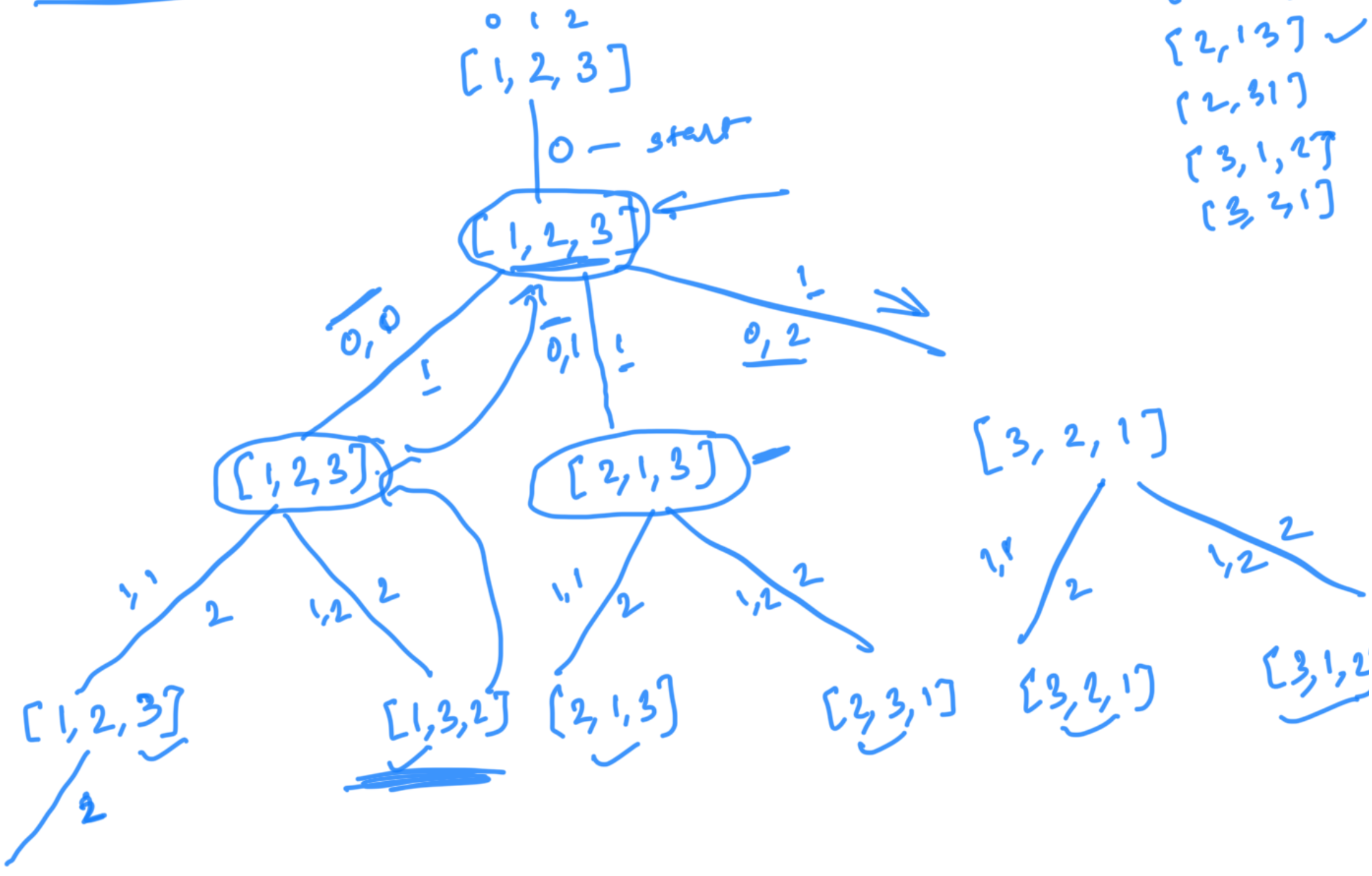
Recursion

Permutation

[1,2,3] —
[1,2,3]
[1,3,2]
[2,1,3] ✓
[2,3,1]
[3,1,2]
[3,2,1]




```
void dfs ( List<Integer> nums, int start )
{
```

```
if ( start == nums.size() - 1 )
 {     print ( nums );
       return;
 }

for ( int i = start ; i < nums.size() ; i++ )
 {     swap ( nums, start, i );    // do
       dfs ( nums, start+1 );
       swap ( nums, start, i );    // undo
 }
}
```

