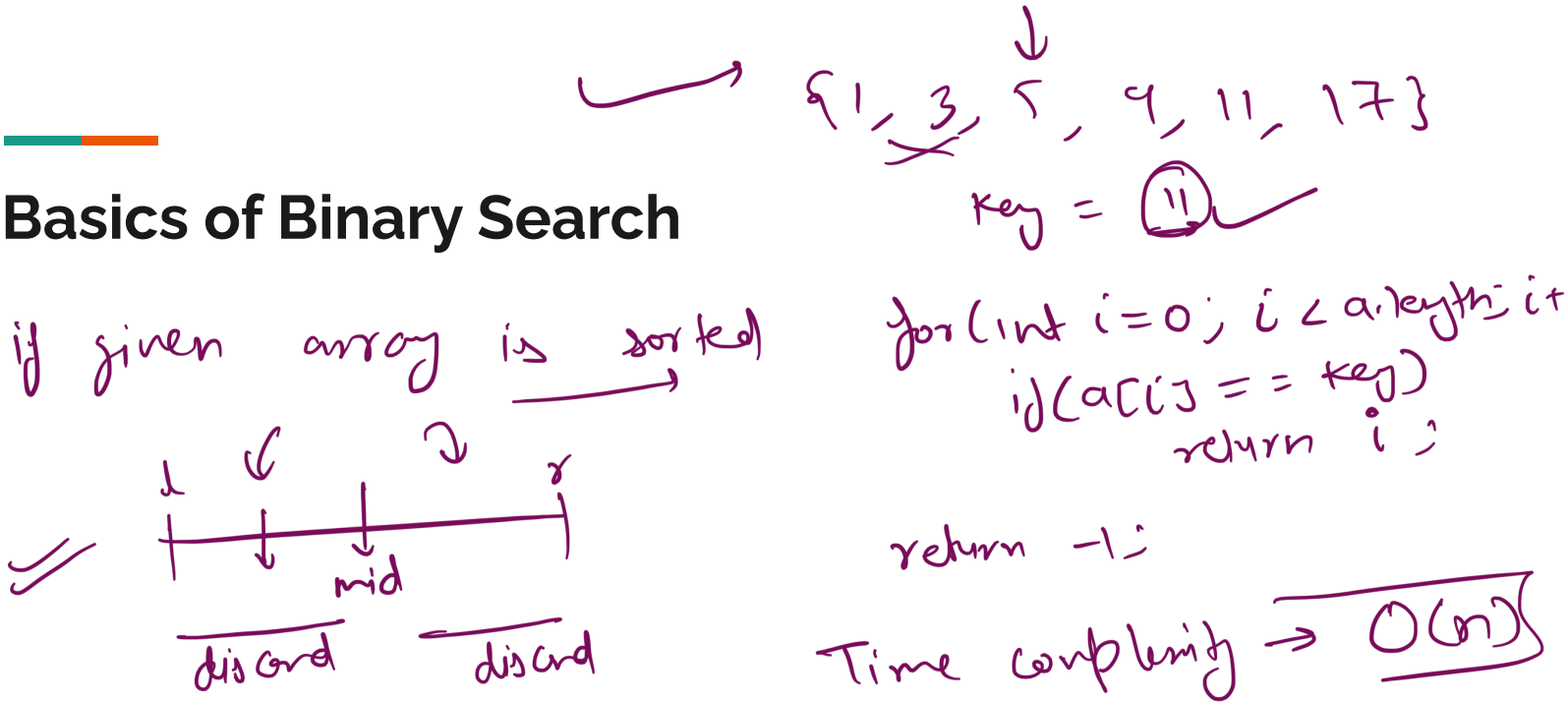
Binary Search Algorithm

Basics

Basics of Binary Search



l → ↓ ← r

{ 1, 3, 5, 9, 11, 17 }

0 1 2 3 4 5

l=0 , r=5

```
int l = 0
int r = a.length-1 ;   key = 8
```

while (l<r)

```
while (l <= r) {
    int mid = (l+r)/2 ;
    if (a[mid] == key) return mid;
    if (key > a[mid]) l = mid+1;
    else r = mid - 1;
}
return -1;
```

l = 3
r = 2

mid = 4 3

$\frac{n}{8}$

$\frac{n}{2^k} = 1 \Rightarrow \boxed{k = \log_2 n}$

k steps

$\boxed{O(\log n)}$

Array size $\rightarrow$ 2^{64} $\longrightarrow$

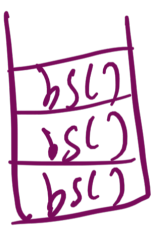
Binary $\rightarrow$ $\log(2^{64}) = 64$.

