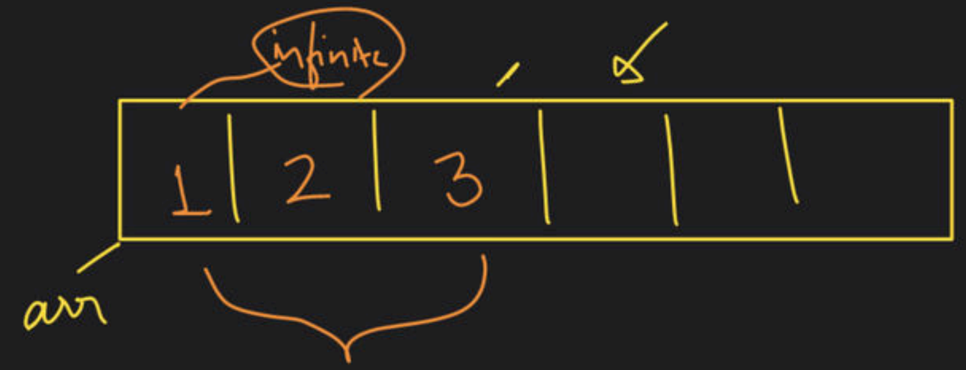
Recursion - Level 4

Special class

Love Babbar • Mar 15, 2023

Q1

i/p →



size = n

n distinct elements

(ii) target = 5

sum

you have to tell the ~~minimum~~ minimum number of elements required to reach target sum.

