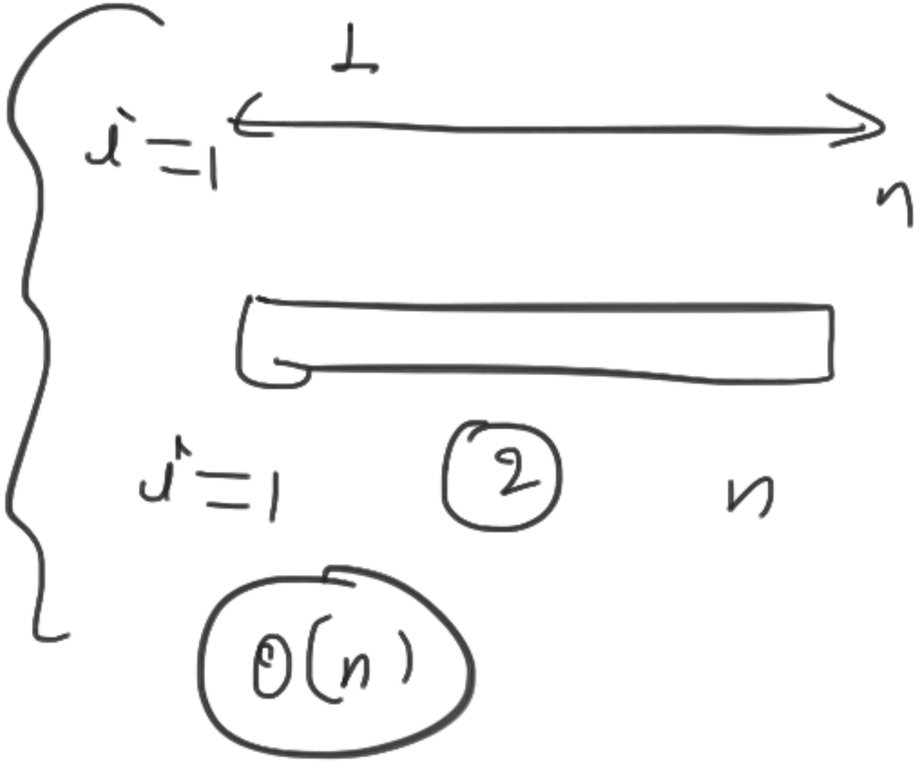
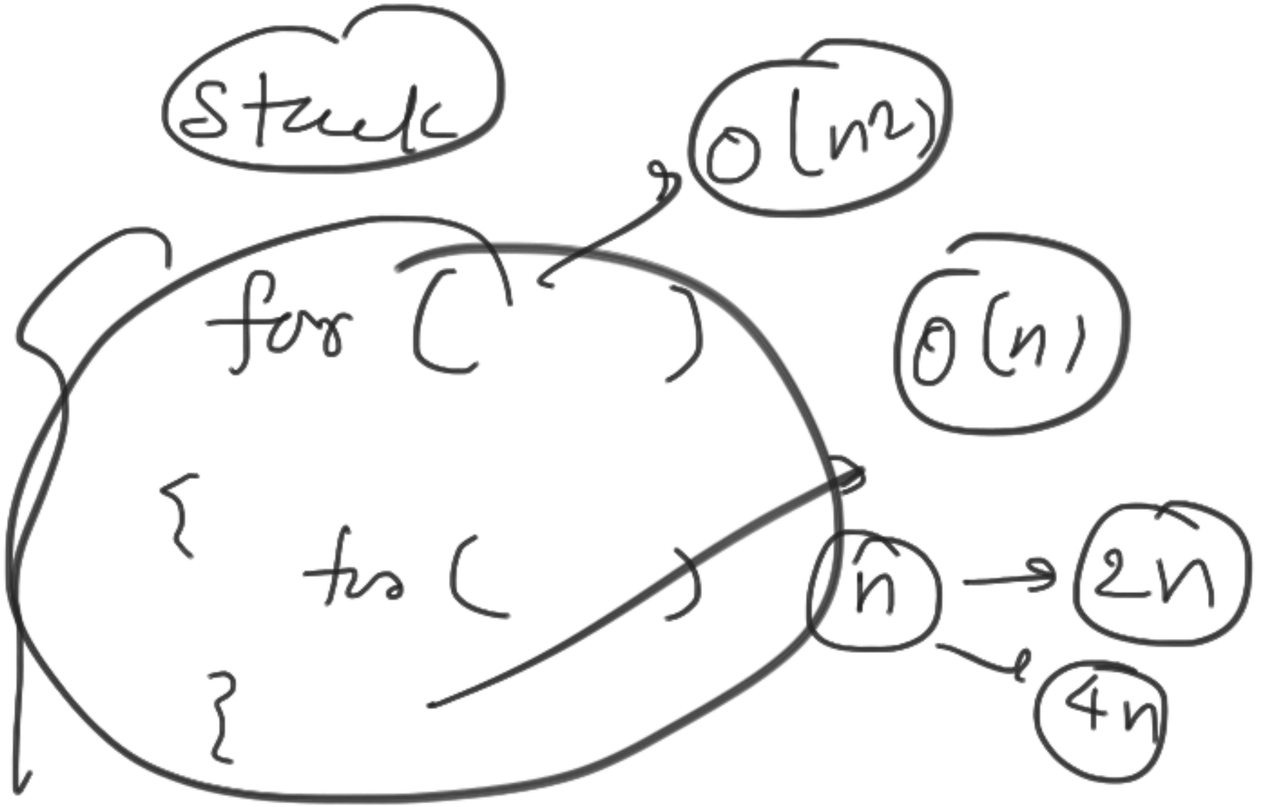
Total no. of iterations × (no. of operations per iteration)

n → time ⇒ O(1) ⇒ n · O(1) = O(n)





for (i = 1 to n)

