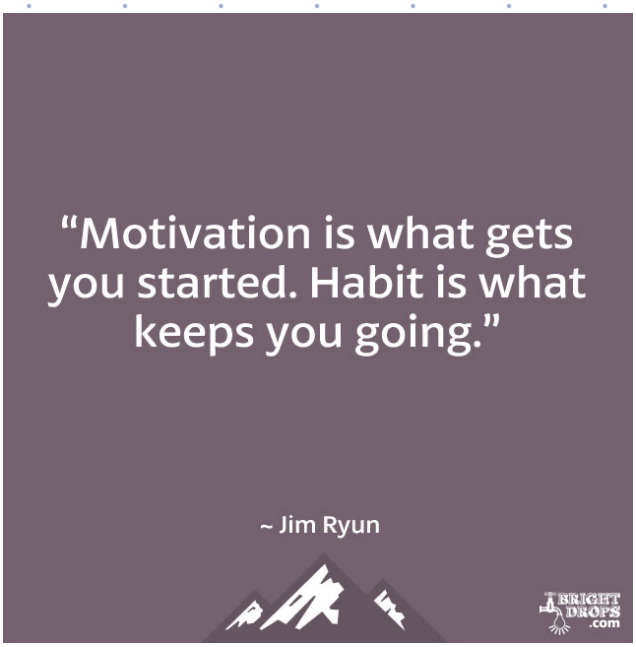
Binary Search Lab



Good Morning

Today's content

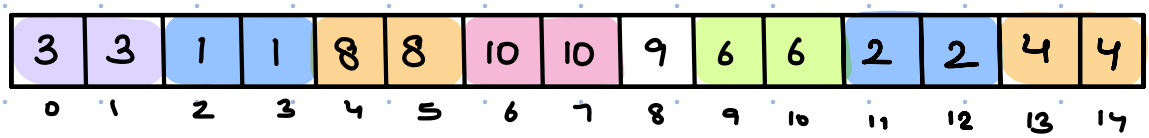
01. Unique element
02. Search in sorted rotated array (Interview ques)
03. Median of an array (Hard)

Unique Elements

Every element in an array occurs twice except for one. Find the unique element.

Note :- Duplicates are adjacent to each other.

arr [ ] =



Ans = 9

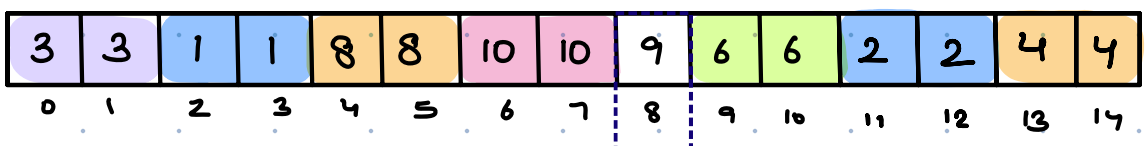
01. Take XOR of all elements

TC: O(N)
SC : O(1)