5 → {1,1,1,1,1} — 5

{1,1,1,1,2} — 4

{1,2,2} — 3

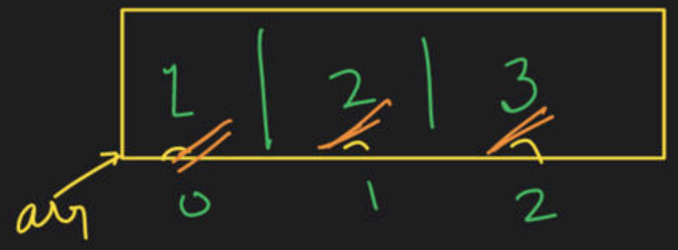
{1,1,3} — 3

{2,3} — 2

(2,3) → target

4, 1

target sum → 5



min no of element req → to reach tar

{1,1,1,1,1}

{2,2,1}

{3,1,1}

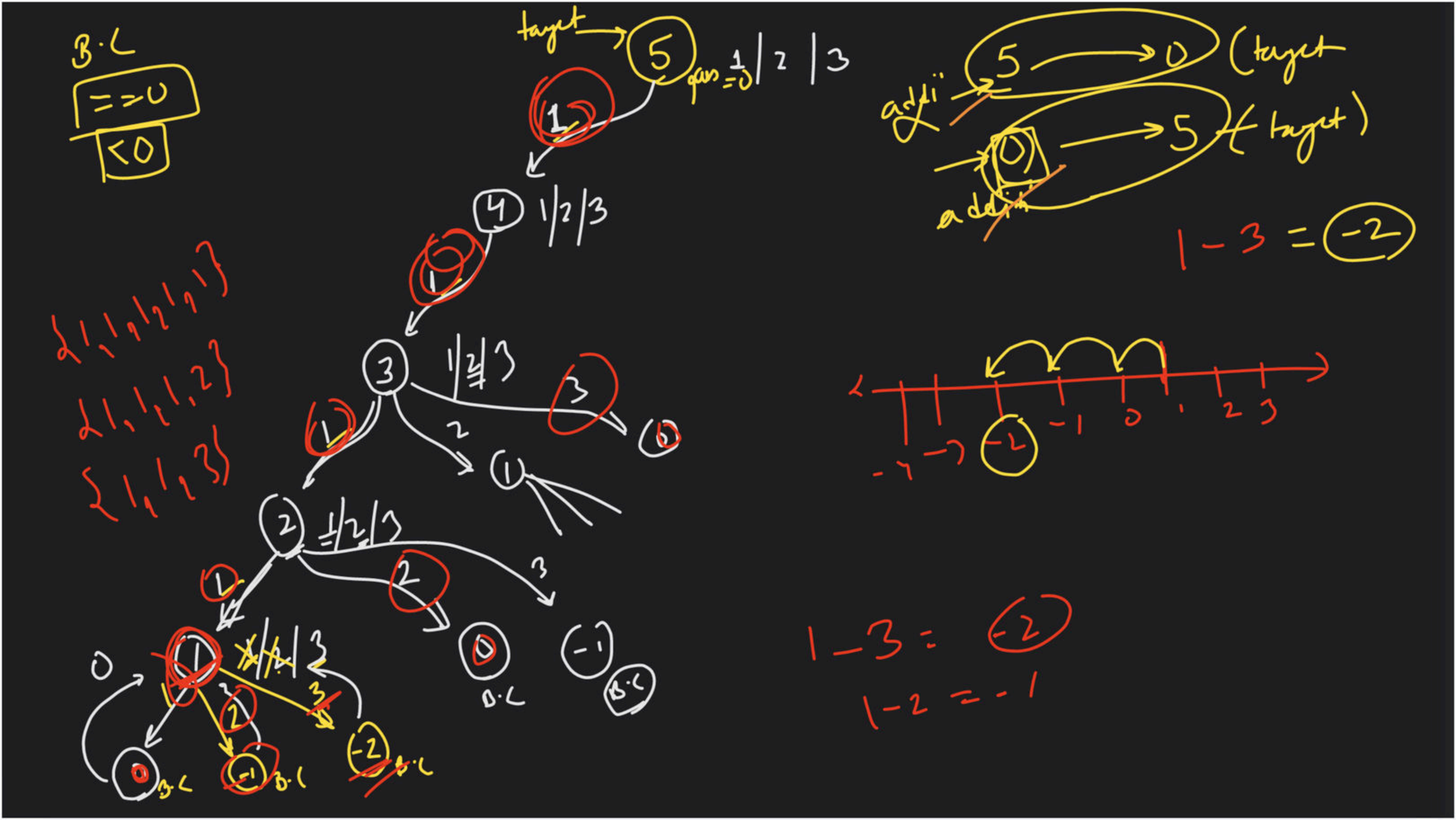
{1,2,1,1}

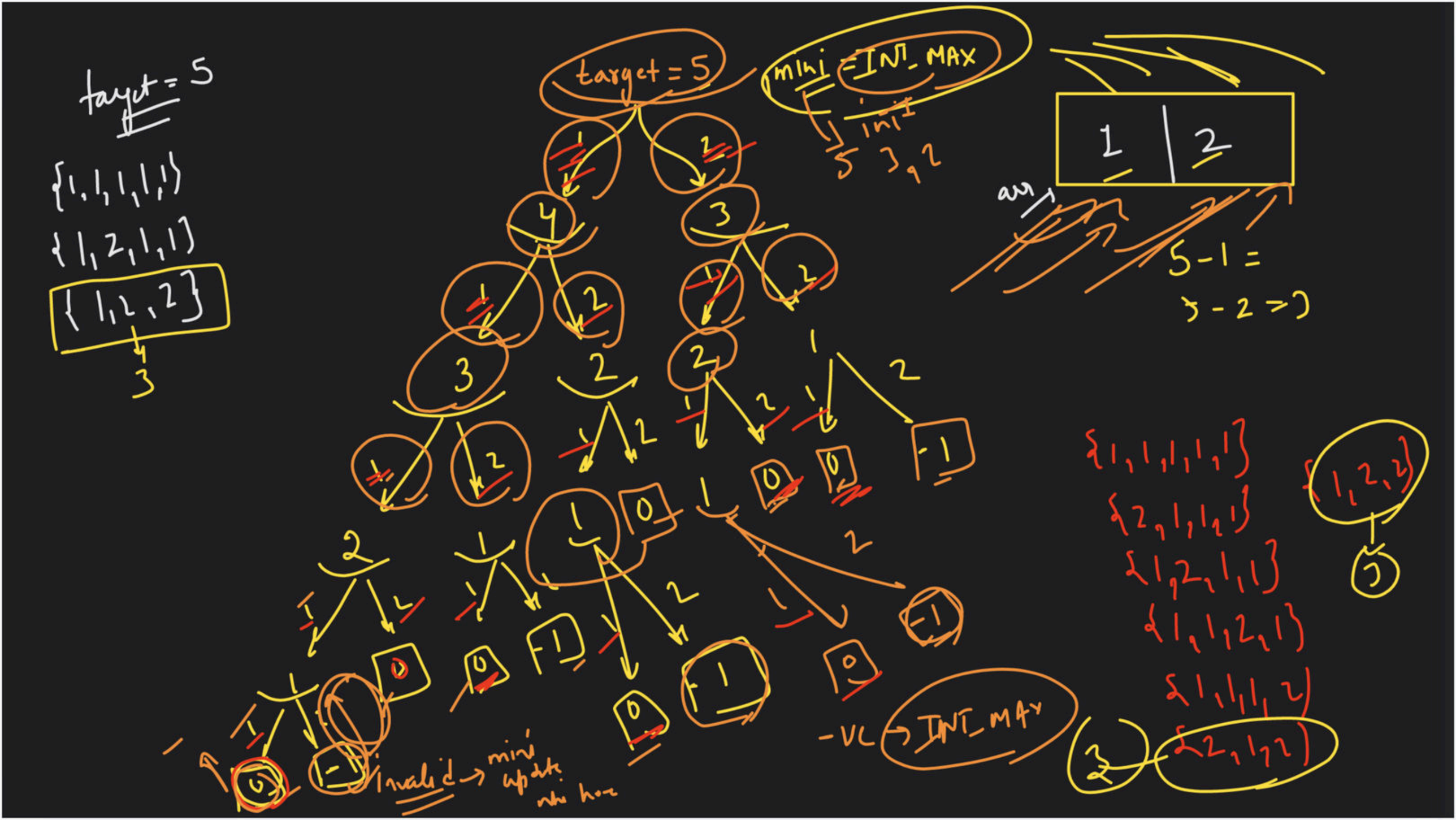
{1,3,1}

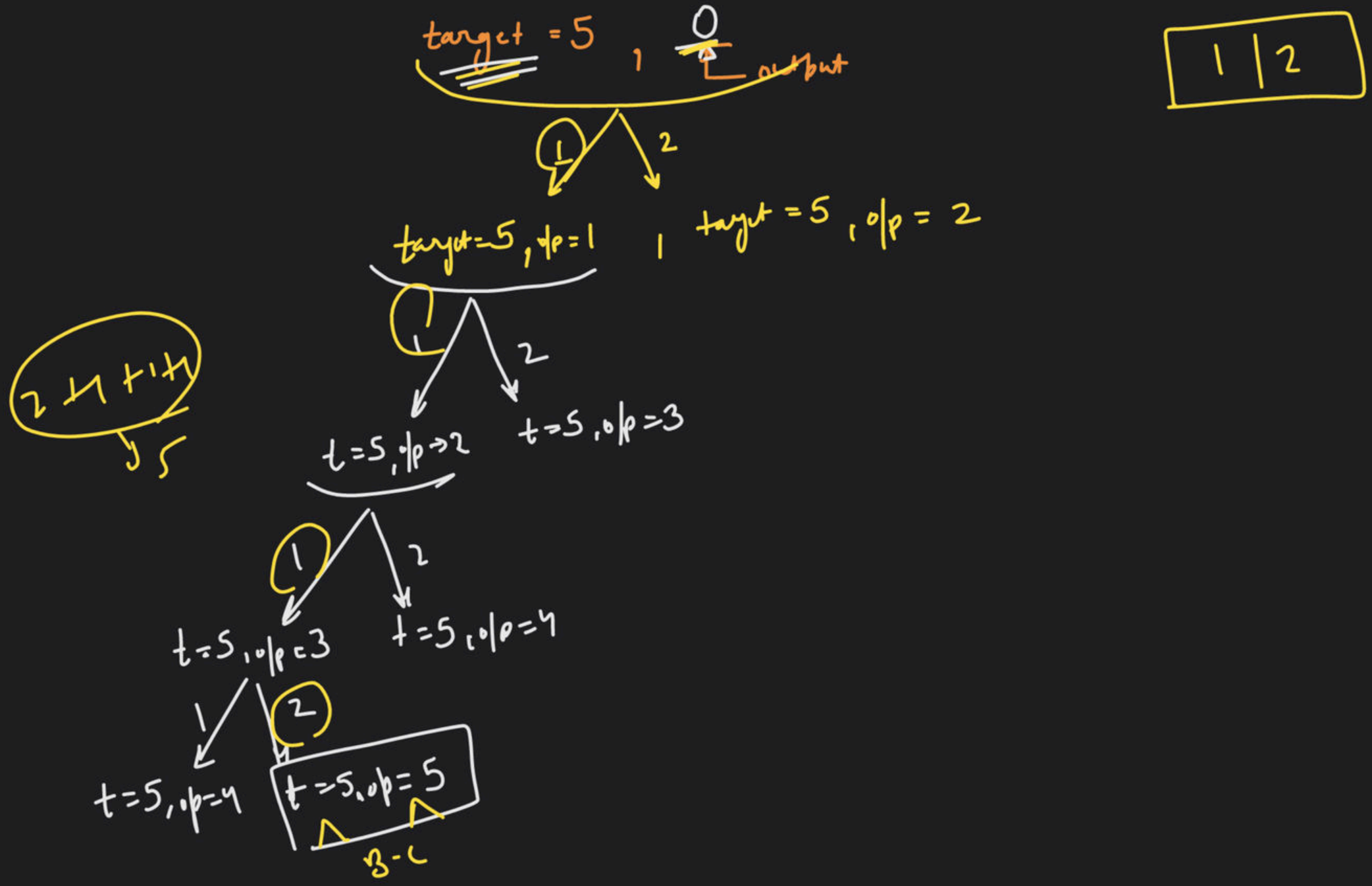
{3,2}

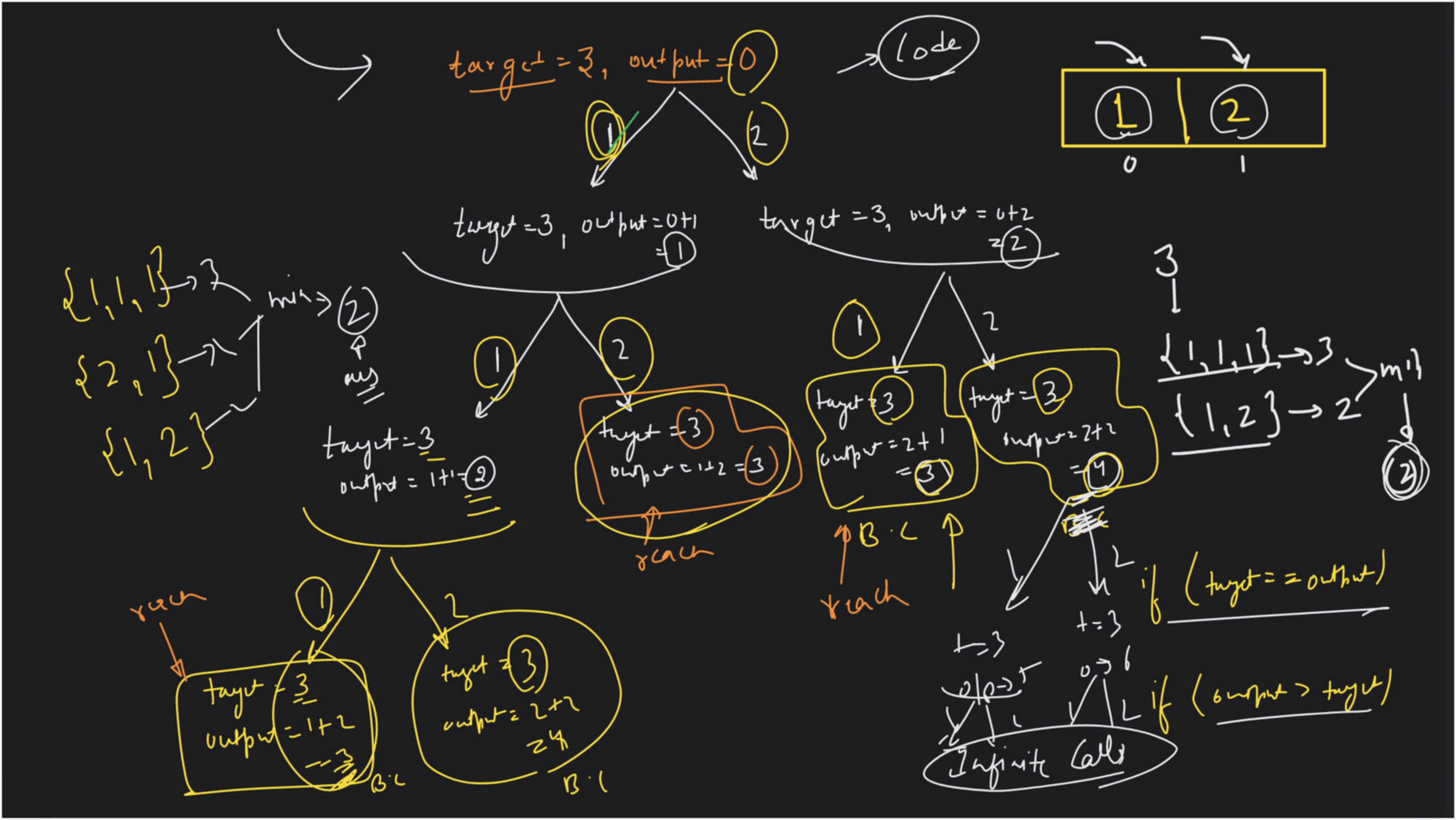
2+3 = 5

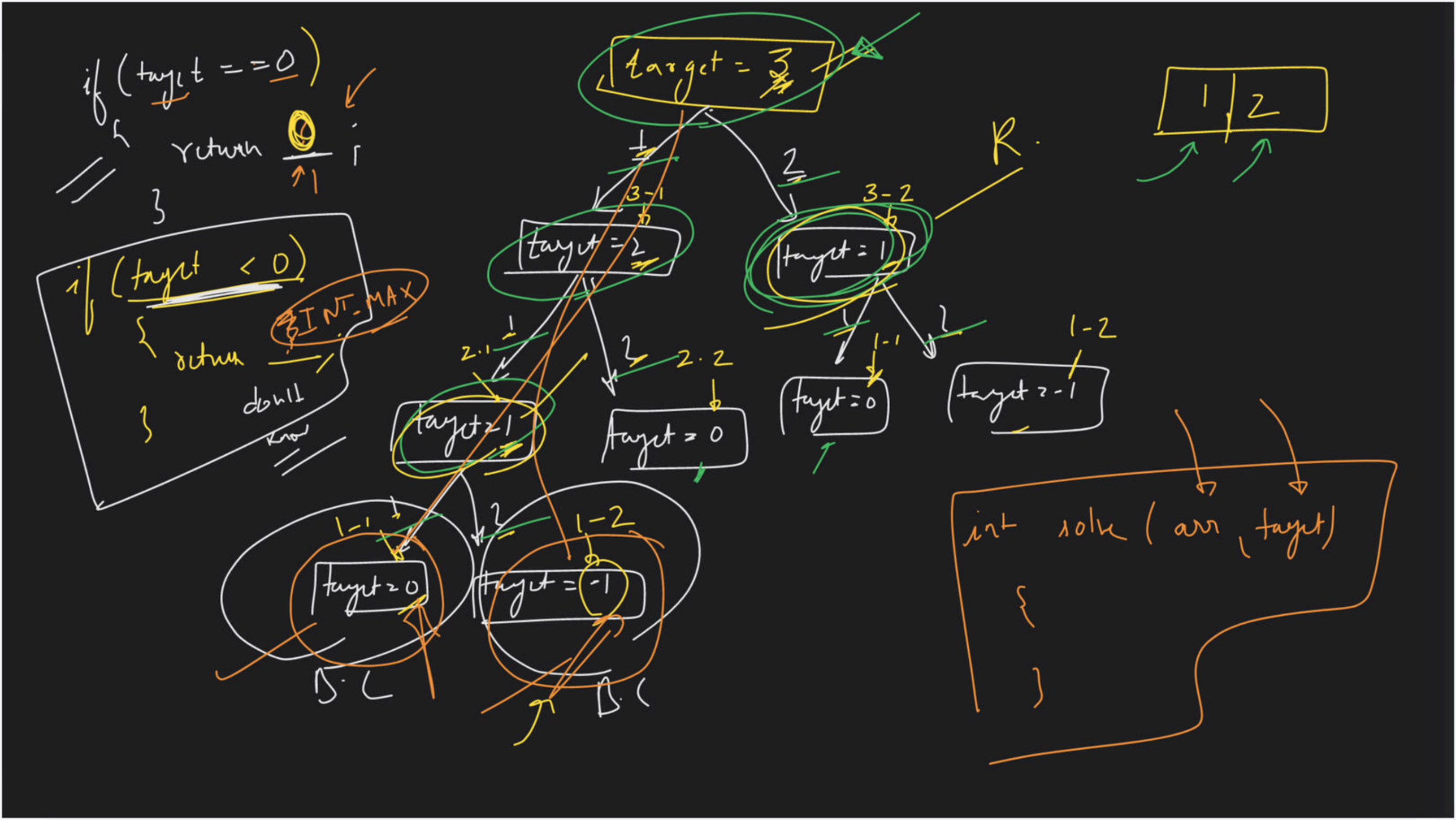
2 element

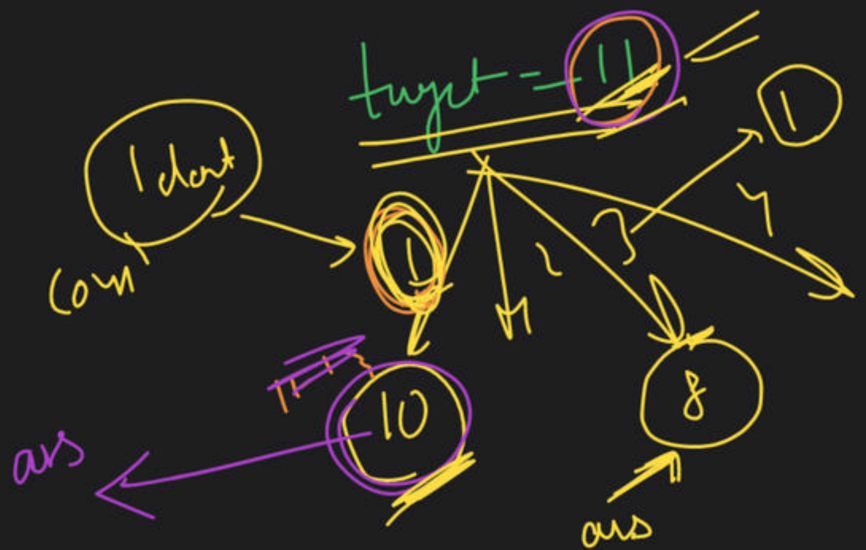

B.C
==0
<0
target
5
coins = 1 | 2 | 3
4
3
2
1
0
-1
-2
B.C
addi
5 → 0 (target
0 → 5 (target)
addi
1-3 = -2
1-3 = -2
1-2 = -1

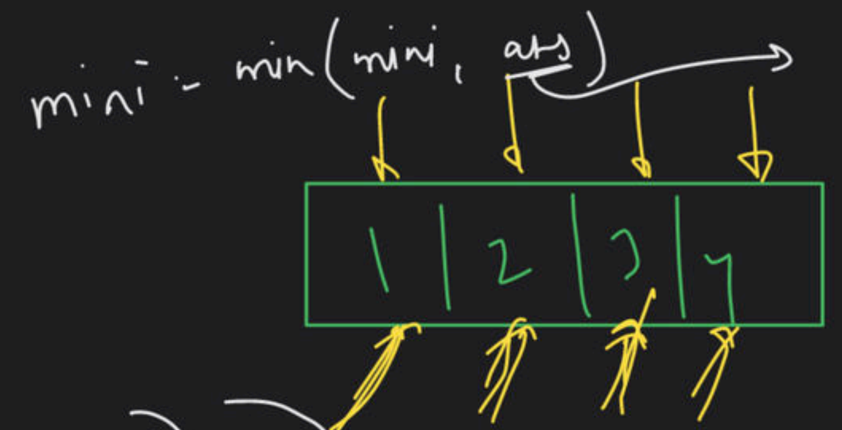

target = 5
{1,1,1,1,1}
{1,2,1,1,1}
{1,2,2}
3
mini = INT_MAX
5
ans
1
2
5-1 =
3-2 =>
Invalid → mini update nahi hoga
-VL → INT_MAX
{1,1,1,1,1}
{2,1,1,1}
{1,2,1,1}
{1,1,2,1}
{1,1,1,2}
{2,1,2}
{1,2,2}
5
2


target = 5
0
output
1
2
target=5, o/p=1
target = 5, o/p = 2
1
2
t=5, o/p=2
