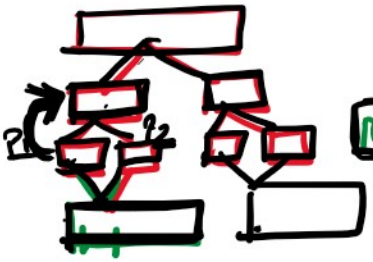
23-09 Merge Sort - 2

23 September 2022 20:11

TC
SC



Merge2Sorted

Recurrance rel^n → Substitution

Reursive Soln

Call Stack

Use Cases — Divide-conquer } Class of Problem }

Median } ⇒

arr1 :- 1 3 4 7 10 12 // Sorted

arr2 :- 2 3 6 15 // Sorted

1 2 3 3 4 6 7 10 12 15 ⇒ 10 elements

even?

Yes

$\frac{n}{2}$ ⇒ $\frac{10}{2}$ ⇒ 5 + 6

n = 10

$\frac{n}{2}$ ⇒ 5 + 6

$\frac{4+6}{2}$ ⇒ $\frac{10}{2}$ ⇒ 5

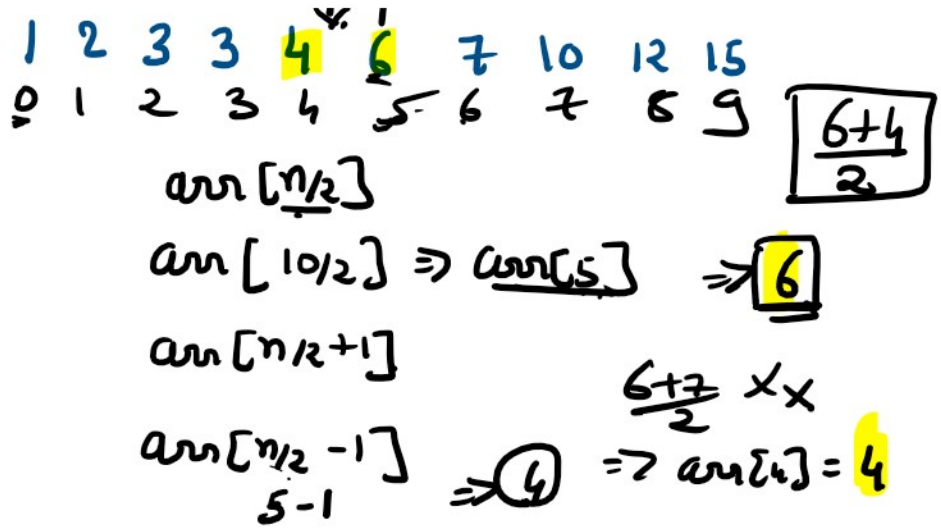
odd 1 2 3 4 5 → 3

AP:1

1. Merged two Sorted arrays
2. Find out no. of elements.

n = no. of elem = 10

1 2 3 3 4 6 7 10 12 15




```
if(n % 2 == 0){ //even
    med = (arr[n/2] + arr[(n/2)-1])/2;
}
else{
    med = arr[parseInt(n/2)]
}
```

new space

```
let arr = MergeTwoSortedArrays([1,3,4,7,10,12],[2,3,6,15]);
console.log(arr);

let med;
let n = arr.length;
if(n % 2 == 0){ //even
    med = (arr[n/2] + arr[(n/2)-1])/2;
}
else{
    med = arr[parseInt(n/2)]
}
console.log(`Median is ${med}`);
```

m+n

m n

m+n

m<n xx

except variables

DS => array

Space complexity :- O(m+n) => O(n)

```
function MergeTwoSortedArrays(a1, a2){
    let res = [];
    let p1 = 0;
    let p2 = 0;
    let k = 0;

    while(p1 < a1.length && p2 < a2.length){
        if(a1[p1] < a2[p2]){
            res[k] = a1[p1];
            p1++;
            k++;
        }
        else{
            res[k] = a2[p2];
            p2++;
            k++;
        }
    }

    if(p1 === a1.length){
        while(p2 != a2.length){
            res[k] = a2[p2];
            k++;
            p2++;
        }
    }
    if(p2 === a2.length){
        while(p1 != a1.length){
            res[k] = a1[p1];
            k++;
            p1++;
        }
    }
    }

    return res;
}

let arr = MergeTwoSortedArrays([1,3,4,7,10,12],[2,3,6,15]);
console.log(arr);

let med;
let n = arr.length;
if(n % 2 == 0){ //even
    med = (arr[n/2] + arr[(n/2)-1])/2;
}
else{
    med = arr[parseInt(n/2)]
}
console.log(`Median is ${med}`);
```