02. Linear search

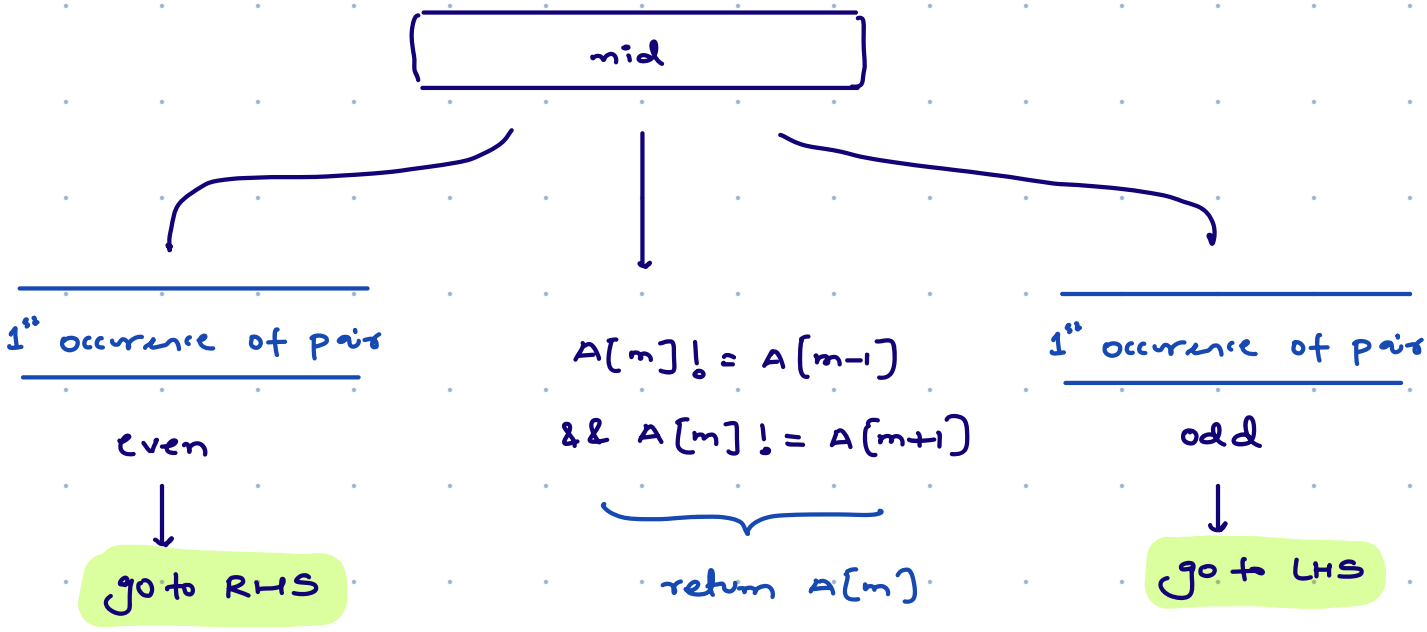
TC: O(N)
SC : O(1)

arr [ ] =



1st occurrence of every pair will be at even index

1st occurence of every pair will be at odd idx

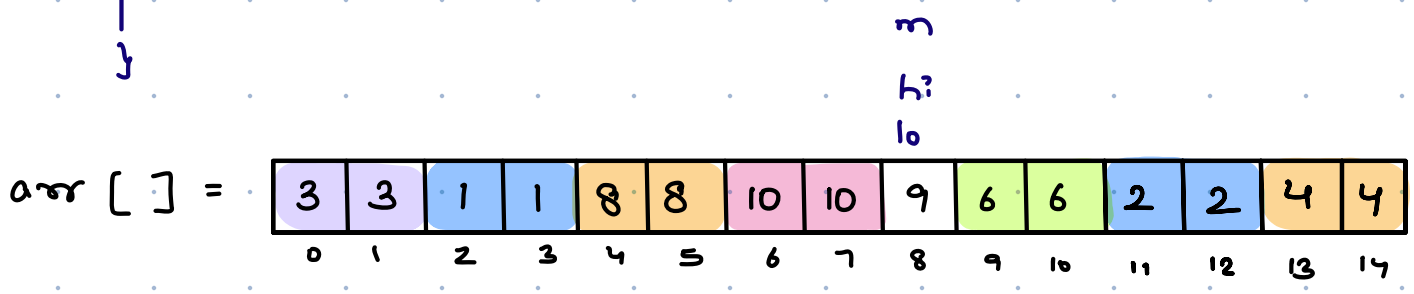


7:26 → 7:30

```
if (N==1) return A[0]
if (A[0] != A[1]) return A[0]
if (A[N-1] != A[N-2]) return A[N-1];
lo = 1
hi = N-2
while (lo <= hi)
    int m = (lo+hi)/2
    if (A[m] != A[m-1] && A[m] != A[m+1]) return A[m]
    if (A[m] == A[m-1])
        m = m-1          // move mid to 1st occurrence of dup
    if (m%2==0) lo = m+2
    else hi = m-1
```

else $hi = m-1$

}



lo	hi	mid	first occ	
1	13	7	6	→ $lo = m+2$
8	13	10	9	→ $hi = m-1$
8	8	8		→ return A[8]

Q Sorted array A[] = {3 9 14 16 20}

Given K, can this K be present inside array?

K = 17 → Yes, it is a possibility

K = 23 → No

Claim → If I have a sorted array, then I can figure out if a particular element is present or not.

* Rotated sorted Array

A[] = {1 2 3 4 5 6}

K = 1 = {6 1 2 3 4 5}

K = 2 = {5 6 1 2 3 4}

K = 3 = {4 5 6 1 2 3}

K = 4 = {3 4 5 6 1 2}

Claim → If I have a sorted rotated array, based on a particular idx, one half is always going to be sorted.

Search in sorted rotated Array

* Given an input ar[], formed by rotating a distinct sorted array right to left.

