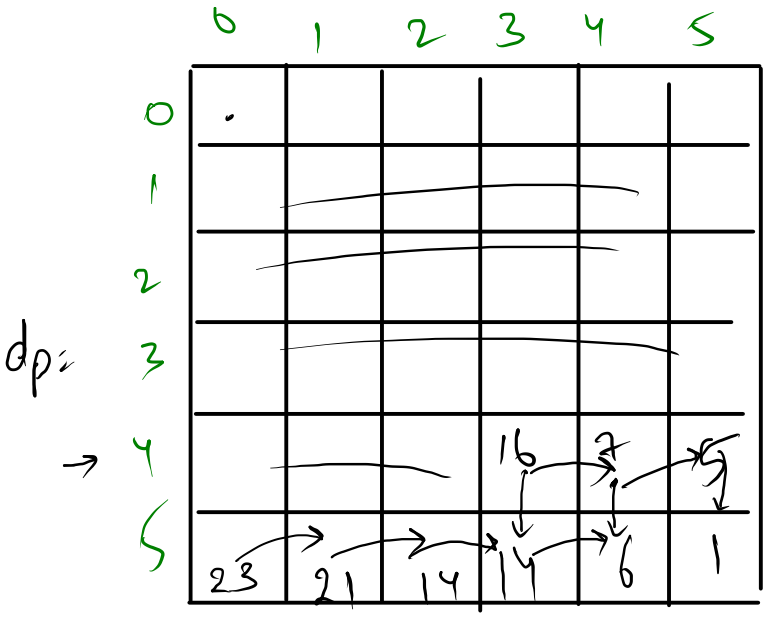
	0	1	2	3	4	5
0	0	1	4	2	8	2
1	4	3	6	5	0	4
2	1	2	4	1	4	6
3	2	0	7	3	2	2
4	3	1	5	9	2	4
5	2	7	0	8	5	1

