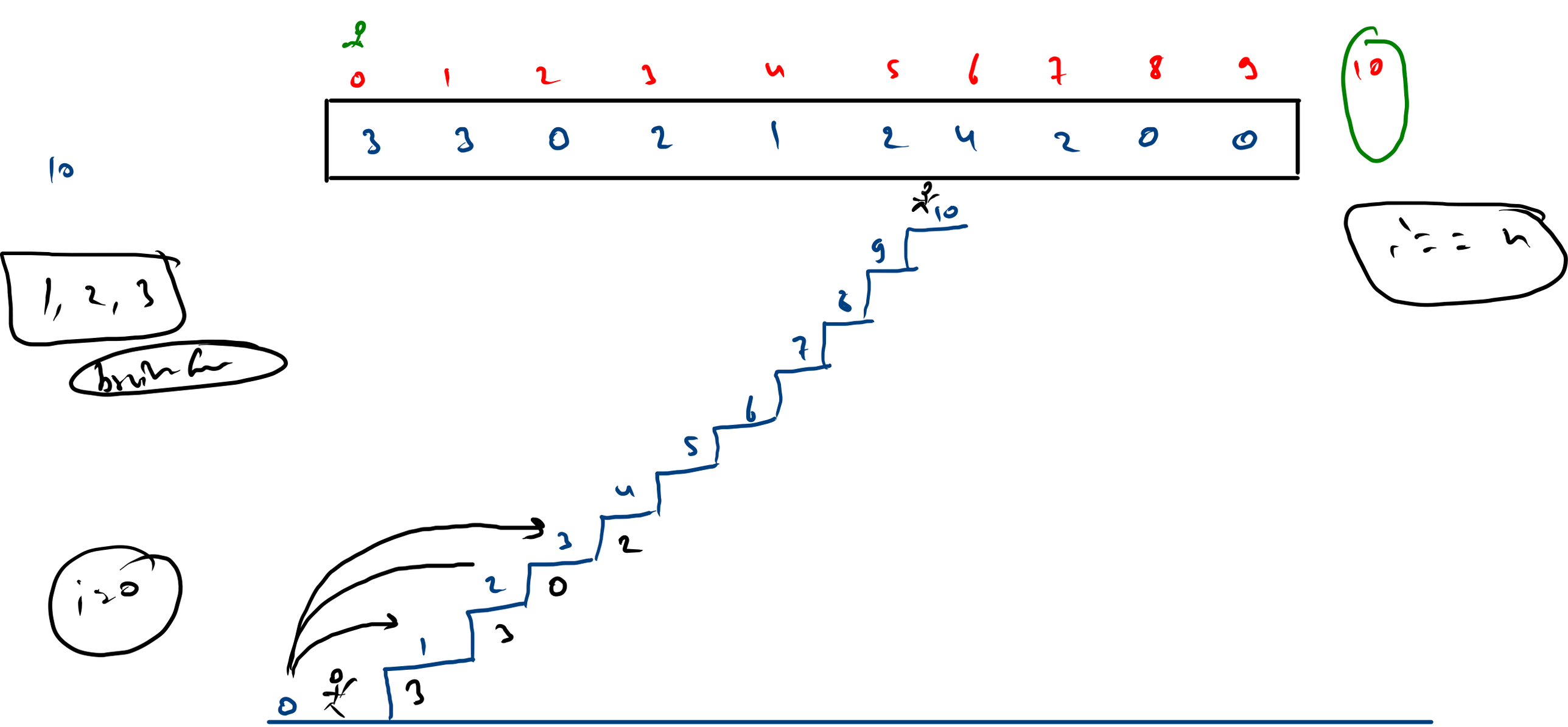
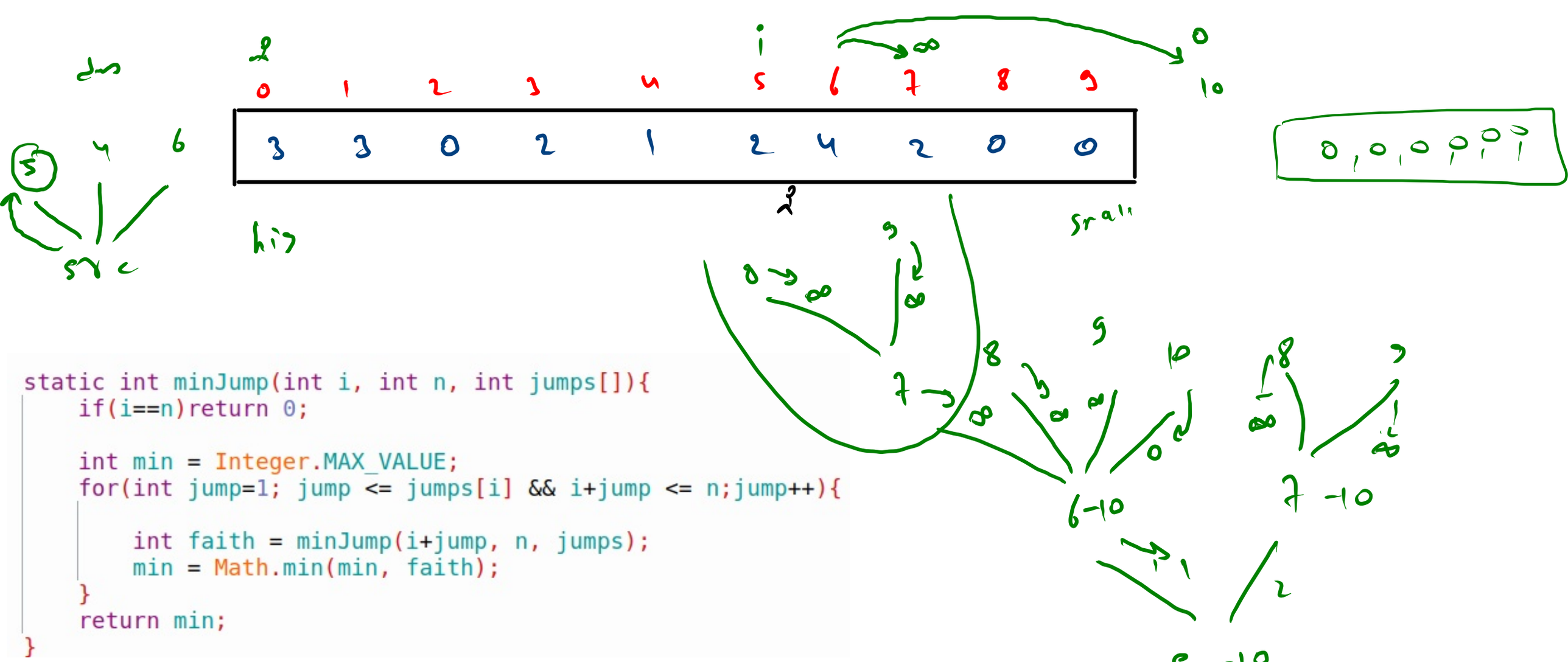

i
0 1 2 3 4 5 6 7 8 9
3 3 0 2 1 2 4 2 0 0
10
10
1, 2, 3
i >= 0
0
1
2
3
4
5
6
7
8
9
10
3
2
2
0
1
3
0
3
2




```java
static int minJump(int i, int n, int jumps[]){
    if(i==n)return 0;

    int min = Integer.MAX_VALUE;
    for(int jump=1; jump <= jumps[i] && i+jump <= n;jump++){

        int faith = minJump(i+jump, n, jumps);
        min = Math.min(min, faith);
    }
    return min;
}
```