Recursive

Memory → O(log n)

Recursive stack

TC → O(log n)



Overhead → Slower

Iterative

Constant memory
O(1)

O(log n)

Recommended method.

✗ int mid = (l+r)/2; int overflow.

✓ int mid = l + (r-l)/2;

int → 2×10^9. 1.8×10^9

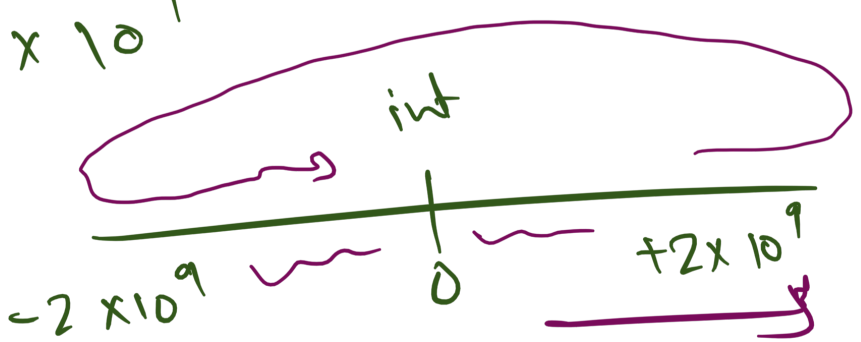
$l = 0.9 \times 10^9$

$r = 1.8 \times 10^9$

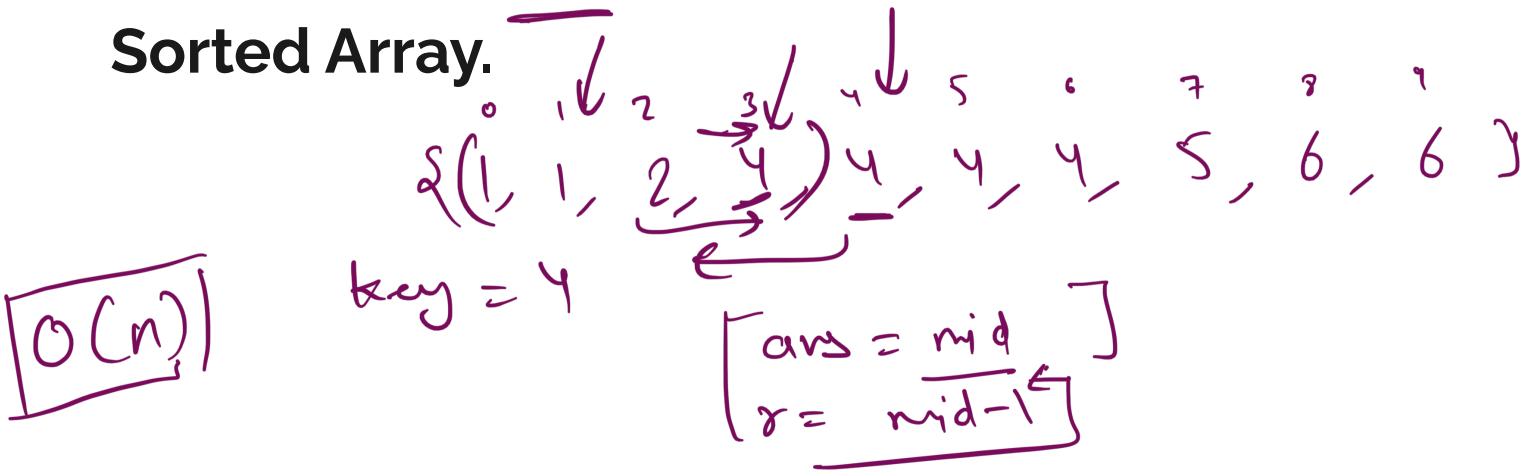
① $(l+r)/2 = \boxed{2.7} \times 10^9$

② $0.9 \times 10^9 + 0.45 \times 10^9$

$= 1.35 \times 10^9$



Index of the First occurrence of an element in a Sorted Array.

