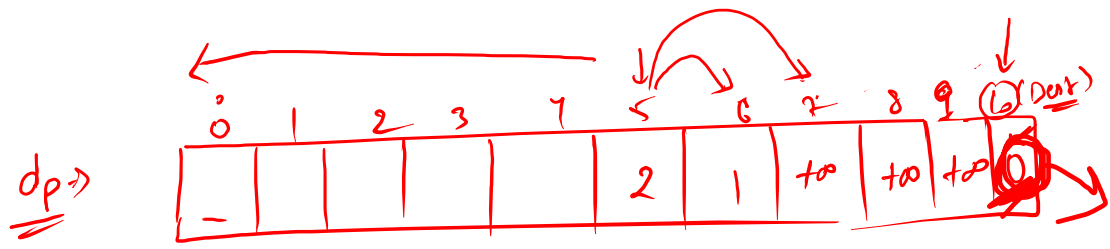
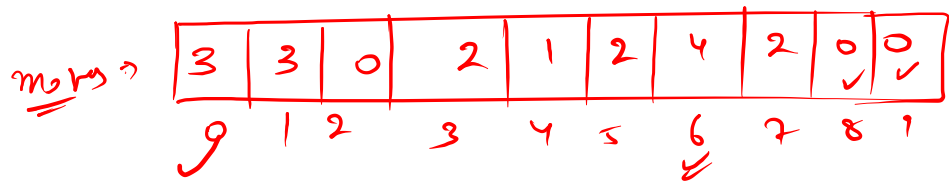


10
3
3
0
2
2
1
2
4
2
0
0
0
0

→ n=10



```

for(int i = n-1 ; i >= 0 ; i--){
    int maxJmp = moves[i];

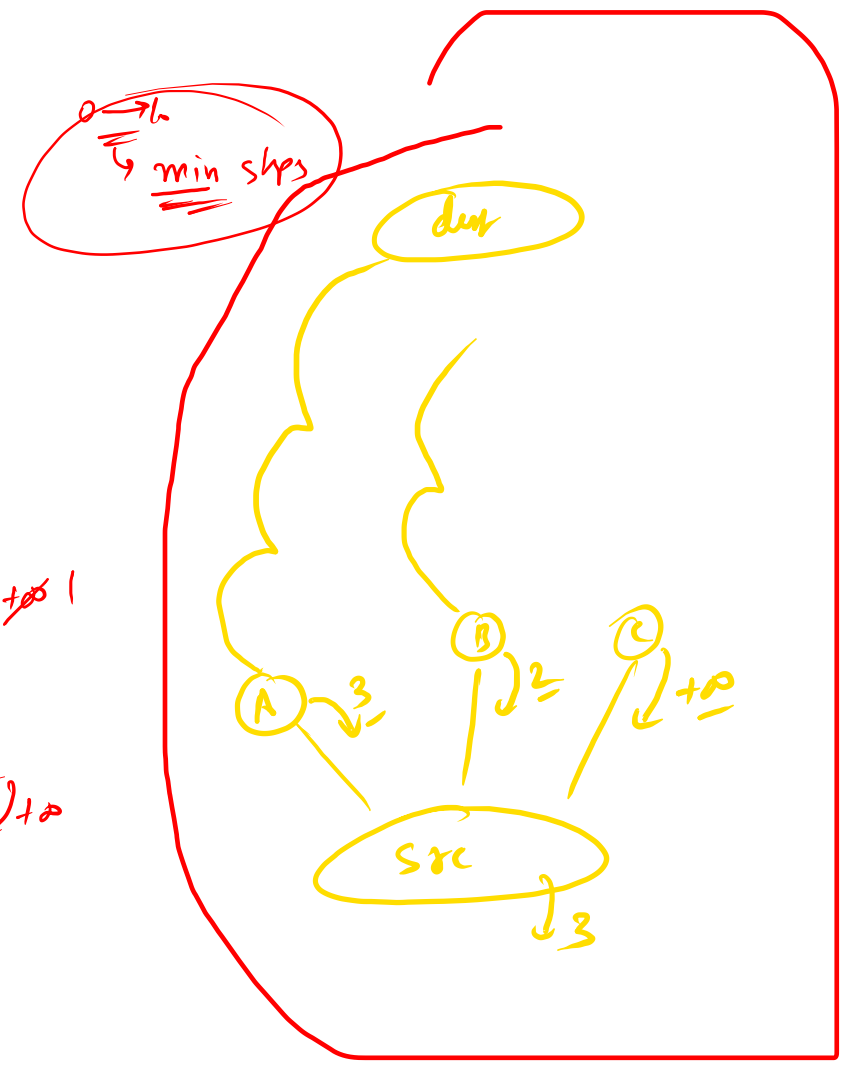
    if(maxJmp == 0){
        qb[i] = Integer.MAX_VALUE;
    }else{
        int minSteps = Integer.MAX_VALUE;
        for(int jmp = 1 ; jmp <= maxJmp && i+jmp <= n ; jmp++){
            minSteps = Math.min(minSteps, qb[i+jmp]);
        }

        if(minSteps == Integer.MAX_VALUE){
            qb[i] = Integer.MAX_VALUE;
        }else{
            qb[i] = minSteps + 1;
        }
    }
}

