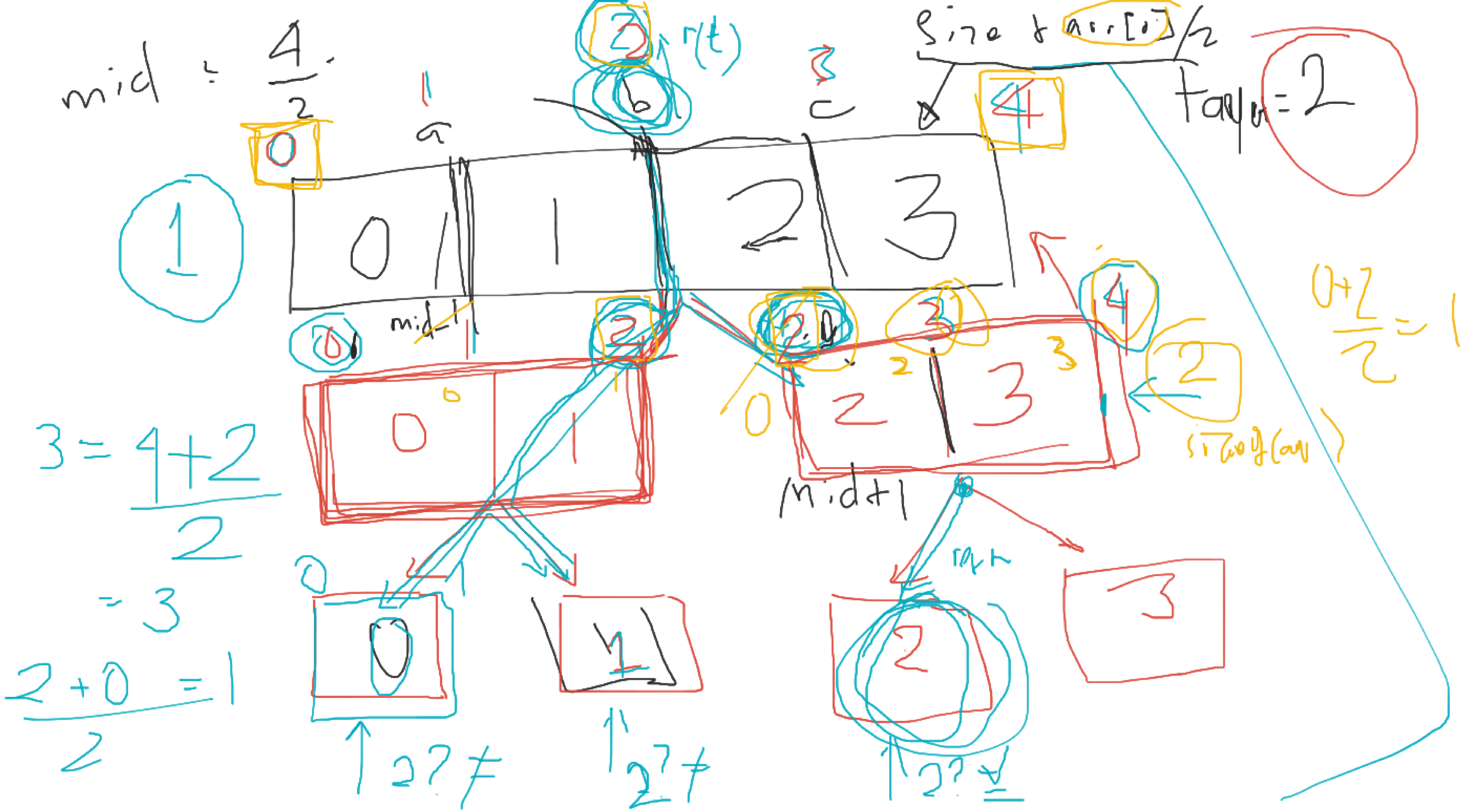
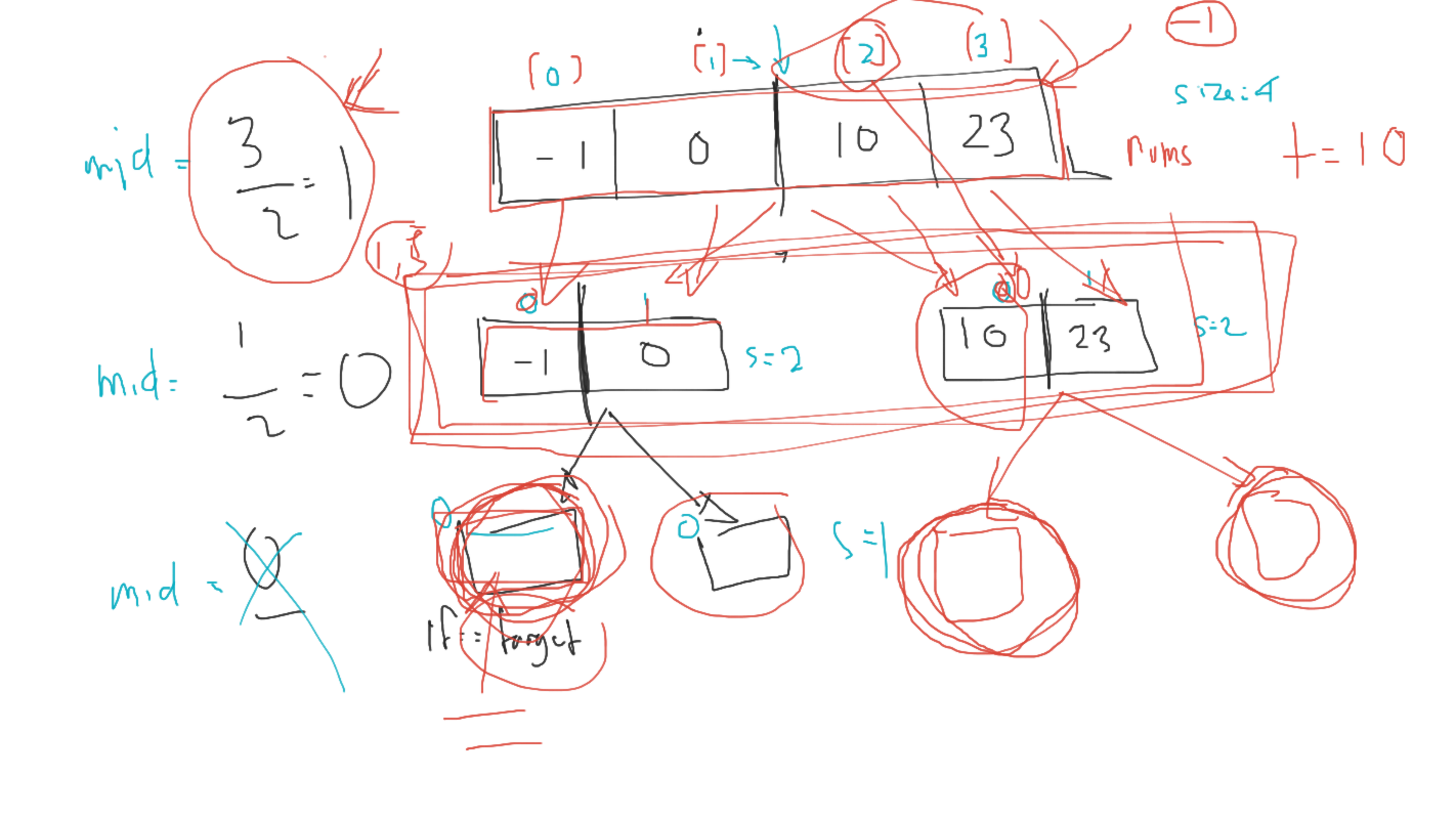