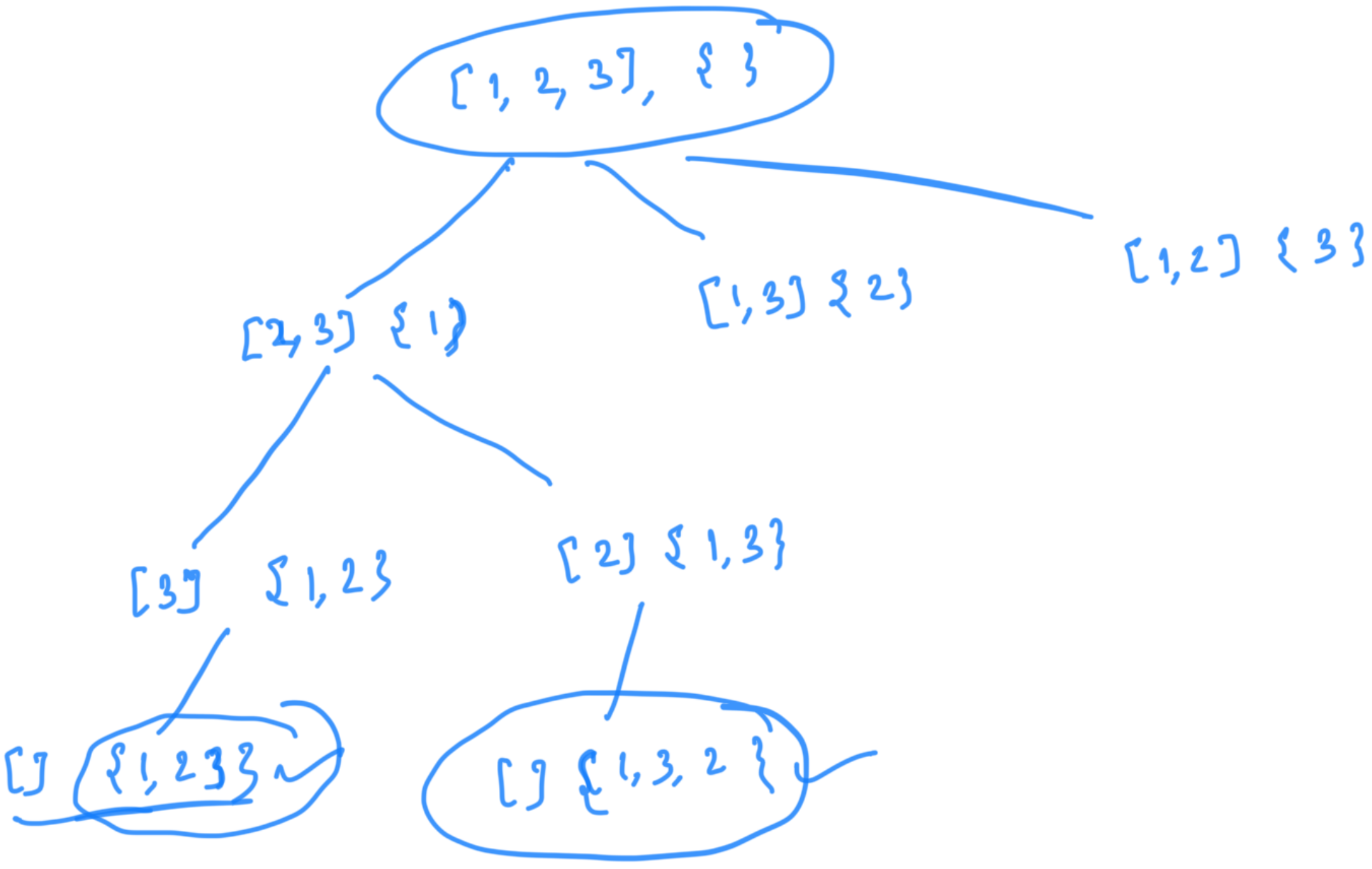
[1, 2, 3]

templist

1 — [2, 3]

2 — [1, 3]

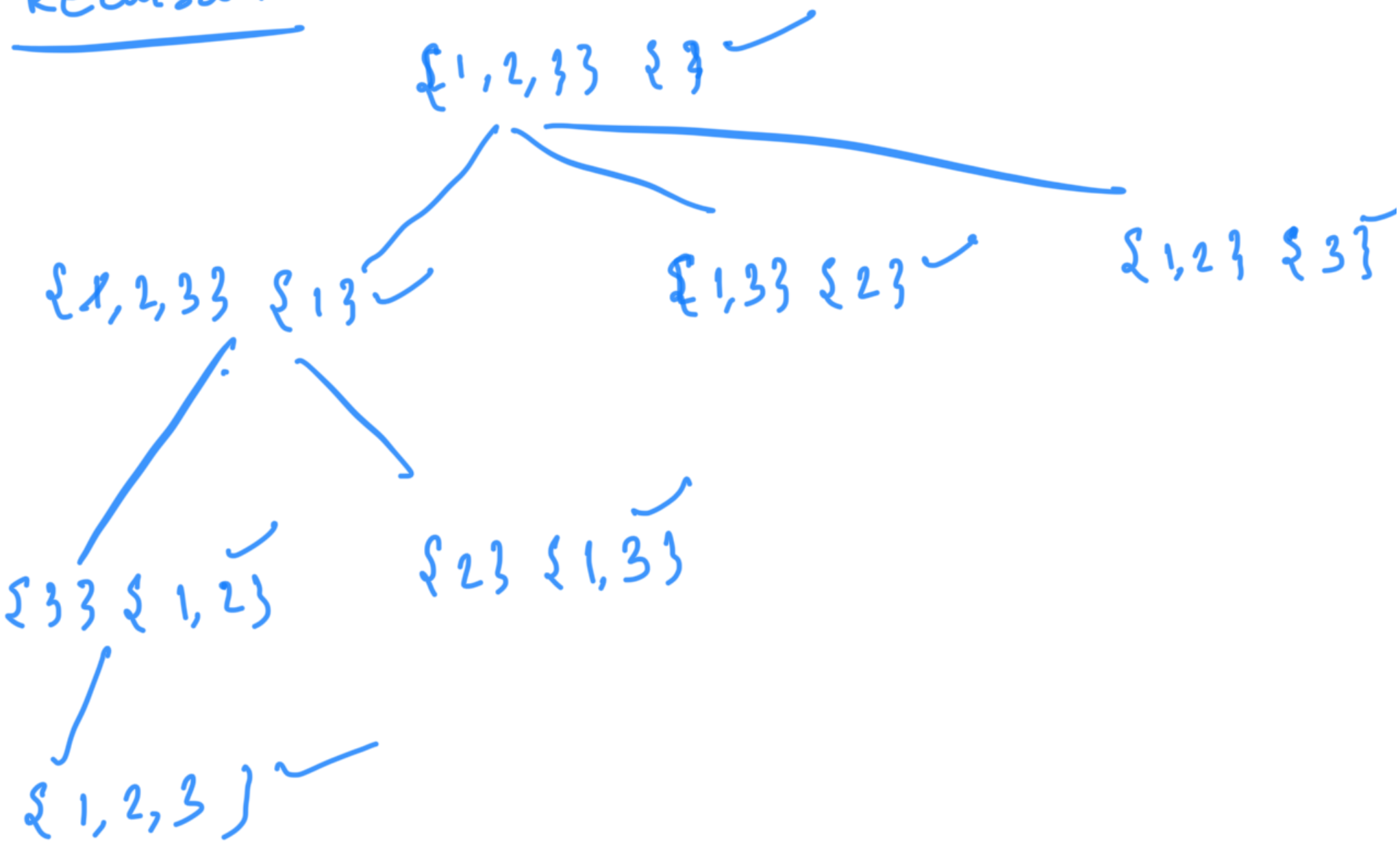
3 — [1, 2]



