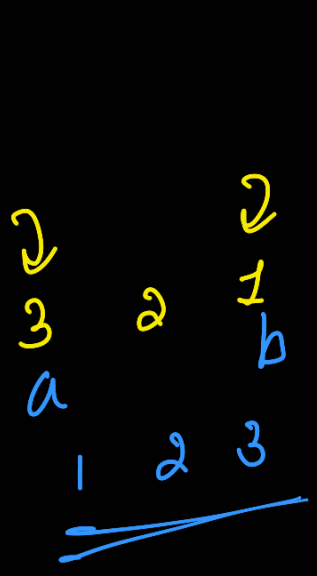
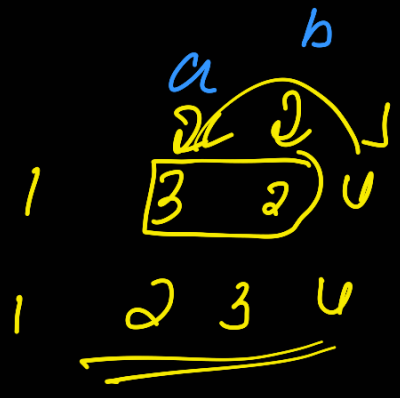




A A A B B B n = 2

A → 3

B → 3

A

A

A B C D A B

A → 1

B → 1

C → 0

D → 1

A → 3

B → 3

A B # # A B # # A B

A B #

n = 2

n = 2

1
A B

A

4

A B C A B d

3

pq : 4 , A
5 , B

A → 3

B → 3

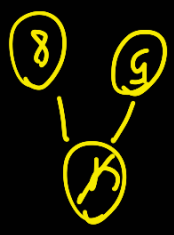
n = 3

5 , B

A B # # A B

1.

2



2 8 5 n

Quickselect : $n \log k$

$O(n)$

$O(1)$

$T(n) = T(n/2) + O(n)$

```
QS(0, n-1, k)
{
    p = partition(0, n-1)
    if (p == k-1)
        return
    else if (p < k)
        Qs(p+1, n, k)
    else
        Qs(1, p-1, k)
```

}

Partition (l, r)

{

Pivot = nums [r]

lastIdx = r

r = r - 1

while (l ≤ r)

{

if (a_l ≤ pivot)

l++

else

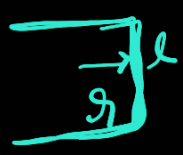
swap (a_l , a_{r-})

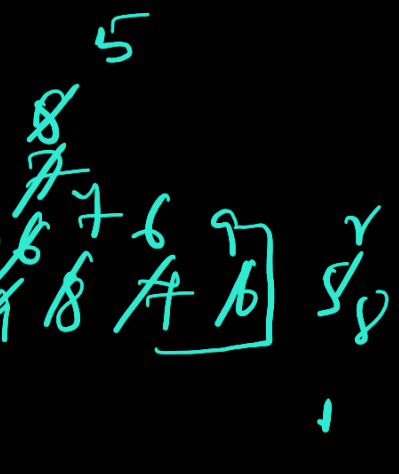
}

if (l < nums.size ?)

swap (a_l, a[last])

$l \leq r$





$l \leq r$

O(n)

$n^2 \times n = O(n^3)$

$n^2 \log(n^2)$

$n^2 \log n$

min heap = Push all 1st col elements into min heap.

Pop each one from heap.