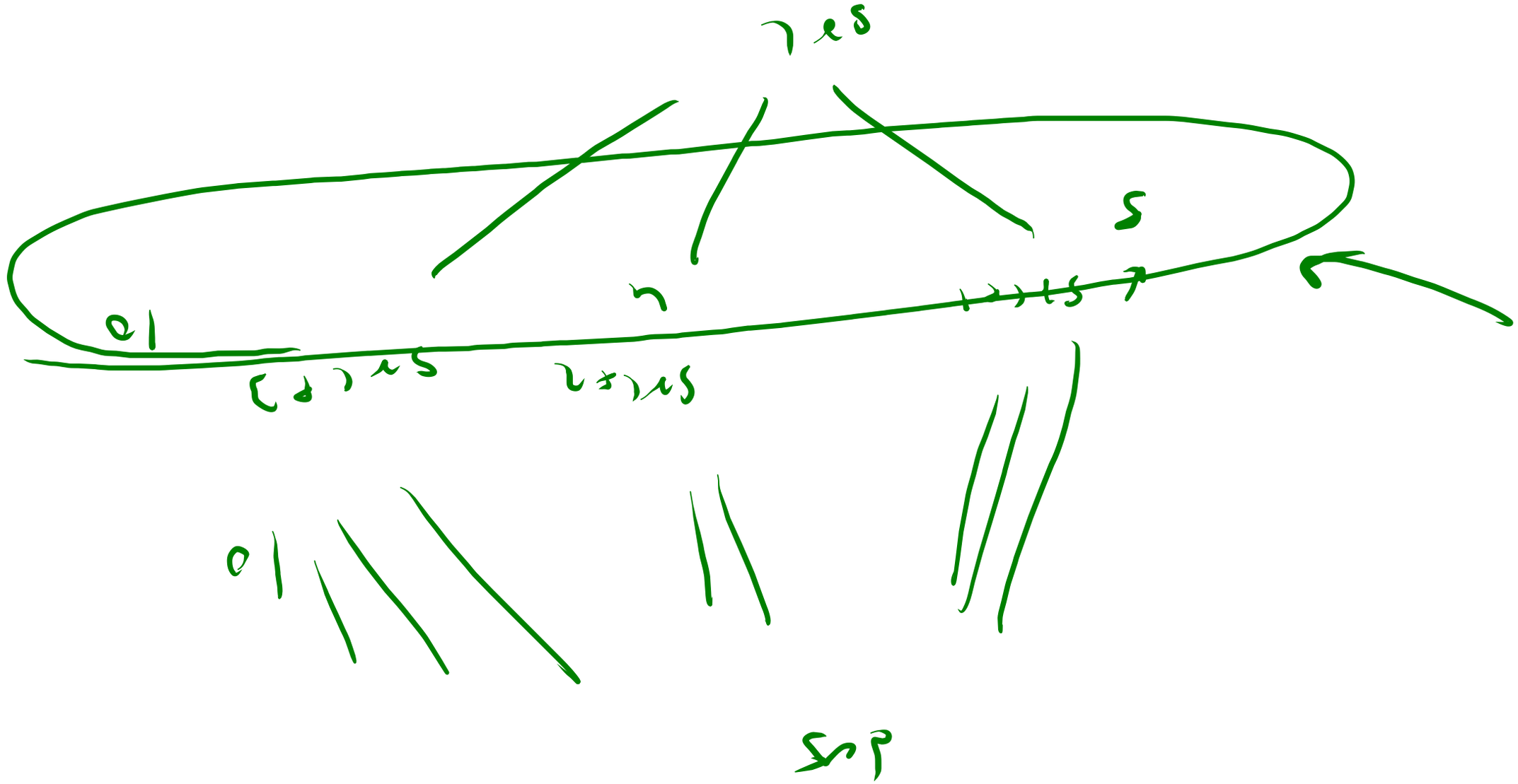
memo → Tab

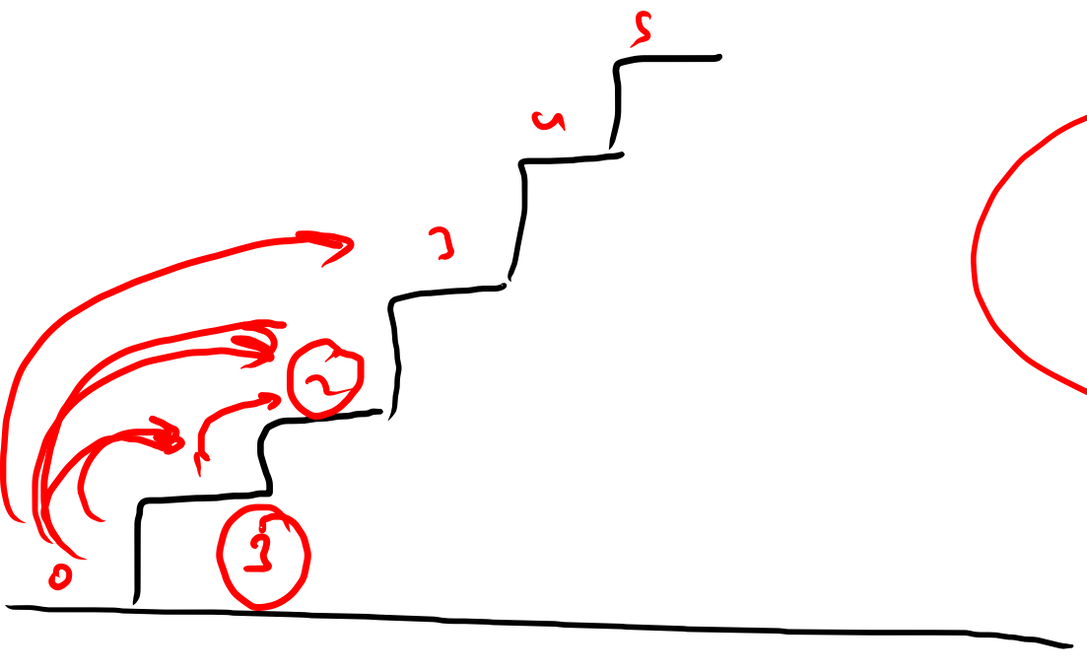
memo → return 2

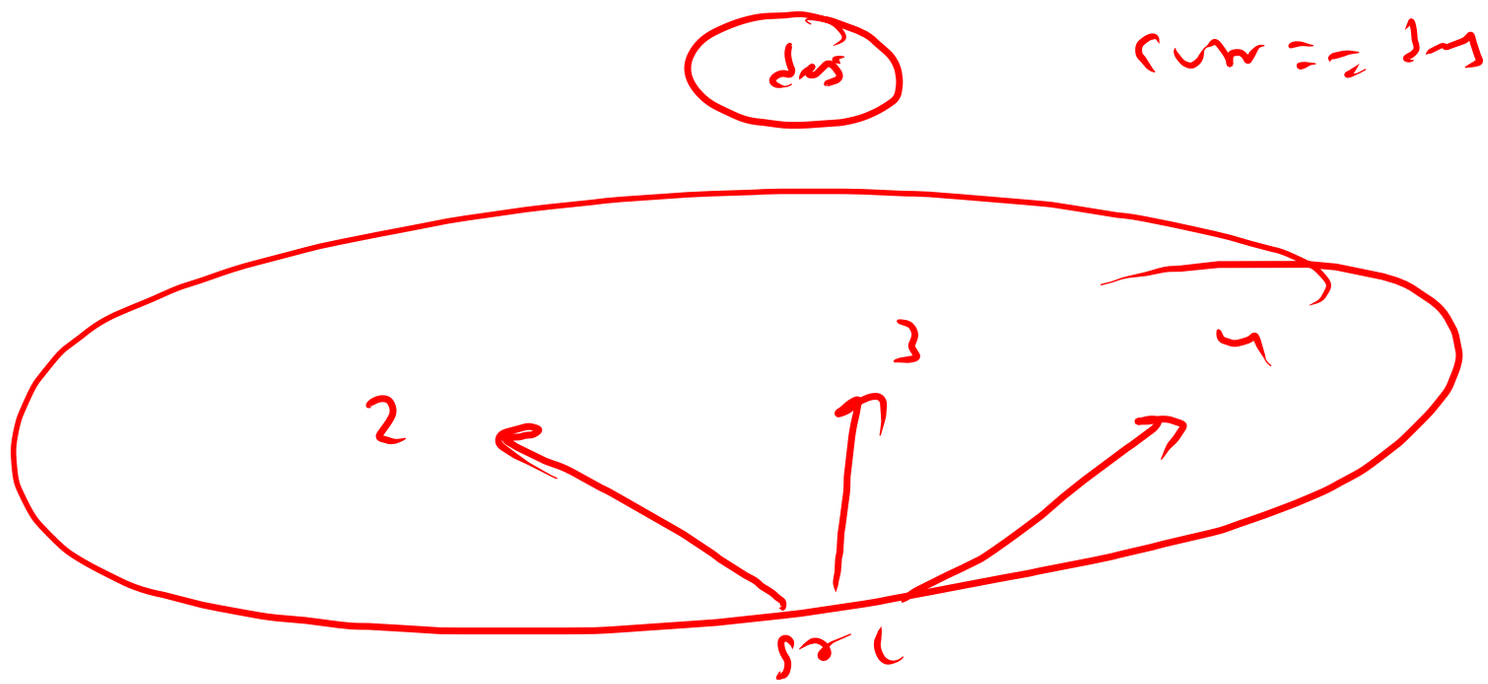
change n to i

change return to ans, continue

calls



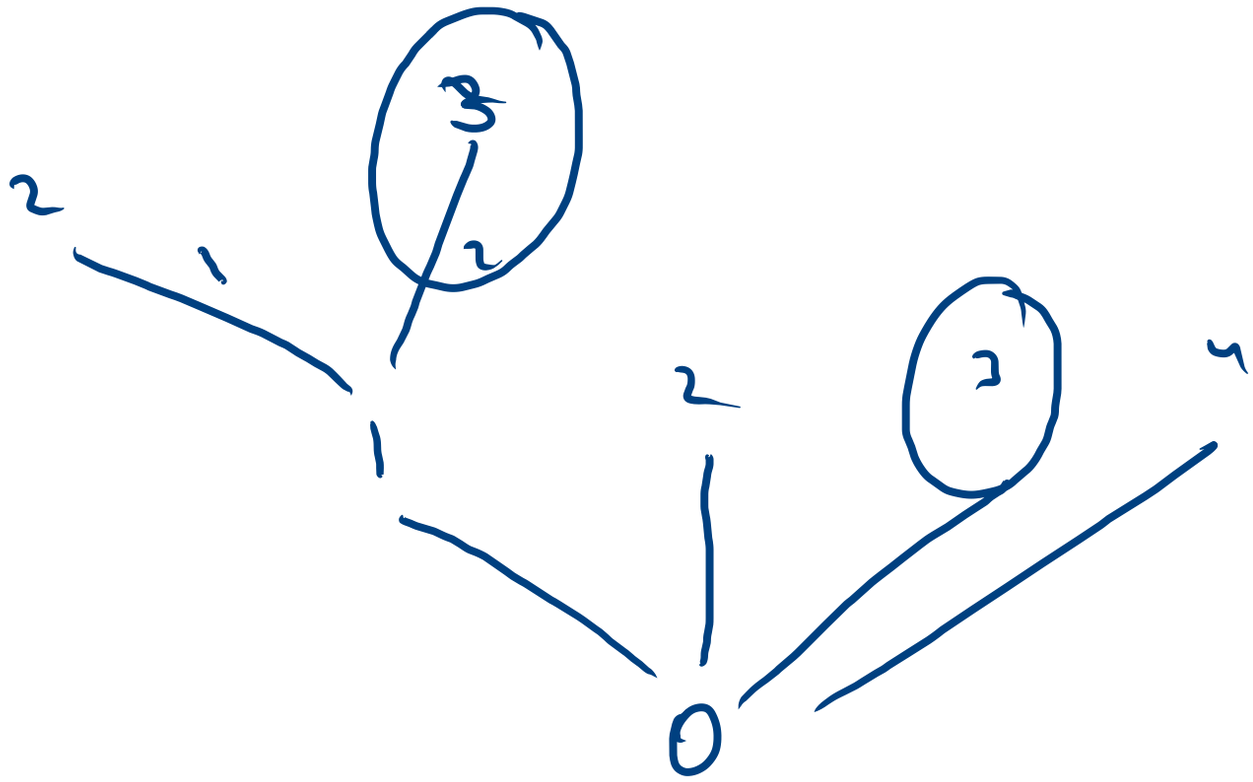

0
1
2
3
4
5

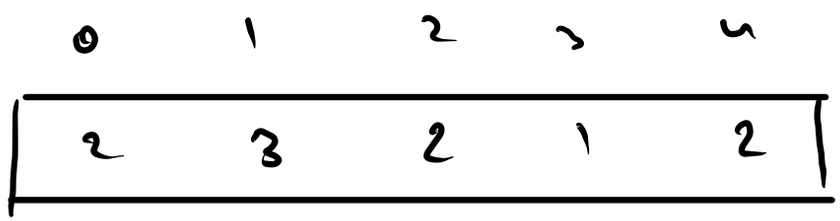

1
2
3
4
5

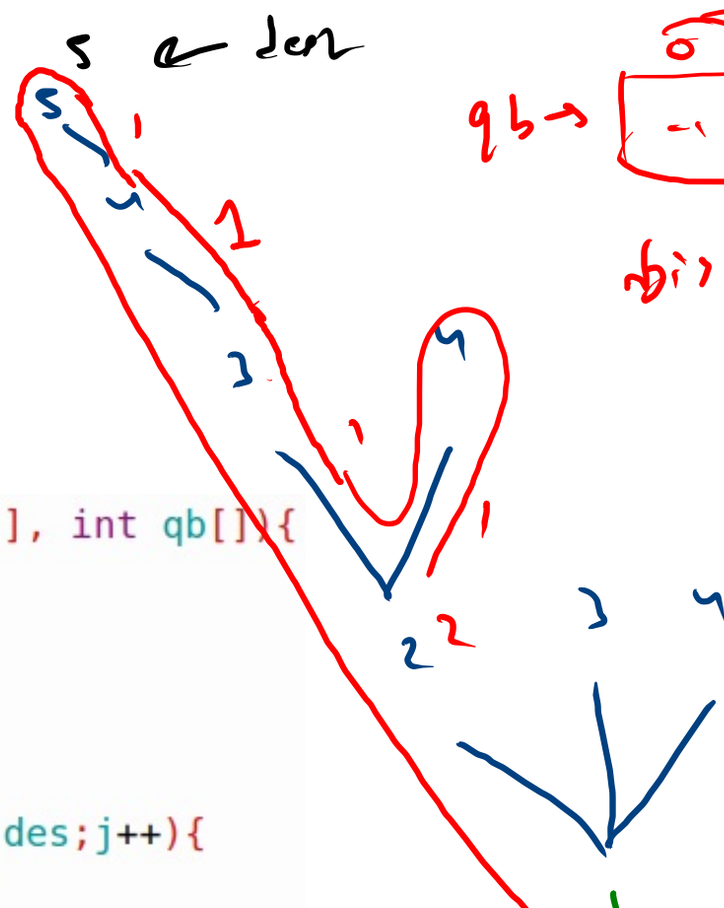
0 1 2 3 4

4 0 0 0 dell

I

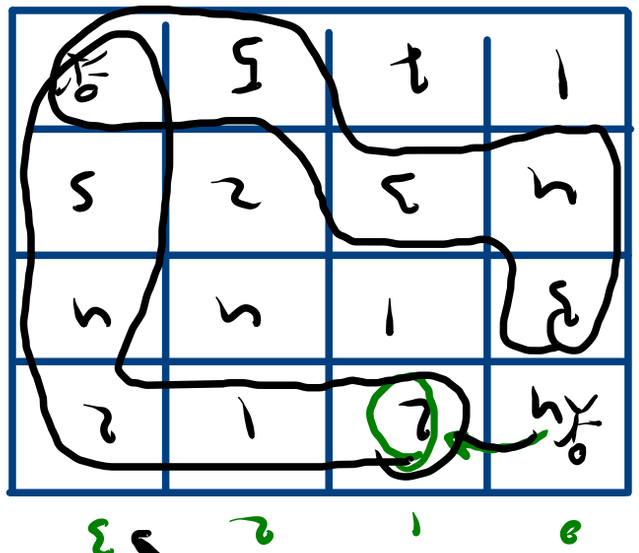