t=5, o/p=3
1
2
t=5, o/p=3
t=5, o/p=4
1
2
t=5, o/p=4
t=5, o/p=5
3-C
1 | 2


target = 3, output = 0
code
1
2
0
1
target = 3, output = 0+1 = 1
target = 3, output = 0+2 = 2
{1,1,1} → 3
{2,1} → 3
{1,2} → 2
min → 2
ans
target = 3
output = 1+1 = 2
target = 3
output = 1+2 = 3
reach
target = 3
output = 2+1 = 3
target = 3
output = 2+2 = 4
B.C
reach
target = 3
output = 1+2 = 3
target = 3
output = 2+2 = 4
B.C
B.C
3
{1,1,1} → 3
{1,2} → 2
min
2
if (target == output)
t = 3
t = 3
Infinite Calls
if (output > target)


if (target == 0)
return 0;
}
if (target < 0)
{
return INT_MAX
doubt
Know
}
target = 3
1
2
3-1
3-2
target = 2
target = 1
2.1
2.2
1-1
1-2
target = 1
target = 0
target = 0
target = -1
target = 0
target = -1
R.
1
2
int solve (arr , target)
{
}






```
int mini = INT_MAX;
for (i=0; i<n; i++)
```

ans = INT_MAX → invalid

```
if (ans != INT_MAX)
{
  mini = min(mini, ans)
}
```

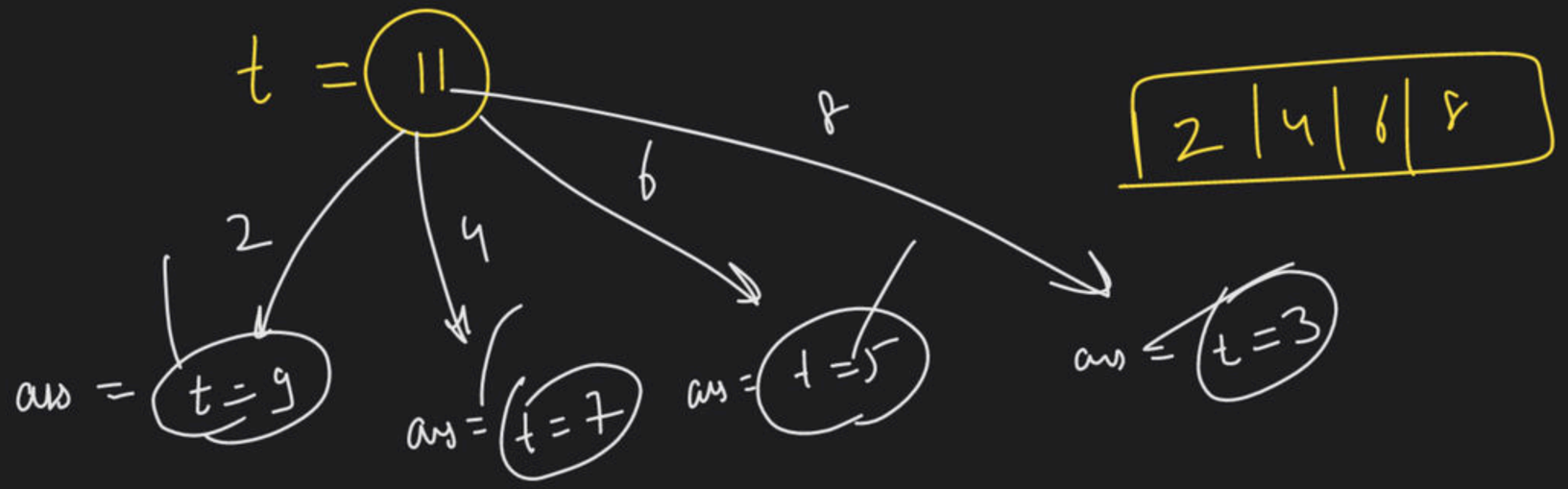
```
int ans = solve(arr, target - arr[i])
```