Search ele & return index if it is present else return -1.

ar[] =

10	11	12	13	17	20	23	25	26	1	3	5	6	8
0	1	2	3	4	5	6	7	8	9	10	11	12	13

K = 17

01. Linear Search

TC: O(N)
SC: O(1)

02. **Binary Search**

→ Find max ele idx of array

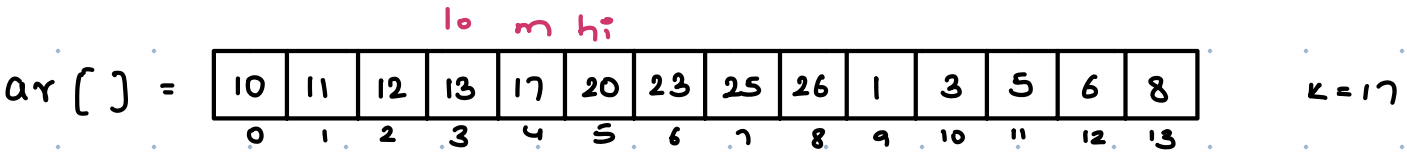
→ Split arry into two subarrays

→ Figure out if K will be present in first subarray or second subarray

→ Apply Binary search again in this array where we have possibility of finding K.

TC: O(log N)
Sc: O(1)

* Use One iteration of Binary search



lo	hi	mid	
0	13	6	: left half is sorted : check if k will be present in lo to m ⟶ yes hi = m-1
0	5	2	: left half is sorted : check if k will be present in lo to m ⟶ No lo = m+1
3	5	4	⟶ return idx

m

hi lo

ar[ ] =

10	11	12	13	17	20	23	25	26	1	3	5	6	8
0	1	2	3	4	5	6	7	8	9	10	11	12	13

k = 4

lo	hi	m	
0	13	6	: left half is sorted : check if k will be present in lo to m → No lo = m+1
7	13	10	: right half is sorted check if k will be present in m to hi → Yes lo = m+1
11	13	12	: left half is sorted : check if k will be present in lo to m → No lo = m+1
13	13	13	
14	13	→	Exhaust search space

```
lo = 0          hi = N-1

while (lo ≤ hi) {
    int m = (lo+hi)/2
    if (A[m] == K) return m;
    else if (A[lo] ≤ A[m])        {lo...m is sorted}
        if (A[lo] ≤ K && K < A[m]) hi = m-1
        else lo = m+1
    else {                        {hi...m is sorted}
        if (A[m] < K && K ≤ A[hi]) lo = m+1
        else hi = m-1
    }
}
return -1
```

* Median of Two sorted Arrays

A[] = {1 4 5}

B[] = {2 3}

A[] = {1 2 3 4 5}

Ans = 3

A[] = {4}

B[] = {1 2 3}

A[] = {1 2 3 4}

Ans = $\frac{2+3}{2}$ = 2.5

Q Given two sorted arrays, figure out median of sorted merged array.

Idea 1 → Merge both the sorted arrays in one array & return the median

TC: O(N+M)
SC: O(N+M)