{

for (j = 1 to i)

{

O(1)

}

}

for (i = 1 to n)

{

for (j = i-1 to 0)

{

}

}

for (i = 1 to n)

for (j = i+1 to n)

;

O(1)

i = 1 ⇒ 2 to n (n-1)

i = 2 ⇒ 3 to n (n-2)

$(n-1) + (n-2) + (n-3) \; - - - + 1$

$= \frac{n(n-1)}{2}$

$O(1)$

$O(1)$

2^i

$\Rightarrow O(n^2)$

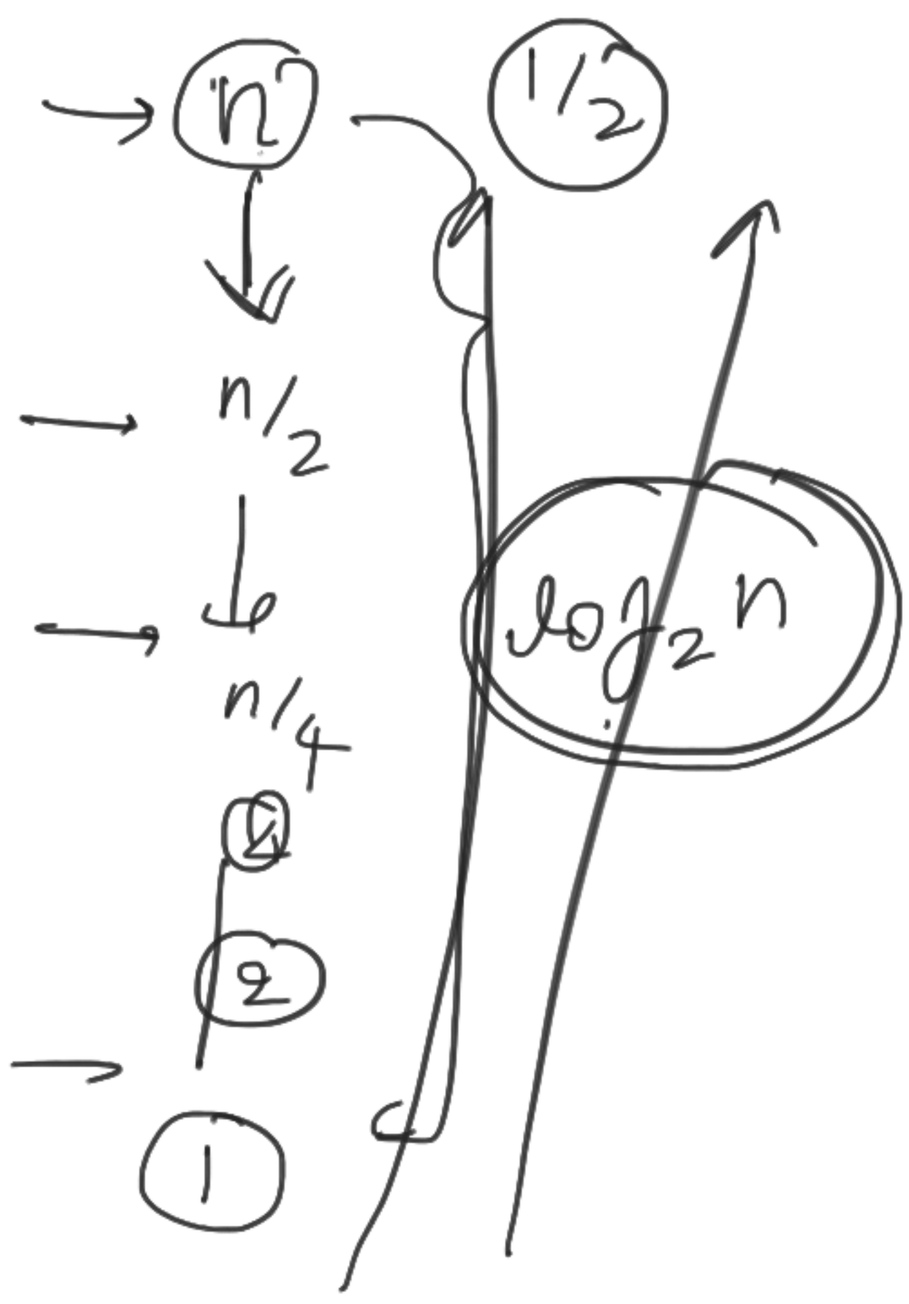
$O(n^2) \cdot O(1)$

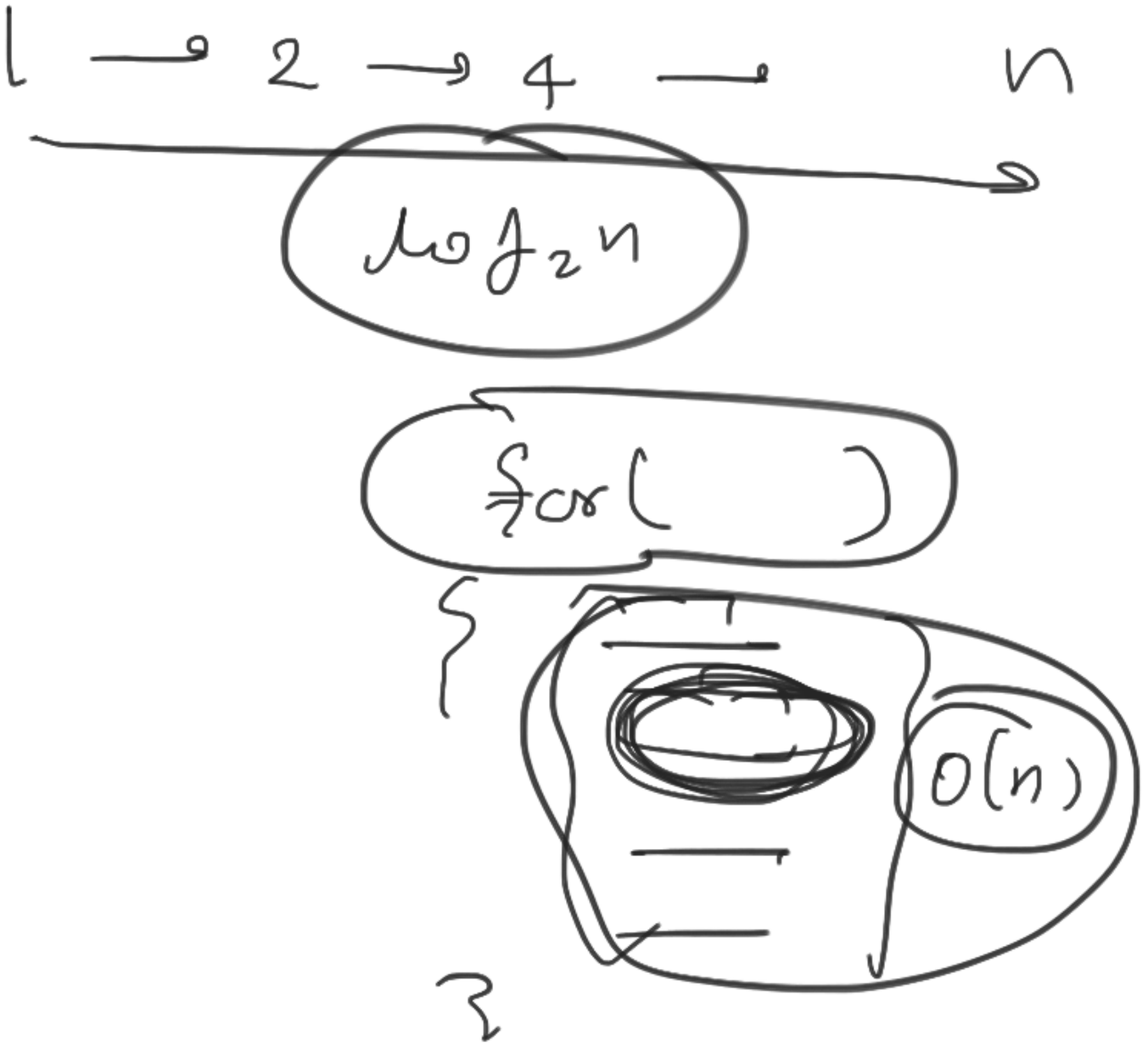
$= O(n^2)$

$i = 1, 2, 4, 8 - n$

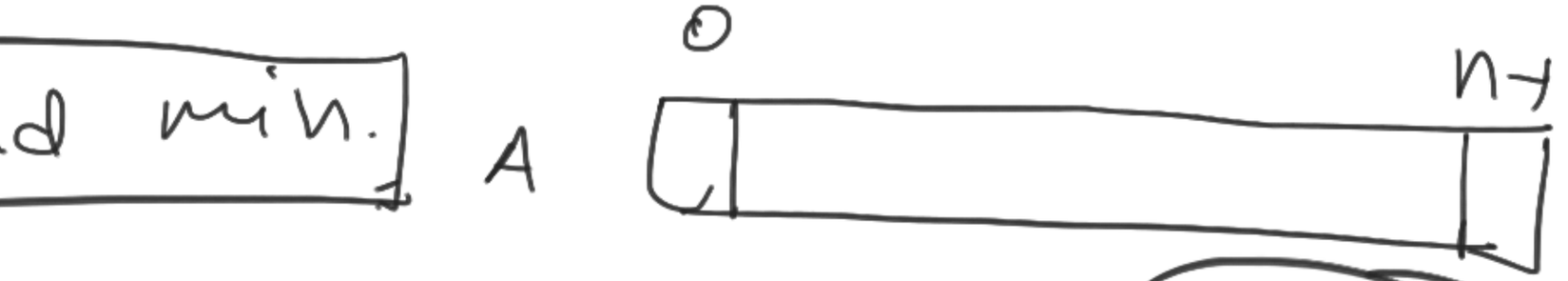