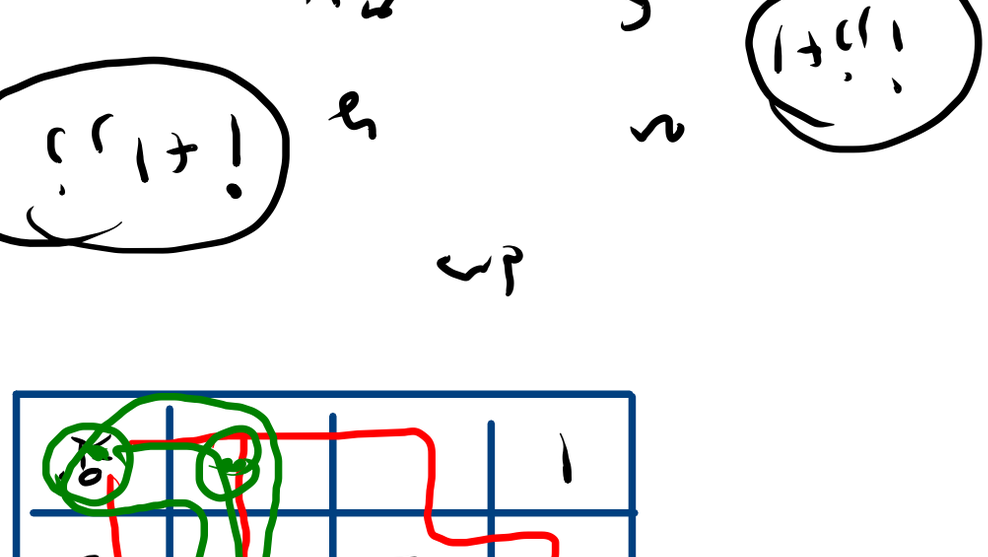


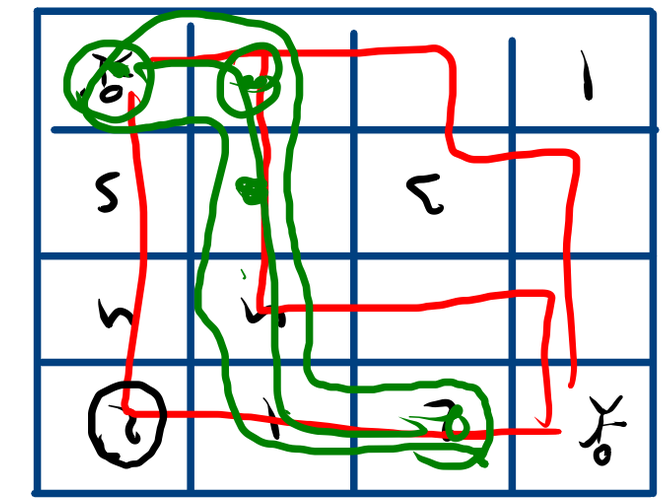


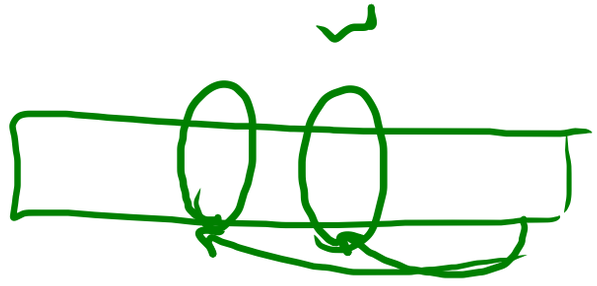

```java
static int rec(int curr, int des, int jumps[], int qb[]){
    if(curr == des)return 1;
    // if(curr>des)return 0;

    if(qb[curr] != -1)return qb[curr];

    int ans=0;
    for(int j=1;j<=jumps[curr] && j+curr <= des;j++){
        ans += rec(curr+j, des, jumps, qb);
    }

    qb[curr] = ans;
    return ans;
}
```