nr = 6

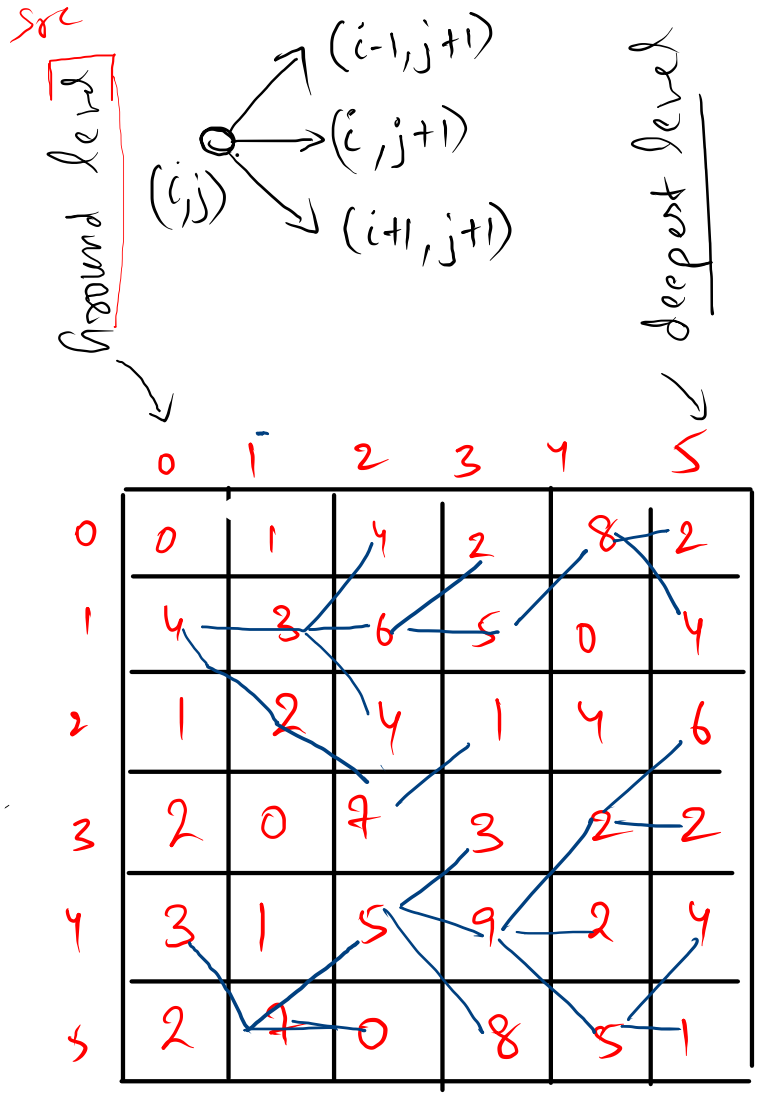
nc = 6

r	c
5	5 4 3 2 1 0
4	5 4 3

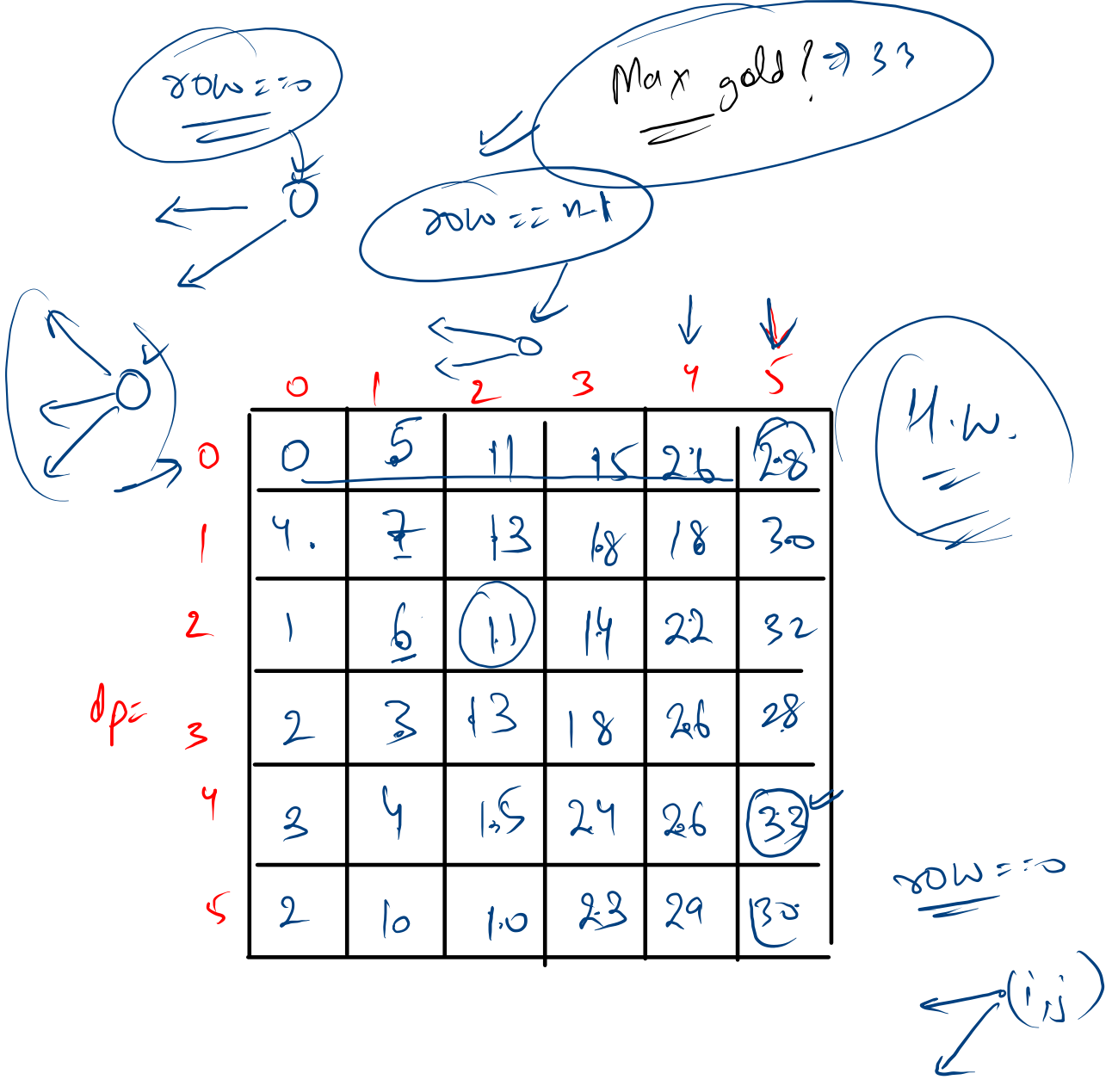
dp:

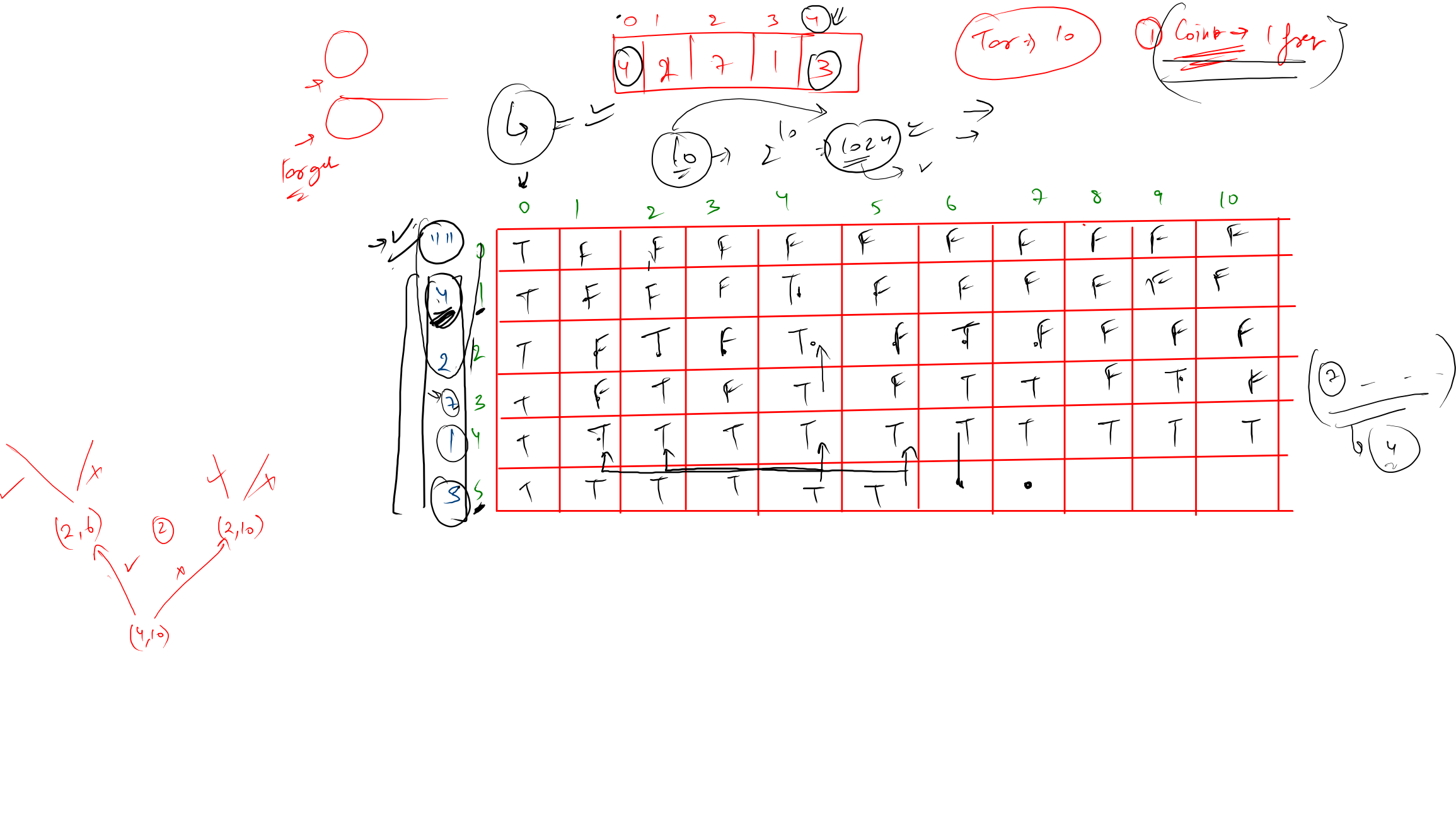


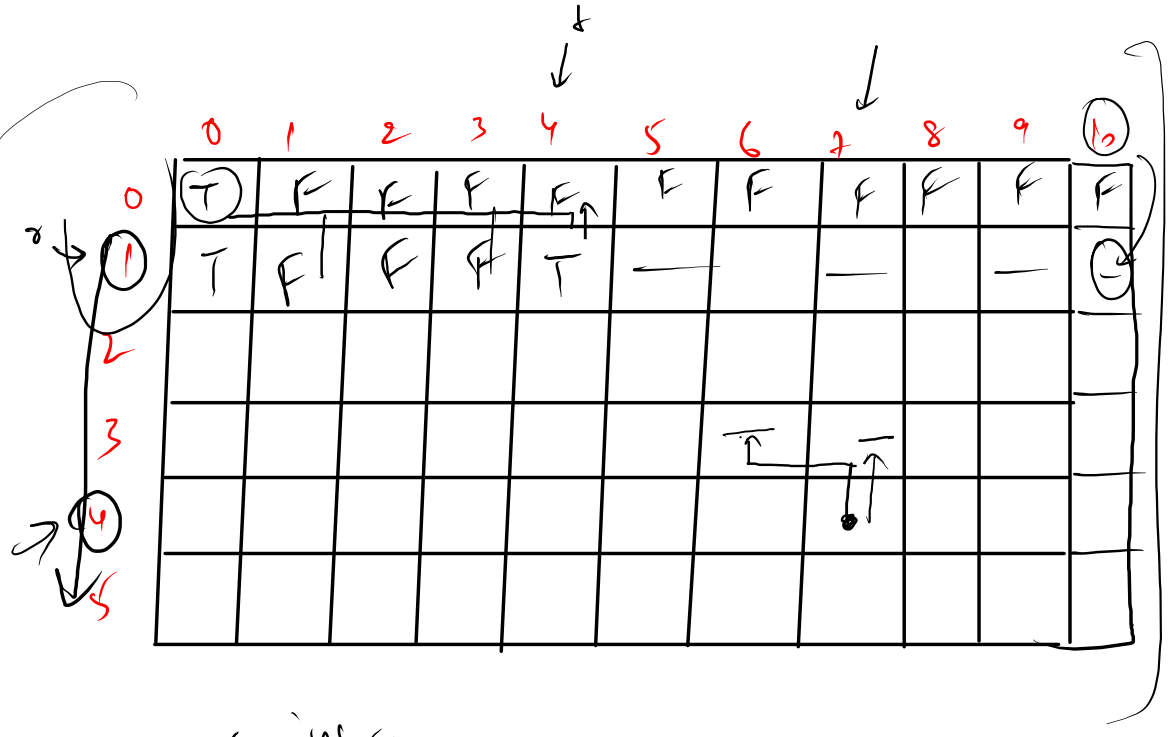

```
int dp[][] = new int[nr][nc];

for(int r = nr-1 ; r >= 0 ; r--){
    for(int c = nc - 1; c >= 0 ; c--){
        if(r == nr-1 && c == nc-1){
            dp[r][c] = cost[r][c];
        }else if(r == nr-1){
            dp[r][c] = dp[r][c+1] + cost[r][c];
        }else if(c == nc-1){
            dp[r][c] = dp[r+1][c] + cost[r][c];
        }else{
            dp[r][c] = Math.min(dp[r][c+1],dp[r+1][c])+cost[r][c];
        }
    }
}

System.out.println(dp[0][0]);
```


src
ground level
deepest level
(i,j)
(i-1,j+1)
(i,j+1)
(i+1,j+1)
0 1 2 3 4 5
0 0 1 4 2 8 2
1 4 3 6 5 0 4
2 1 2 4 1 4 6
3 2 0 7 3 2 2
4 3 1 5 9 2 4
5 2 1 0 8 5 1