A[] = {1 3 4 7 10 12}

B[] = {2 3 6 15}

size = 10

merge[] = {1 2 3 3 4 6 7 10 12 15}

largest ele of first bucket

smallest ele of second bucket

$ans = \frac{l+s}{2}$

sele sele

A[] = {1 3 4 7 10 12}

B[] = {2 3 6 15}

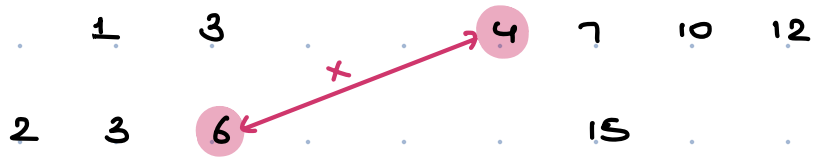
n = size of 1st arr
m = size of 2nd arr

01. Scenario : 4 elements of first array

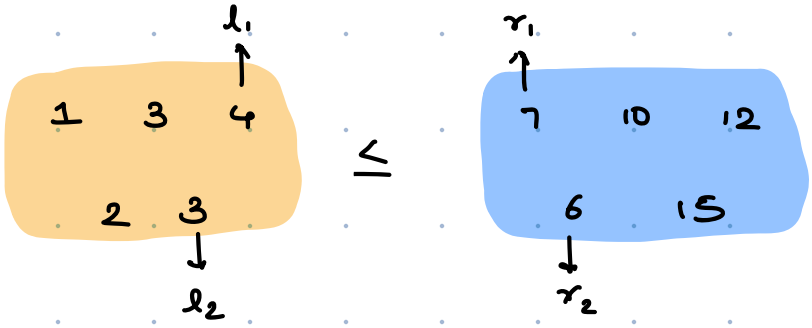
1 3 4 7 10 12

2 3 6 15

02. Scenario : 2 elements from first array



03. Scenario 3 : 3 elements from 1st array

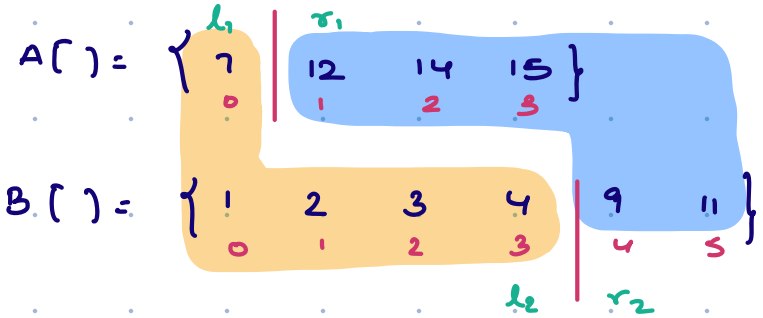


$l_1 \le r_2$

$l_2 \le r_1$

} correct bucketisation

Idea → Apply Binary search on smaller array & figure out how many elements we have to pick from array for left bucket.



Total ele = 10

Ele in one bucket = 5

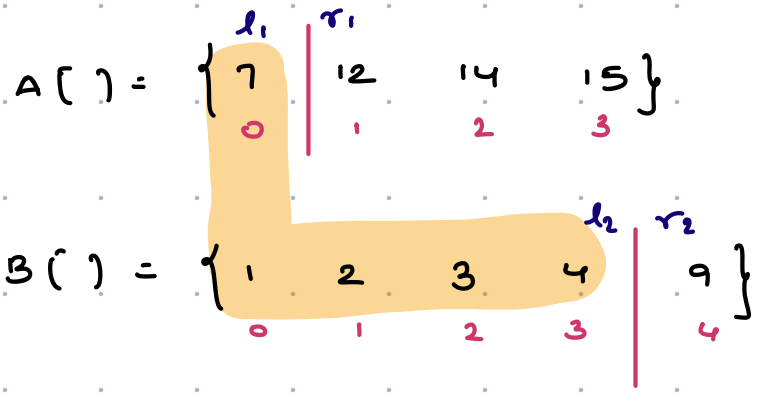
search space = [0 4]

lo	hi	mid/cut 1	
0	4	2	→ 2 ele from A for LHS & 3 ele from B $l_1 \le r_2$ ✗ ∴ go to LHS hi = mid − 1
0	1	0	→ 0 ele from A for LHS & 5 ele from B $l_1 \le r_2$ ✓ $l_2 \le r_1$ ✗ ∴ go to RHS lo = mid + 1
1	1	1	→ 1 ele from A for LHS & 4 ele from B $l_1 \le r_2$ ✓ $l_2 \le r_1$ ✓

valid distribution

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

```
lo = 0

hi = length of smaller array A

while (lo ≤ hi)
{
    int cut1 = (lo + hi) / 2          //no. of ele from A on LHS

    int cut2 = ((n+m+1)/2) - cut1     //no. of ele from B on LHS

    l1 = (cut1 == 0) ? -∞ : A[cut1 - 1];
    r1 = (cut1 == n) ? ∞ : A[cut1]
    l2 = (cut2 == 0) ? -∞ : B[cut2 - 1]
    r2 = (cut2 == ∞) ? ∞ : B[cut2]

    if (l1 ≤ r2 && l2 ≤ r1)
    {
        if ((n+m)%2 == 0)
        {
            return (max(l1, l2) + min(r1, r2)) / 2
        }
        else
        {
            return max(l1, l2)
        }
    }
    else if (l1 > r2)  hi = cut1 - 1
    else               lo = cut1 + 1
}
```



Total elc = 9

for LHS = 5

for RHS = 4

lo	hi	cut1	cut 2	
0	4	2	3	
0	1	0	5	
1	1	1		→ correct splitty

Ans = max (7, 4) = 7