```

minSteps = too 1

6/2 5/1 7/too



gob

	0	1	2	3	4	5
0	0+12 = 23	1+22 = 23	4+20 = 24	2+18 = 20	8+13 = 21	2+10 = 19
1	4+20 = 24	3+19 = 22	23	18	13	12
2	20	19	17	13	13	13
3	2+19 = 21	0+15 = 19	7+12 = 19	3+9 = 12	2+7 = 9	7
4	3+20 = 23	1+19 = 20	5+14 = 19	9+7 = 16	2+5 = 7	4+1 = 5
5	2+21 = 23	7+14 = 21	0+14 = 14	8+6 = 14	6	1

Cost =

11/25
(4,9)

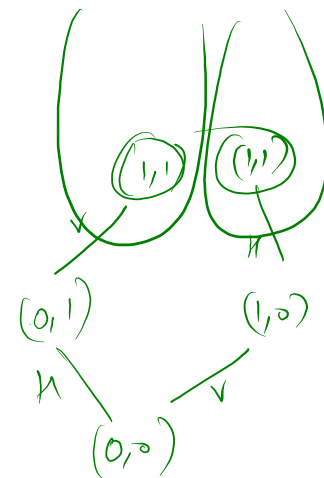
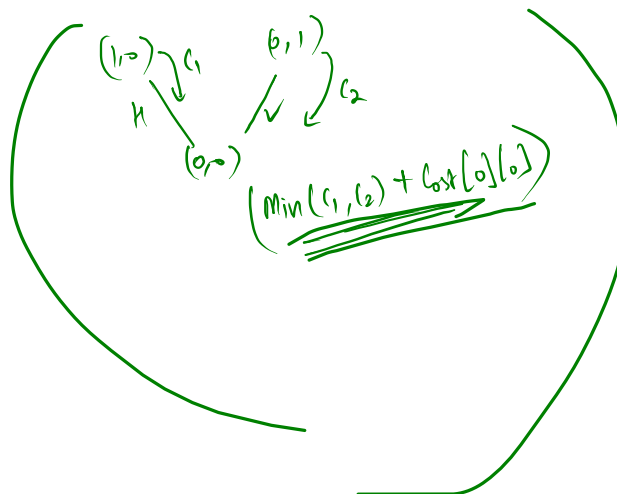
	0	1	2	3	4	5
0	0	1	4	2	8	2
1	4	3	6	5	0	4
2	1	2	4	1	4	6
3	2	0	7	3	2	2
4	3	1	5	9	2	4
5	2	7	0	8	5	1

Cost[i][j] → cost required to visit (i,j) block

(0,0) → (5,5)

Problem stmt → find min cost to move from (0,0) → (nr-1, nc-1)

gob[i][j] ⇒ Min cost to reach destination from (i,j) = Sol → gob[0][0]



	0	1	2	3	4	5
0	0	1	4	2	8	2
1	4	3	<u>6</u>	5	0	4
2	1	2	4	1	4	6
3	2	0	7	3	2	2
4	3	1	5	9	<u>2</u>	4
5	<u>2</u>	7	<u>0</u>	8	5	<u>1</u>

	0	1	2	3	4	5
0	0					
1						
2						
3						
4					2+5 7	u+1 =5
5	28	2+4 =21	0+4 =14	8+6 =14	6	1