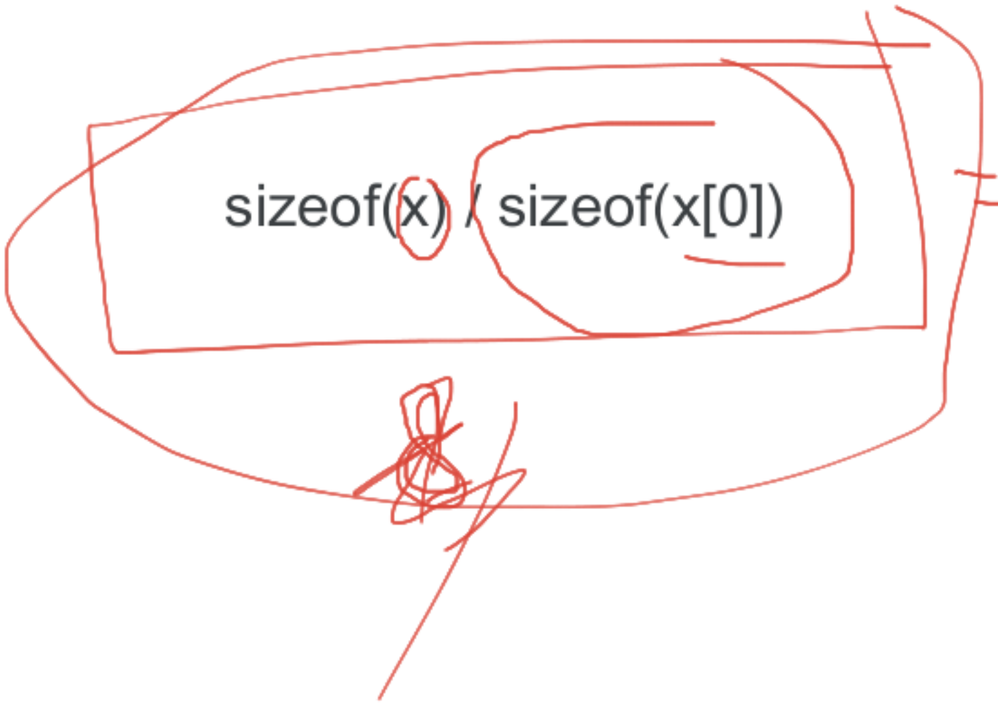
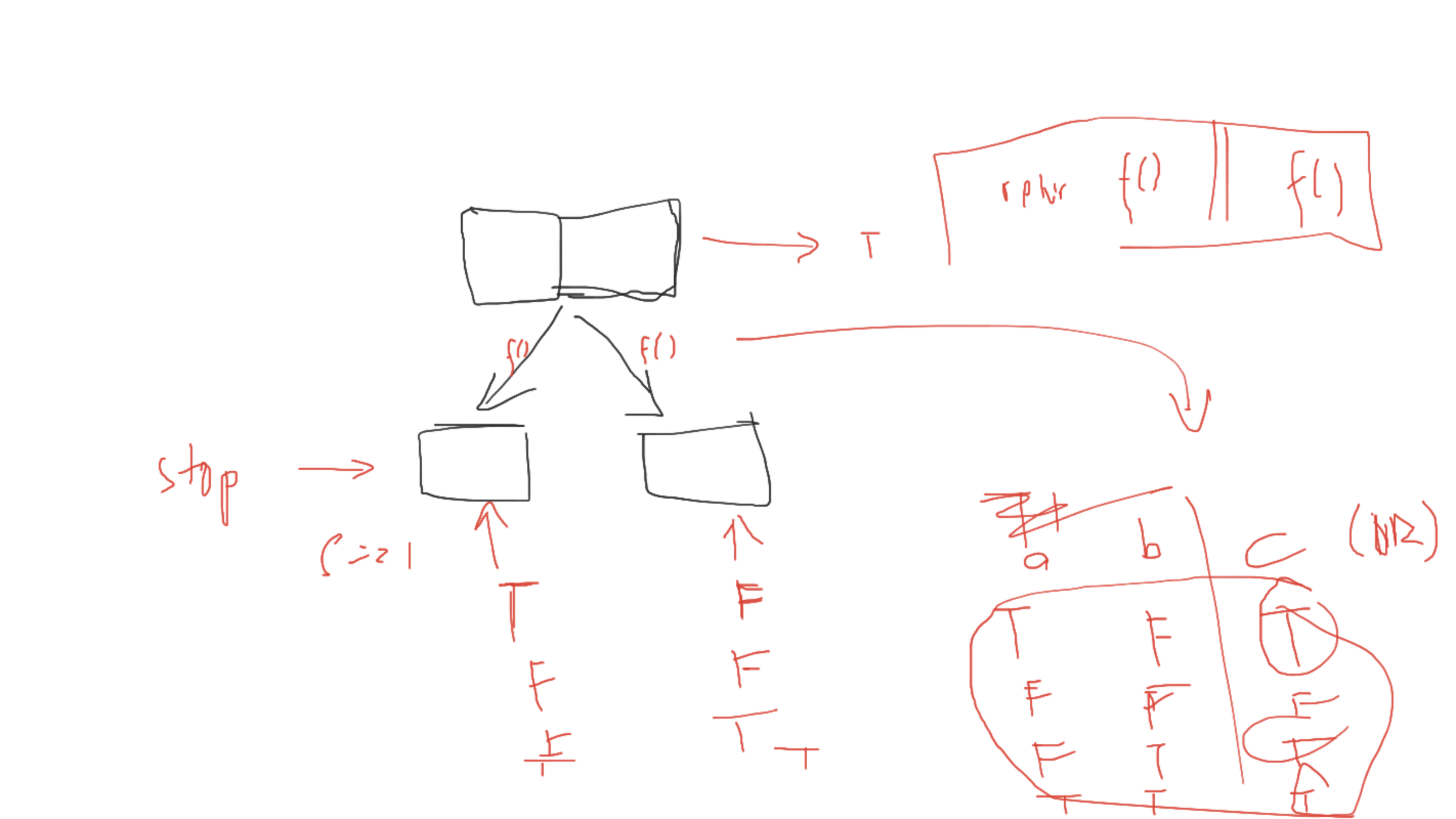