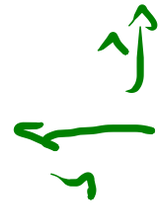
92427085 1
20732231 5
50412414 6
01428243 6

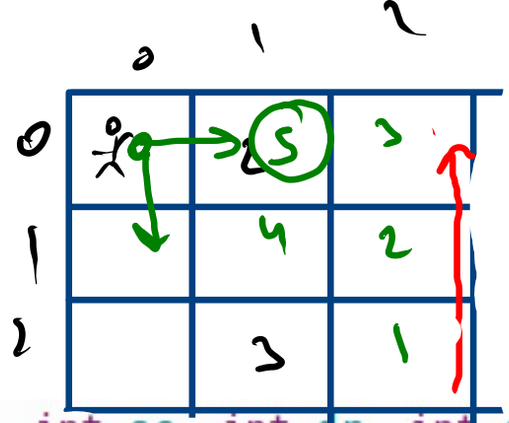


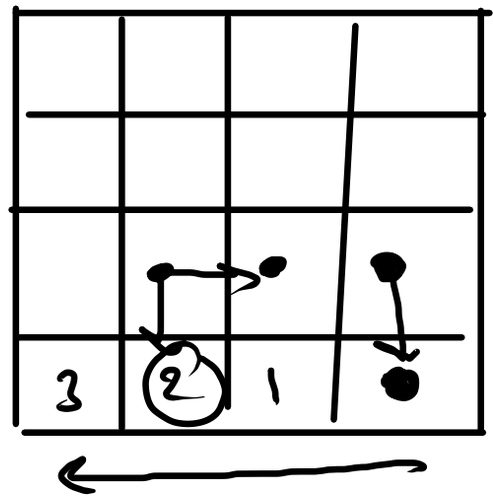




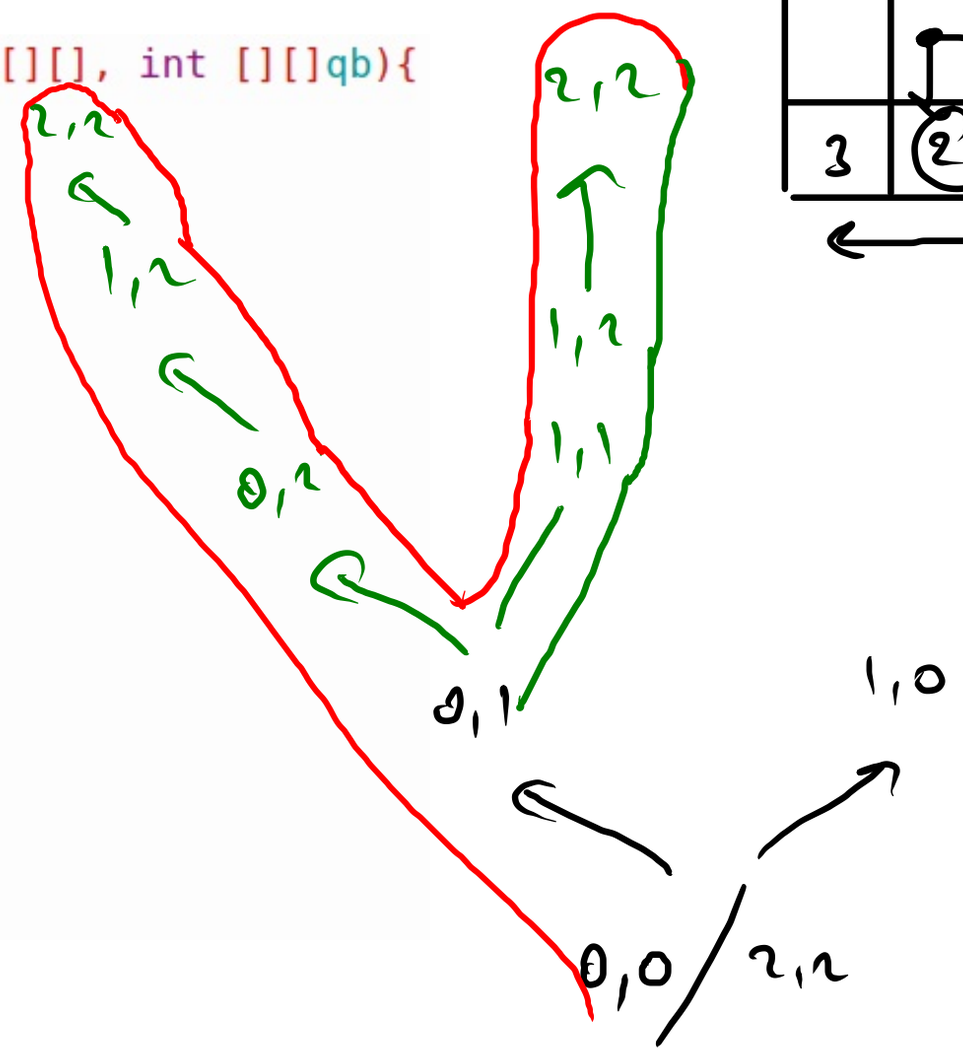









```java
static int rec(int sr, int sc, int dr, int dc, int cost[][], int [][]qb){
    if(sr == dr && sc == dc)return cost[dr][dc];

    if(qb[sr][sc] != -1)return qb[sr][sc];

    int hFaith = Integer.MAX_VALUE;
    int vFaith = Integer.MAX_VALUE;

    if(sc != dc){
        hFaith = rec(sr, sc+1, dr, dc, cost, qb);
    }
    if(sr != dr){
        vFaith = rec(sr+1, sc, dr, dc, cost, qb);
    }

    int ans = Math.min(vFaith, hFaith) + cost[sr][sc];
    qb[sr][sc] = ans;
    return ans;

}
```