$O(1)$

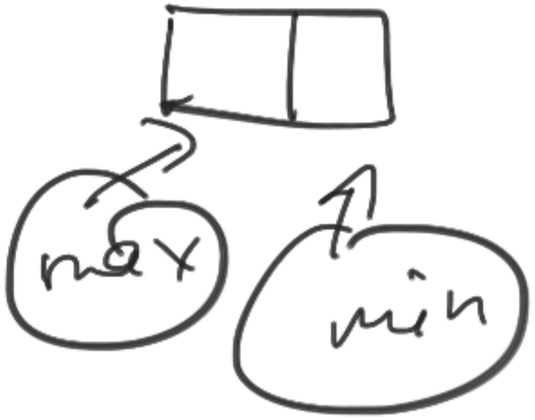
$\log n$


n
1/2
n/2
n/4
2
1
log2n


1 → 2 → 4 → n
log2n
for( )
{
O(n)
}

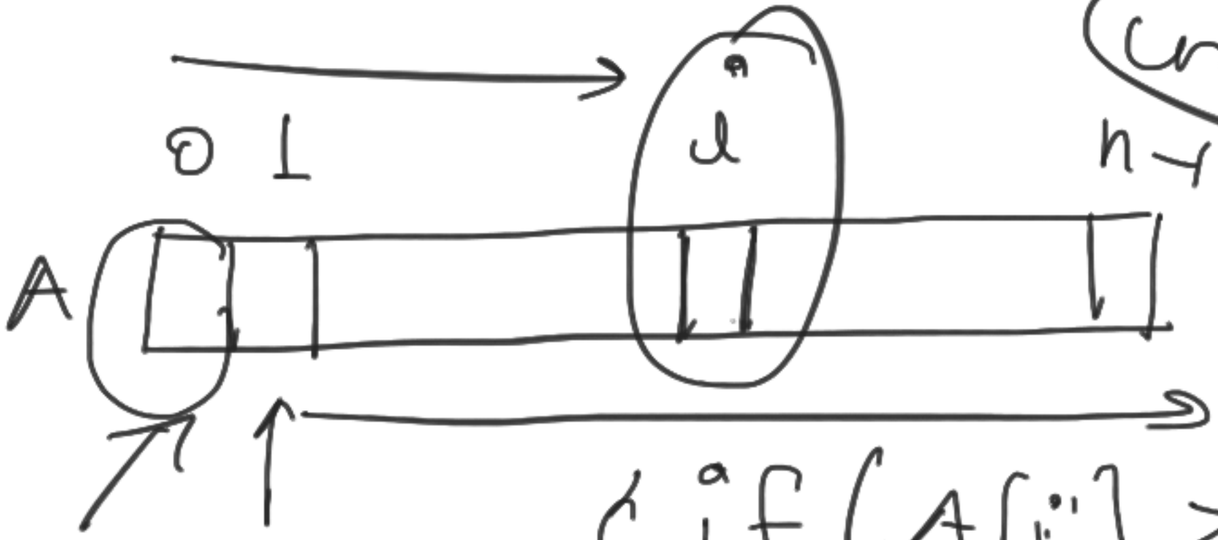
find max and min.





if

else



max = A[0]