TC :- O( )?

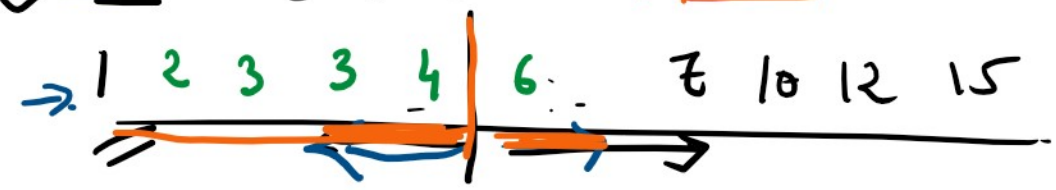
SC :- O(m+n)

O(1)!!

APP => 2

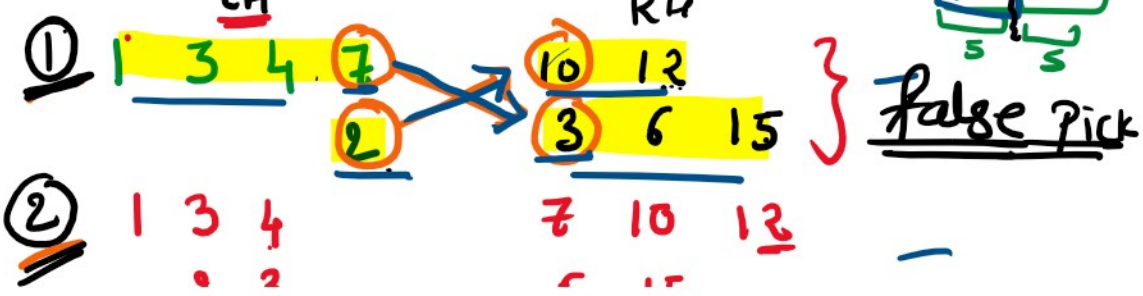
arr1 :- 1 3 4 7 10 12 // Sorted

arr2 :- 2 3 6 15 // Sorted

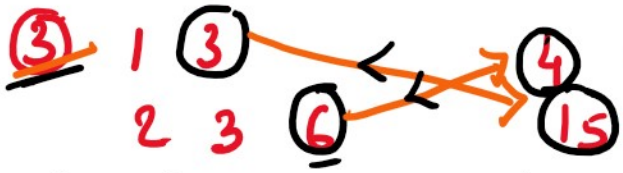


arr1 :- 1 3 4 7 10 12 // Sorted

arr2 :- 2 3 6 15 // Sorted



(2) 1 3 4 7 10 12
2 3 6 15

(3) 

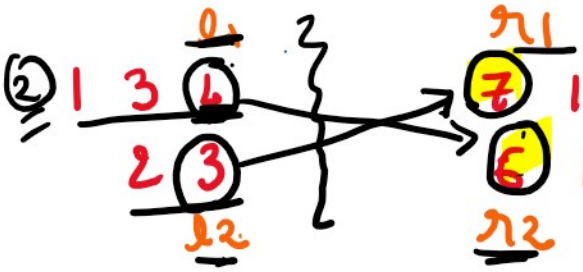
7 10 12 } false
Pick

→ If my picked half is a valid one, then 7 has to be lesser than 3

$7 \not< 3 \Rightarrow$ false.