nr=6
nc=6

r=4
c=4

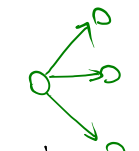
```

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] = cost[r][c] + dp[r][c+1];
            }else if(c == nc-1){
                dp[r][c] = cost[r][c] + dp[r+1][c];
            }else{
                dp[r][c] = cost[r][c] + Math.min(dp[r][c+1],dp[r+1][c]);
            }
        }
    }
}

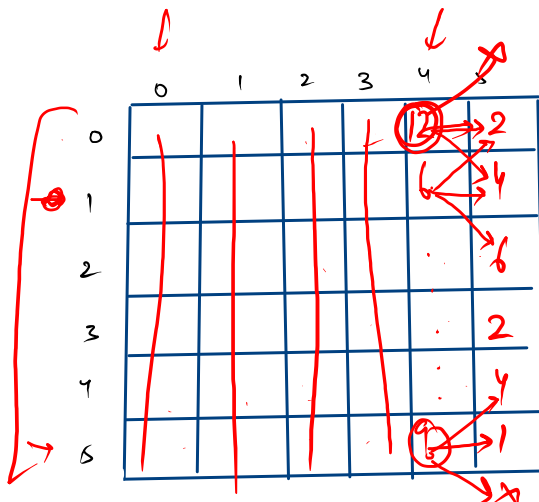
```

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

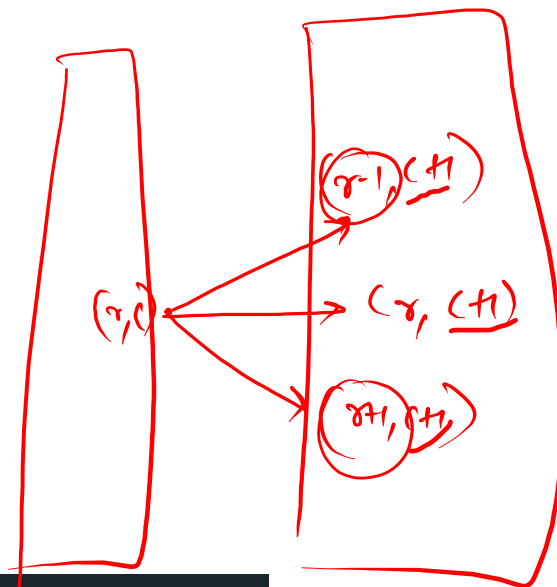
src



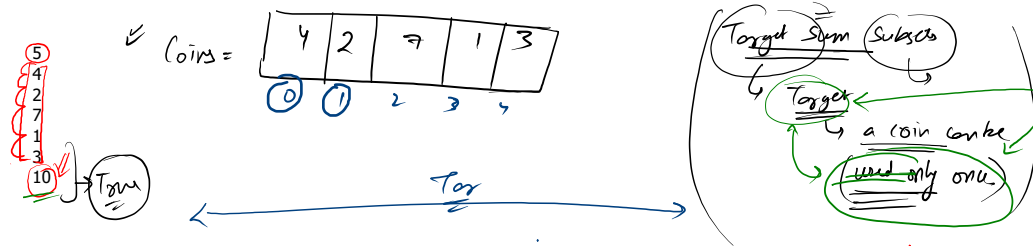
	0	1	2	3	4	5
0	0	1	4	2	8	2
1	4	3	6	5	0	4
2	1	2	4	1	4	6
3	2		0	7	3	2
4	3	1	5	9	2	4
5	2	7	0	8	5	1



	0	1	2	3	4	5
0					12	
1					6	
2						
3						2
4						4
5					9	



```
for(int c = nc-1 ; c>=0 ; c--){
    for(int r = 0 ; r < nr ; r++){
        if(c == nc-1){
            dp[r][c] = mine[r][c];
        }else{
            if(r == 0){
                dp[r][c] = Math.max(dp[r][c+1], dp[r+1][c+1]) + mine[r][c];
            }else if(r == nr-1){
                dp[r][c] = Math.max(dp[r][c+1], dp[r-1][c+1]) + mine[r][c];
            }else{
                dp[r][c] = Math.max(dp[r][c+1], Math.max(dp[r-1][c+1], dp[r+1][c+1])) + mine[r][c];
            }
        }
    }
}
```



DP

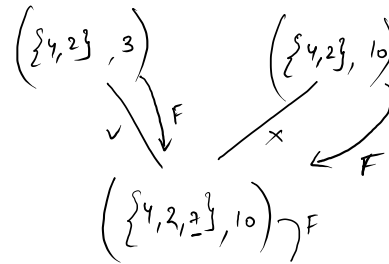
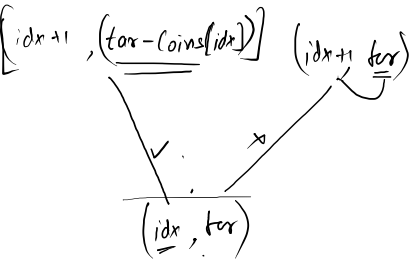
Coins