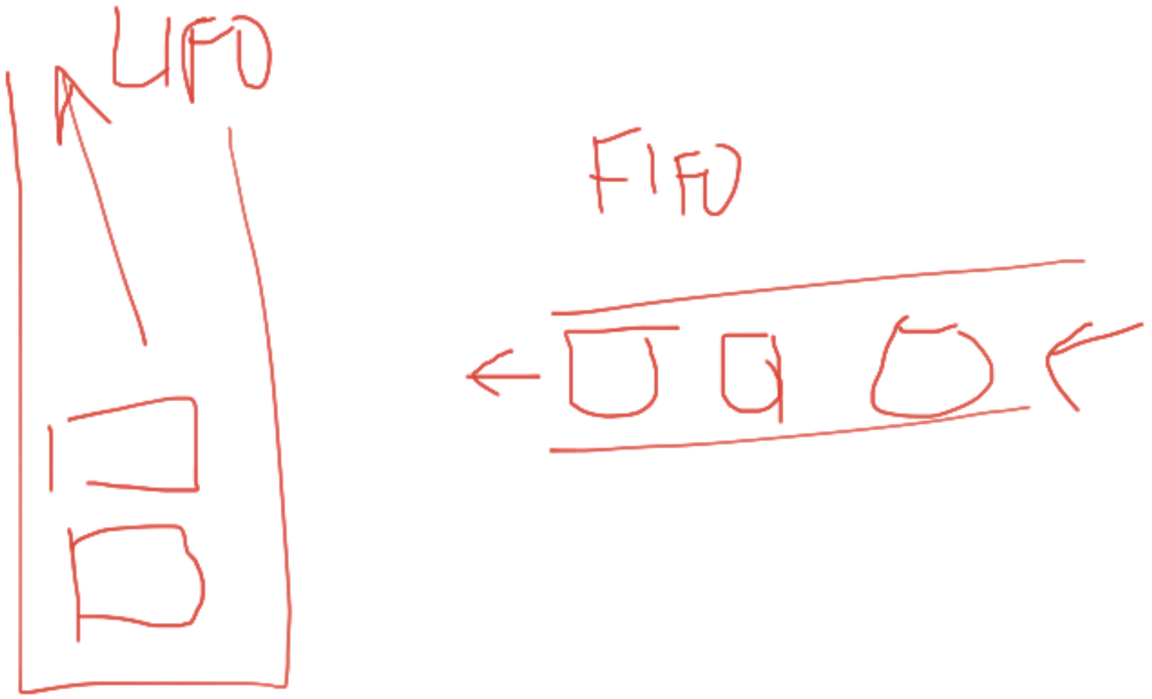
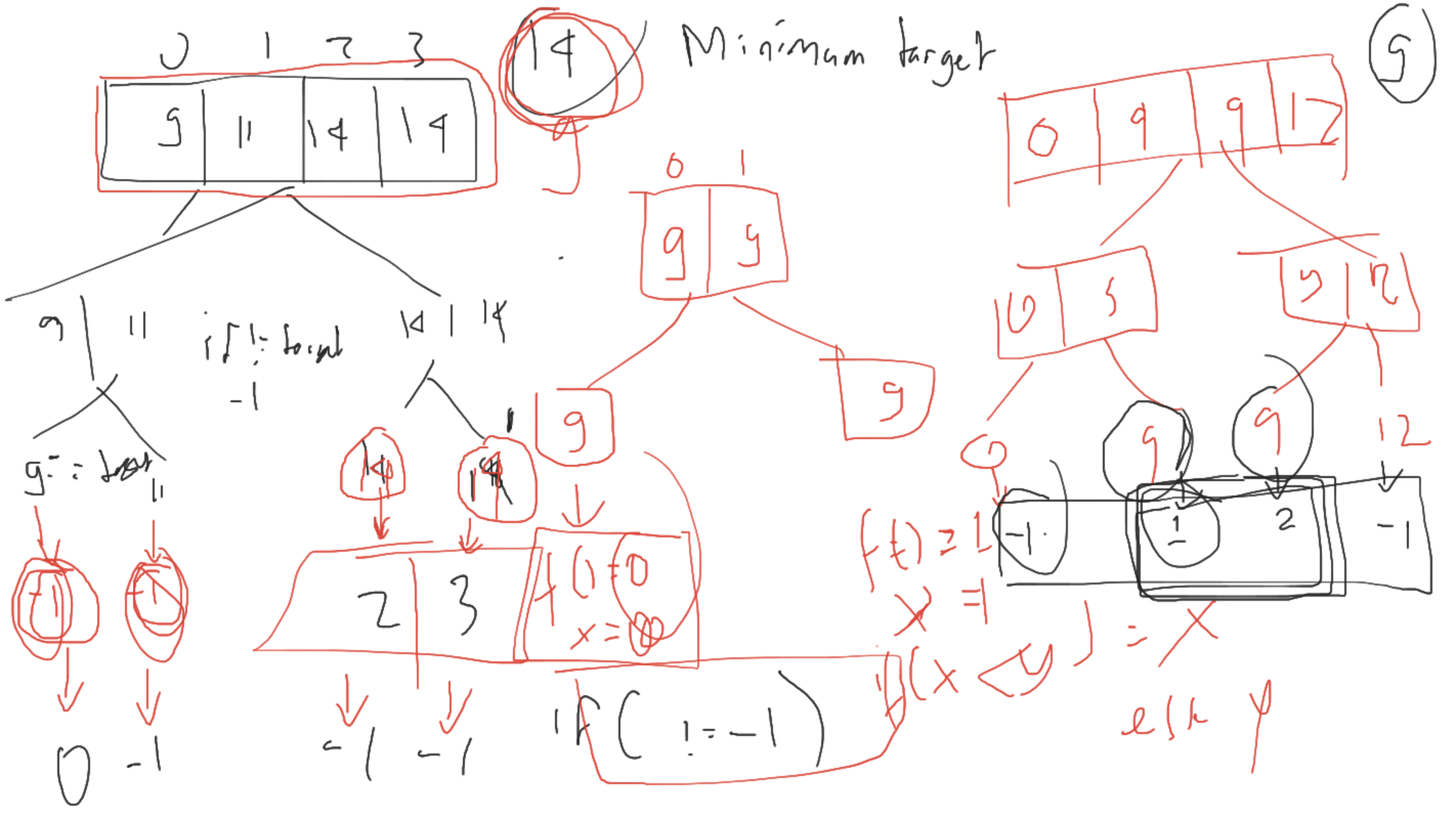