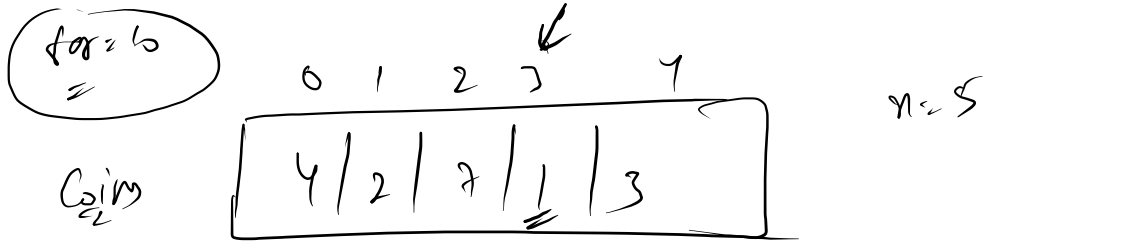
mine :-


row == 0
row == n-1
Max gold ? → 33
0 1 2 3 4 5
0 0 5 11 15 26 28
1 4 7 13 18 18 30
2 1 6 11 14 22 32
3 2 3 13 18 26 28
4 3 4 15 24 26 33
5 2 10 10 23 29 30
dp :-
H.W.
row == 0
(i,j)


Target → 10
Coins → 1 per
forget
0 1 2 3 4
4 2 7 1 3
10
1024
0 1 2 3 4 5 6 7 8 9 10
1 11
4
2
3
1
5
T F F F F F F F F F F
T F F F T F F F F F F
T F T F T F T F F F F
T F T F T F T T F T F
T T T T T T T T T T T
T T T T T T
(2,6)
(2,10)
(4,10)




```java
boolean dp[][] = new boolean[n + 1][tar + 1];

for (int r = 0 ; r <= n ; r++) {
  for (int t = 0 ; t <= tar ; t++) {
    if (r == 0 && t == 0) {
      dp[0][0] = true;
    } else if (r == 0) {
      dp[r][t] = false;
    } else if (t == 0) {
      dp[r][t] = true;
    } else {
      int coin = coins[r - 1];
      boolean inc = (t - coin >= 0) ? dp[r - 1][t - coin] : false; // inc
      boolean exc = dp[r - 1][t]; // exc

      dp[r][t] = inc || exc;
    }
  }
  if (dp[r][tar] == true) {
    System.out.println(true);
    return;
  }
}

System.out.println(false);
```