→ In my 3rd obv,

3 < 15 ⇒ true. ☺
6 < 4 ⇒ false. ☹



} true

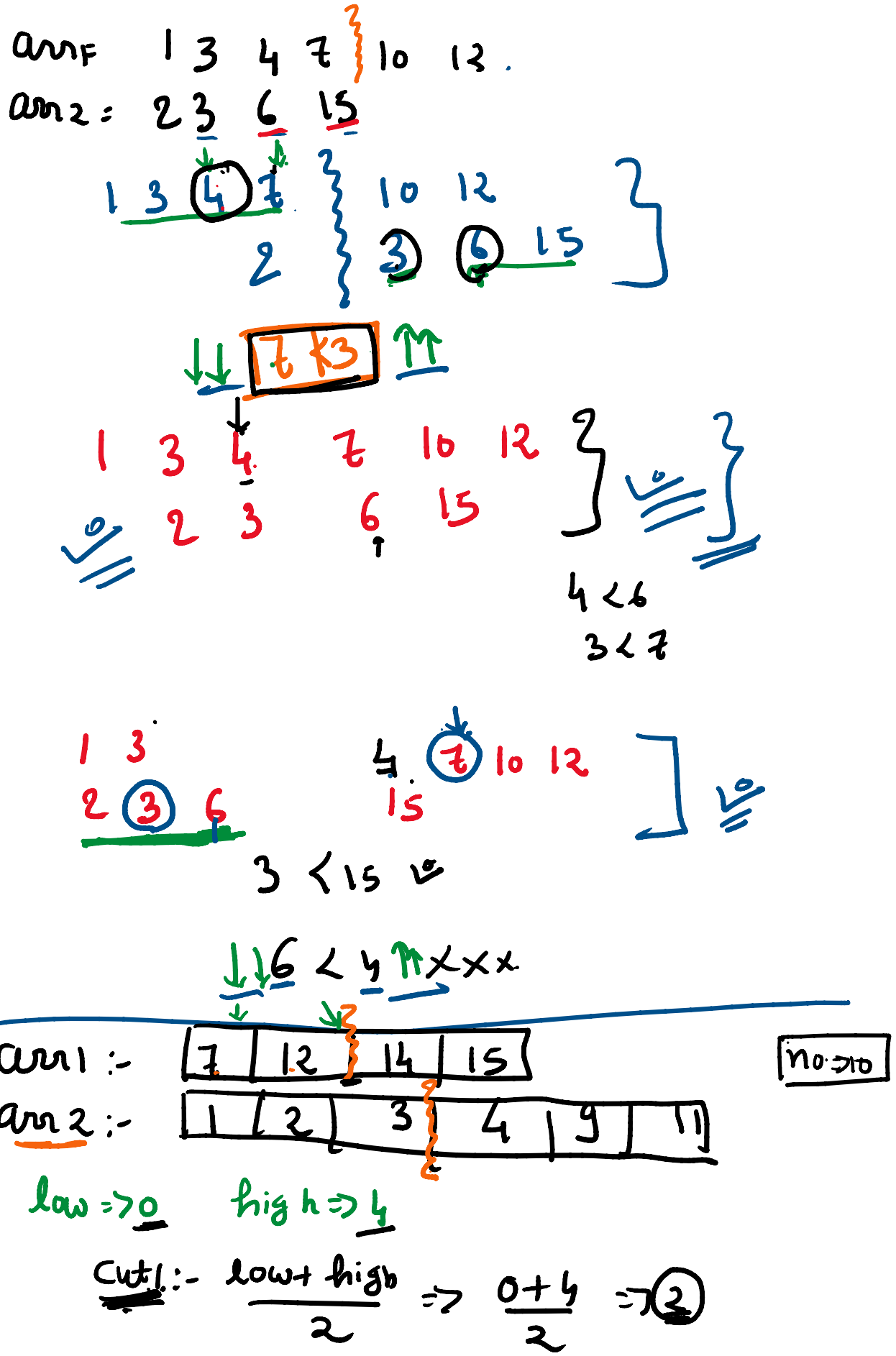
4 < 6 ⇒ true.
&& 3 < 7 ⇒ true. } Perfect Pick

