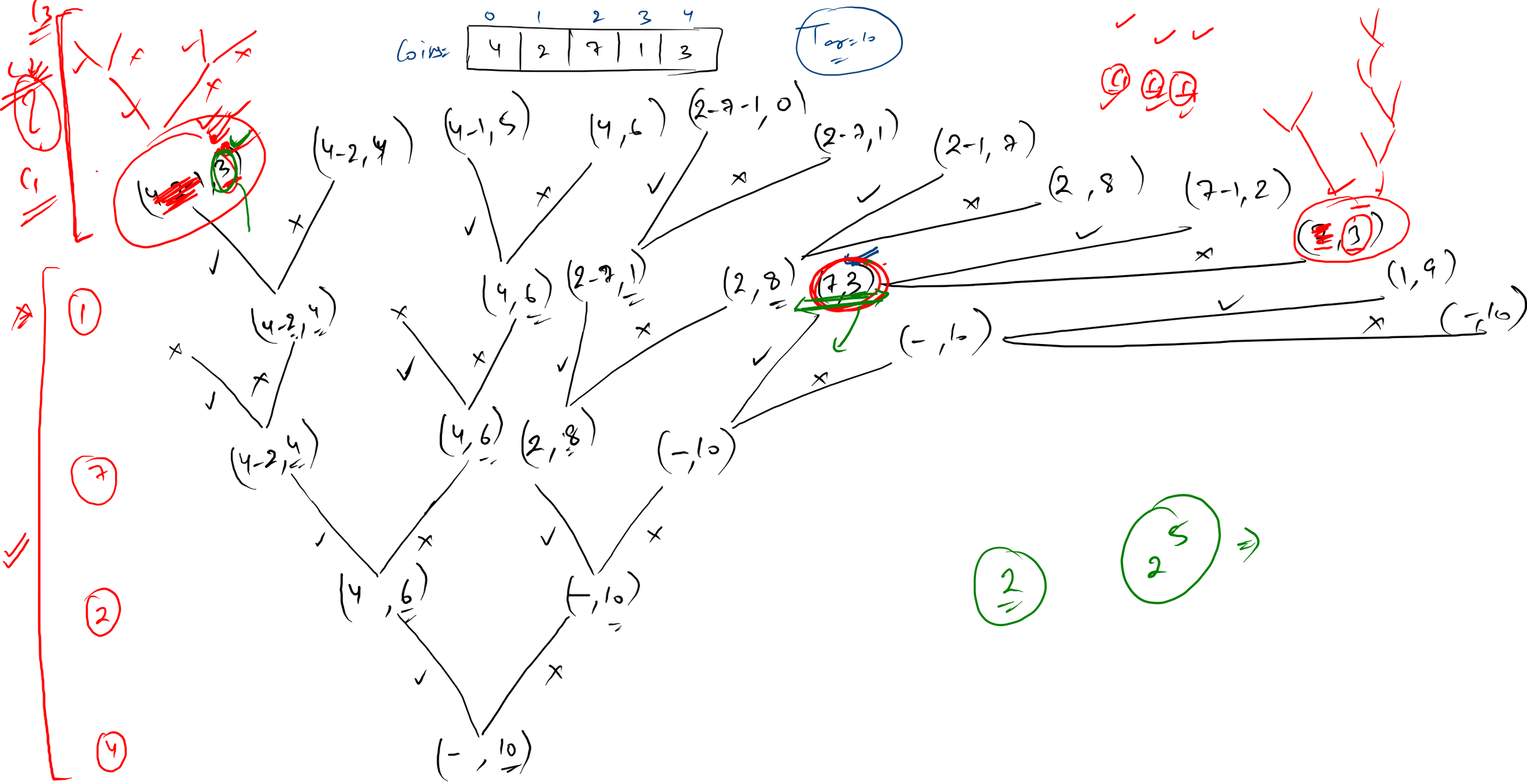
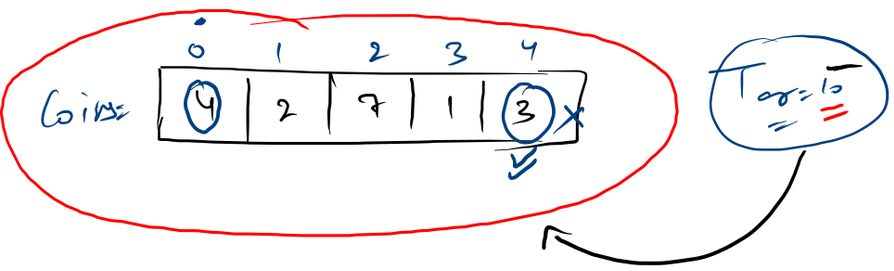
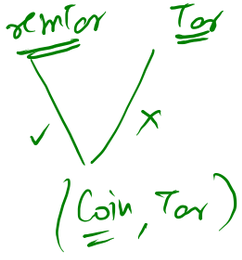
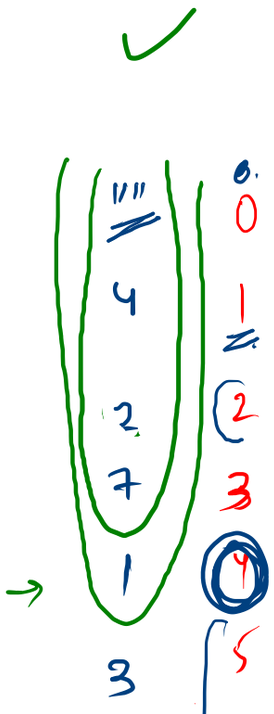