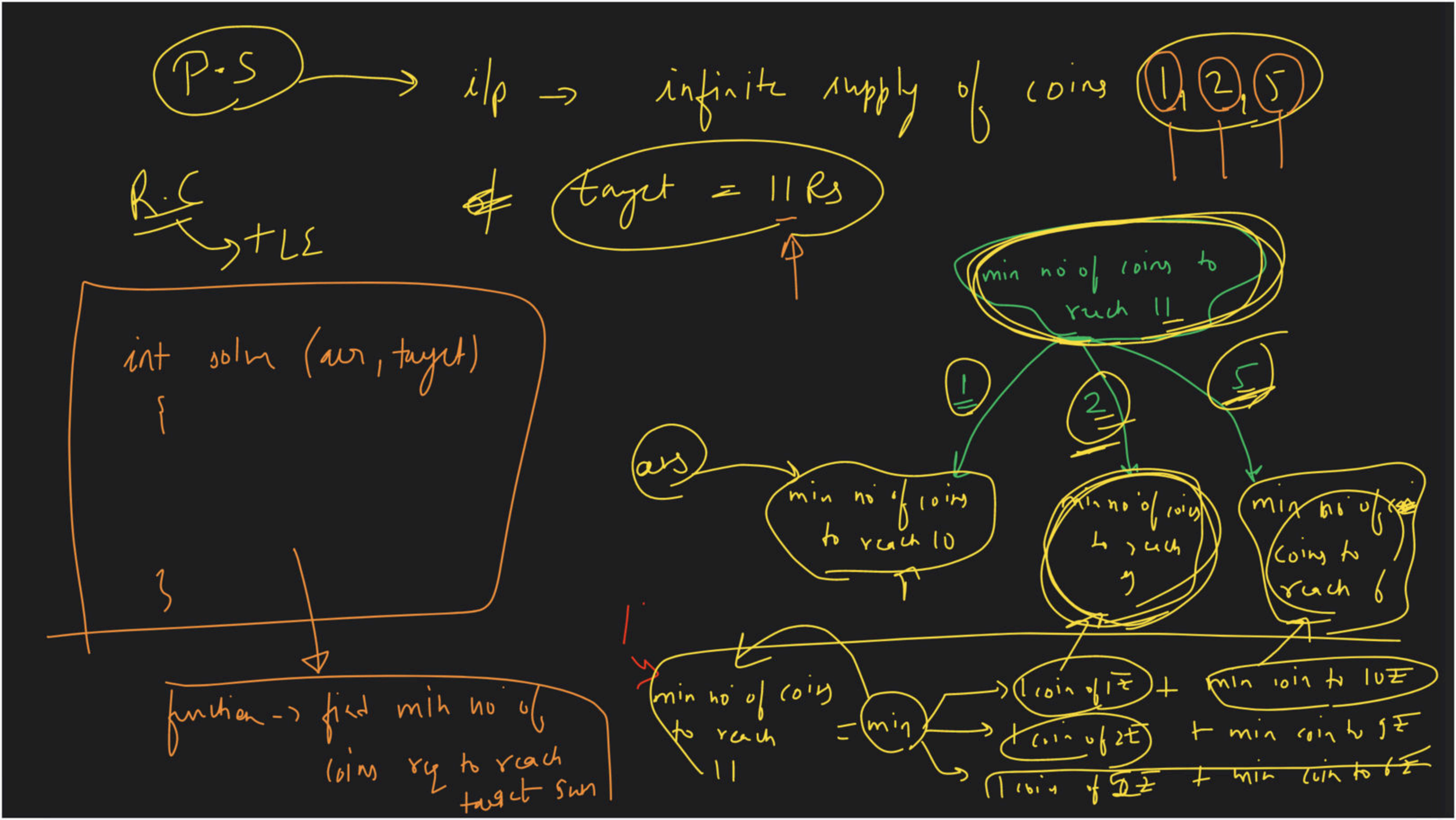
ans → valid → mini update

ans → invalid → ignore

valid

```
if (ans != INT_MAX)
{
  mini = min(ans+1, mini)
}
```


t = 11
2
4
6
8
ans = t = 9
ans = t = 7
ans = t = 5
ans = t = 3
2 | 4 | 6 | 8


P.S
i/p → infinite supply of coins
1 2 5
R.C
↳ TLE
target = 11 Rs
int solve (arr, target)
{
}
function -> find min no of coins req to reach target sum
min no of coins to reach 11
arr
1
2
5
min no of coins to reach 10
min no of coins to reach 9
min no of coins to reach 6
min no of coins to reach 11
= min
1 coin of 1₹ + min coin to 10₹
1 coin of 2₹ + min coin to 9₹
1 coin of 5₹ + min coin to 6₹