---

l_1, l_2, r_1, r_2

$$\frac{(\max(l_1, l_2) + \min(r_1, r_2))}{2}$$

$l_1 <= r_2$
$l_2 <= r_1$


arr1 1 3 4 7 10 12.
arr2: 2 3 6 15
1 3 4 7 10 12
2 3 6 15
7 k3
1 3 4 7 10 12
2 3 6 15
4 < 6
3 < 7
1 3
2 3 6
4 7 10 12
15
3 < 15
6 < 4 xxx
arr1 :- 7 12 14 15
arr2 :- 1 2 3 4 9 11
no => 10
low => 0
high => 4
Cut! :- low + high / 2 => 0+4 / 2 => 2

$l_1 = 12$ → $r_1 = 14$
$l_2 = 3$ → $r_2 = 4$

$12 <= 4$ false

Not a valid partition

↓ $12 <= 4$ ↑

high = mid - 1

low => 0 high => 2 - 1 = 1

cut 1 => $\frac{low + high}{2}$ => $\frac{0+1}{2}$ => 0

arr1 :- 7 | 12 | 14 | 15

arr2 :- 1 | 2 | 3 | 4 | 9 | 11

n = 10

cut 2 = 5 - 0 => 5

= $\frac{n}{2}$ - cut1

X

1 2 3 4 9 11

7 12 14 15

$l_1 = -1$ $r_1 = 2$

$l_2 = 9$ $r_2 = 11$

$-1 <= 11$ ✓

$9 <= 2$ XX

↓↓ 9 < = 7 ↑↑

low = mid + 1 => 0+1 = (1)

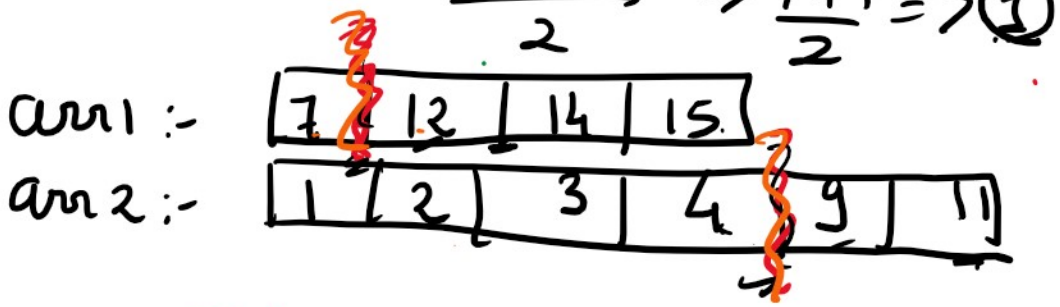
arr1 :- | 7 | 12 | 14 | 15 |

arr2 :- | 1 | 2 | 3 | 4 | 9 | 11 |

low => mid+1 => 0+1 = 1

high => 1

mid => $\frac{low + high}{2}$ => $\frac{1+1}{2}$ => (1)