permutation
combination

subset

Recursion



Bit masking

n element 2^n

2^{n-1}

[1,2,3]

	0 0 0	→	{ }
0	0 0 1	→	{3}
1	0 1 0	→	{2}
2	0 1 1	→	{2,3}
3	1 0 0	→	{1}
4	1 0 1	→	{1,3}
5	1 1 0	→	{1,2}
6	1 1 1	→	[1,2,3]
7			

Iterative solution

{1,2,3}

{ } , {1,} {2} {3} ---

3

{ { } }

{ { } } → { { }, { 1 }, { 2 }, { 3 } }

{ { 1 }, { 1 } { 2 }, { 3 } }

{ { 1 }, { 3 }, { 1, 2 }, { 1, 3 } }

N number

[1, 1, 1 2]

[1, 1, 1, 2]

[1, 2, 1 1]

[1, 1 2 1]

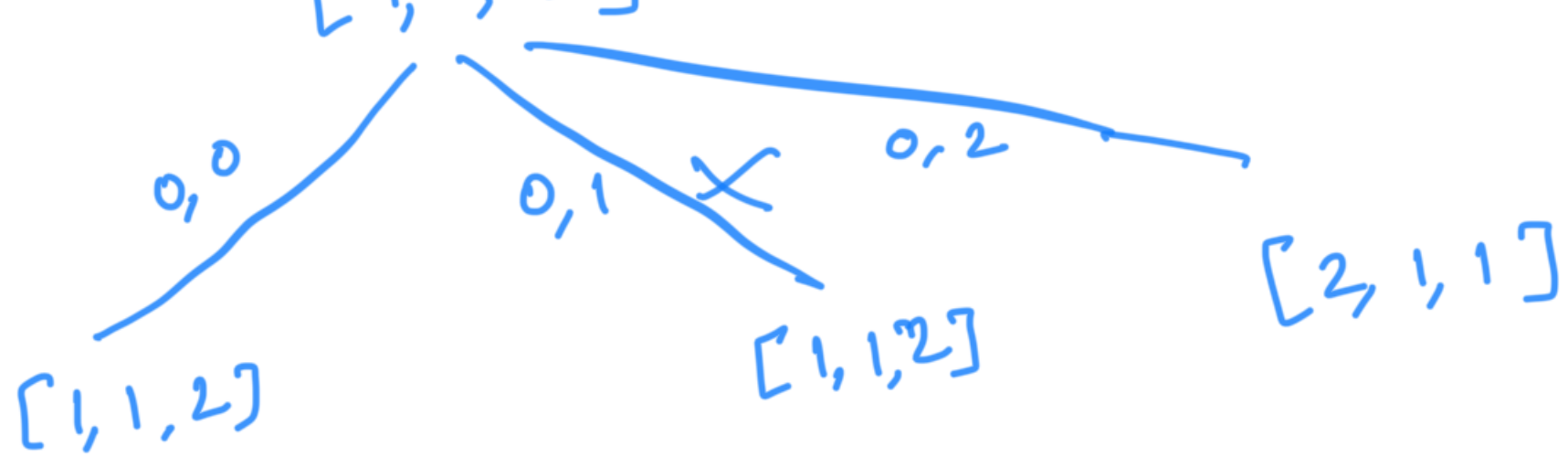
$\frac{4!}{3!} = \frac{4 \cdot 3!}{3!}$

[2,1,1] /

[1,1,1,2,2,3,3,3,4] ✓

$$\frac{9!}{3!\,2!\,3!} =$$

[1,1,2] ✓



combination sum

[2,3,5,8], 8

[2 2 2 2] ✓

[3 3 2] ✓

8, 6, 4 ... 0

2, 4, 6, (8)