mid = 4/2
size + arr[0] / 2
fayu = 2
mid-1
mid+1
3 = 4+2/2
= 3
2+0/2 = 1
0+2/2 = 1
r(t)

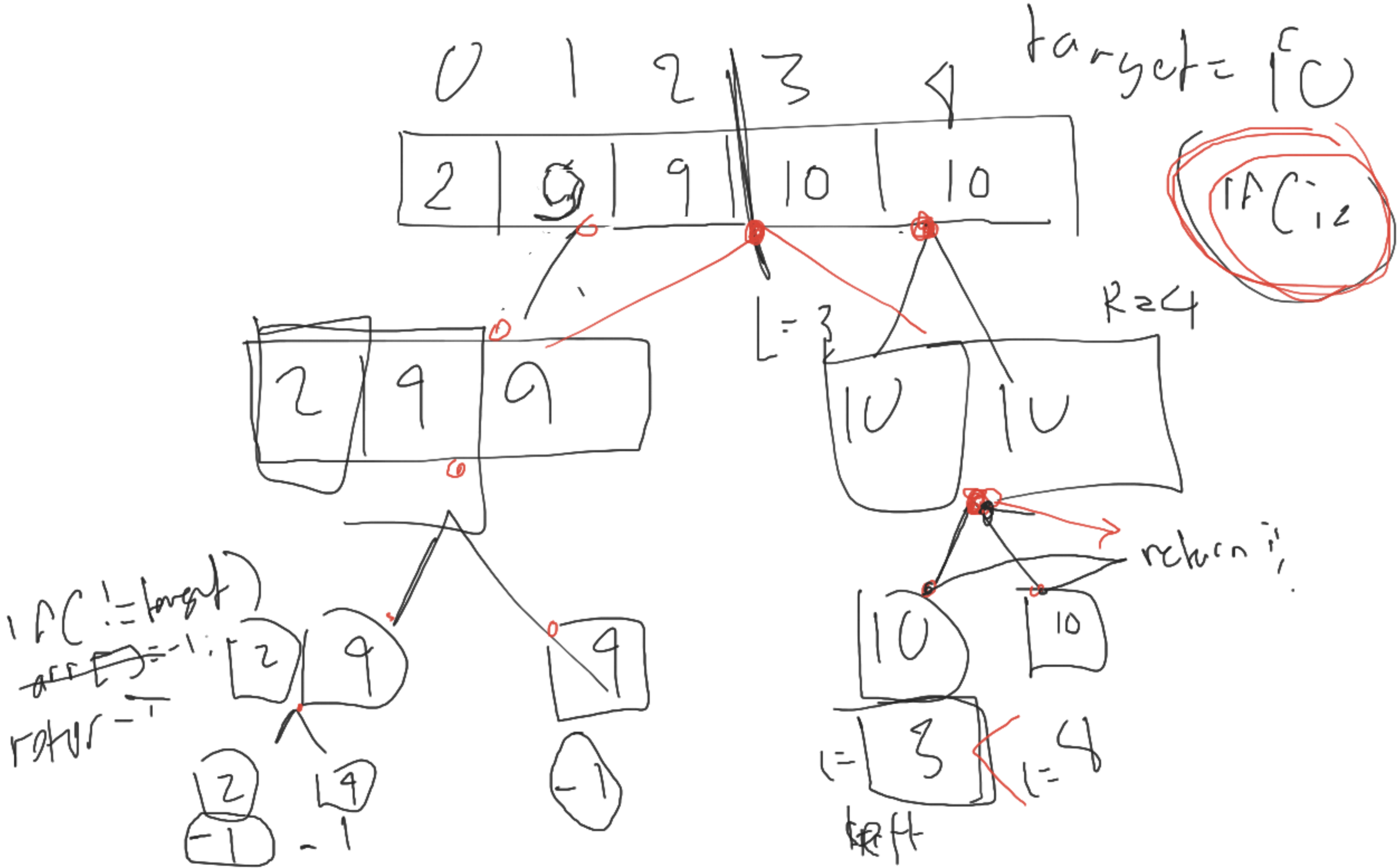

mid = 3/2 = 1
[0]
[1]
[2]
[3]
-1
size: 4
-1
0
10
23
nums
t = 10
mid = 1/2 = 0
0
1
-1
0
s=2
0
1
10
23
s=2
mid =
0
0
s=1
if = target