Q => n

✓ Coins = | 0: 2 | 1: 3 | 2: 5 | 3: 6 |

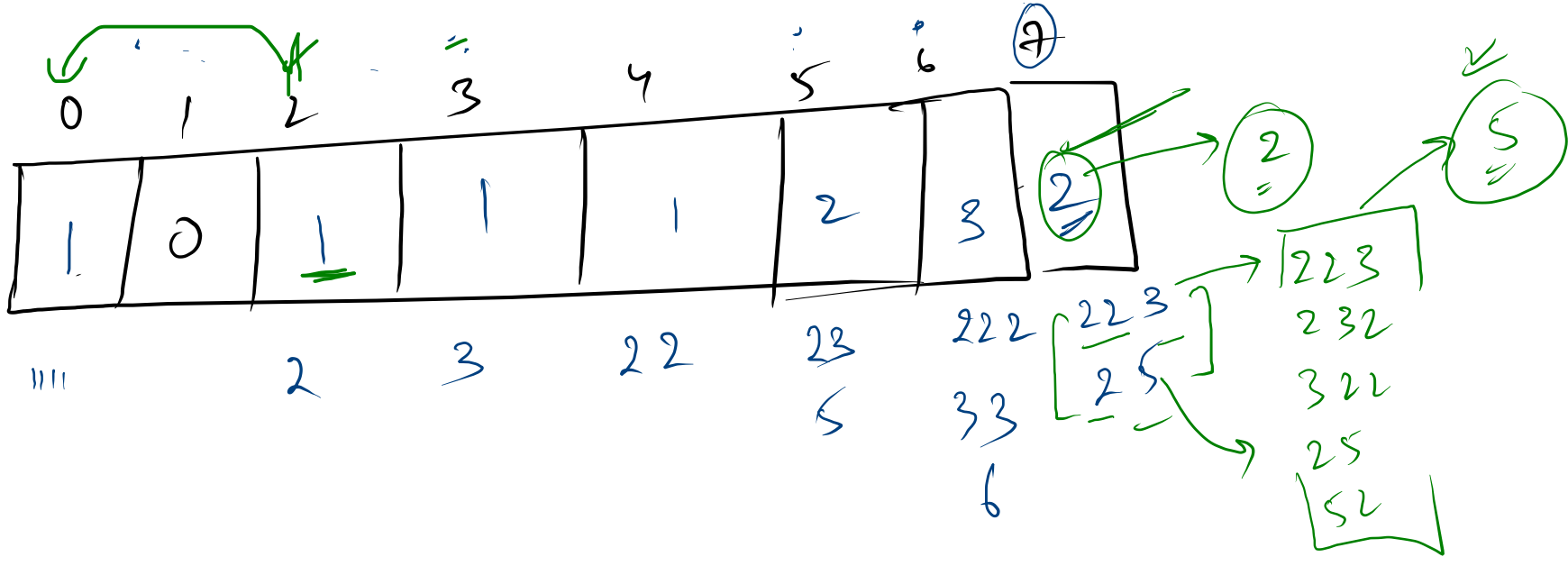
✓ for = 7

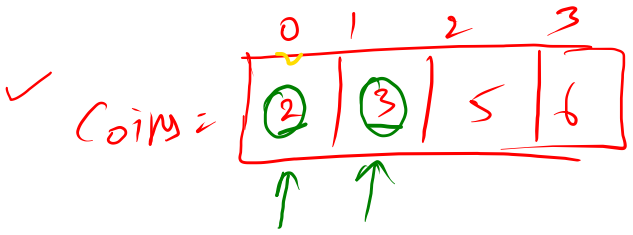
* Give (unlimited denominations)
  - inc / exc