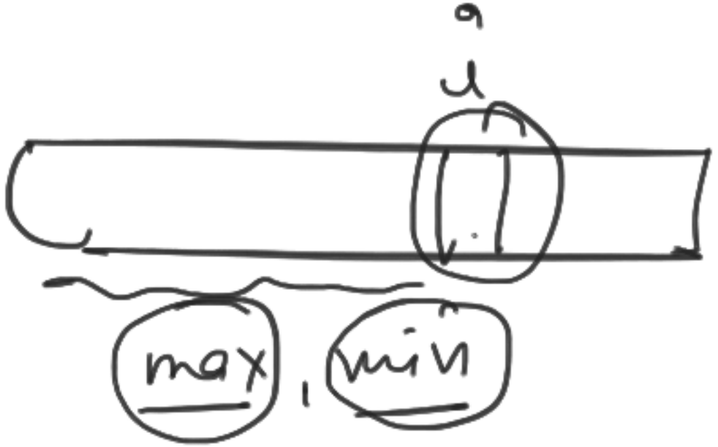
min

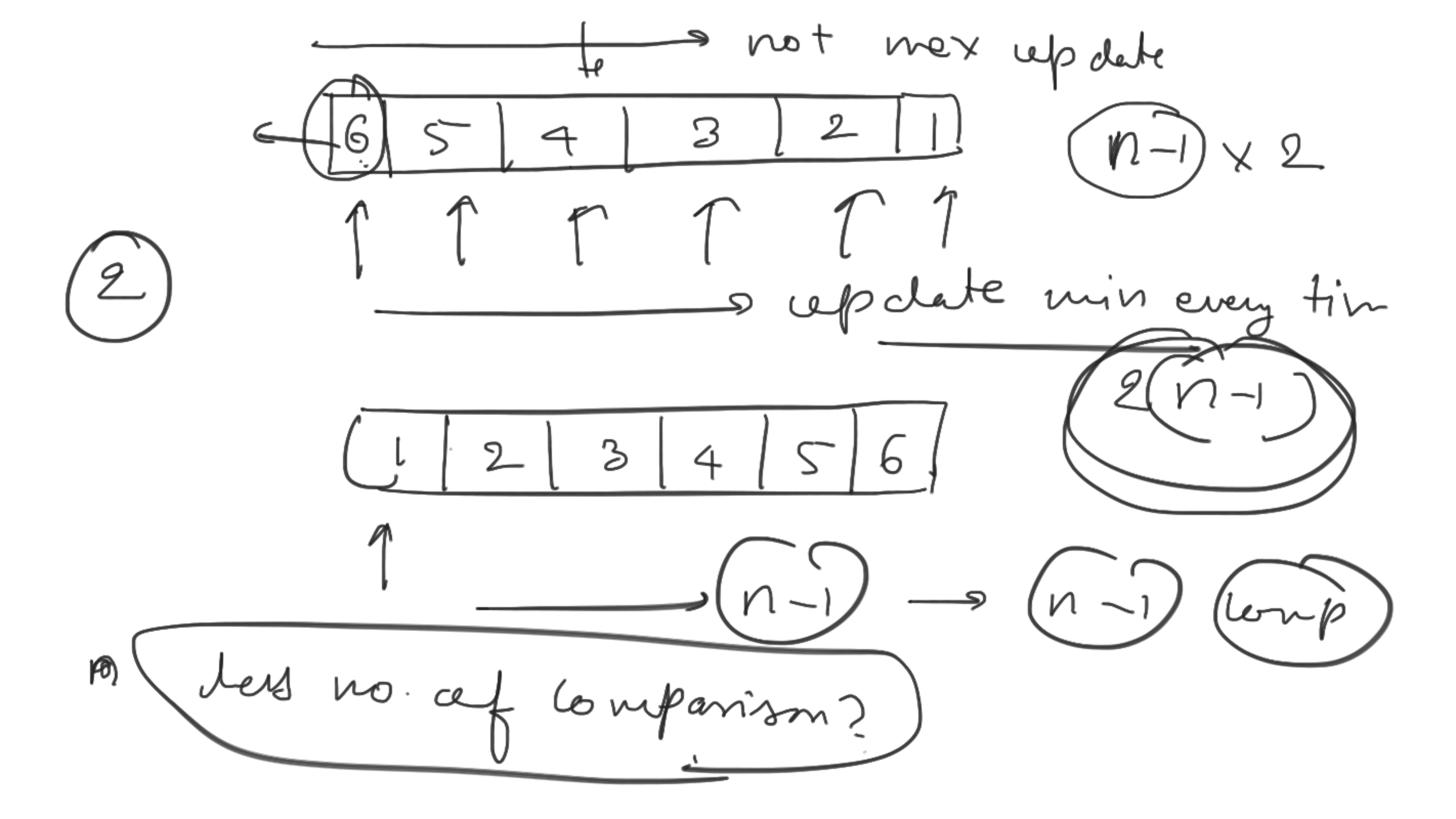
if (A[i] > max)

max = A[i]

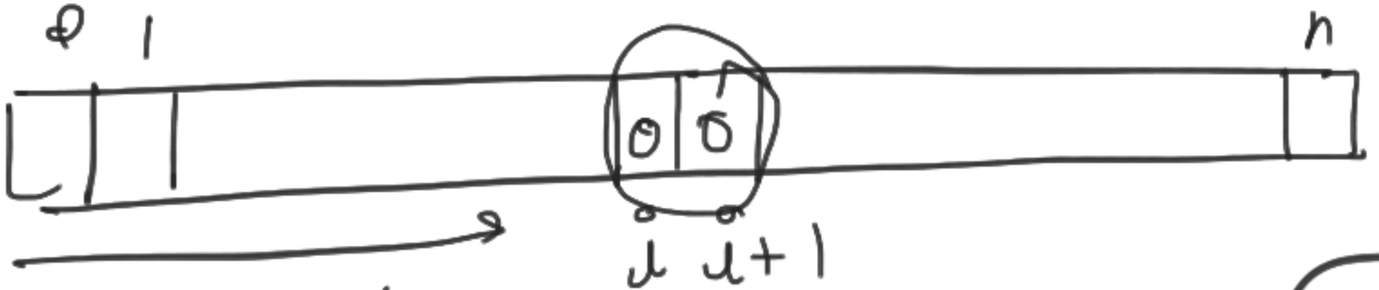
else (A[i] < min)

min = A[i]




not mex update
6 5 4 3 2 1
n-1 x 2
2
update min every time
2 (n-1)
1 2 3 4 5 6
n-1
n-1
loop
lets no. of comparison?

O(n)



n → 3n/2

n/2

i i+1

2

max, min

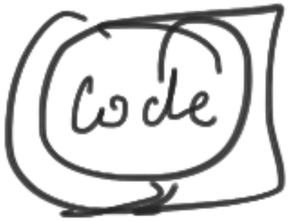
```
if( A[i] > A[i+1] )
{
    if ( A[i] > max )
        max = A[i]
    if ( A[i+1] < min )
        min = A[i+1]
}
else{
    if( A[i+1] > max )
        max = A[i+1]
    if ( A[i] < min )
        min = A[i]
}
```