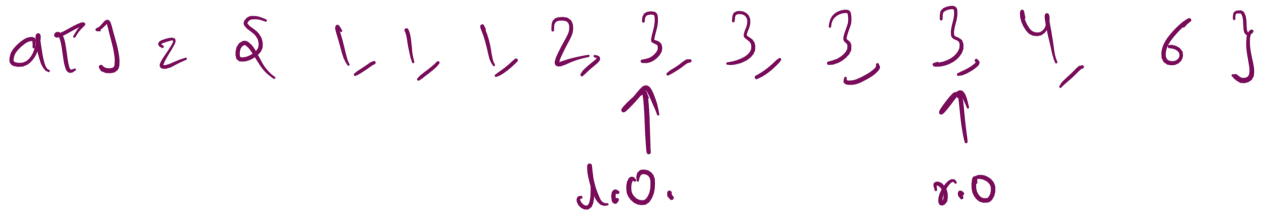


Homework

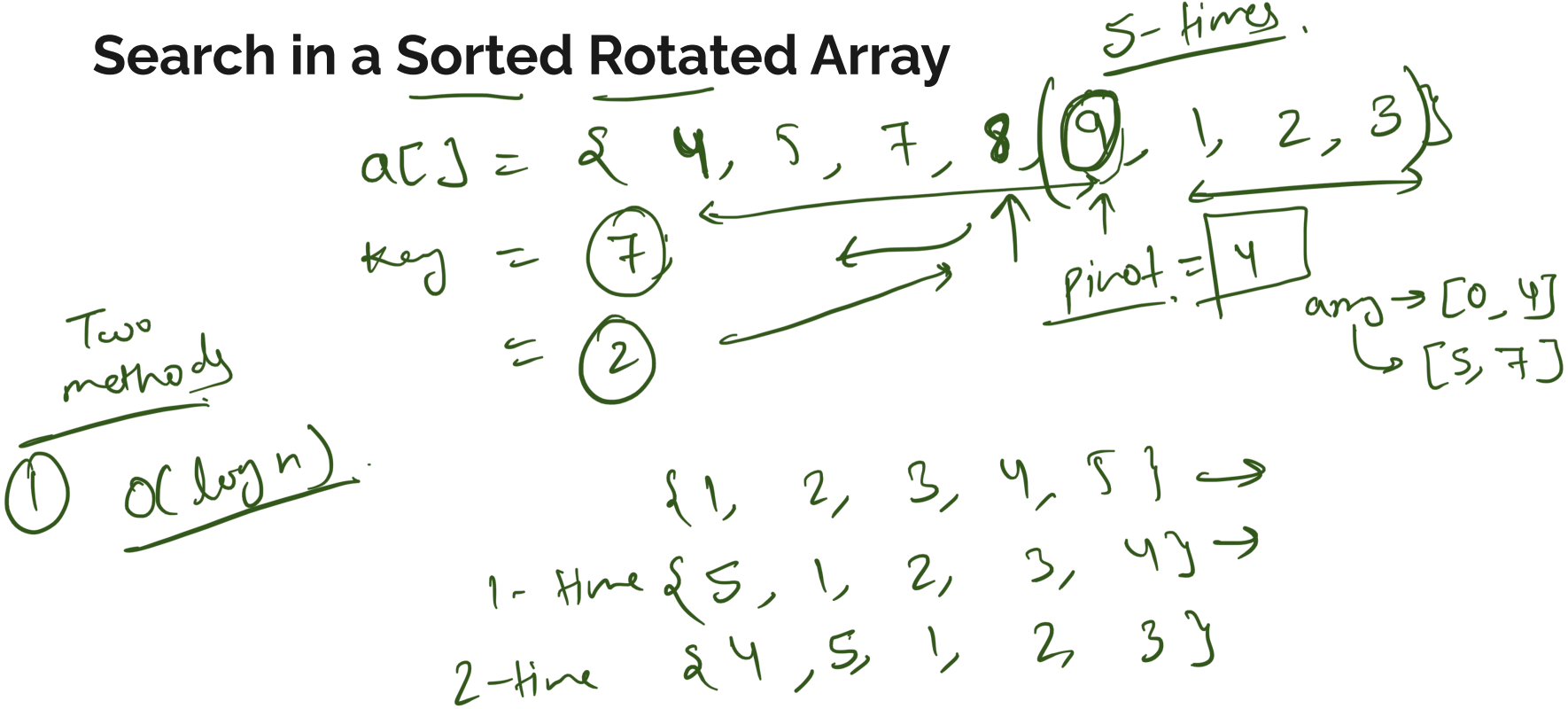
Index of the Last occurrence of an element in a Sorted Array.