Add to ans. if popped element is from row k push nums[i][j+1] if $j+1 < n$

$T(n) = O(n \log n)$

at any time heap contains only k elements

$O(\log k)$

$k^2 \log k$

in an array of size n
median can be obtained
by storing the
current top

1 2

1 2 3 4 5

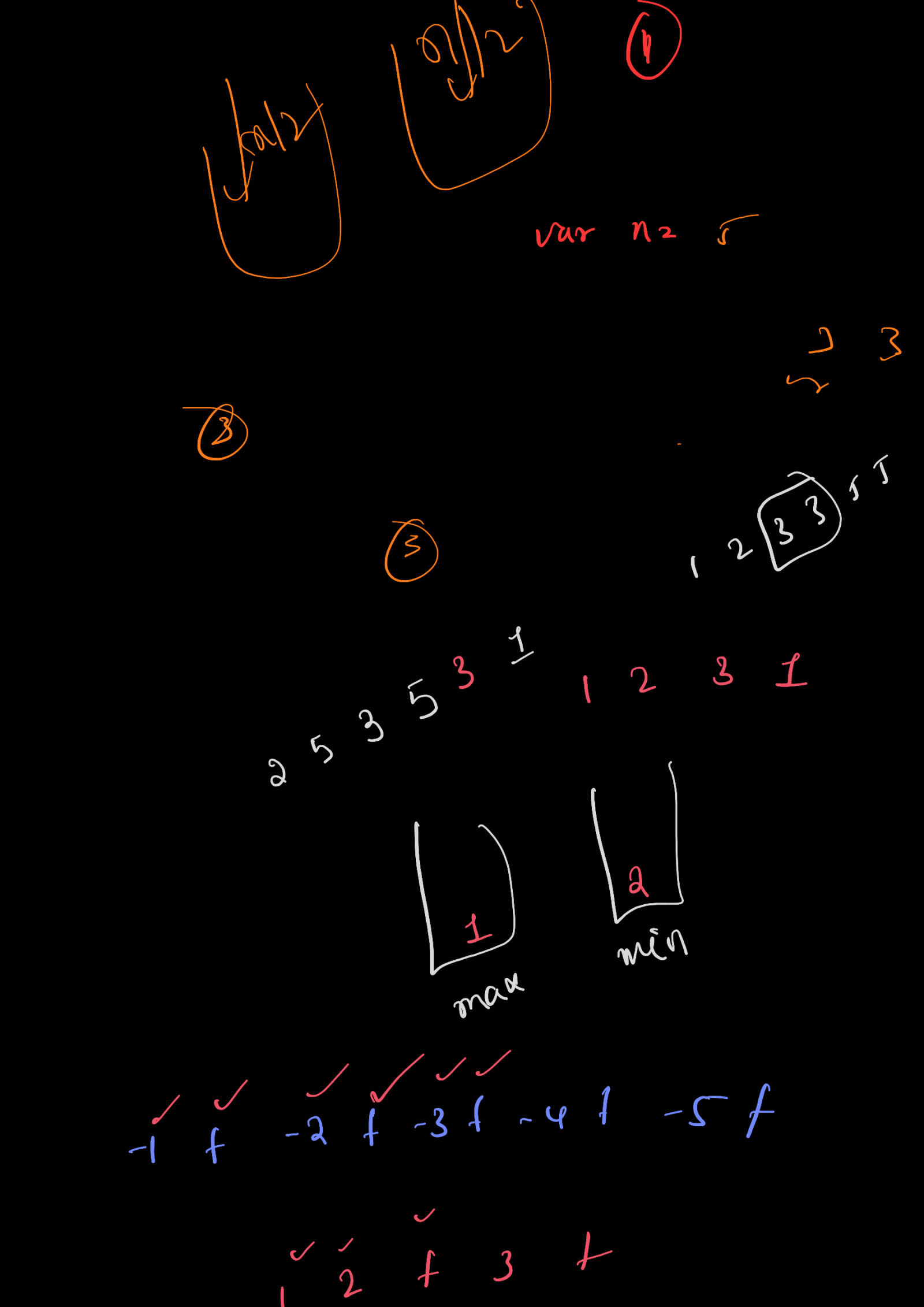
1 ② 3 4 5

6 9 8 7 5 3 10

7 8
5 6 9

3 5 (6 7) 8 9

1	4 to .
max heap	min heap
sm n/2	


var n2 s
1 2 3 3
2 5 3 5 3 1
1 2 3 1
1
max
2
min
-1 f -2 f -3 f -4 f -5 f
2 f 3 f