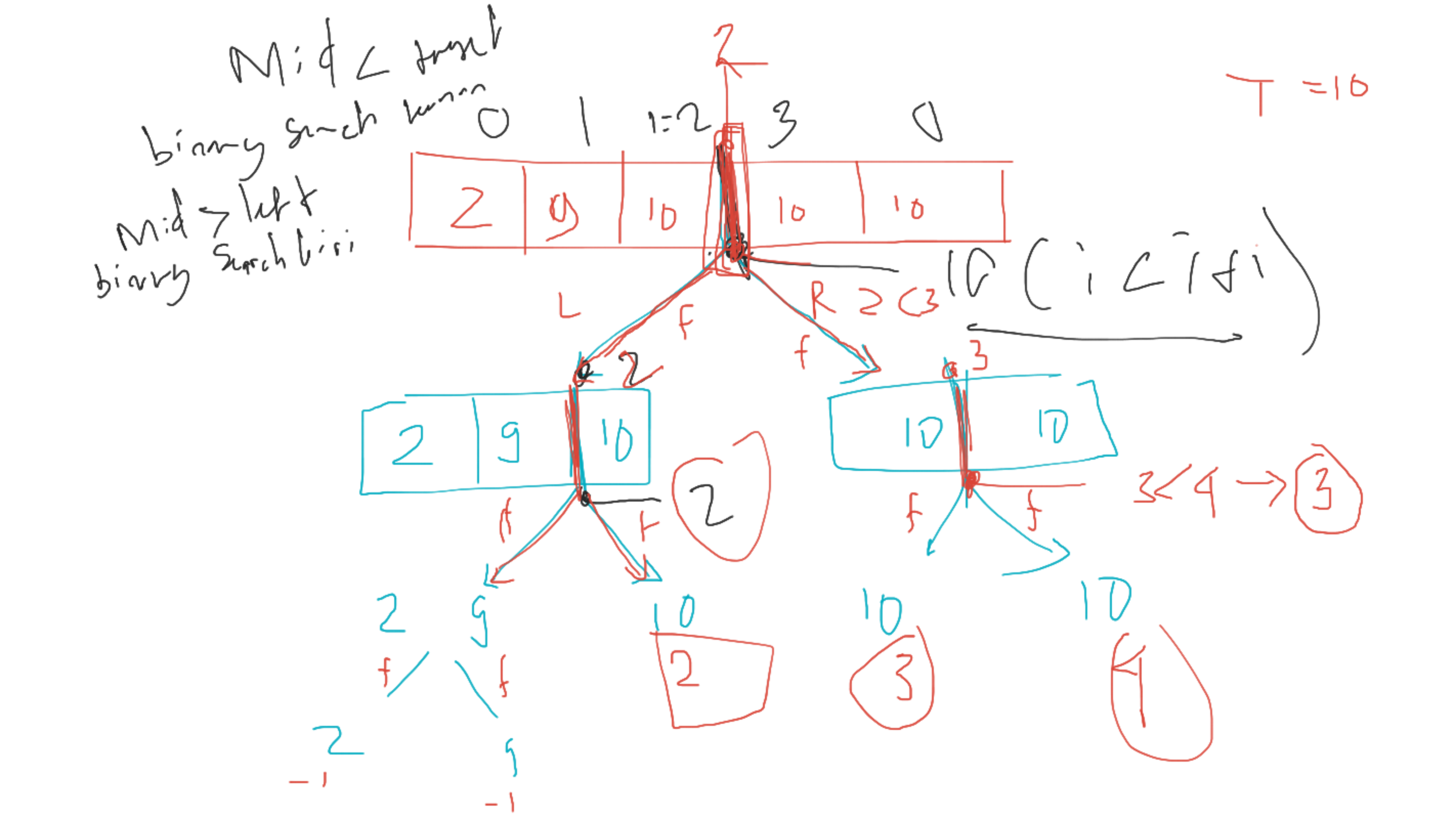

sizeof(x) / sizeof(x[0])


T
f()
f()
stop
c=2
T
F
F
F
F
T
T
T
a
b
c
(OR)
T
F
F
F
F
T
T
T
F
F
T
T


LIFO
FIFO


Minimum target
9
0 1 2 3
9 11 14 14
14
9 11
14 14
if != target
-1
0 1
9 9
9
14 14
2 3
1 0 = 0
x = 0
if (1 != -1)
0 -1
-1 -1
0 9 9 12
0 9
9 12
-1 1 2 -1
f(1) = 1
x = 1
if(x < y) = x
else y


0 1 2 3 4
target = 10
2 9 9 10 10
L = 3
R = 4
2 9 9
10 10
return i;
10 10
L = 3
L = 4
-1 -1


Mid < first
binary search turn
Mid > left
binary search bili
T =10
0 1 1:2 3 4
2 9 10 10 10
2
L
f
R ≥ 3
10 (i < ifi)
f
2
2 9 10
3
10 10
f f
f f
2
3<4 → 3
2 9
f f
10
2
10
3
10
4
2
-1
9
-1