3 → 3n/2

2 → 3

n → 3n/2

Recursive approach



max
min

2

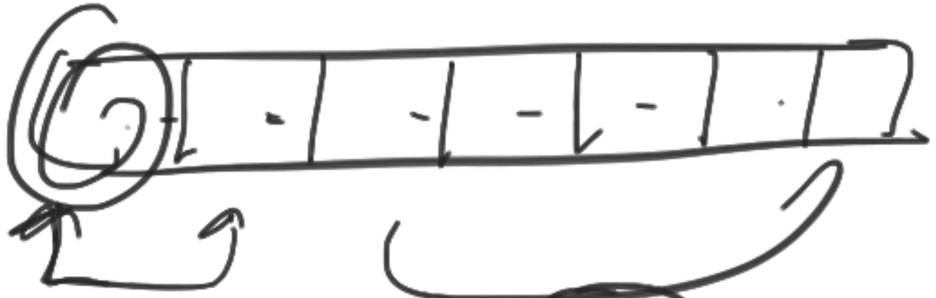
n even

n odd

n → odd

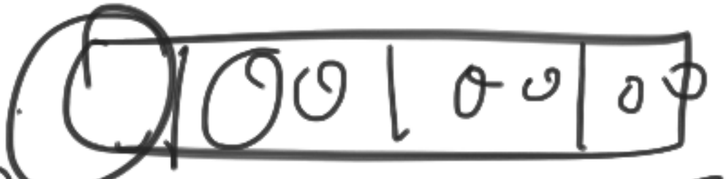
{

{



5 ⇒ 2

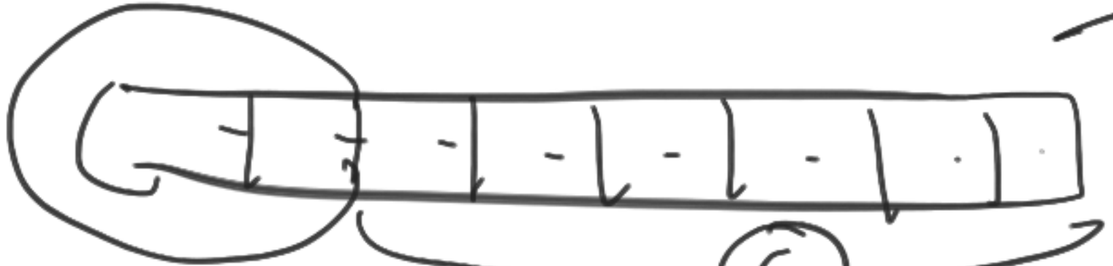
7



6

new

00

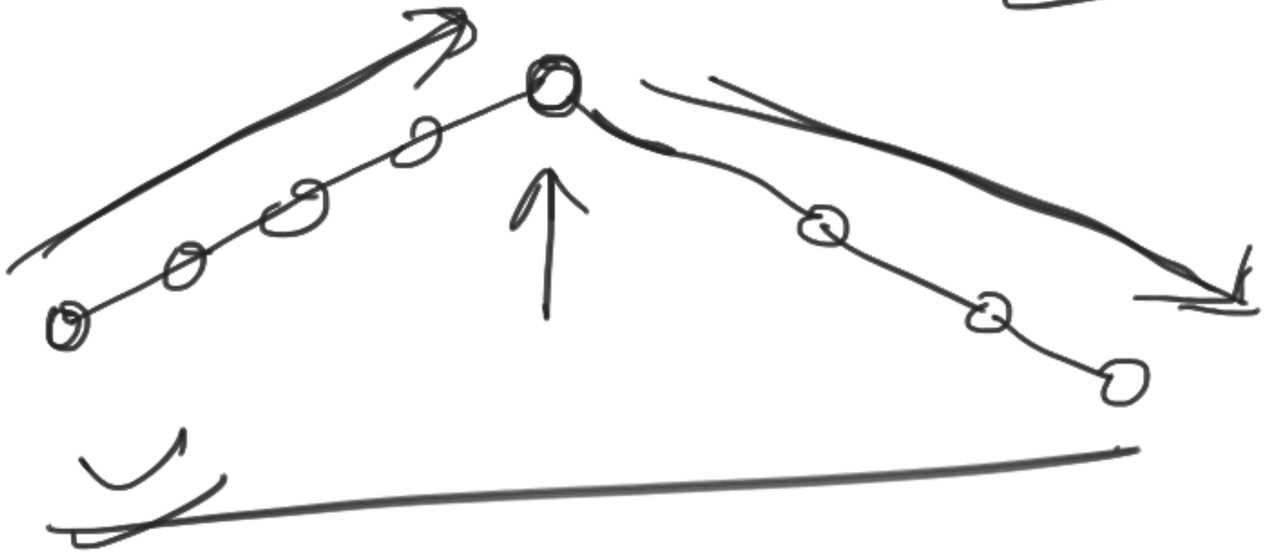


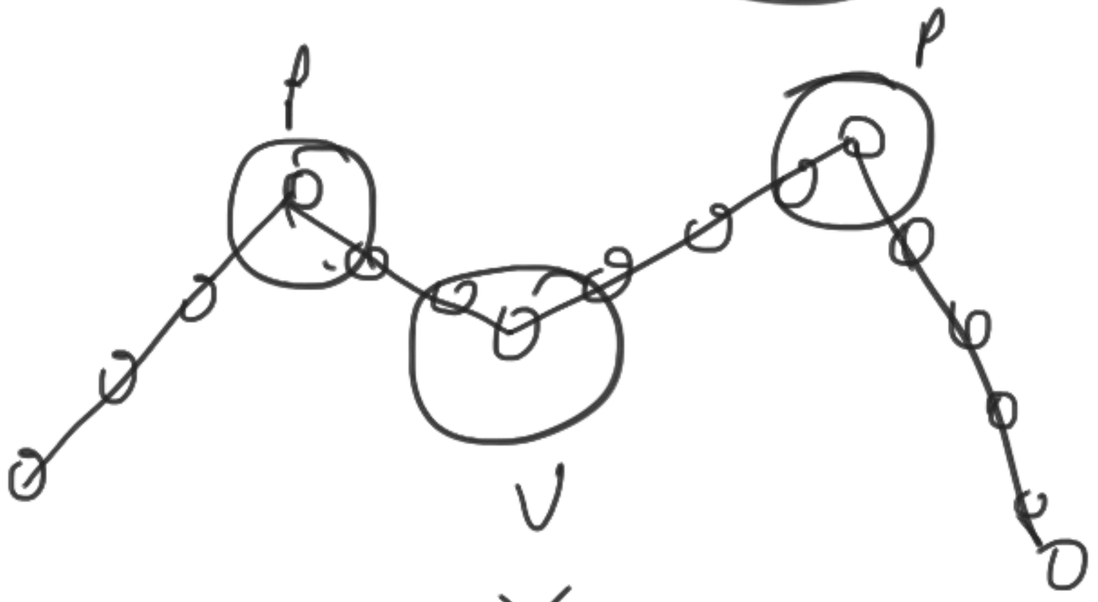
6

$(6)_2 = 3$ th...