[3, 5]

[8]

Start 0 take the number and go to next

or

do not take this no. go to next.

dfs (arr[], int start, int target, list)

{

if (target == 0)

{ print list;

return;

}

if (target < 0) return ;

if (start >= arr.length) return ;

list.add (arr[start]);

taken —— dfs (arr, start, target - arr[start], list)

list.remove (list.size()-1);

Not taken —— dfs (arr, start+1, target, list)

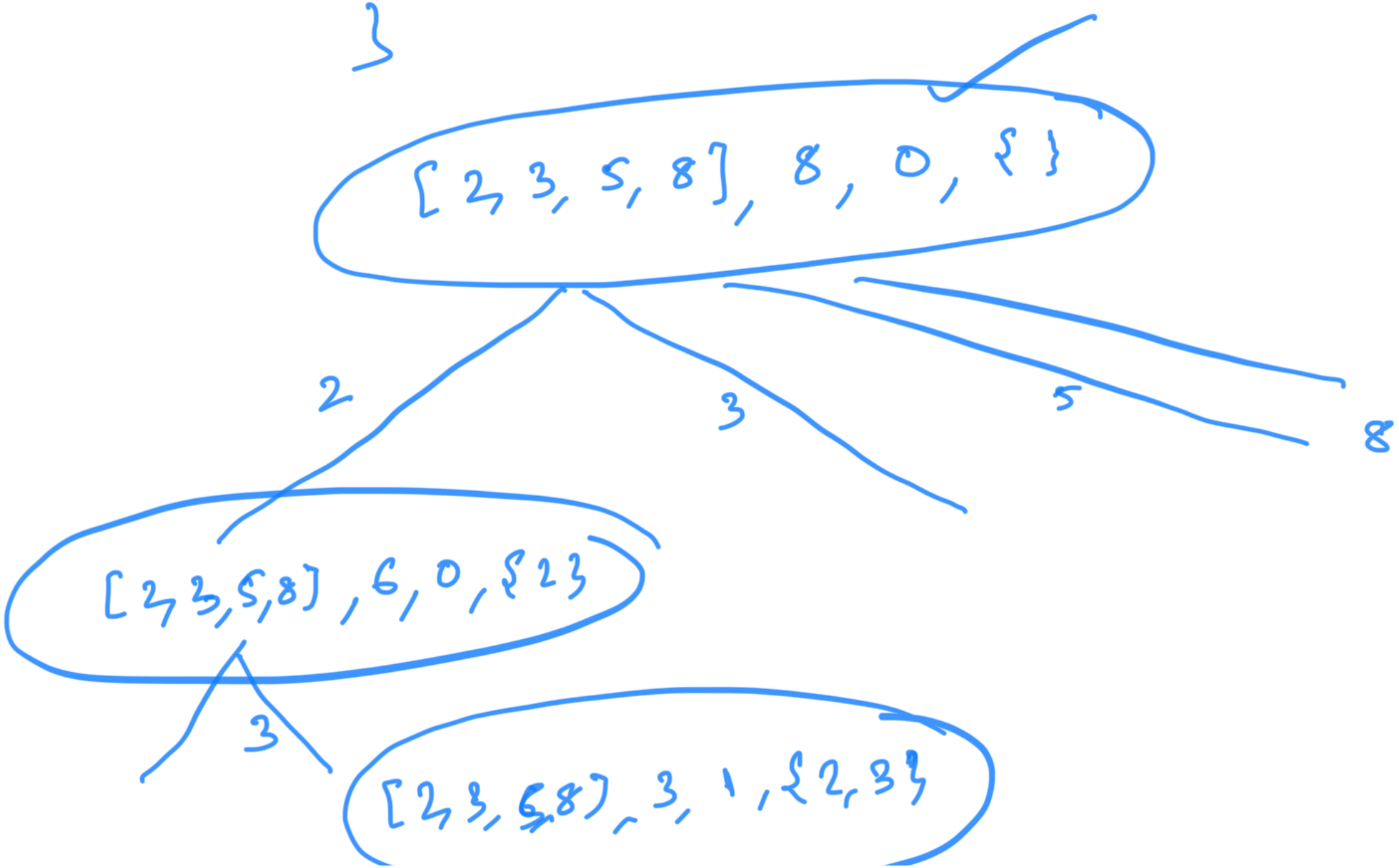
}

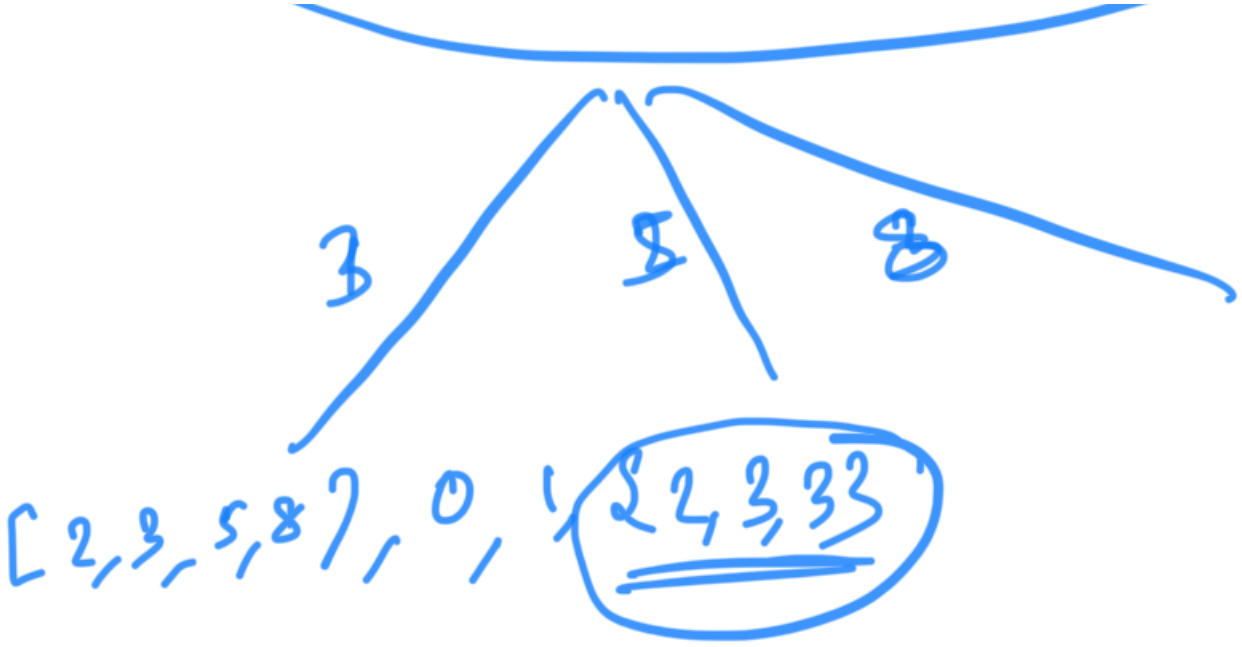
dfs (

)

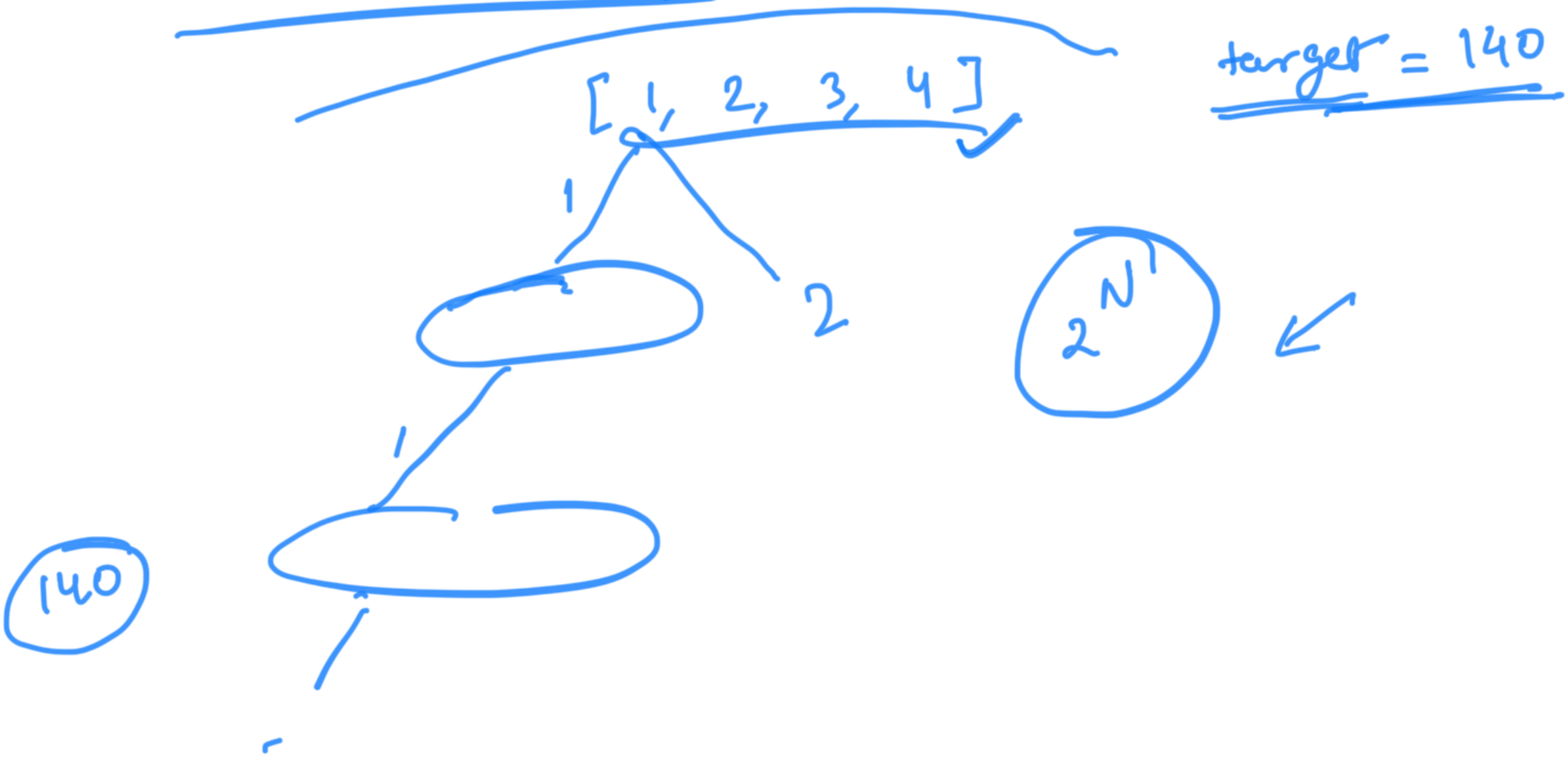
{

```
for (i: start ; i< arr.length;i++)
{       list.add(arr[start]);
    dfs(arr, i, target- arr[start],
        list.removeO;
}
```