f(x) {

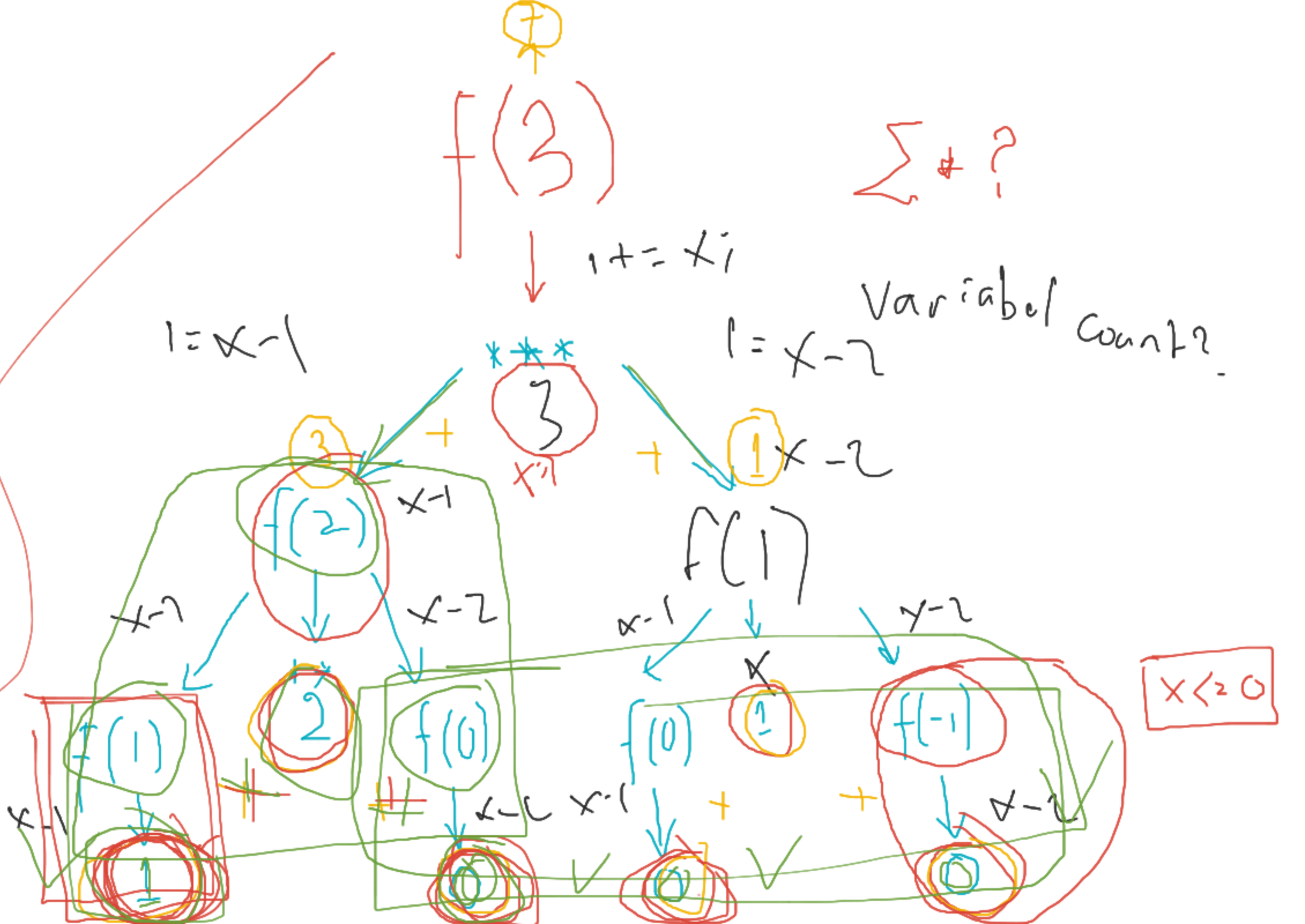
f(x-1) -

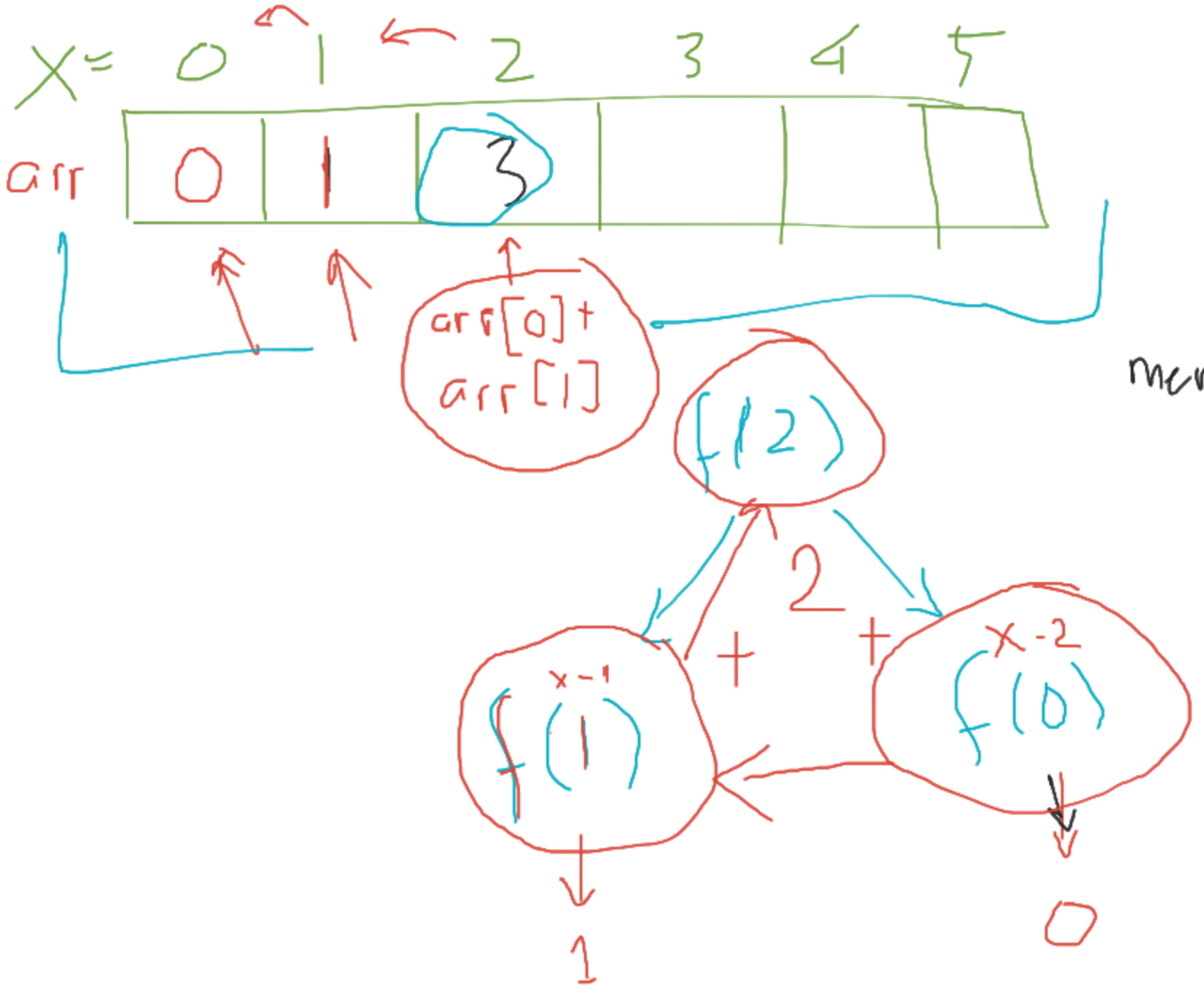
print(x) x 2

f(x-2) -

}

```
int fraktal_B(int x) {
  int i;
  if (x <= 0) return 0;
  else {
    int bintang = x;
    bintang += fraktal_B(x - 1);
    bintang += fraktal_B(x - 2);
    return bintang;
  }
}
```





f(2) = arr[0] + arr[1]