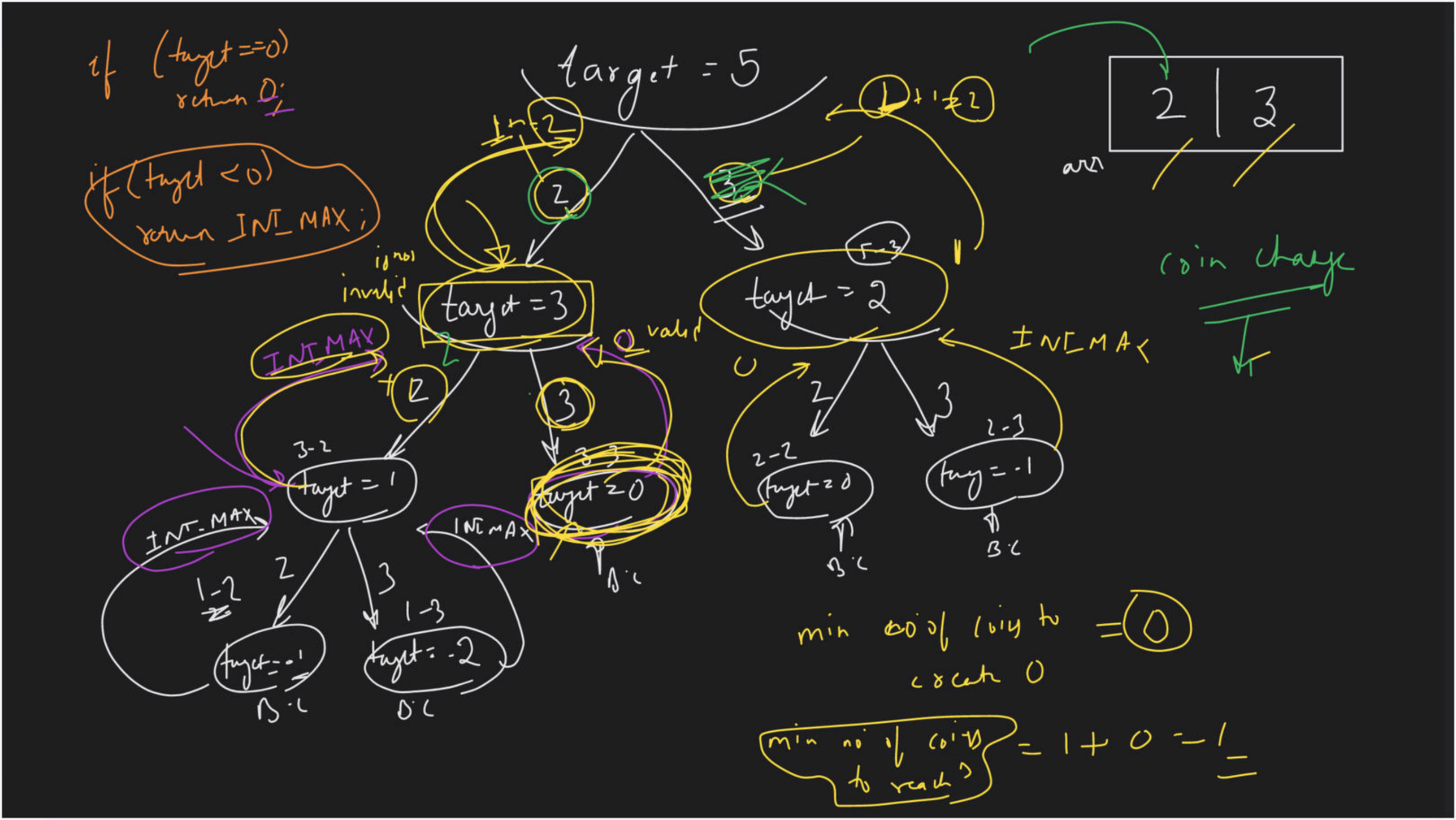
smallest x in list

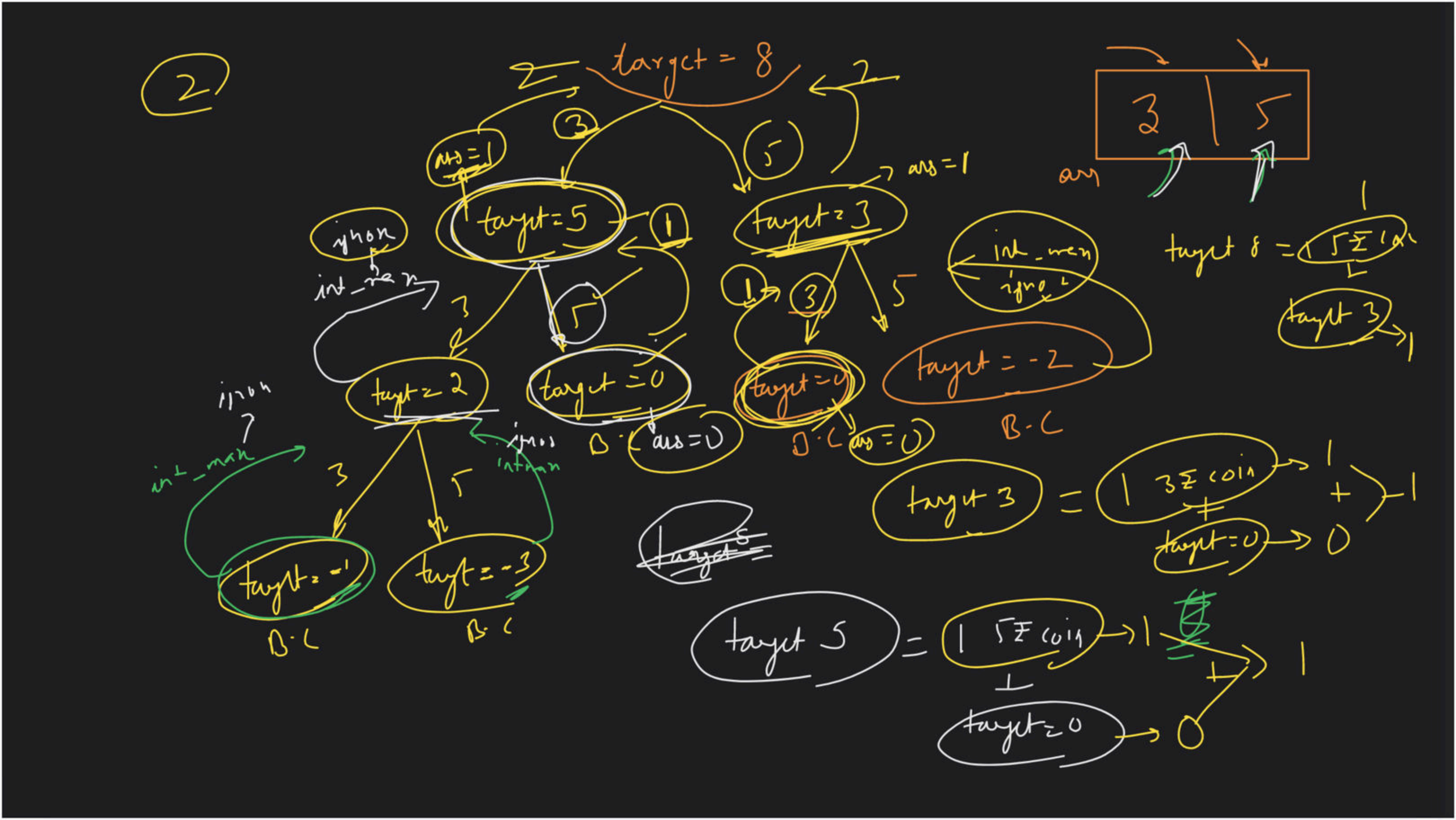
wildcard pattern matching

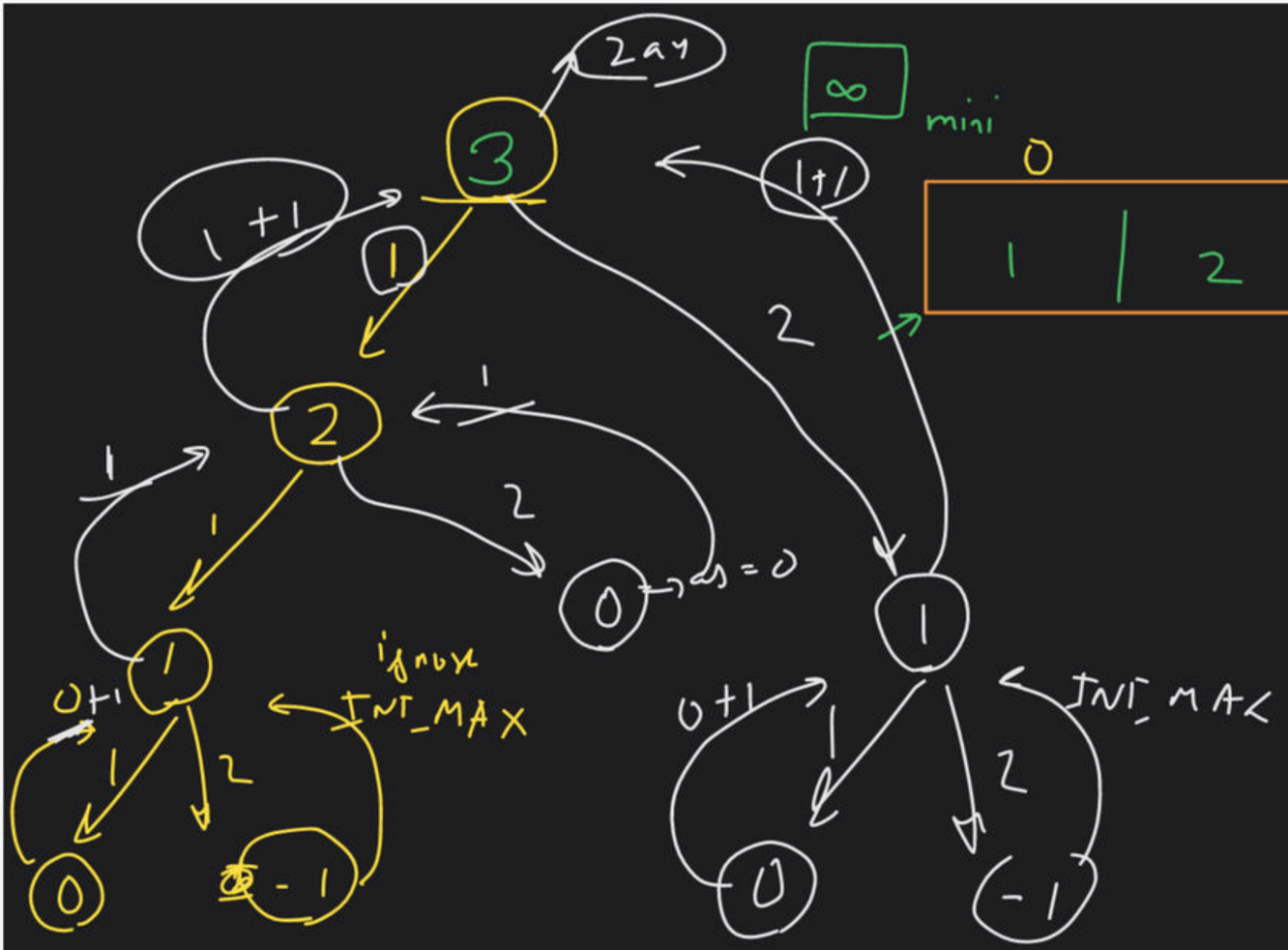
Median in a Stream

Ryder expression matching

2 min
Break


if (target == 0)
return 0;
if (target < 0)
return INT_MAX;
target = 5
1
1+1=2
i=2
2
3
arr
2 | 3
coin change
F
if not invalid
target = 3
target = 2
i=3
valid
INT_MAX
INT_MAX
2
3
3-2
target = 1
target = 0
target = 2
2-2
2-3
target = 0
target = -1
INT_MAX
INT_MAX
INT_MAX
1-2
2
3
1-3
target = -1
target = -2
B.C
B.C
B.C
B.C
B.C
min no of coin to create 0
= 0
min no of coin to reach 5
= 1 + 0 = 1


2
target = 8
3
5
ans = 1
target = 5
target = 3
ans = 1
1
3
5
python
int_max
target = 2
target = 0
ans = 0
B.C
target = 1
target = -2
B.C
ans = 0
int_max
target = -1
target = -3
B.C
B.C
target 5 = 1 코인 → 1
target = 0 → 0
target 3 = 1 3코인 → 1
target = 0 → 0
arr
2
5
target 8 = 1 코인 → 1
target 3 → 1




```
int solve (arr, target)
{
    if (target == 0)
        return 0;

    if (target < 0)
        return INT_MAX;

    int mini = INT_MAX;
    for (i = 0; i < n; i++)
    {
        int ans = solve(arr, target - arr[i]);
        if (ans != INT_MAX)
        {
            mini = min(mini, ans+1);
        }
    }
    return mini;
}
```