Coins = 6

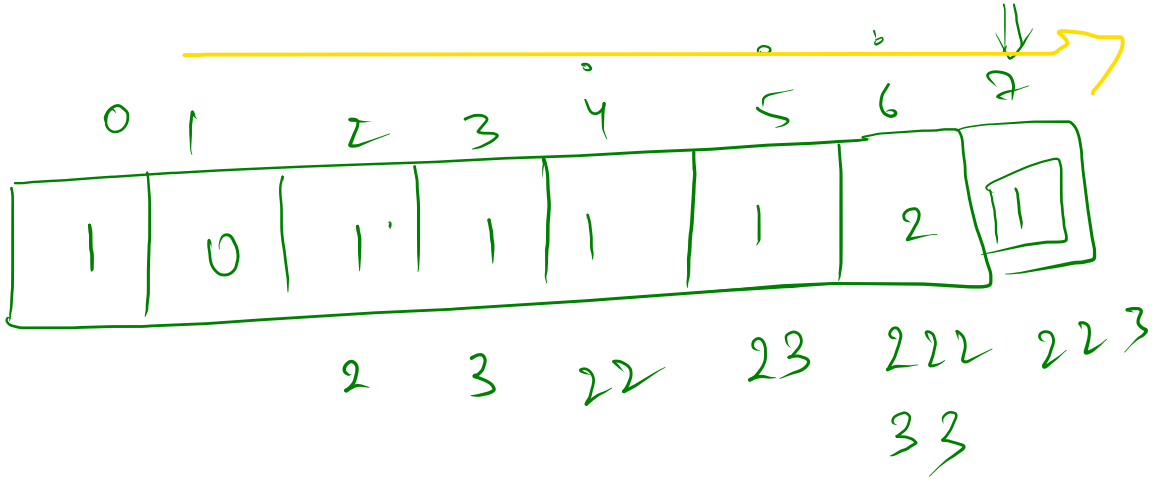
Permutation ✓
- Array
  - select

Combination
- select

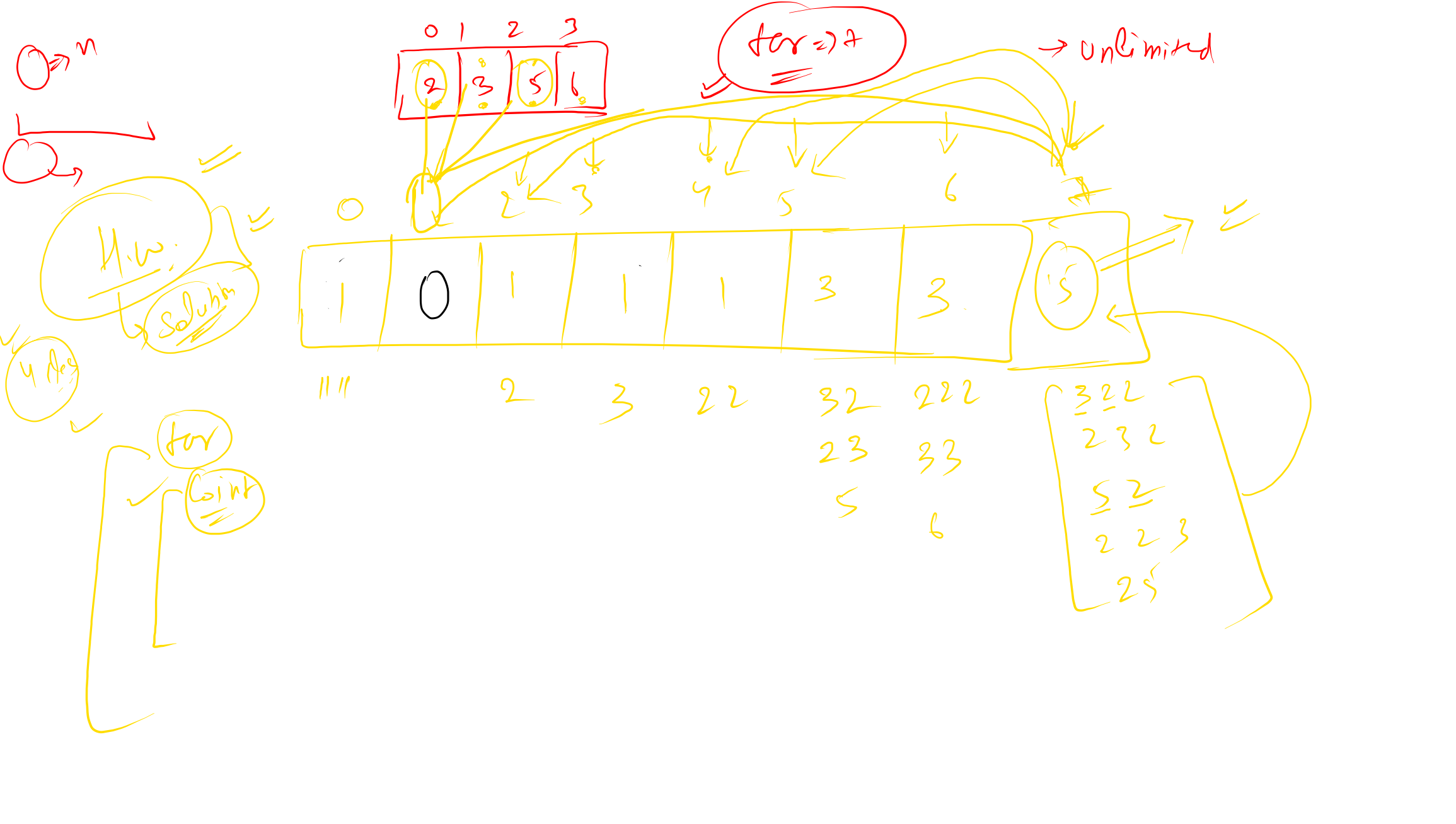


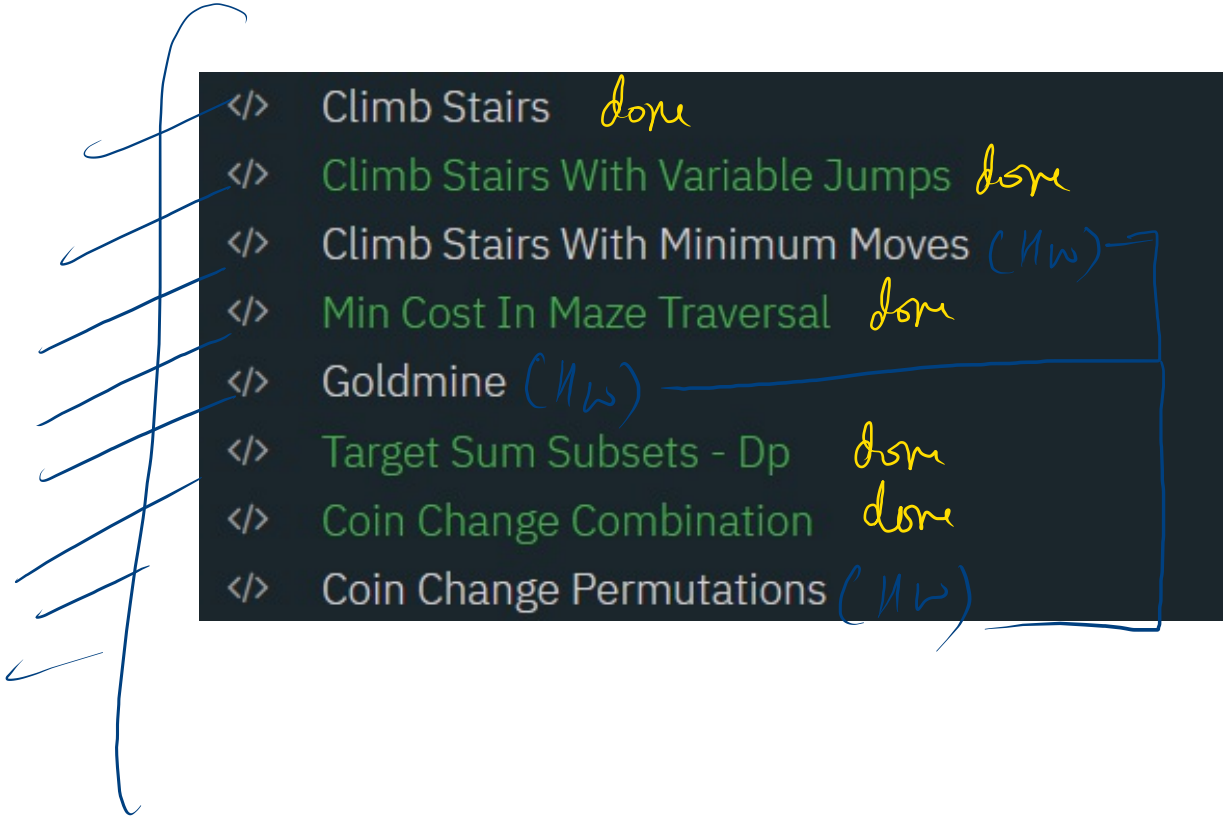


tar = 7




```java
int dp[] = new int[tar+1];

dp[0] = 1;

for(int coin : coins){
    for(int t = coin ; t <= tar ; t++){
        dp[t] += dp[t-coin];
    }
}

System.out.println(dp[tar]);
```


0 1 2 3
2 3 5 6
for
unlimited
0 1 2 3 4 5 6 7
1 0 1 1 1 3 3 5
H.W:
Solution
4 Aug
for
coin
111
2
3
22
32
23
5
222
33
6
322
232
52
223
25


Climb Stairs done
Climb Stairs With Variable Jumps done
Climb Stairs With Minimum Moves (HW)
Min Cost In Maze Traversal done
Goldmine (HW)
Target Sum Subsets - Dp done
Coin Change Combination done
Coin Change Permutations (HW)

8 → 5 + 3 =