Time complexity



[1, . . - - 140]

140

a_1 , a_2 , a_3 - - - a_N

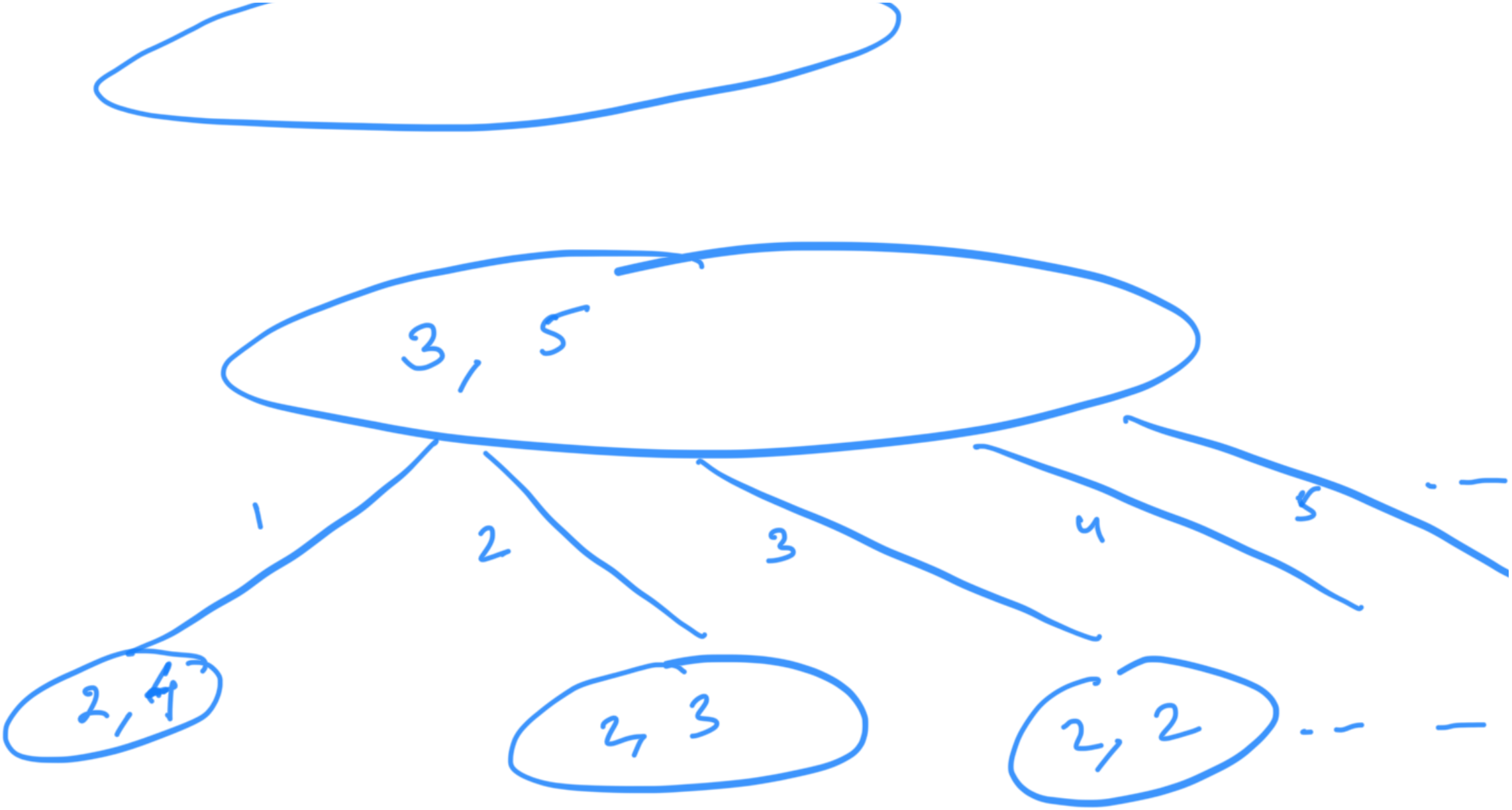
take

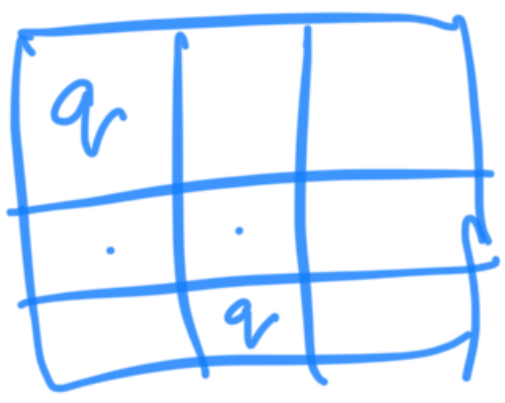
don't

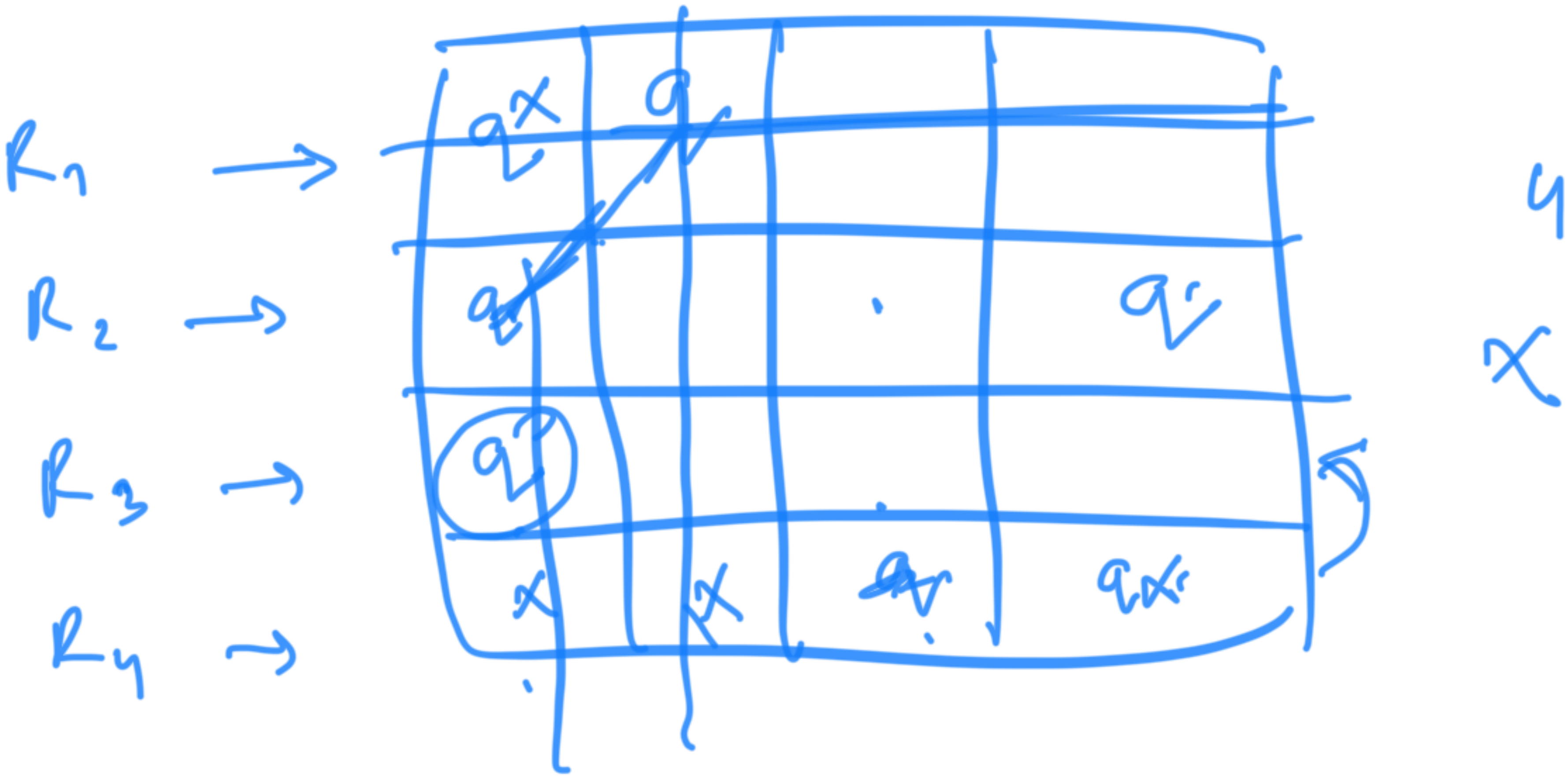
2^N

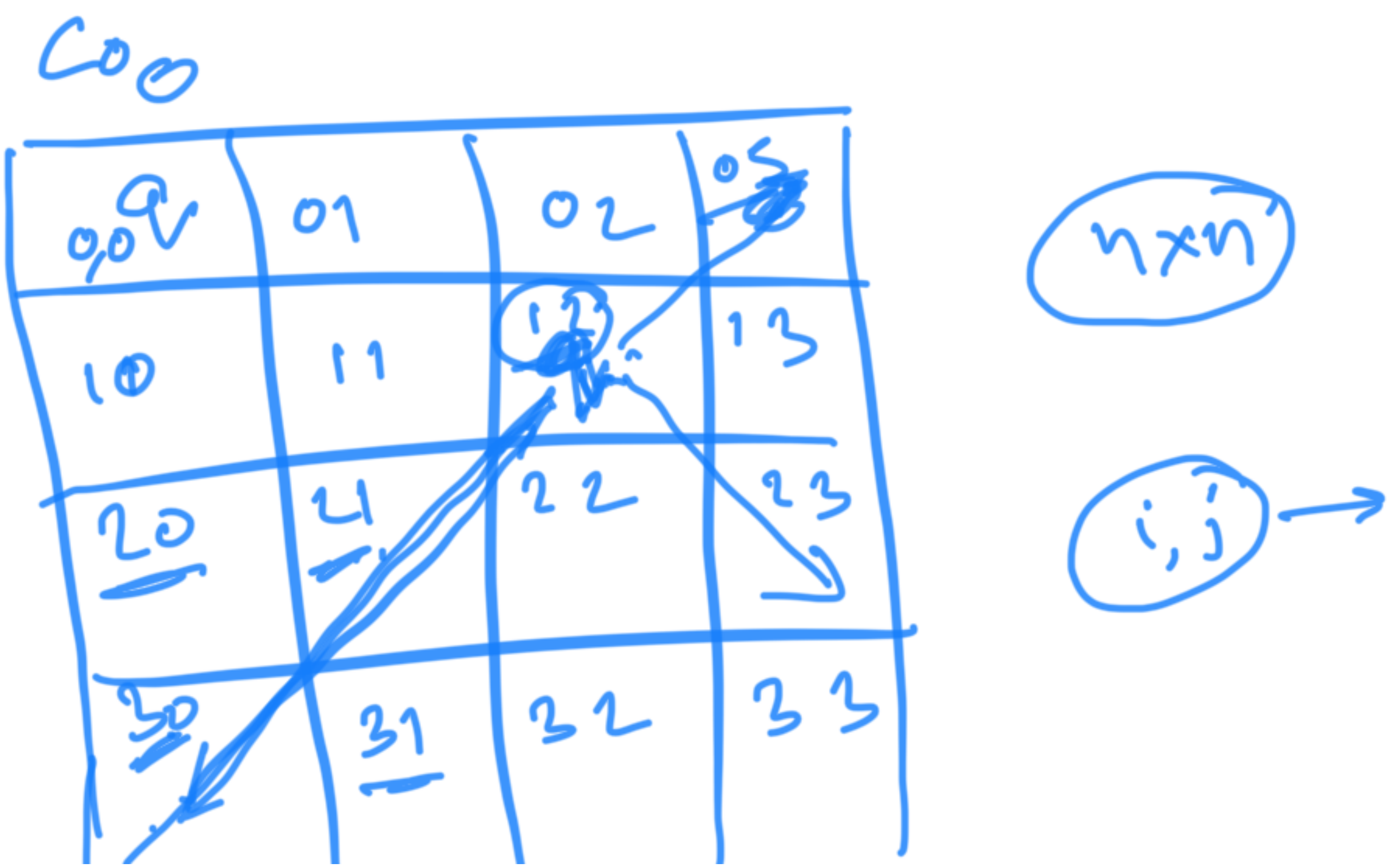
2^N and target

$O(2^N)$


3, 5
1
2
3
4
5
2, 4
2, 3
2, 2


q