target = 7

1	2	3

solve

3 i

3 mid

Break

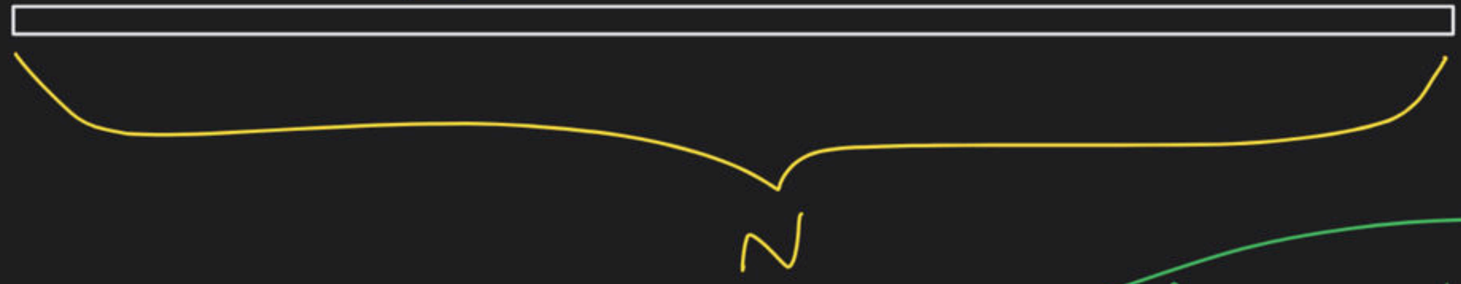

(1) incl – excl
(2) element → for loop
call
or
0 → ans = 0

Cut into segments

i/p → N → rod length

maximum no of segments

x, y, z



N

N = 7

x = 5

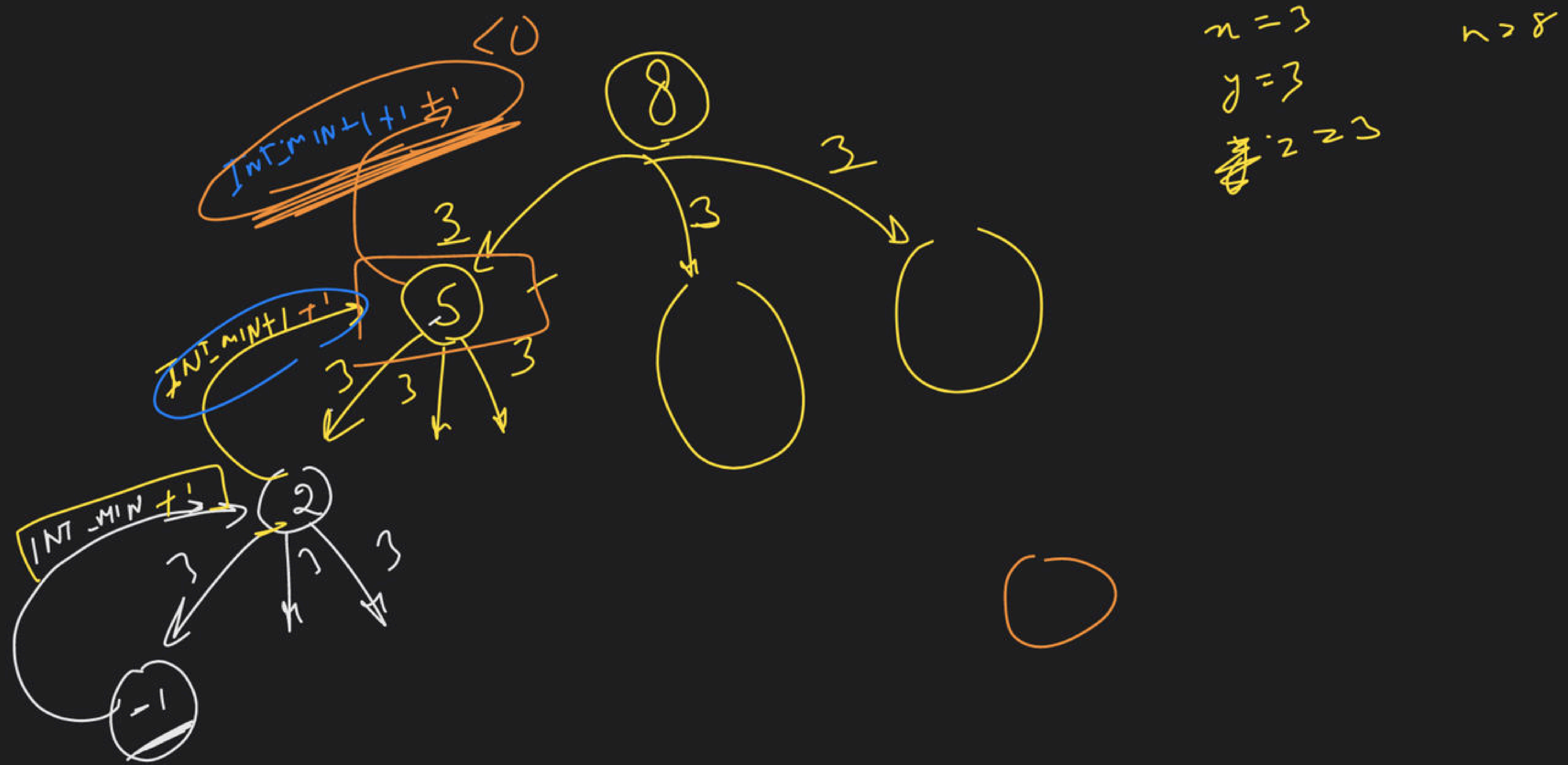
y = 2

z = 2

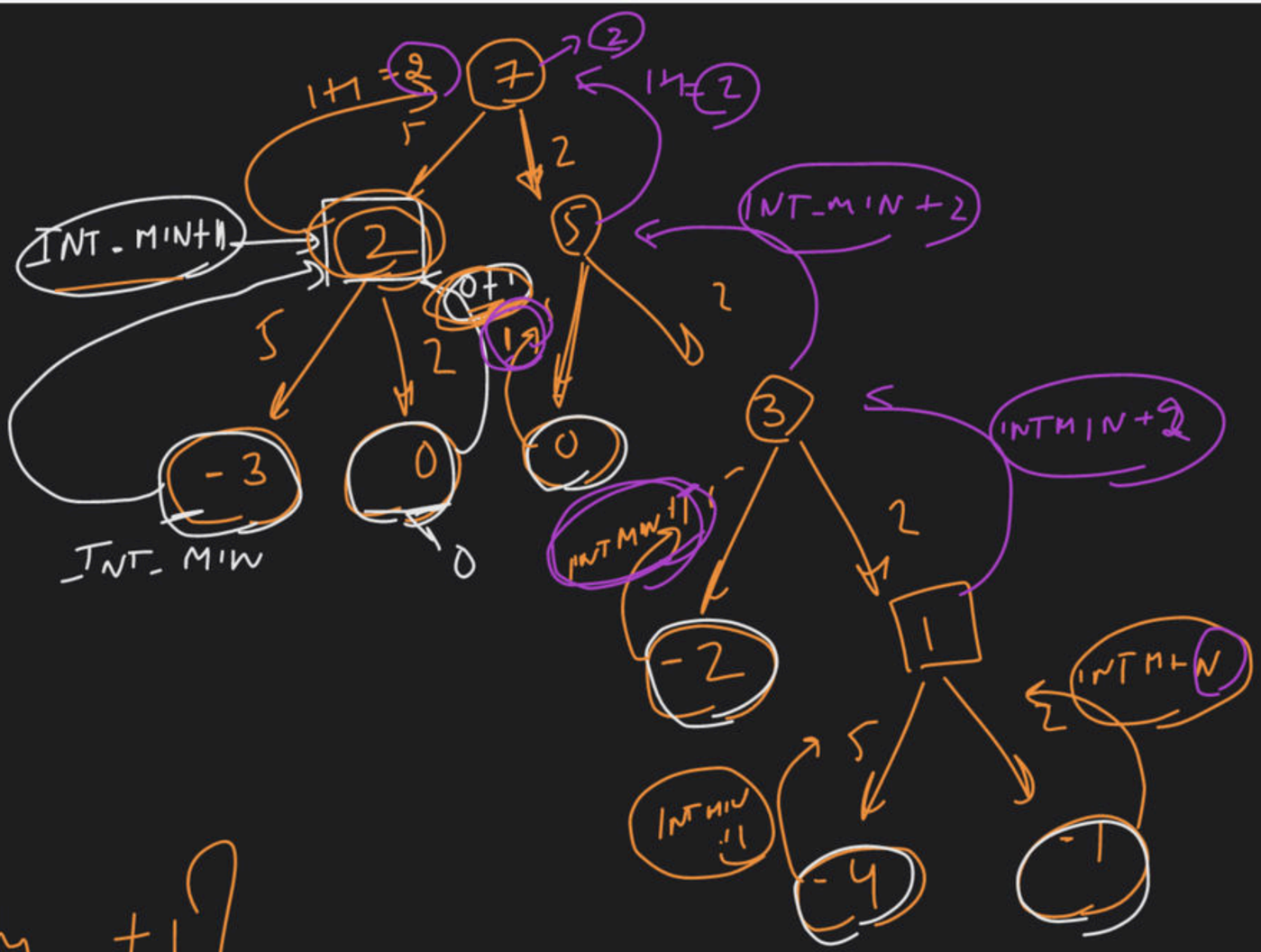
House-Robbery!