0 1 2 3 4
Coins
4 2 7 1 3
Target = 10
(4-2, 4)
(4-1, 5)
(4, 6)
(2-7-1, 0)
(2-7, 1)
(2-1, 7)
(2, 8)
(7-1, 2)
(1, 9)
(-, 10)
(4-2, 4)
(4, 6)
(2-7, 1)
(2, 8)
(7, 3)
(4-2, 4)
(4, 6)
(2, 8)
(-, 10)
(4, 6)
(-, 10)
(-, 10)
1
7
2
4

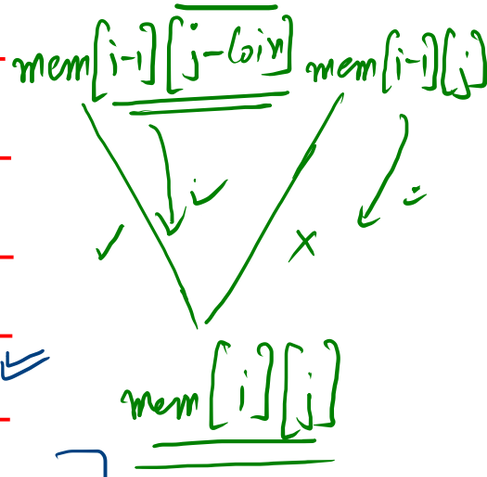