min con max pair

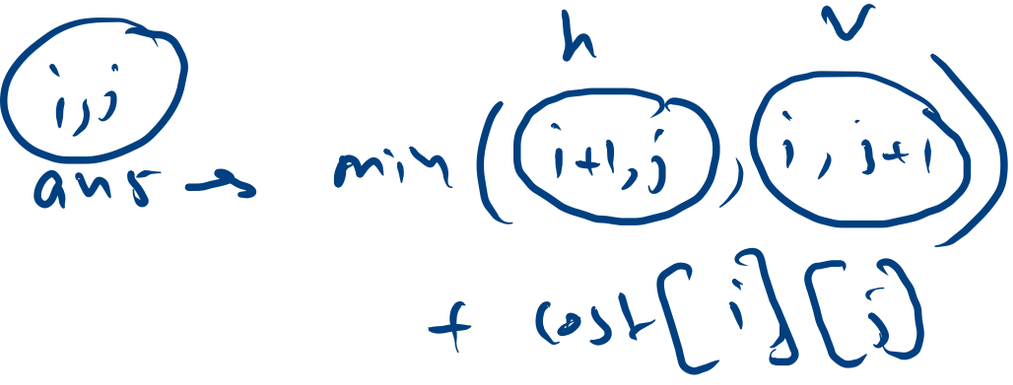
cost

4
4
4 0 1 5
4 2 2 1
1 0 1 0
3 3 2 3

4	0	1	5
4	2	2	1
1	0	1	0
3	3	2	3

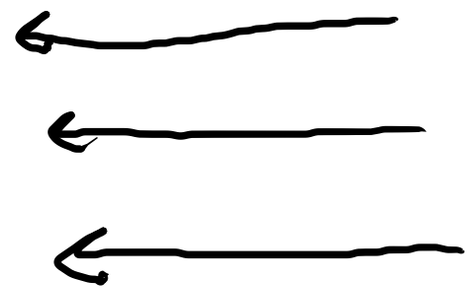
qb

10	6	7	9
9	6	6	4
5	4	4	3
11	8	5	3



qb

9	6	6	4
5	4	4	3
11	8	5	3



h x n

Tabulation

Storage

meaning

direction

memo → Tab

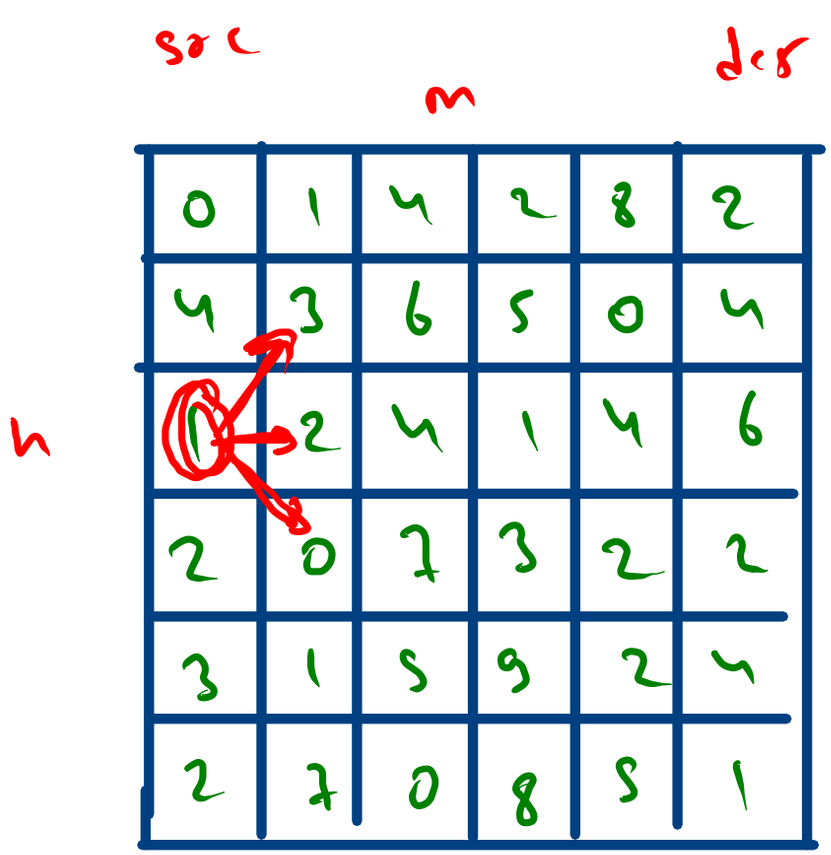
memo → recursion 2

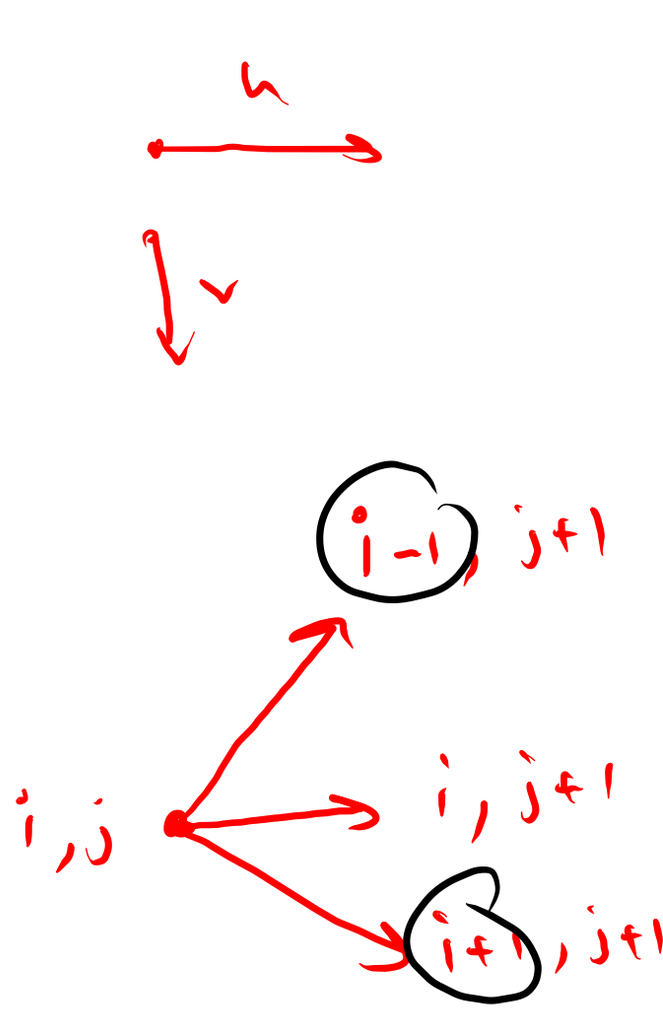
change h to i

change return to ans, continue

calls

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

dst

n

h

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

qb

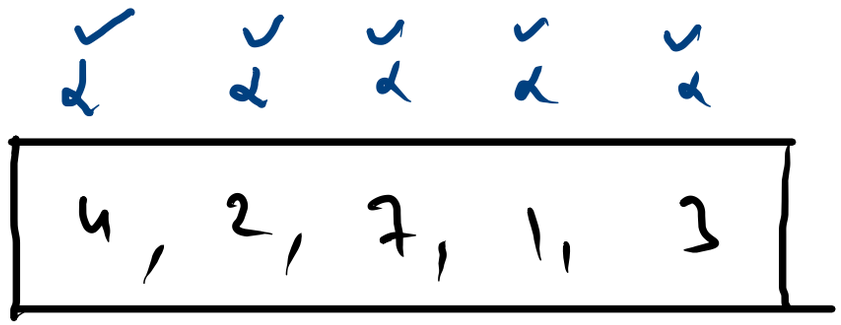
h x m

c

r

4	-	-	4	12	2
6	-	-	-	6	4
1	-	-	11	10	1
0	-	-	-	8	2
1	-	-	-	6	4
3	-	-	-	0	1

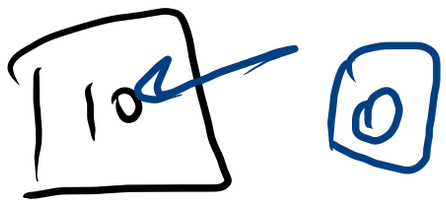
```
qb = new int [h][w]

j ← c to 0 {
    i ← 0 to r {
        init max → gold[i][j]
    }
}
```