cut2 = 5 - 1 => 4 l1 = cut1 - 1

l1 = 7 { r1 = 12 14 15

1 2 3 l2 = 4 { 9 r2 11

7 <= 9 ✓

4 <= 12 ✓ } true

(max(l1, l2) + min(r1, r2)) {

$$\frac{(\max(l_1, l_2) + \min(r_1, r_2))}{2}$$

$\frac{7 + 9}{2} \Rightarrow \frac{16}{2} \Rightarrow 8$

$l_1 = arr1[cut1 - 1]$

$l_2 = arr2[cut2 - 1]$

$r_1 = arr1[cut1]$

$r_2 = arr2[cut2]$

$cut1 = (low + high)/2$

$cut2 = \frac{(n1 + n2)}{2} - cut1$

4

1 2 3 4 7 9 | 11 12 14 15 16

$n1 + n2$

$ln \Rightarrow 11$

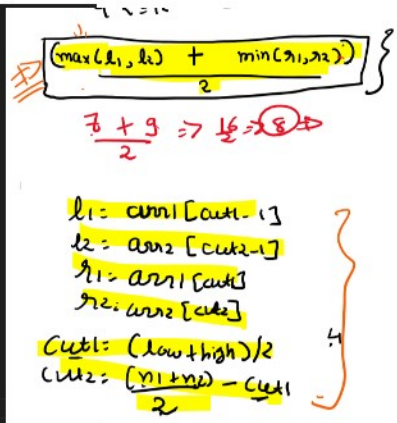
$n1 \Rightarrow 6$

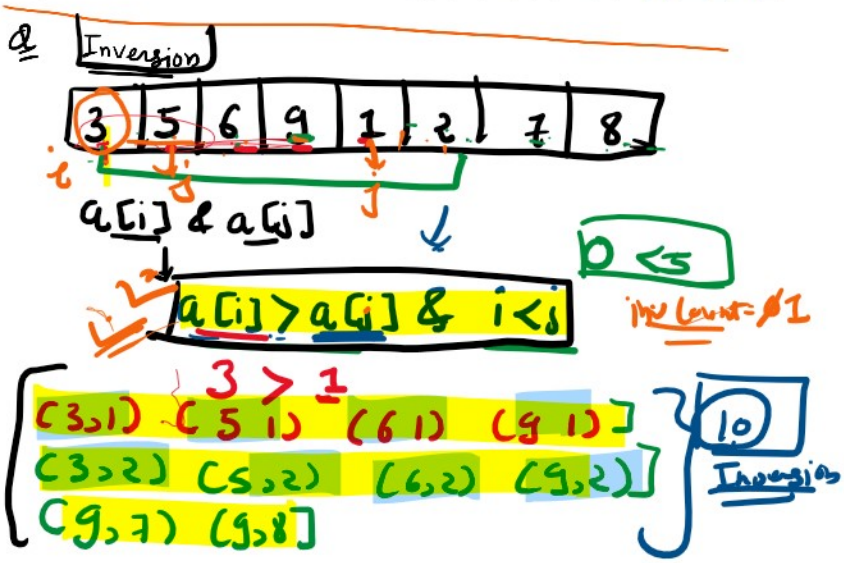
$n2 \Rightarrow 5$

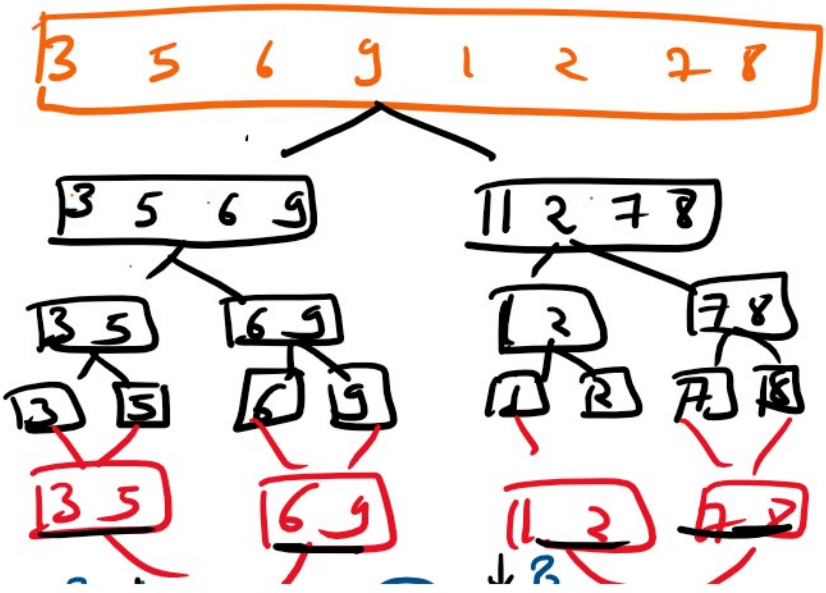
$\Rightarrow \left(\frac{n1 + n2 + 1}{2}\right) \Rightarrow \frac{12}{2} \Rightarrow 6$