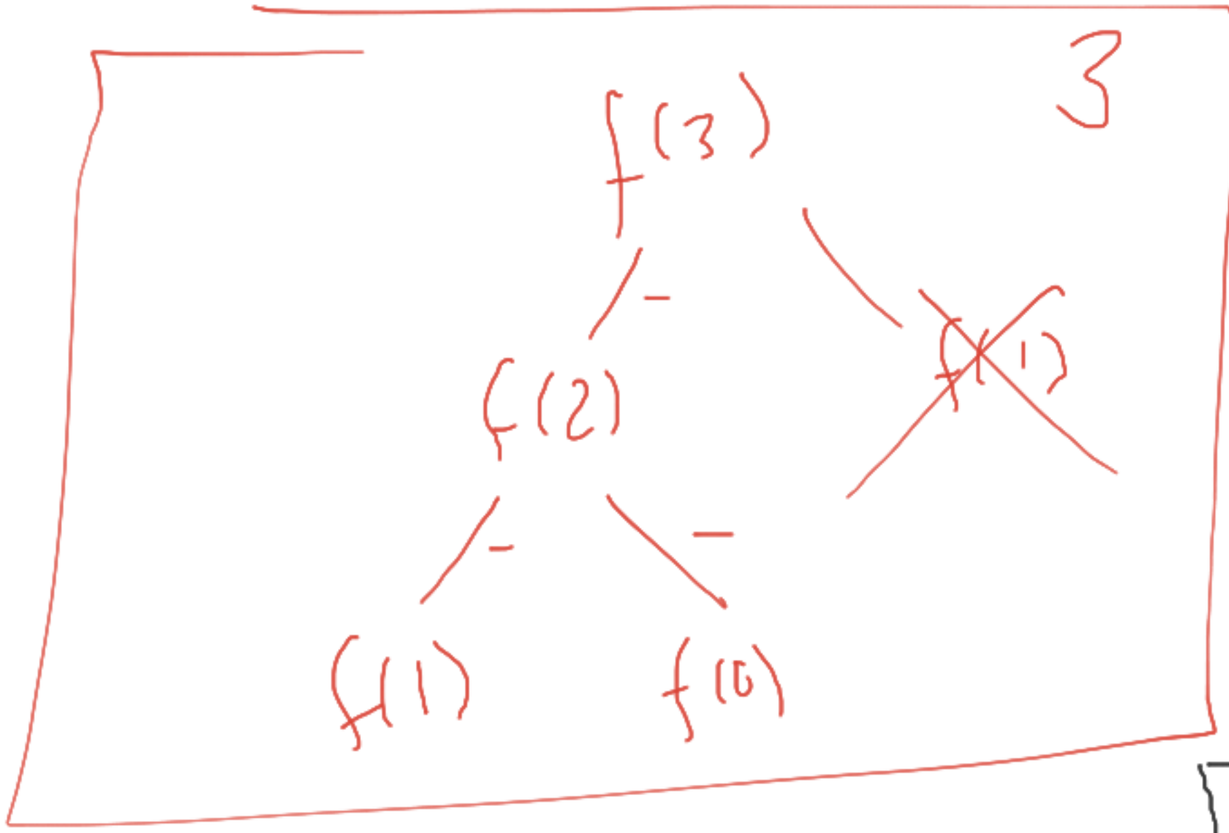
memo[x] =

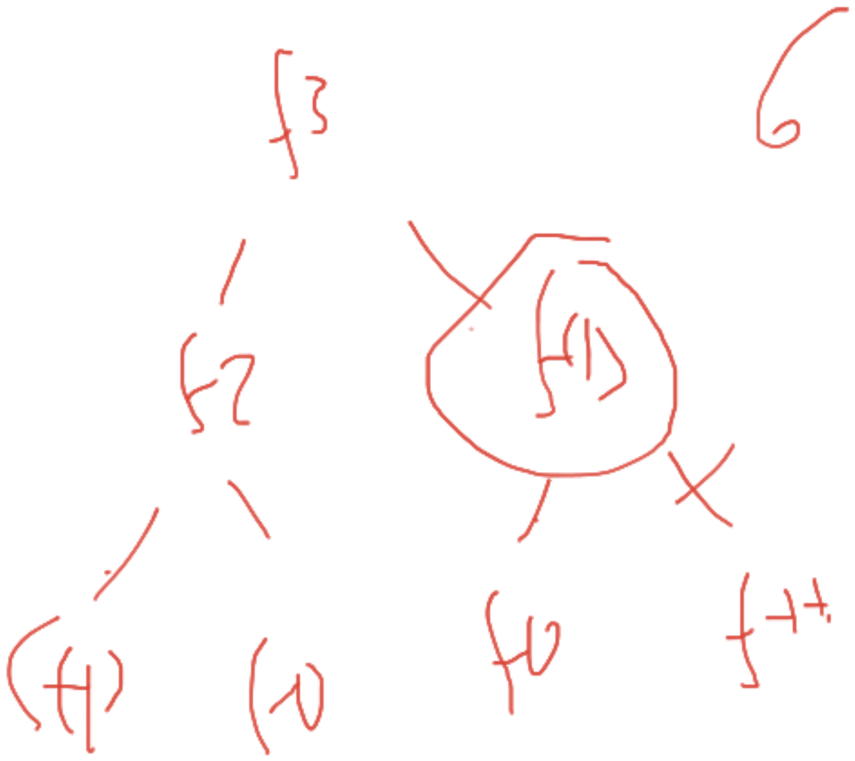
f(2) = f(1) + f(0)

f(2) = arr[0] + arr[1]

if(arr[x] != 0)
return arr[x]

else {
f(x)