	0	1	2	3	4	5	6	7	8	9	10
- 0	T	F	F	F	F	F	F	F	F	F	F
{4}	T	F	F	F	T	F	F	F	F	F	F
{4, 2}	T	F	T	F	T	T	T	F	F	F	F
{4, 2, 7}	T	F	T	F	T	T	T	T	F	T	F
{4, 2, 1}	T	T	T	T	T	T	T	T	T	T	T
{4, 2, 1, 3}	T	T	T	T	T	T	T	T	T	T	T

Target = 10

(idx, tar - coins[idx])

(idx, tar)



4	2	7	1	3
0	1	2	3	4

$\eta = 5$


```
int n = coins.length;  
boolean dp[][] = new boolean[n+1][tar+1];
```

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

 $\gamma = 3,$

$C \approx 8$

Coin = 7

$$\underline{-6} \geq 0$$
[illegible]

Coin \Rightarrow

2	3	5	4
---	---	---	---

tar \Rightarrow 7

0	1	2	3	4	5	6	7
1	0	\emptyset	$1\emptyset$	$1\emptyset$	$1\emptyset$	$\emptyset\emptyset$	0

\uparrow
 \uparrow
 \uparrow

Coin \Rightarrow 3

```
int dp[] = new int[tar+1];
dp[0] = 1;

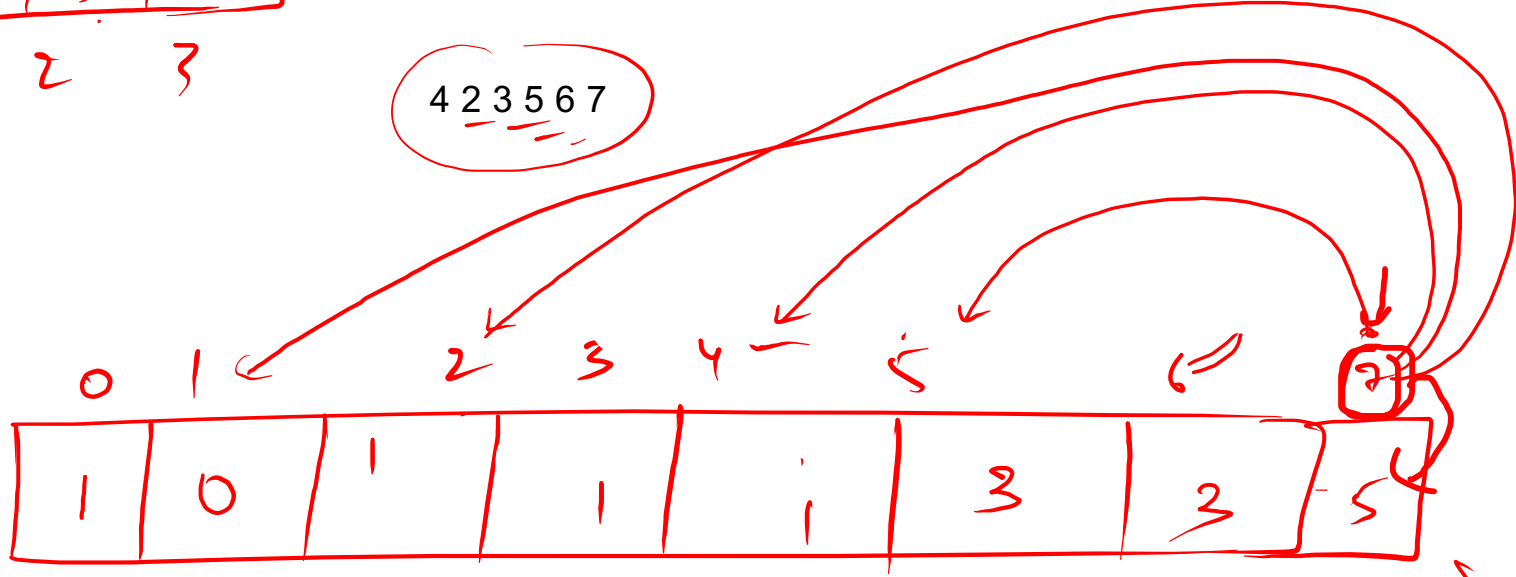
for(int i = 0 ; i < coins.length ; i++){
    int coin = coins[i];
    for(int j = coin ; j <= tar ; j++){
        dp[j] += dp[j-coin];
    }
}

return dp[tar];
```


2	3	5	6
---	---	---	---

0 1 2 3

4 2 3 5 6 7



111

2

3

22

32

222

322

23

33

232

5

6

52

223

25