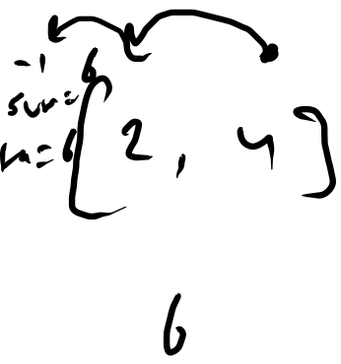


7+7 == True

False

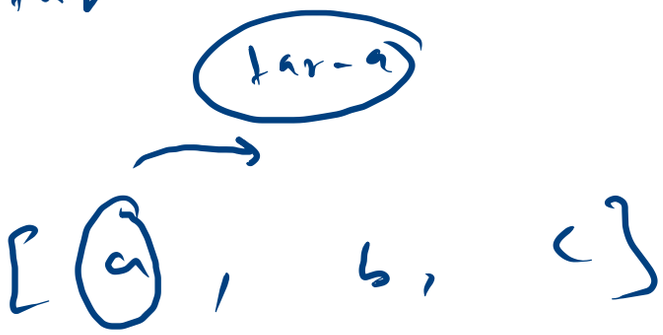
5 4 2 7 1 3 10



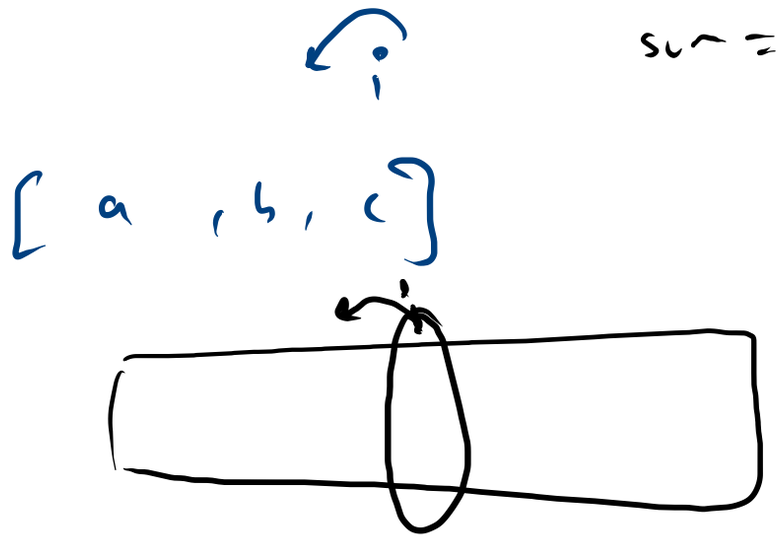


6

tar



sum, tar

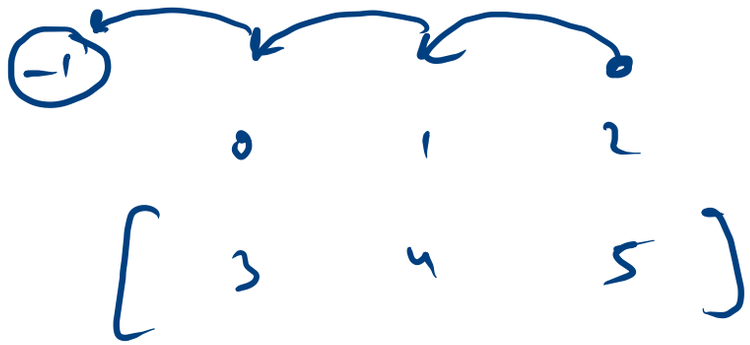


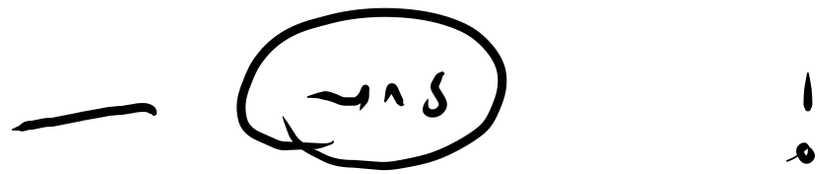
sum == tar

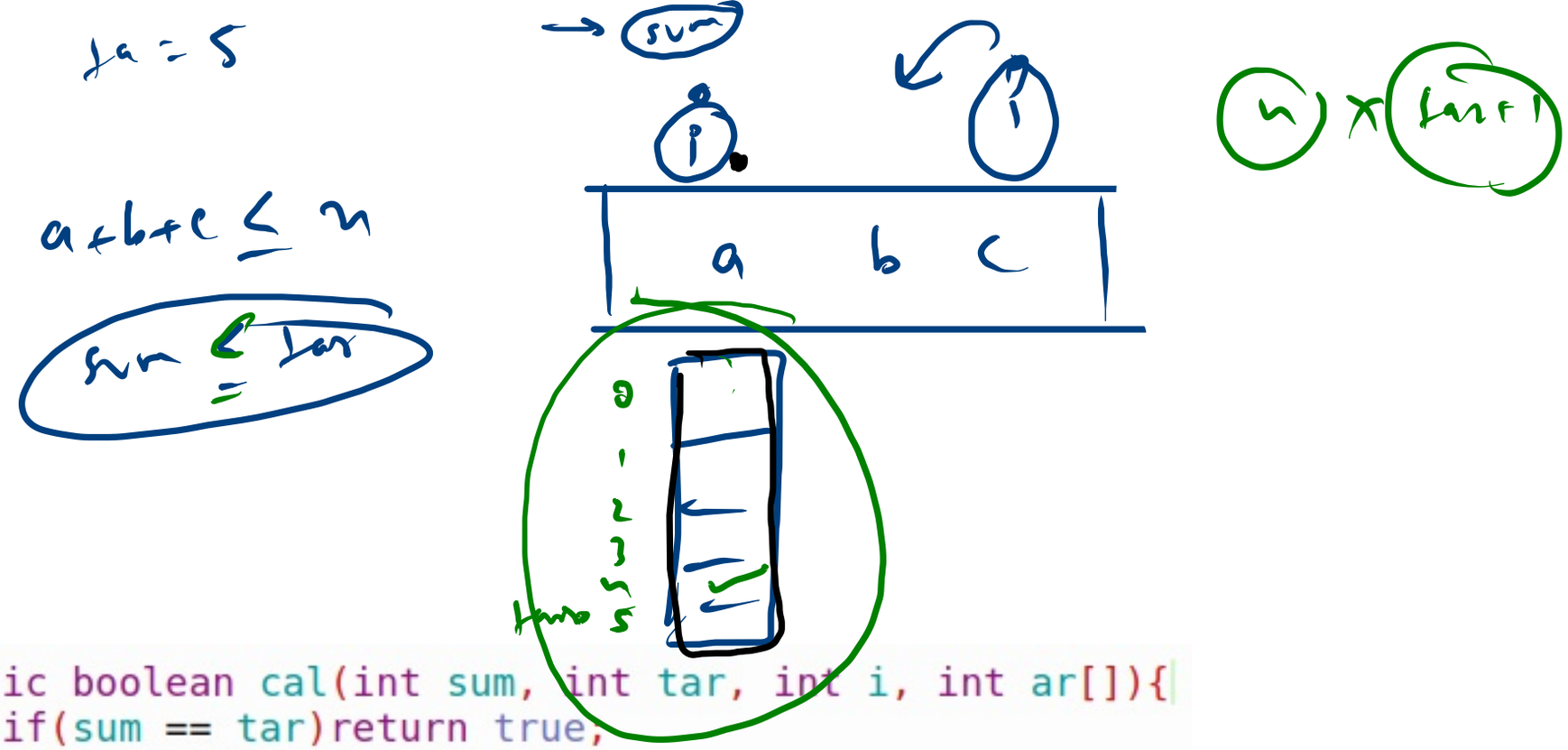
1 / 5 / 100

2 / 0 / 10

10







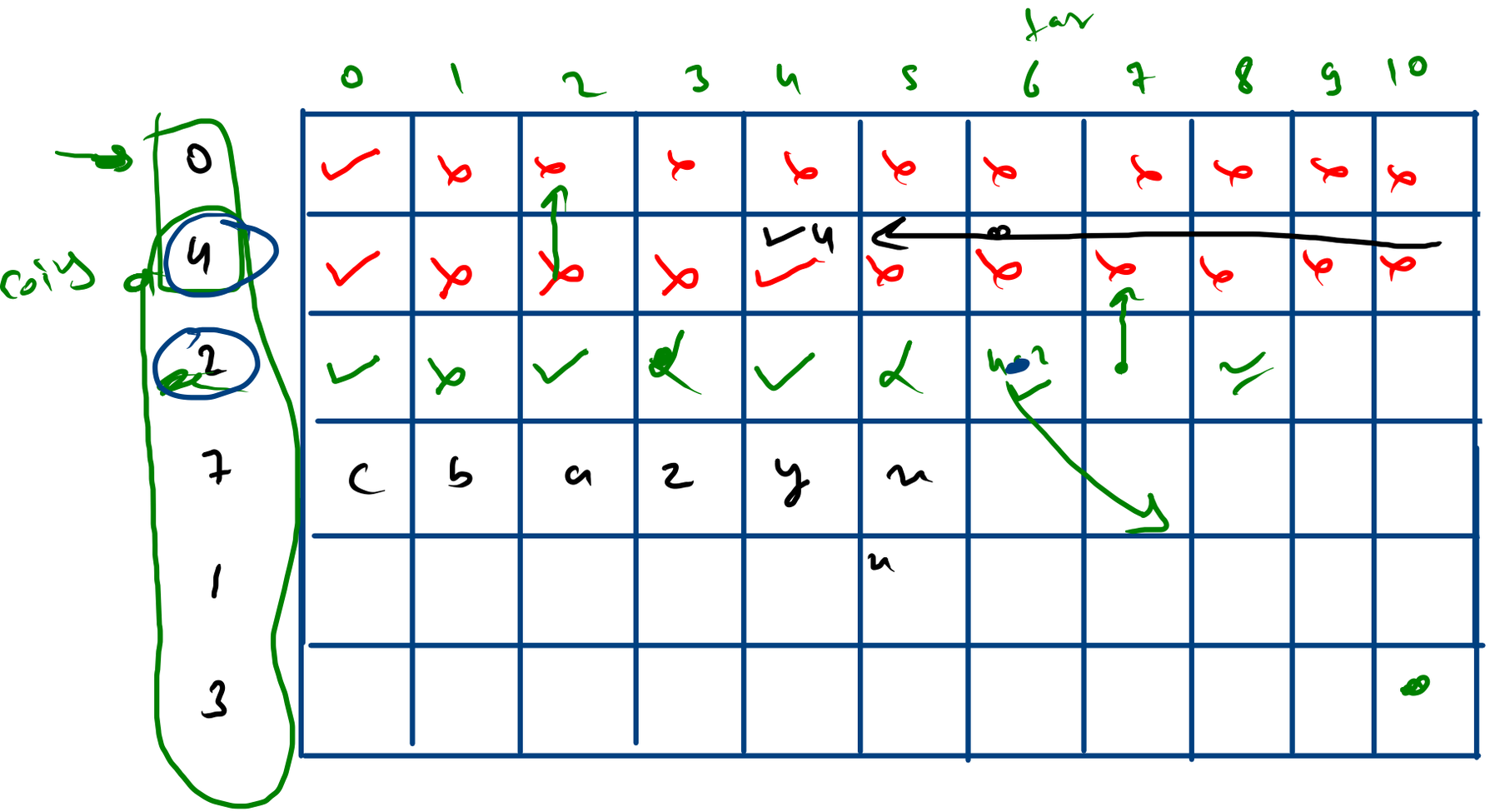

```
static boolean cal(int sum, int tar, int i, int ar[]){
    if(sum == tar)return true;
    if(i == -1)return false;

    if(sum + ar[i] <= tar){
        boolean can = cal(sum+ar[i], tar, i-1, ar);
        if(can)return true;
    }
    return cal(sum, tar, i-1, ar);
}
```

$$\begin{bmatrix} 4 & 2 & 2 & 1 & 3 \end{bmatrix}$$

$\tan x = 10$

coin = [4 2 7 1 3] tar = 10

strms = hr(tar+1)



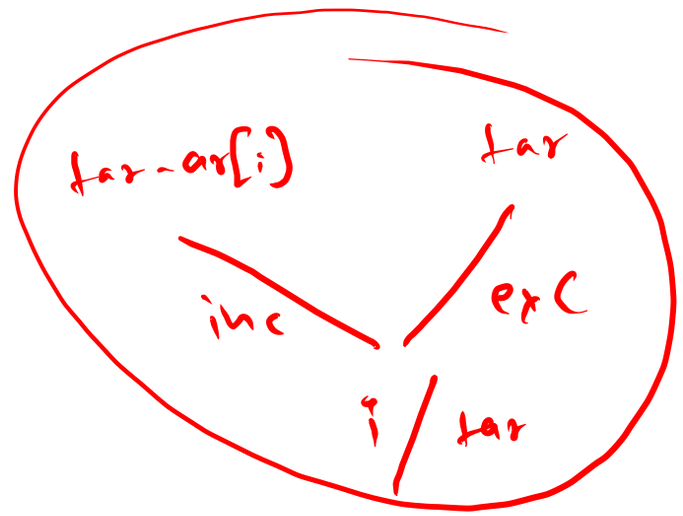
meaning

Travel sol

include

qb[i-1][t-arr[i]]

excl qb[i-1][t]



coins → [2 3 5 6]