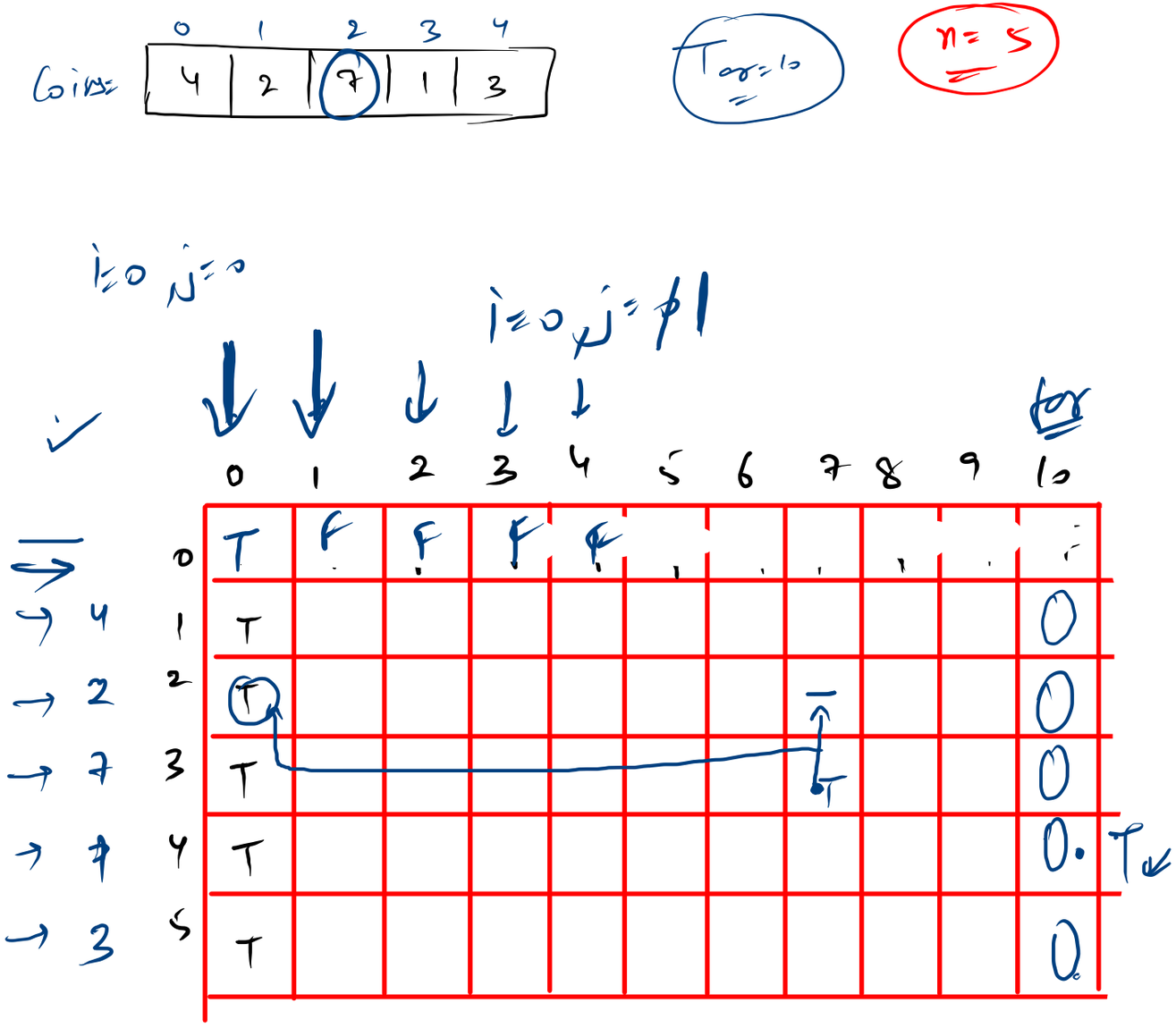


Target

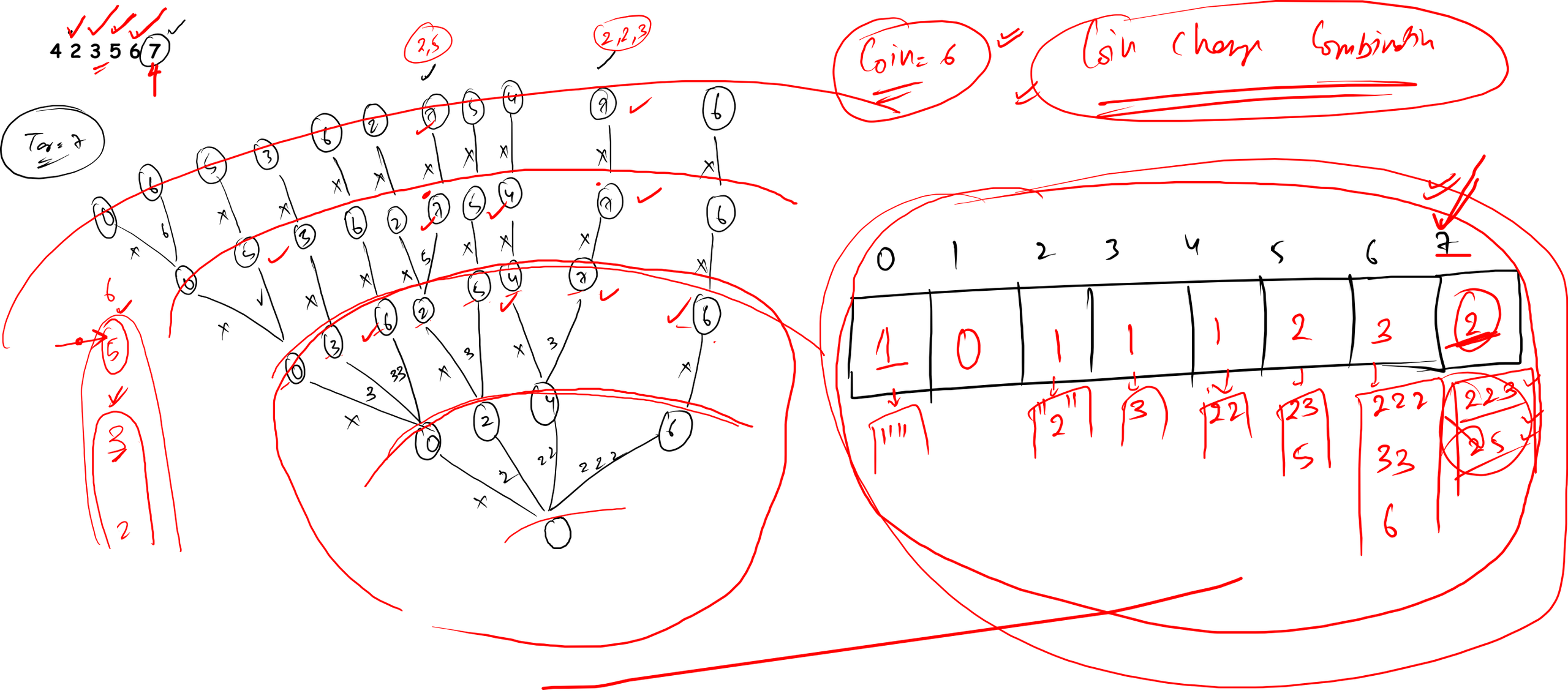
	0	1	2	3	4	5	6	7	8	9	10
0	T	F	F	F	F	F	F	F	F	F	F
1	T	F	F	F	T	F	F	F	F	F	F
2	T	F	T	F	T	F	T	F	F	F	F
3	T	F	T	F	T	F	T	T	F	T	F
4	T	T	T	T	T	T	T	T	T	T	T
5	T										