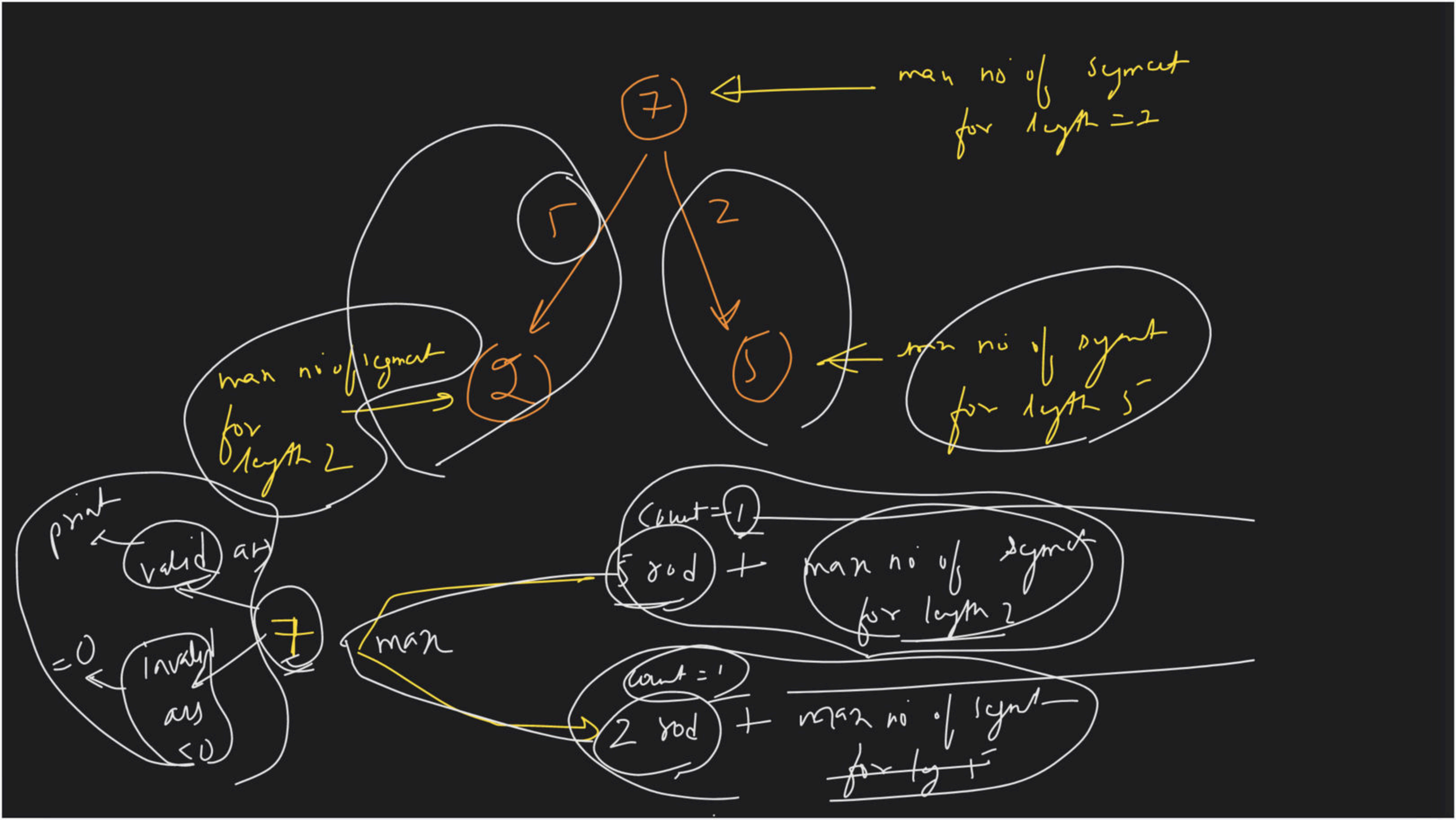

INT_MAX

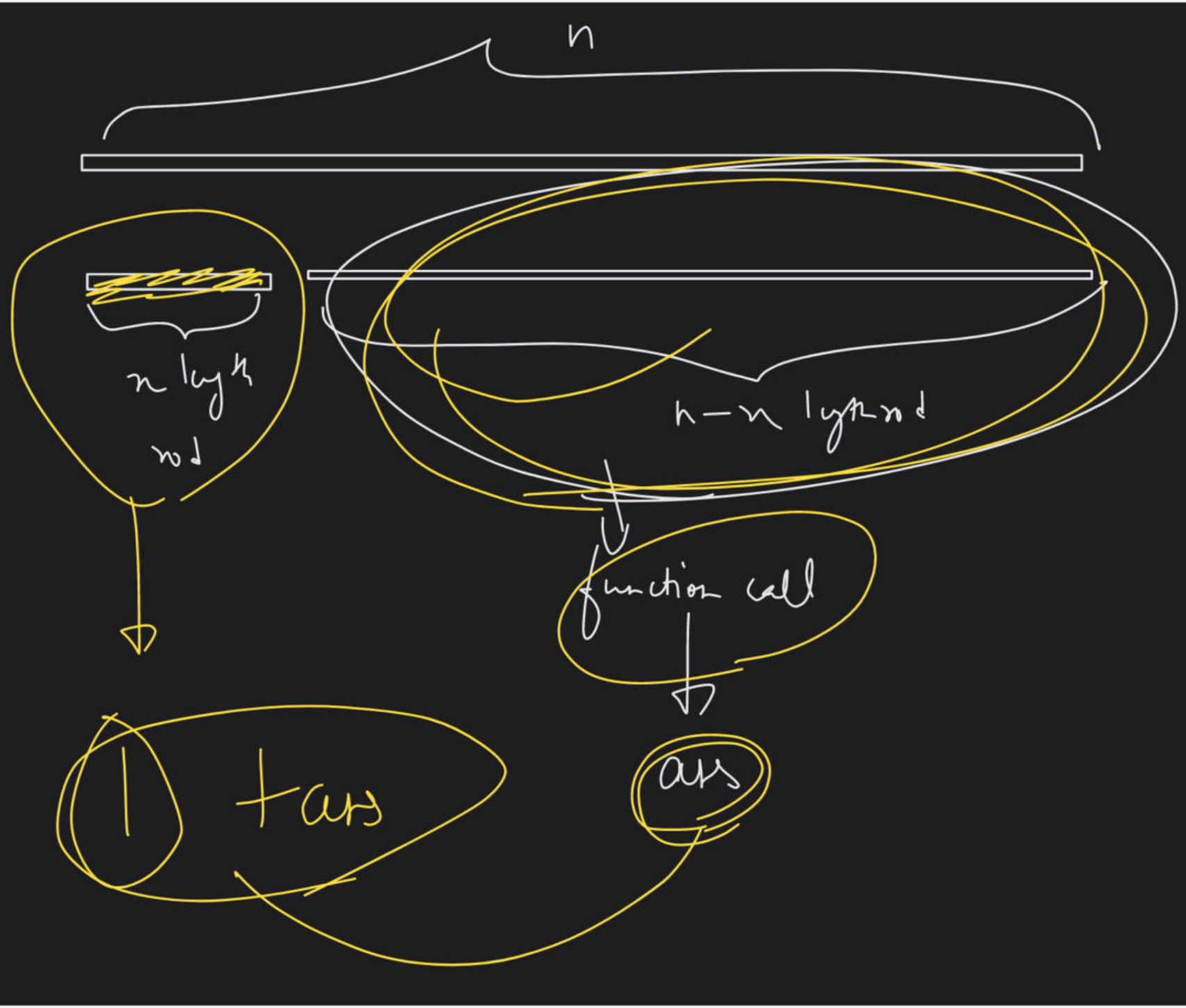

8
5
2



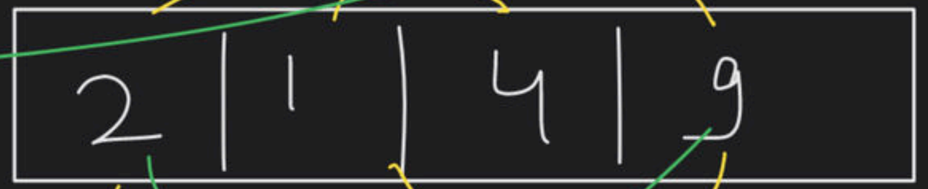
x = 3
y = 3
z = 3
n > 8
INT_MIN+1+1+3
INT_MIN+1+1
INT_MIN+1
<0
8
5
2
-1
3


n = 7
x = 5
y = 2
INT_MIN+1
7
2
INT_MIN+2
5
2
INT_MIN+2
2
0+1
1
5
2
2
-3
0
0
3
INTMIN+2
INTMIN+1
INT_MIN
0
2
-2
1
INTMIN
5
-4
-1
why +1?


7
max no of symcut for length = 2
5
2
2
max no of symcut for length 2
5
max no of symcut for length 5
print
valid ans
7
=0
invalid ans <0
max
count = 1
5 rod
+
max no of symcut for length 2
count = 1
2 rod
+
max no of symcut for leg +


n
n log n
n-n log n
function call
1 + ans
ans

Max sum of non-adjacent elements:



i/p →

maxi

2+9 = 11 ← max sum

2+4 = 6

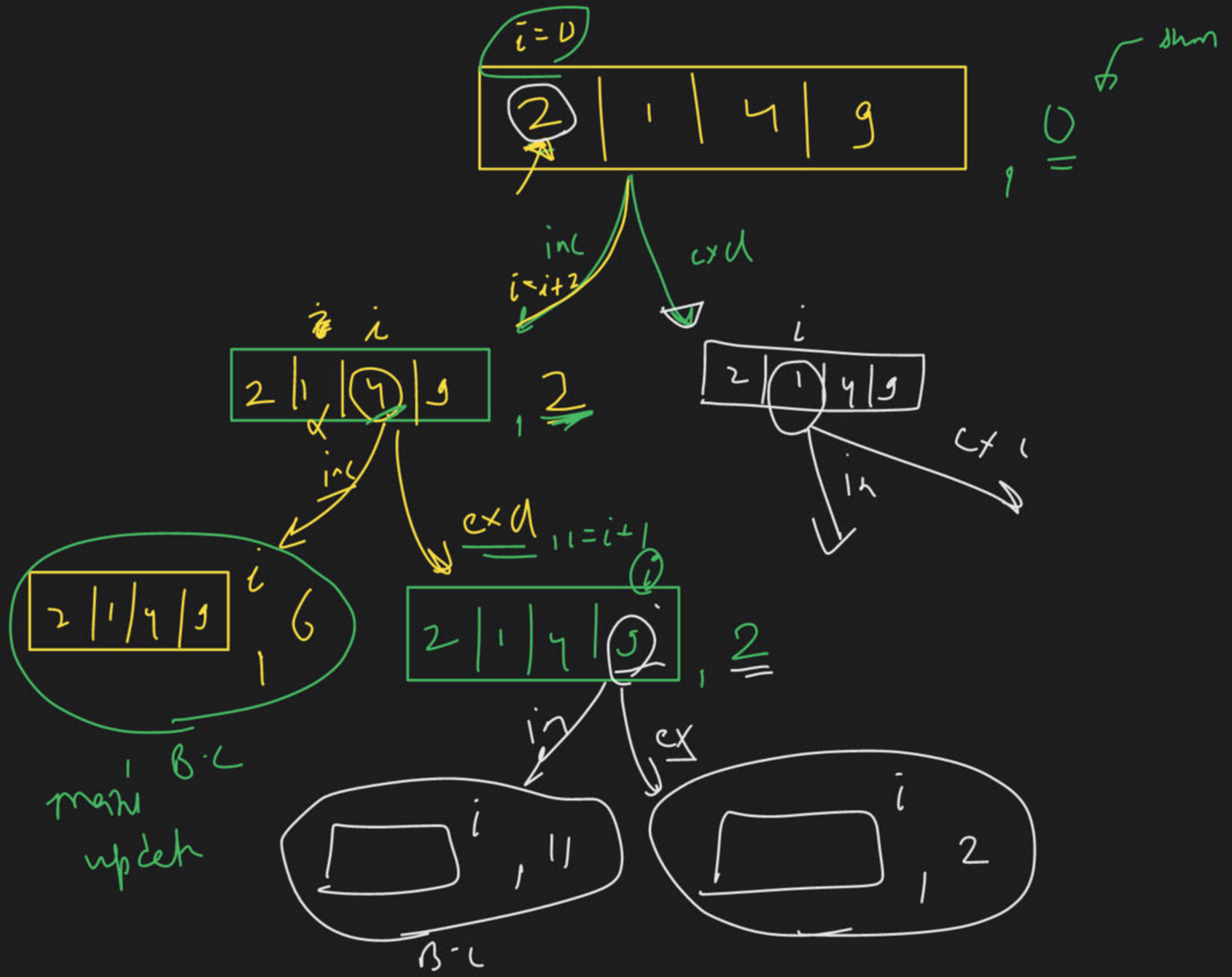
1+9 = 10

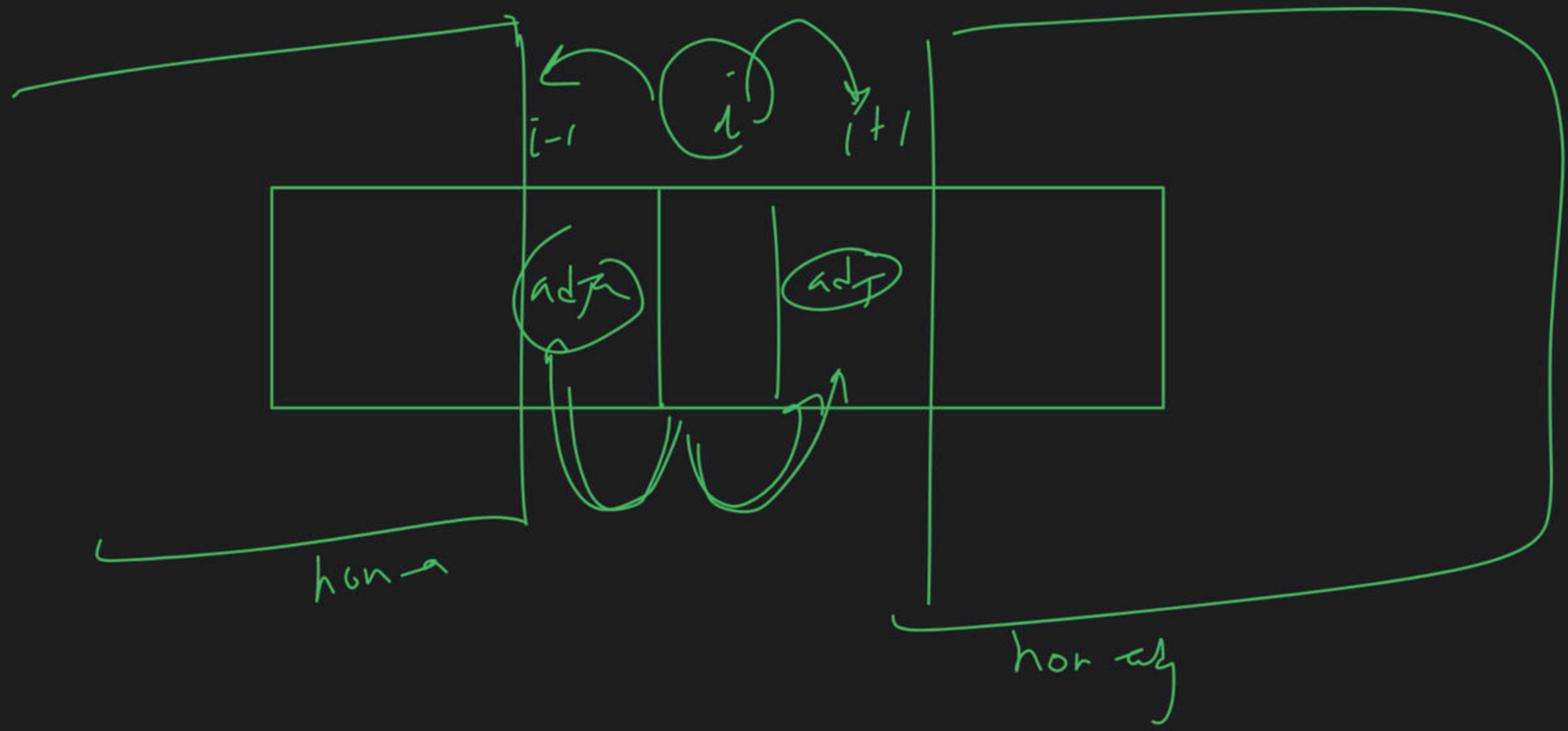
2+9 = 11

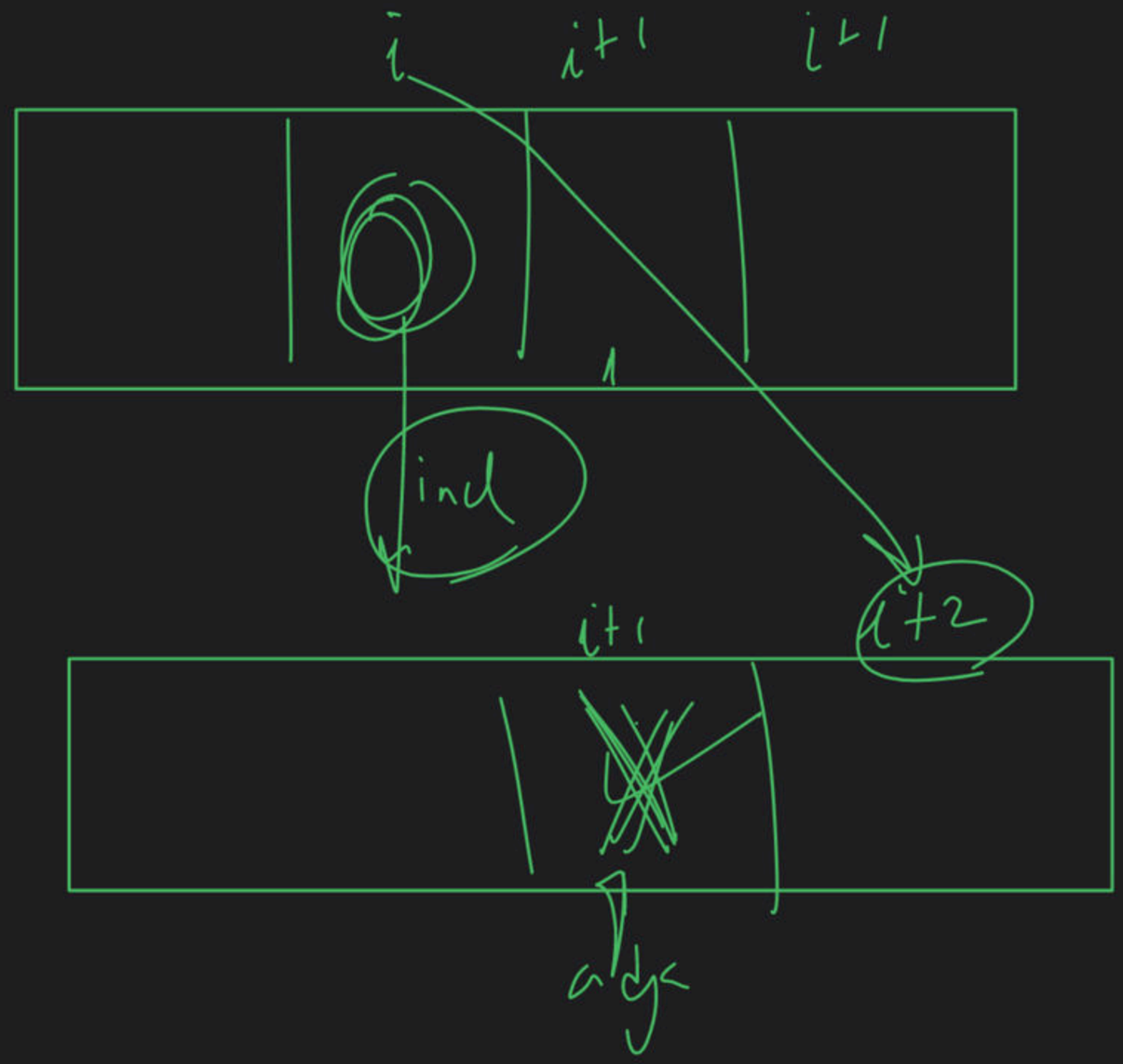
(11) ans

Return the maximum sum of subsequence in which no two elements are adjacent

incl/excl


i=0
2 | 1 | 4 | 9
0
dum
inc
i=i+2
exit
i
2 | 1 | 4 | 9
2
i
2 | 1 | 4 | 9
inc
exit
i=i+1
2 | 1 | 4 | 9
2
i
2 | 1 | 4 | 9
6
B.C
main
update
i
11
i
2
B.C


i-1
i
i+1
adjx
adjy
hor_x
hor_adj


i
i+1
i+1
1
inc
i+2
i+1
adr