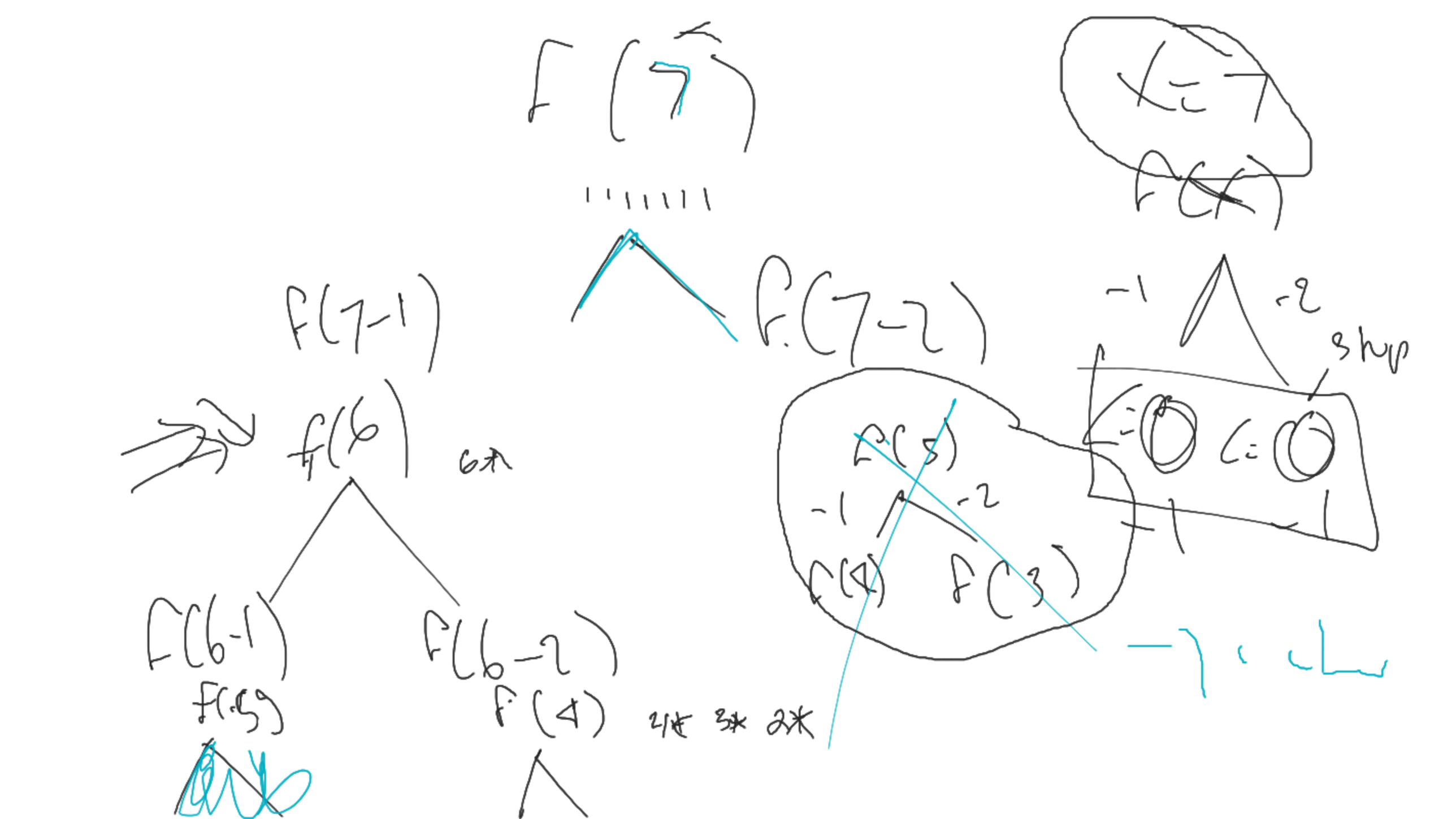


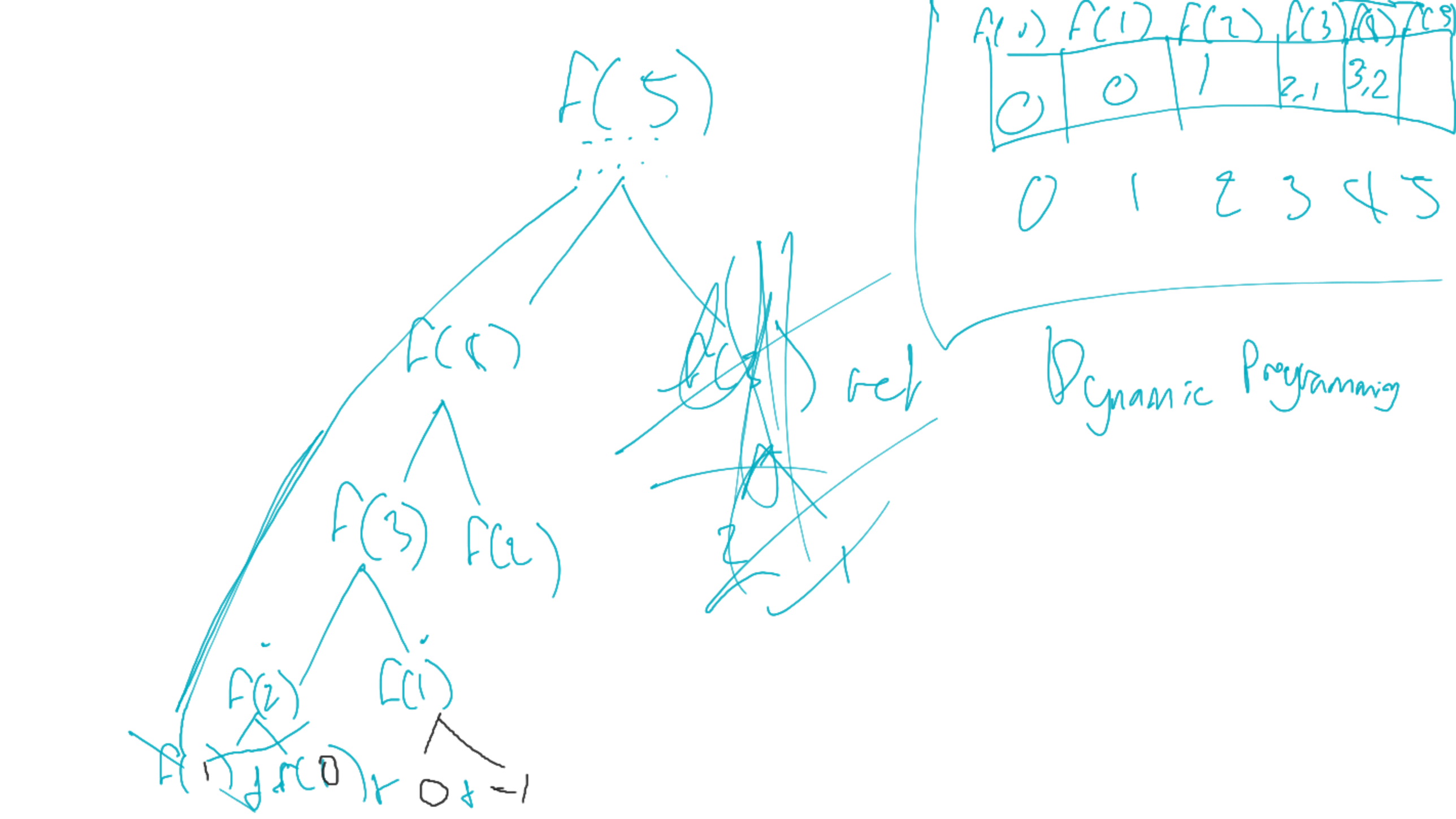


Dynamic Programming !

Memorization

80


f(7)
f(7-1)
f(7-2)
f(6)
f(6-1)
f(6-2)
f(5)
f(4)
f(5)
f(4)
f(3)
-1
-2
stop
6개
2개 3개 2개


F(5)
F(4)
F(3)
F(2)
F(2)
F(1)
F(1)
F(0)
0
1
F(0)
F(1)
F(2)
F(3)
F(4)
F(5)
0
0
1
2,1
3,2
0 1 2 3 4 5
Dynamic Programming