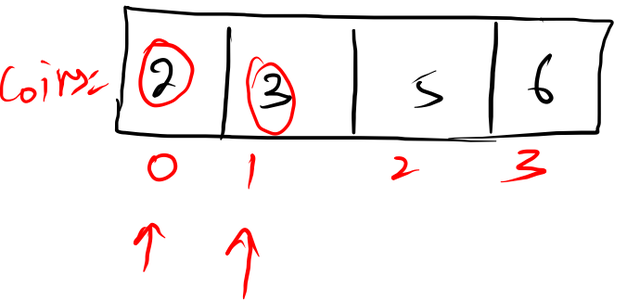


$$mem[i][j] = mem[i-1][j-coin] \;||\; mem[i-1][j]$$

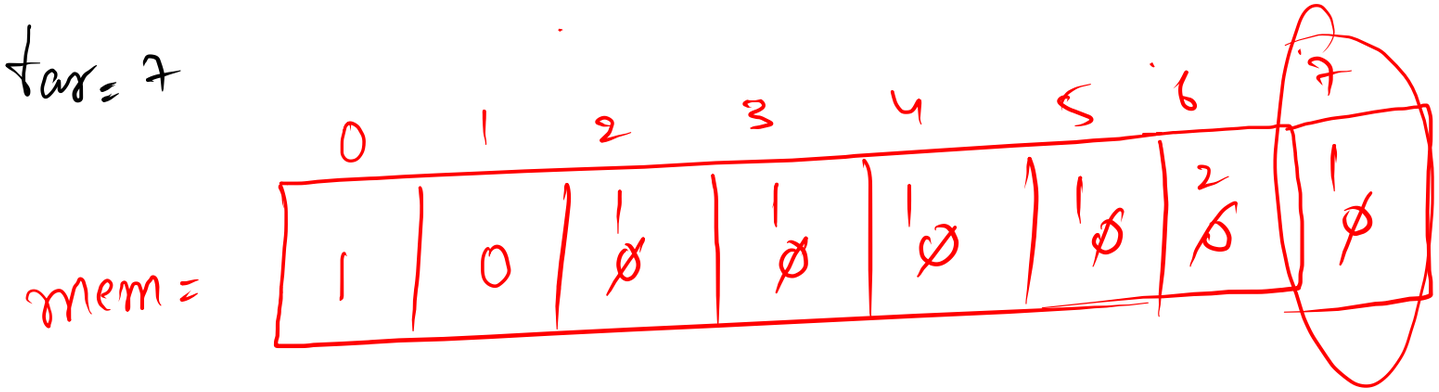



```java
public static boolean tarSumSubset(int coins[],int tar){
    int n = coins.length;
    boolean mem[][] = new boolean[n+1][tar+1];

    for(int i = 0 ; i <= n ; i++){
        for(int j = 0 ;  j <= tar ; j++){
            if(j == 0){
                mem[i][j] = true;
            }else if(i == 0){
                mem[i][j] = false;
            }else{
                int coin = coins[i-1];
                mem[i][j] = mem[i-1][j];
                if(j-coin >= 0){
                    mem[i][j] = mem[i][j] || mem[i-1][j-coin];
                }
            }
        }

        if(mem[i][tar]){
            return true;
        }
    }

    return false;
}
```


4 2 3 5 6 7
Tar = 7
2,5
2,2,3
Coin = 6
Coin change Combination
0 1 2 3 4 5 6 7
1 0 1 1 1 2 3 2
1111
2"
3
22
23
5
222
33
6
223
25