x 1, 2, 3, 4, 5

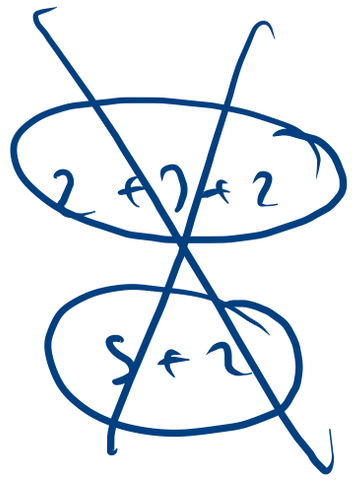
target → 7

2+2+2+1

8

2 + 2 + 3

2 + 5

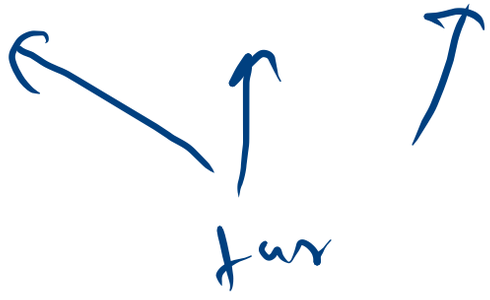


coins x rep

423567

tar = 0

1



coins → [2, 3, 5, 1]

target → 7

0	1	2	3	4	5	6	7
ans → 1	∞ 1	∞ 2	∞ 3	∞ 4	∞ 2	2	∞ 2
•	1	•2 1+1	•3 2+1 1+1+1	•2+2 3+1 2+1+1 1+1+1+1	2+3 5	2+2+1 3+3	2+2+3 2+5

4-2 = 2

6-2 = 4

3-2 = 0

2 - 5

2
1+1

0 1 2 3

coins → [2 3 5 6]

target → 7

2+2+3

2+3+2

3+2+2

2+5

5+2

Com

per

2

combin

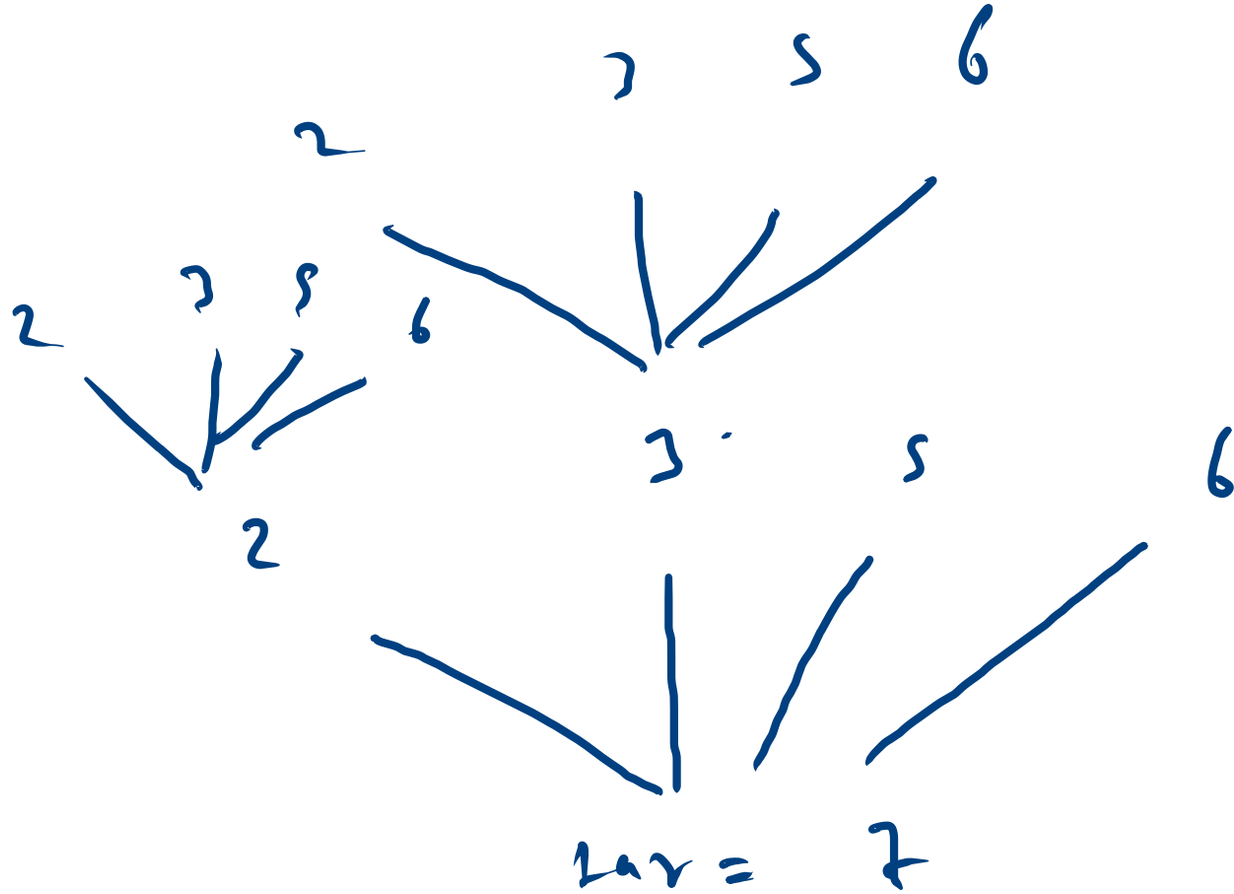
1 → 3

2,3 5 6

3

coins → [2 3 5 6]

target → 7



coins → [2 3 5 1]

0	1	2	3	4	5	6	7
1	~~0~~ 1	x2	x3	x4	x2	2	~~0~~ 2
.		2	3	2+2	2+3	2+2+2 3+3	2+2+3

coins → [2 3 5 1]

2-1 = 1

3-2 = 1

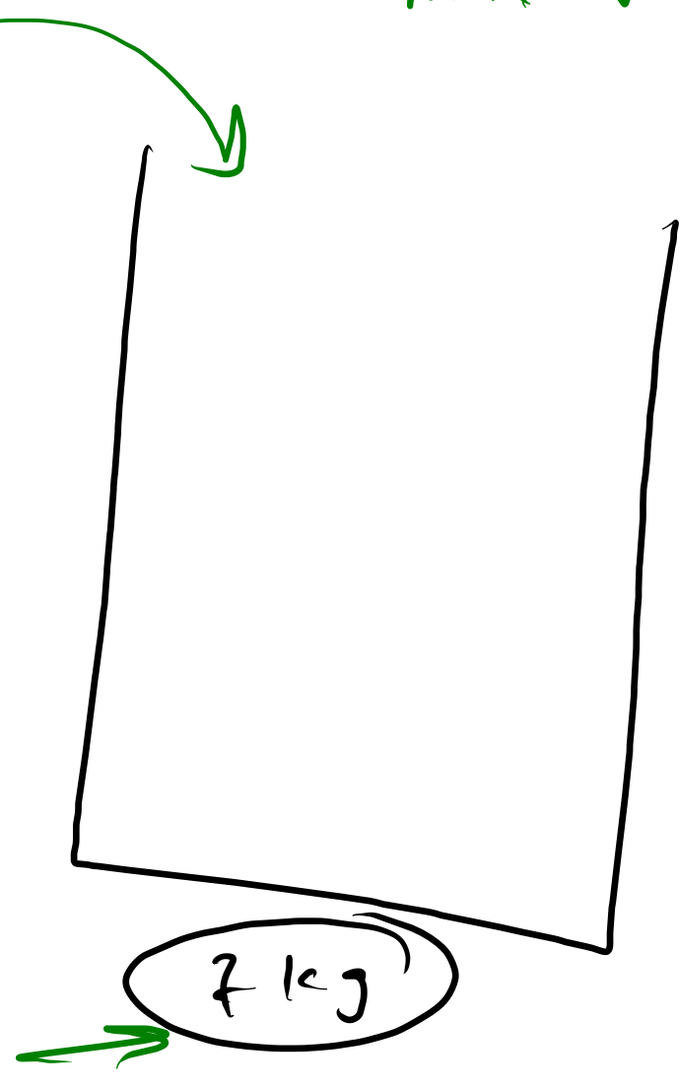
0	1	2	3	4	5	6	7
1	1	1+2	1				
	1	2	1+2	2 2			
		1+1	3	1 1 2			
			2 1	1 3			
			1 1 1	1 2 1			
			2	3 1			
			3	2 1			
			5	1 1 1 1			
			1				