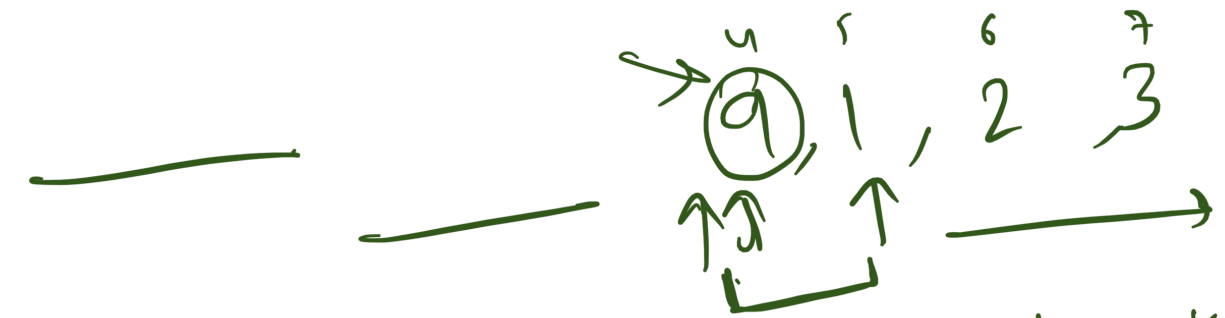
Homework

Count the occurrences of Element in a Sorted Array

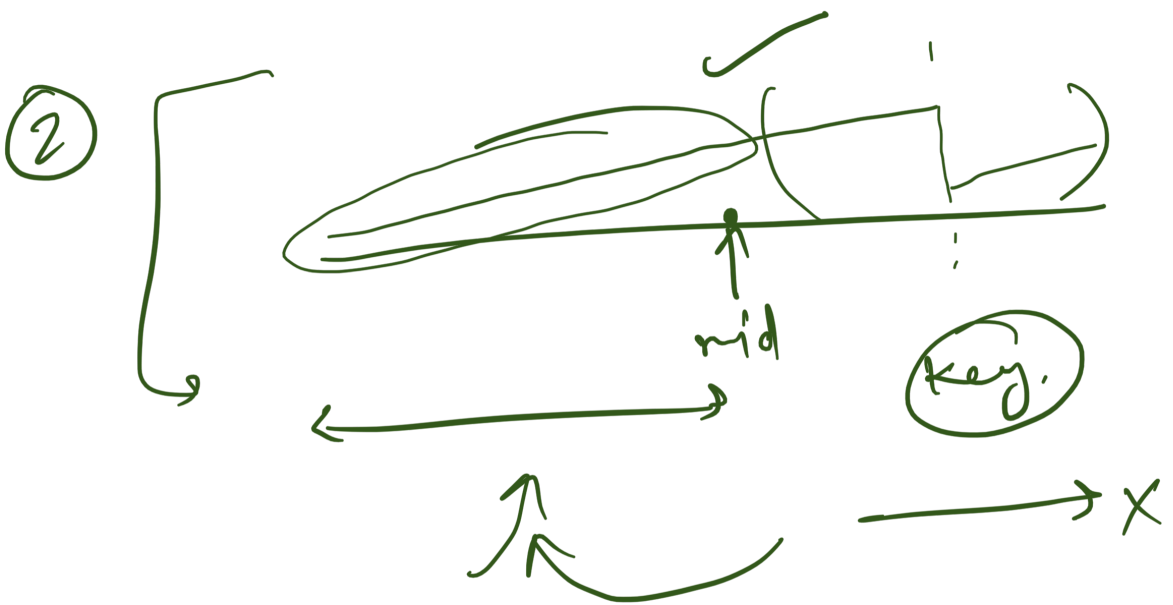


Search in a Sorted Rotated Array

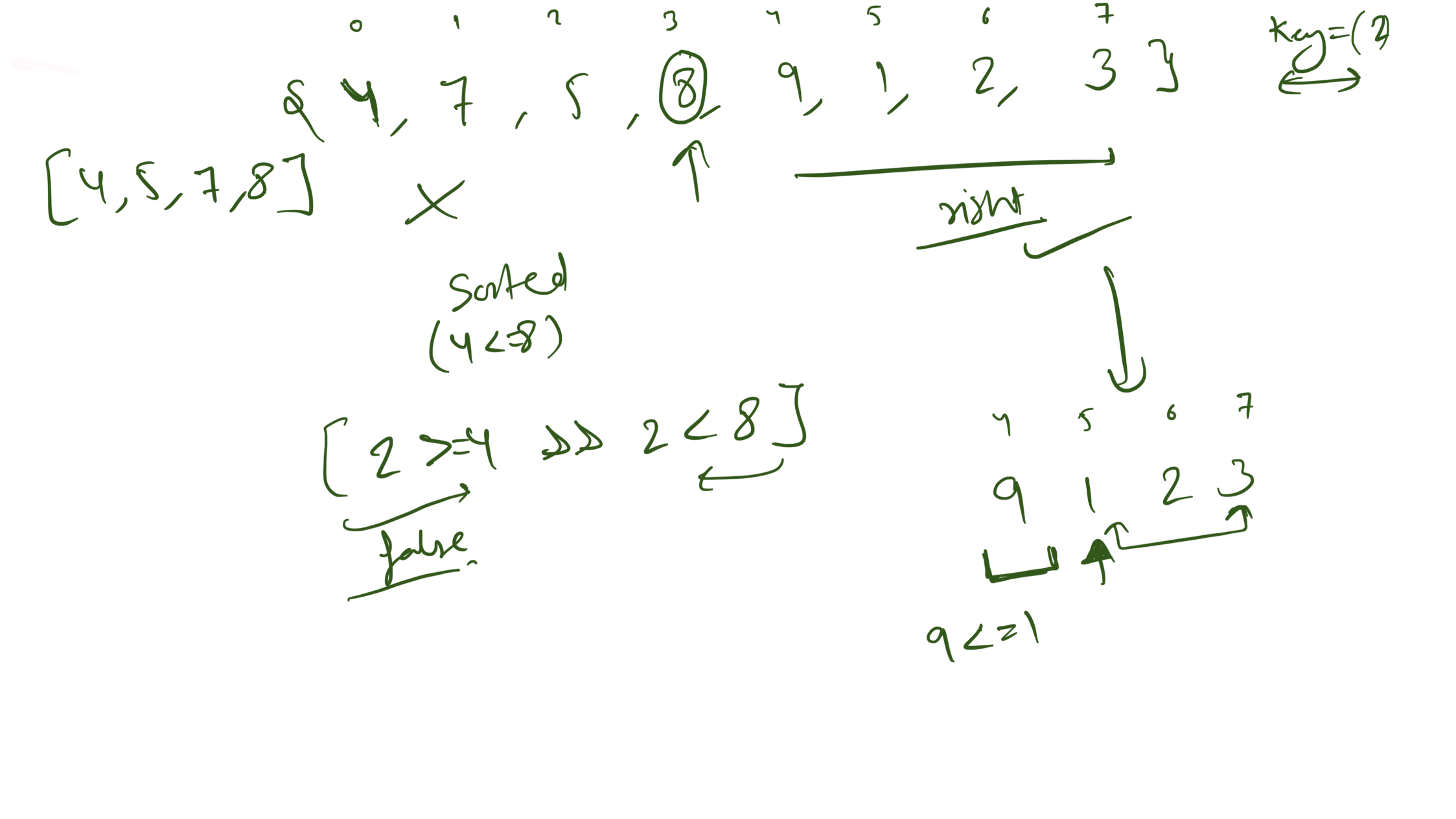


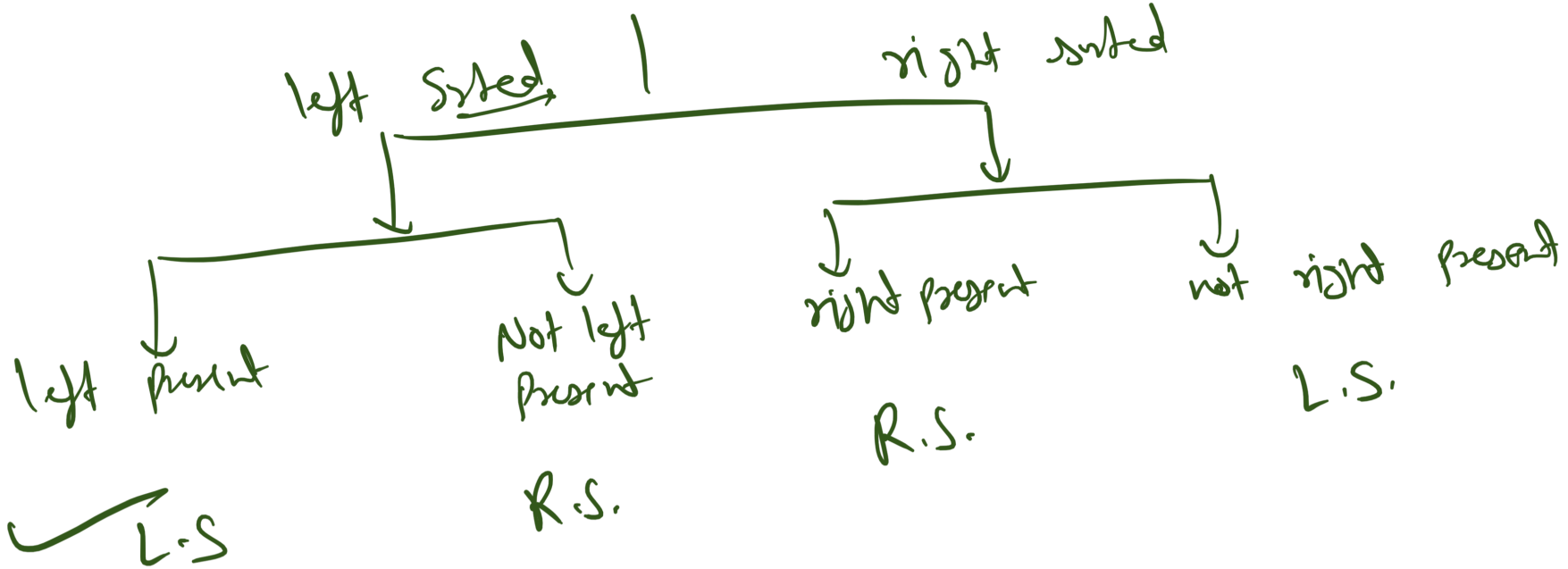


if(mid = = key) return mid;



left sorted -P	Left
left sorted -A	Right
right sorted -P	Right
right sorted -A	Left


0 1 2 3 4 5 6 7
{ 4, 7, 5, 8, 9, 1, 2, 3 }
Key=(
[4,5,7,8]
X
Sorted
(4<=8)
right
[2 >=4 >> 2 <8]
false.
4 5 6 7
9 1 2 3
9<=1


left sided
right sided
left present
Not left present
right present
not right present
L.S
R.S.
R.S.
L.S.

Practice Problems

1. Find the first position of 1 in a sorted array containing only 0s and 1s.
2. Find the upper bound of an element in a sorted Array (upper bound of an element is the next greater element).
3. Given a sorted array with no duplicates A and a target value B, return the index if the target is found. If not, return the index where it would be if it were inserted in order.