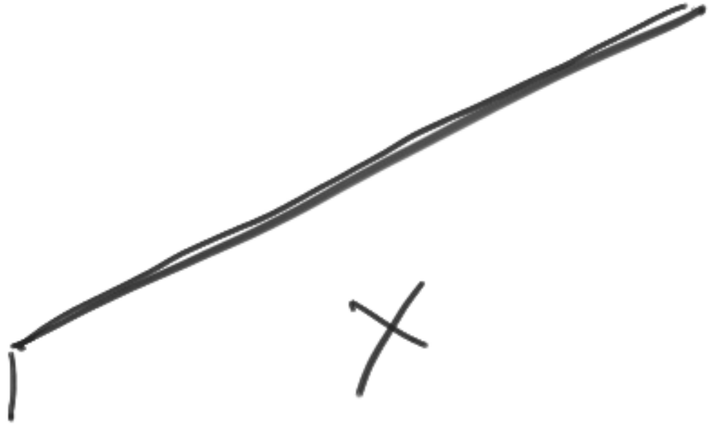
Valid mountain

Only one peak. ⇒ mountain



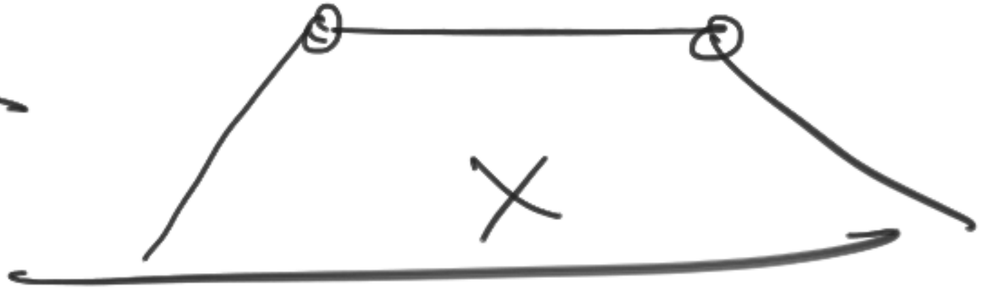


X

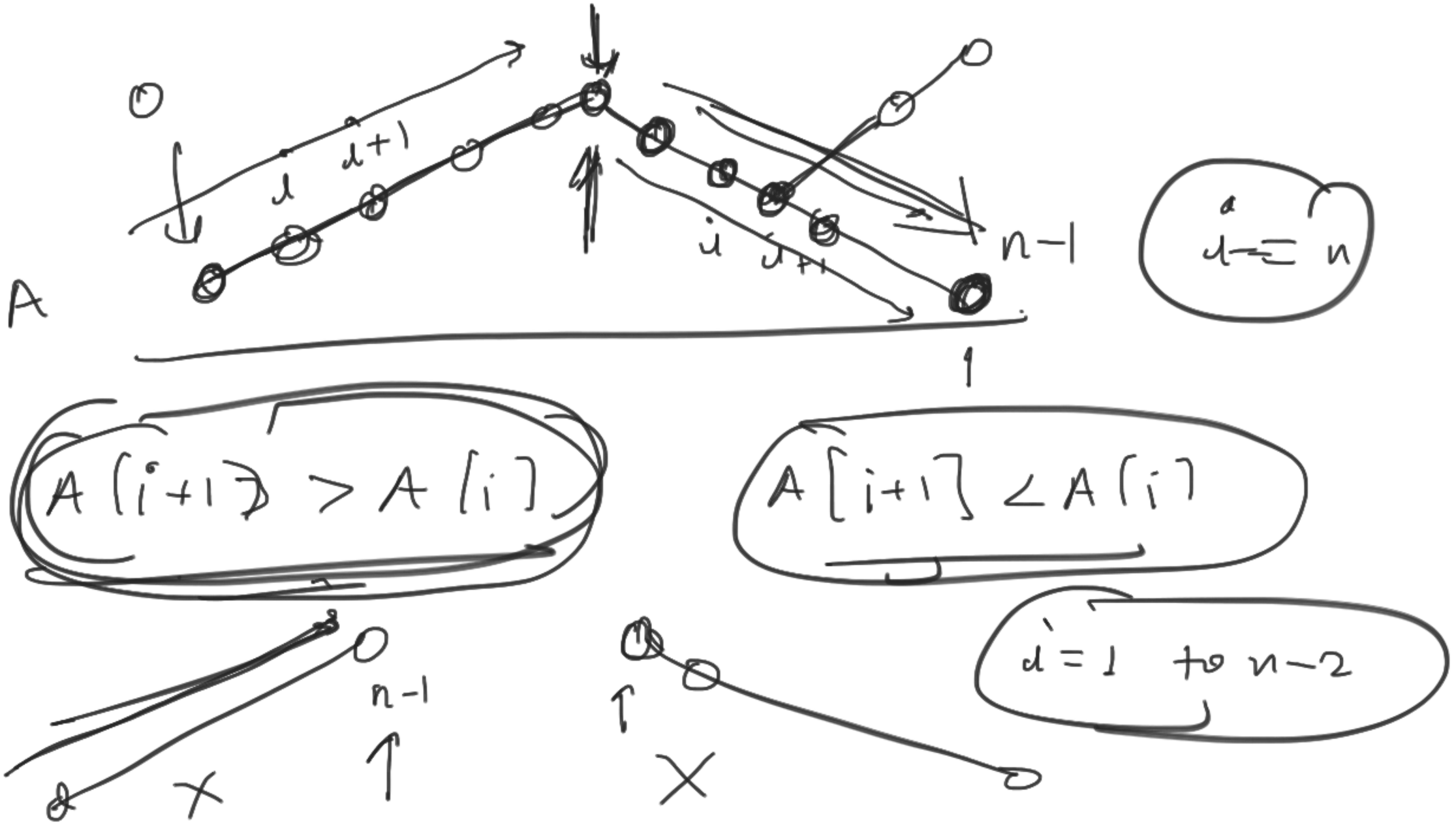