0 1

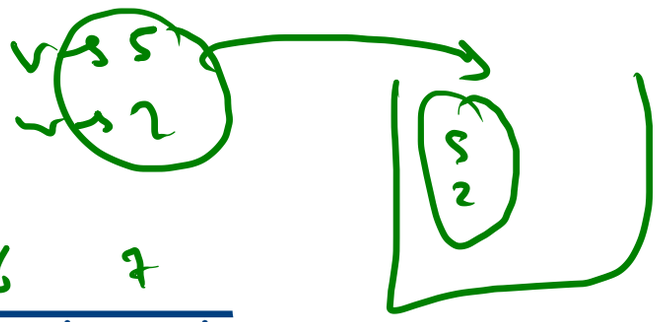
5
15 14 10 45 30
2 5 1 3 4
7

14	45	30	15	10
5	3	4	2	1

max value

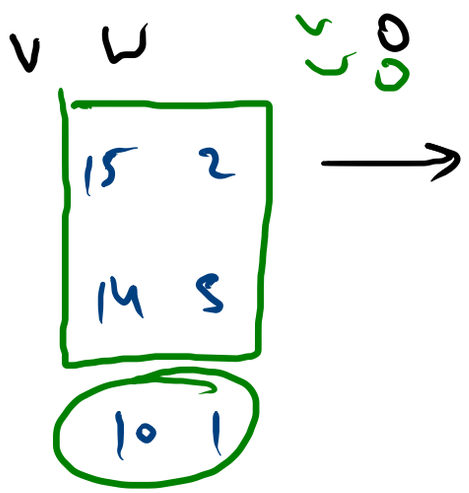


2 5
7 5
5 5



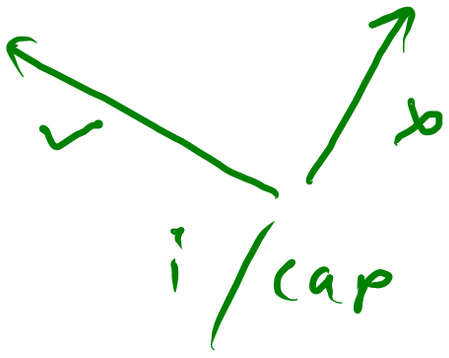
5
15 14 10 45 30
2 5 1 3 4
7

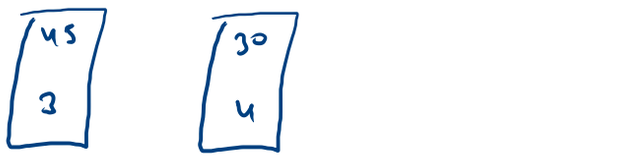
15

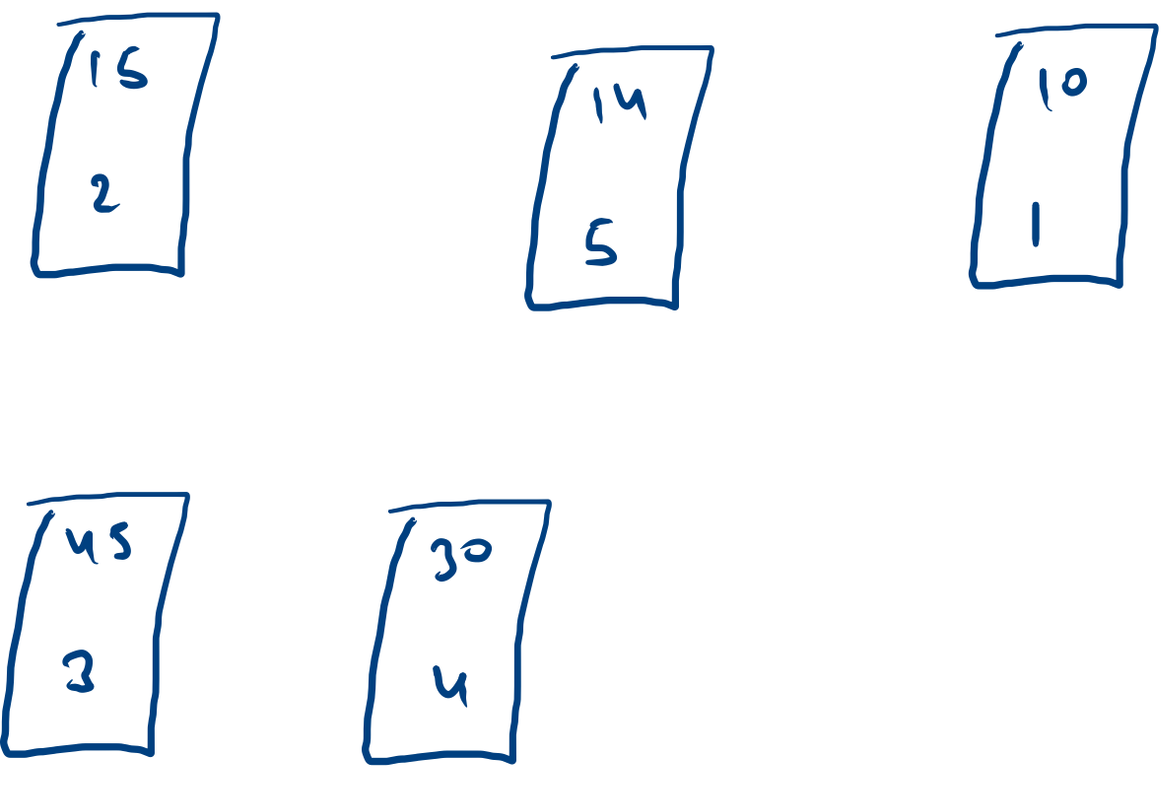


0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0
0	0	15	15	15	15	15	15
0	0	15	15	15	15	15	29
0	10	10, 15 15	10 of 15 25				

10 cap - w[i]






15
2
14
5
10
1
45
3
30
4

Repetition

combi — sub sequence / repeat

4 - 2 =

5 - 2 = 3

15	45	14	30	10
2	3	5	4	1

	0	1	2	3	4	5	6	7
value	0	10	~~15~~ 20	~~0~~ 45	~~30~~ ~~45~~ 55	~~60~~ 65	90 75	~~95~~ 100
	.	10 / 1	15 / 2; 10 / 1, 10 / 1	15 / 2; 45 / 3; 10 10 10 / 1 1 1	15 / 2, 15 / 2; 45 / 0; 30 / 4, 0; 10 / 1, 45 / 3	15 / 2, 15 / 2; 45 / 3, 15 / 2; 14 / 5 + 0; 30 / 4; 10 10 45 / 1 1 1	15 15 15 / 2 2 2; 45 / 3, 45 / 3; 14 / 5; 30 / 4, 15 / 2; 10 / 1	15 15 15 / 2 2 2; 45 / 0, 45 / 0; 14 / 5, 15 / 2; 30 / 4, 45 / 3; 10 / 1, 45 / 3, 45 / 3

sh "01"

"000"

"111"

"01011"

h = 3

5

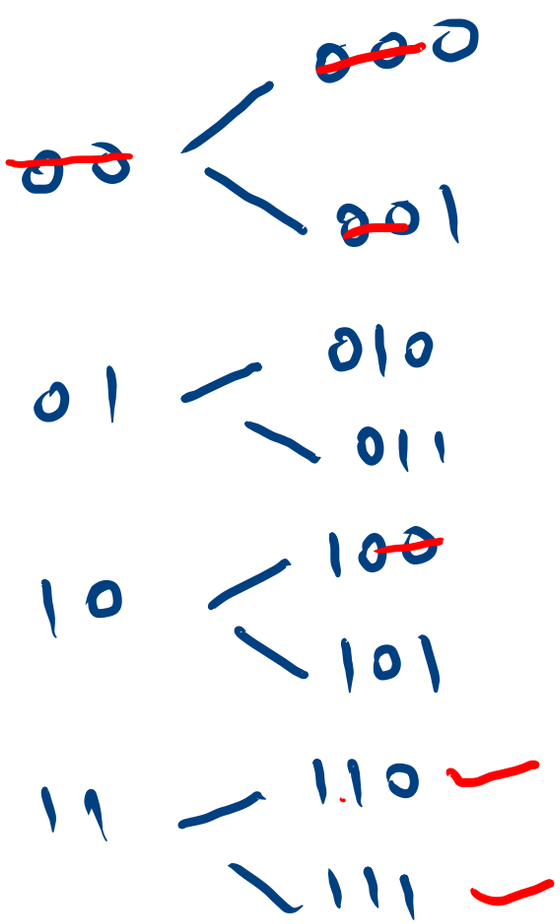
000 ✗
001 ✗
010 ✓
011 ✓
100 ✗
101 ✓
110 ✓
111 ✓

h = 4

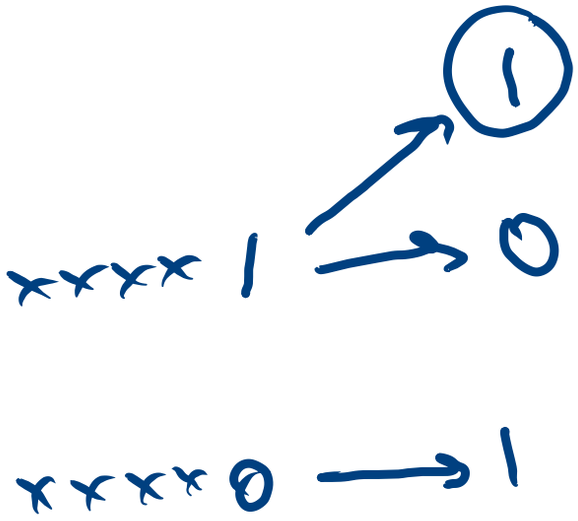
8

0000 ✗
0010 ✗
0100 ✗
0110 ✓
1000 ✗
1010 ✓
1100 ✗
1110 ✓
0001 ✗
0011
0101 ✓
0111 ✓
1001 ✗
1011 ✓
1101 ✓
1111 ✓

2



3



3

0 count

1 count

2 3

000 001 010 011 100 101 110 111

0	1	2	3	4	5
	1	1	2	3	previous + 1
	1	2	3	2+3	max(1 + 3)
	0	10	010	0101	
		01	110	1101	
	1	11	101	1011	
			011	0111	
			111	1111	
				1010	
				0110	
				1110	