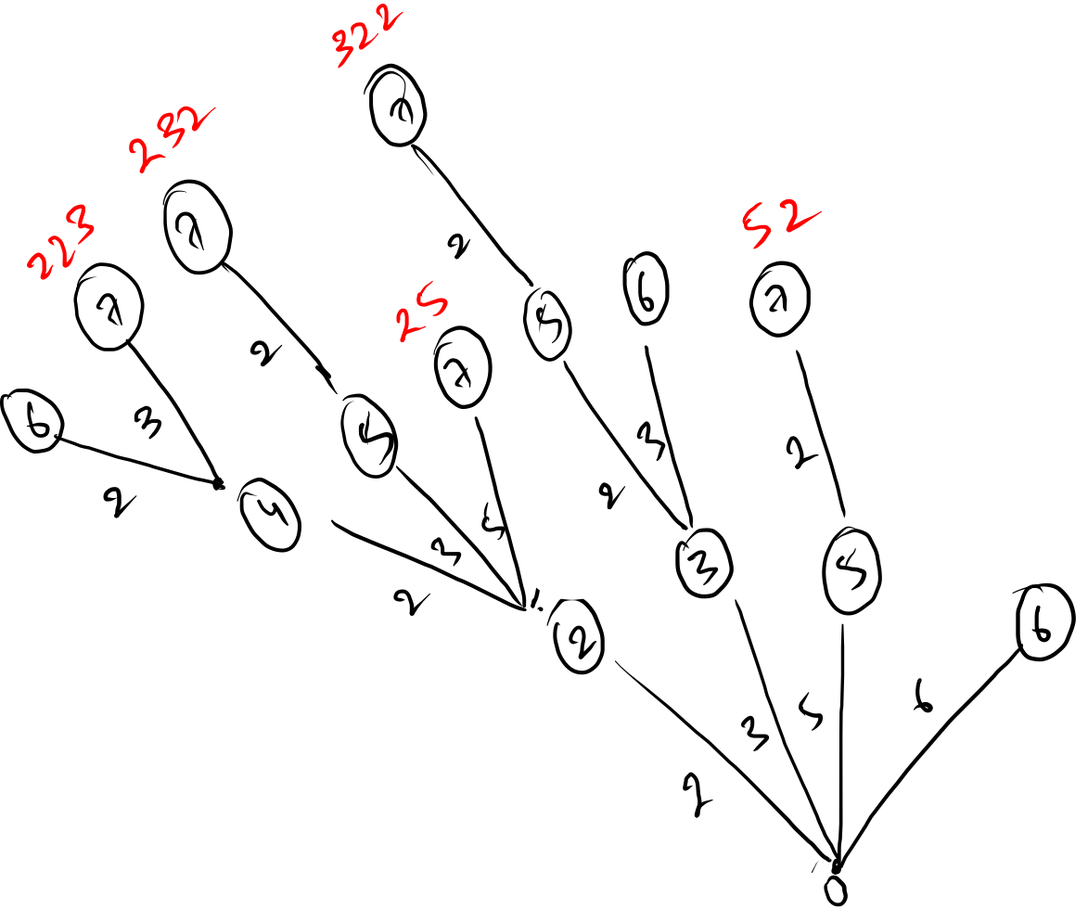


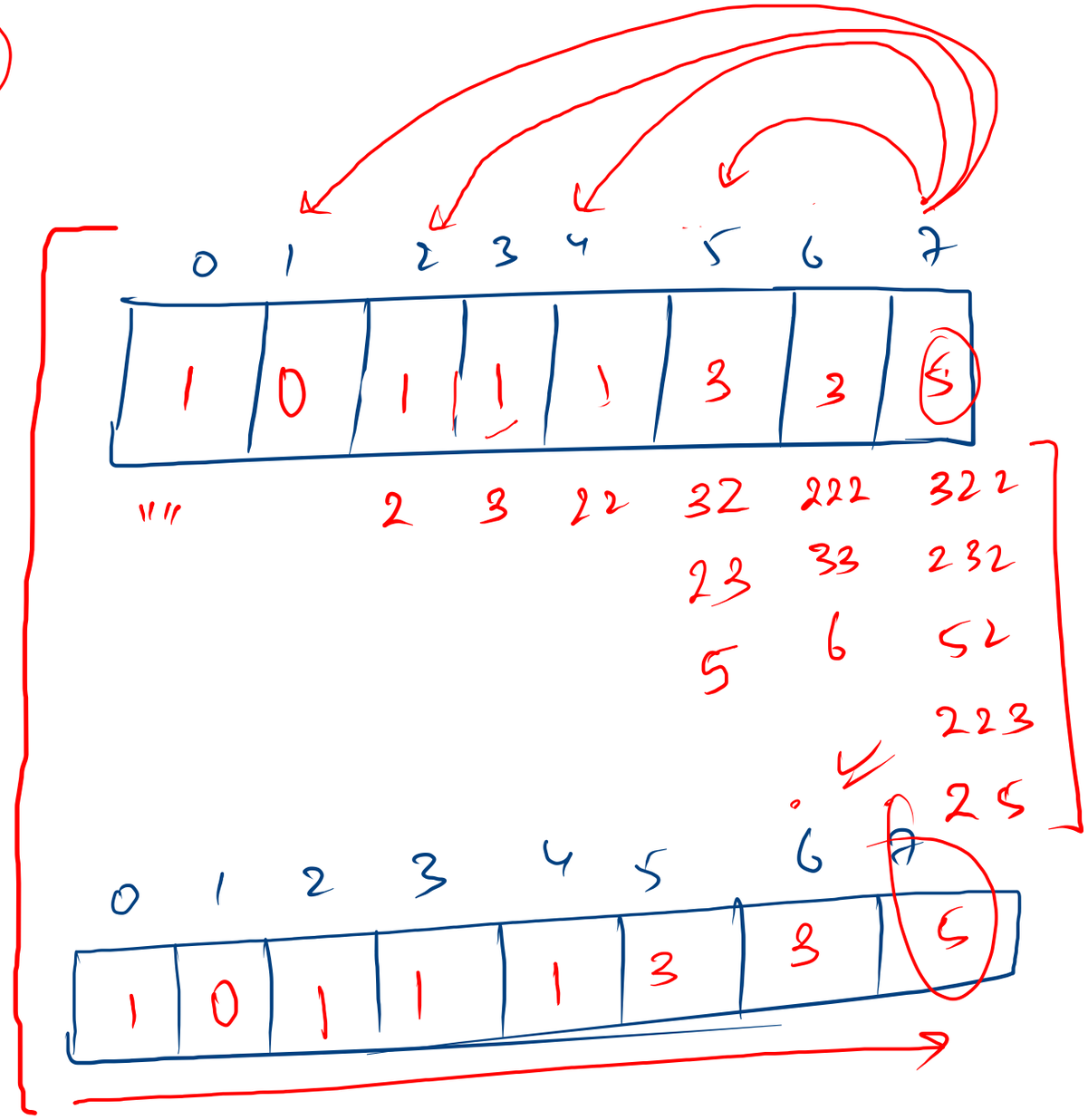
tar = 7




```java
public static int coinChangeCombination(int coins[],int tar){
    int mem[] = new int[tar+1];

    mem[0] = 1;

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

    return mem[tar];
}
```