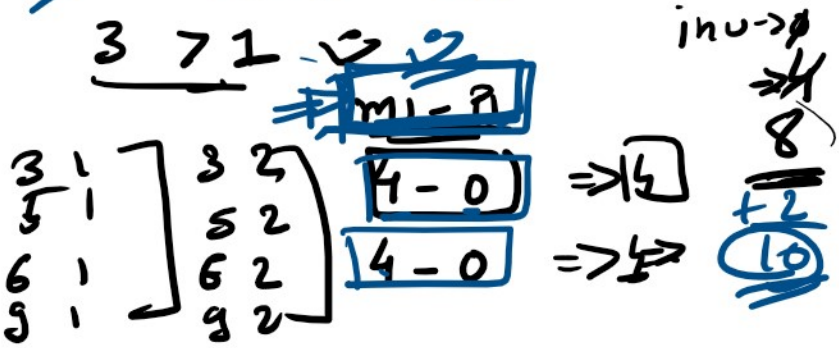
$\max(l_1, l_2)$

```javascript
function findMedianSortedArrays(nums1, nums2) {
    if(nums2.length < nums1.length){
        return findMedianSortedArrays(nums2,nums1)
    }

    let n1 = nums1.length;
    let n2 = nums2.length;
    let low = 0, high = n1;

    while(low<=high){
        let cut1 = (low+high) >> 1;
        let cut2 = parseInt((n1+n2+1)/2)-cut1;

        let left1 = cut1 == 0 ? Number.MIN_VALUE : nums1[cut1-1];
        let left2 = cut2 == 0 ? Number.MIN_VALUE : nums2[cut2-1];

        let right1 = cut1 == n1 ? Number.MAX_VALUE : nums1[cut1];
        let right2 = cut2 == n2 ? Number.MAX_VALUE : nums2[cut2];

        if(left1 <= right2 && left2 <= right1){
            if((n1+n2) % 2 == 0){
                return parseInt((Math.max(left1,left2) + Math.min(right1,right2))/2);
            }
            else{
                return Math.max(left1,left2);
            }
        }
        else if(left1 > right2){
            high = cut1 - 1;
        }
        else{
            low = cut1 + 1;
        }
```







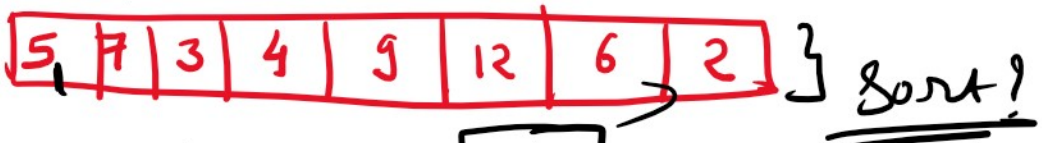


a[i] > a[j] & i < j ⇒ Single ino

a[i] > 2*a[j] & i < j

↓

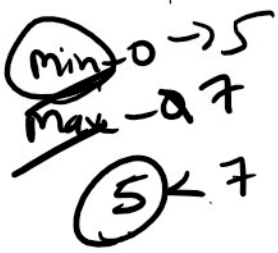
double ino

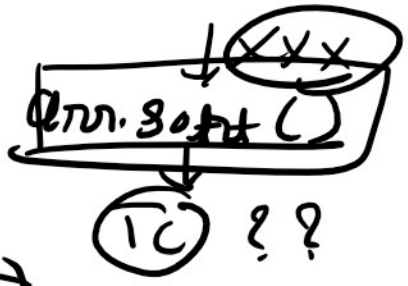


Sort?