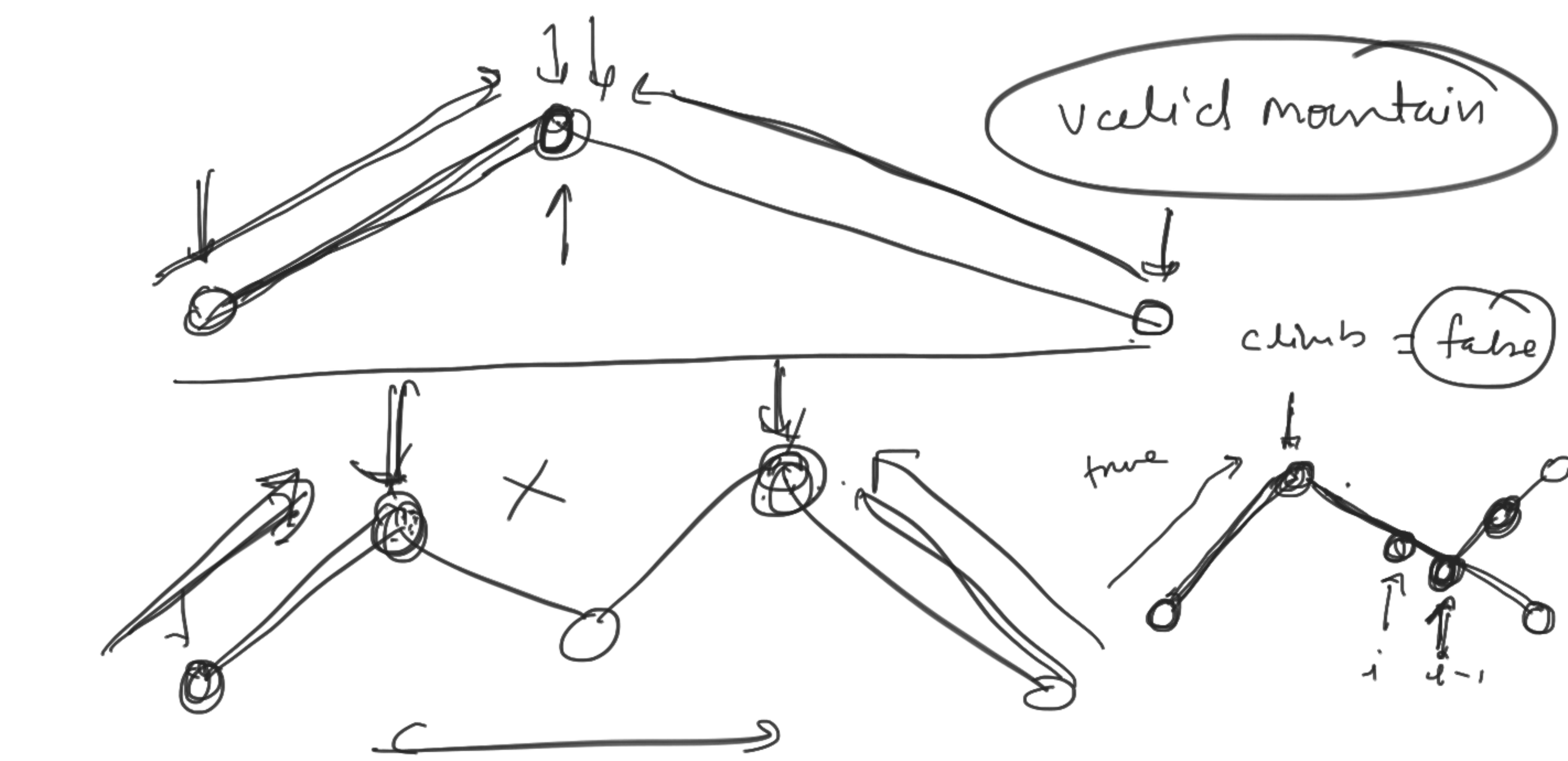
X

X



X


A
i
i+1
n-1
a
i = n
1
A[i+1] > A[i]
A[i+1] < A[i]
i = 1 to n-2
n-1
X
X


valid mountain
climb = false
true
i
i-1