2 3 5 6
Top => 7




0 1 2 3 4 5 6 7
1 0 1 1 1 3 3 5
2 3 22 32 222 322
23 33 232
5 6 52
223
25
0 1 2 3 4 5 6 7
1 0 1 1 1 3 3 5

5 ← n → items

15 14 10 45 30] value

2 5 1 3 4] wts

7] Cap

vls:

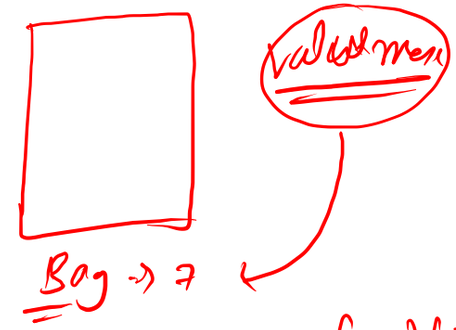
0	1	2	3	4
15	14	10	45	30

wts:

0	1	2	3	4
2	5	1	3	4

Cap = 7

H.W.



$$mem[i][j] = Max(\underbrace{mem[i-1][j]}_{exc}, \underbrace{mem[i-1][j-wt]+vl}_{inc})$$

ans

↳ mem[n][Cap]

15 − 2

14 − 5

10 − 1

45 − 3

30 − 4

i \ j	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	0	15	15	15	15	15	15
2	0	0	15	15	15	15	15	29
3	0	10	15	25	25	25	25	29
4	0	10	15	45	55	60	70	70
5	0	10	15	45	55	60	70	75

mem[i-1][j-wt]

(i-1, j-wt)

mem[i-1][j]

(i-1, j)

(i, j)

✓ X

n/2

n/4

n/n

⇒ O(n)

vl = vls[i-1]

wt = wts[i-1]

Q ≡ Find maximum value provided each item can only bought once & the product should be under net-wt constraint?

5 n → items

15 14 10 45 30] value

2 5 1 3 4] wts

7] Cap

vls:

0	1	2	3	4
15	14	10	45	30

wts:

0	1	2	3	4
2	5	1	3	4

Cap = 7

Unbounded Knapsack

Bag → 7

value/mem

H.W.

mem[i] = Max(mem[i], mem[i-wt] + val)

Items → Infinite time

0	1	2	3	4	5	6	7
0	10	20	45	55	65	90	100

1	2	3	4	5	6	7
1	~~2~~	~~2~~	~~22~~	~~22~~	~~222~~	~~222~~
	11	~~111~~	~~1111~~	~~11111~~	~~111111~~	~~1111111~~
		3	13	113	33	133

313

331

Permutation

Combination

1 + 3 + 3

3 + 1 + 3

3 + 3 + 1