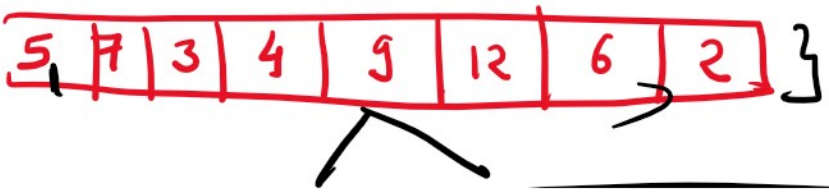
min

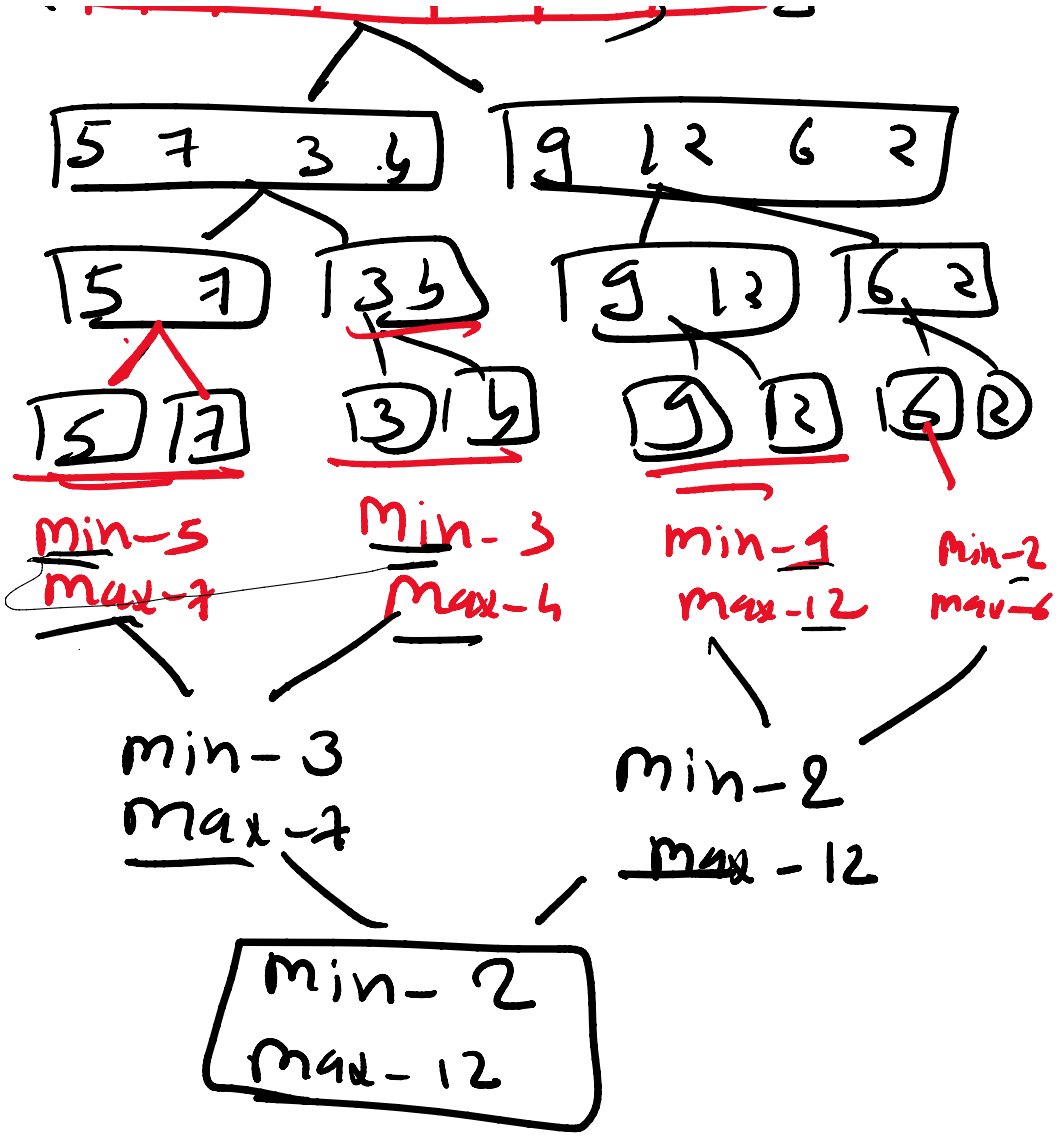
max





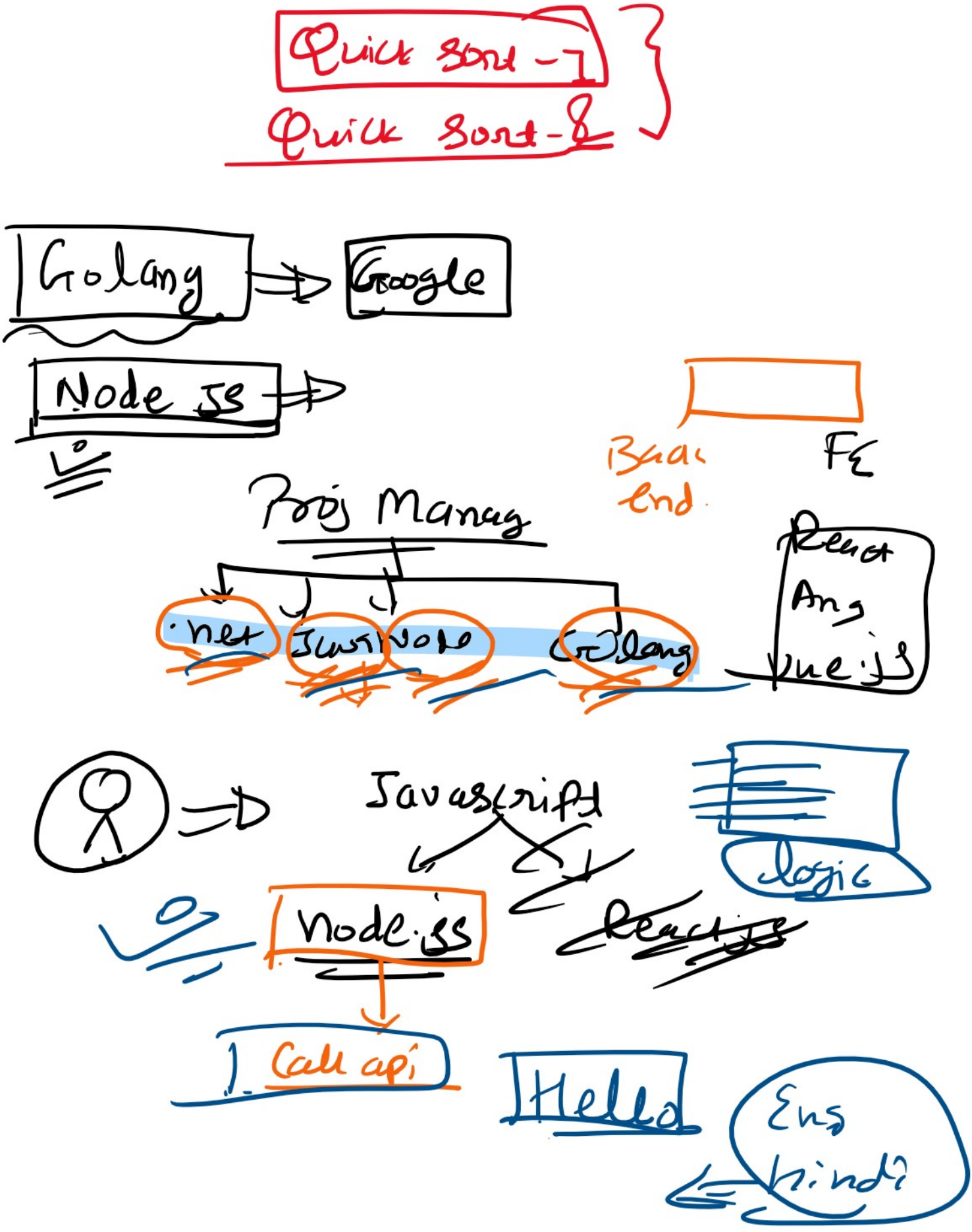
Brute force





1. Median → Hard → leetcode
2. Inversion
3. Double inversion
4. Min-Max.

9 AM


Quick Sort -1
Quick Sort -2
Golang
Google
Node JS
Back end
FE
Proj Manag
.net
Node
Golang
React
Ang
Vue.js
JavaScript
logic
Node.js
React.js
1. Call api
Hello
Eng
hindi