q


R1
R2
R3
R4
4x4
X


Coo
0,0
01
02
10
11
13
20
21
22
23
30
31
32
33
nxn
i, j

boolean[] isQATCOL = [(3,0) (2,1) (1,2) (0,3)]

[(0,0) (1,1) (2,2) (3,3)] → 3 3

3 map to same value

[(0,1) (1,2) (2,3)]

(2,0) (3,1)

-2 -2

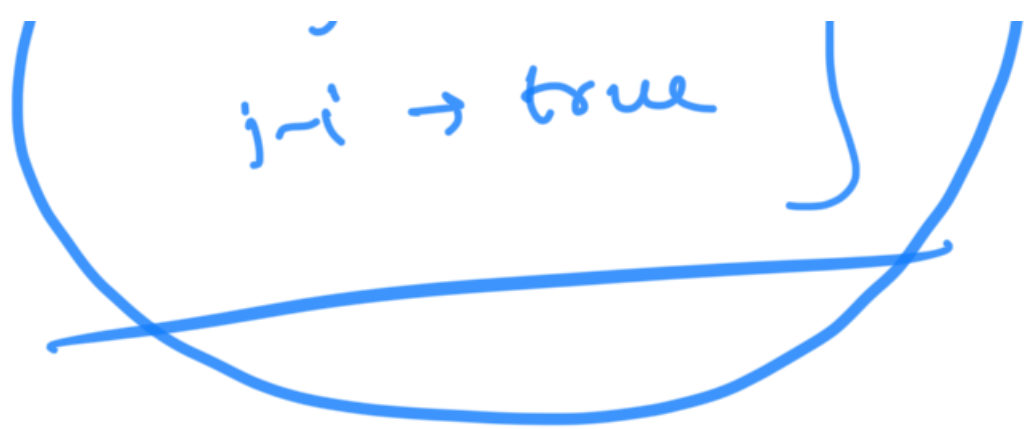
(i,j) → (i+j) (j-i)

col →

(i,j)

j → true

i+j → true



j+i

j-i

dfs(w

F

```
(
    dfs(row, N)
    {
        if(row == N)
        { Print ans; return
        }

    for(col=0; col<N; col++)
    {
        // try to put q at (row, col)
    if(hm1.get(col) == true
        || hm2.get(col+row) == true
        || hm3.get(row-col) == true
            continue;
```