Binary Search 2

Rajat Sharma
Khushi Raj
Yugesh v
sharath r
Naval Oli
Madhan Kumar M S
Subhranil Kundu
Vishal Mosa
Shrikanth
Murali Mudigonda
Saurabh Ruikar
Sarthak
amit khandelwal 100%.
Vasanth
Akansh Nirmal
Rajendra
Aditya
Sushant
Pankaj
Sneha L
PREETHAM
Venkata Sribhavana Nandiraju
suyash gupta
Rajeeb Kumar Nath
Prajwal Khobragade
Sumit Adwani
Gagan Kumar S
KULDEEP PATIDAR
Shani Jaiswal
Gowtham
Shradha Srivastava
Sanket Giri
Abhishek Sharma
Vimal Kumar
Nikhil Pandey

- Search in a rotated sorted array
- Square root
- A^{th} magical number.
- Median of two sorted arrays.

Current PSP

↓

65%.

Important Announcement

Contest 2 — Reattempt 2 is live

Recap

1> Identify the search space

```
l = 0
r = N - 1
while ( l <= r ) {
```

2> check if mid position can be my ans

```
    mid = l + (r - l)/2    // avoids overflow.
    if ( A[mid] == k )  return mid
```

3> Decide whether to go left or right

```
    if (A[mid] > k ) {
        r = mid - 1        // go left
    }
    else { l = mid + 1 }   // go right
}

return -1
```

3 steps to solve any binary search question

1> Identify the search space

2> check if mid position can be my ans

3> Decide whether to go left or right

Q> Search in a rotated sorted array. MSFT Nutanix

clockwise direction

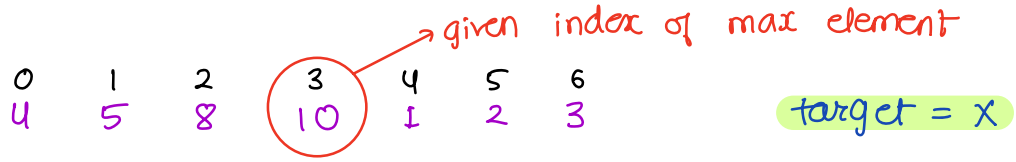
Rotate 1 → 15 2 4 8 10

Rotate 2 → 10 15 2 4 8

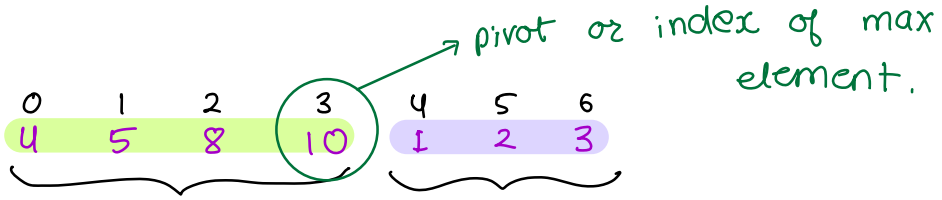
How to figure out if array is already sorted?

Compare first & last value in array

A[0] < A[N-1]

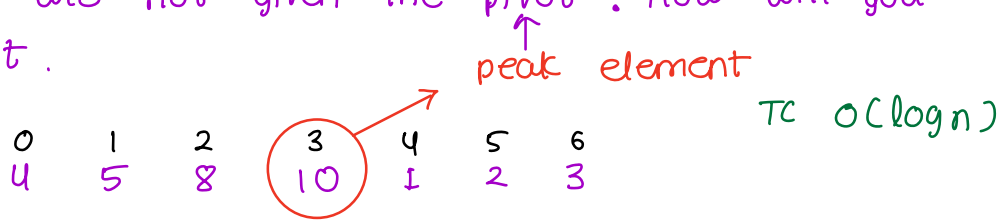


If you are given the index of the largest element How will you solve it in O(logN).



two separate sorted arrays.

- If you are not given the pivot. How will you solve it.

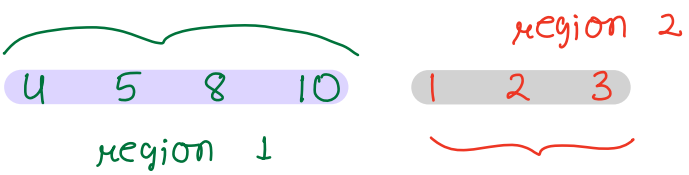


Apply BS on 0 to peak element index

Apply BS on peak element index +1 ⟶ last index

overall TC : O(log N)

* Do it using one single binary search.



How to identify in which region any value lies ?

X = 8 ⟶ region 1

X = 3 ⟶ region 2

```
int getRegion (A[], X) {
    if (X >= A[0]) return 1
    return 2
}
```

target = X

Overall Idea :-

1> Identify the search space ⟶ 0 — N-1

2> check if mid position can be my any
if (A[mid] == X) return mid

3> Decide whether to go left or right

---

region 1 region 2

0	1	2	3	4	5	6	7	8	9	10	11
10	20	30	1	2	3	4	5	6	7	8	9

K=20

left	right	mid	mid Region	target Region	
0	11	5	2	1	go left
0	4	2	1	1	normal BS
0	1	0	1	1	go left
1	1	1	1	1	go right

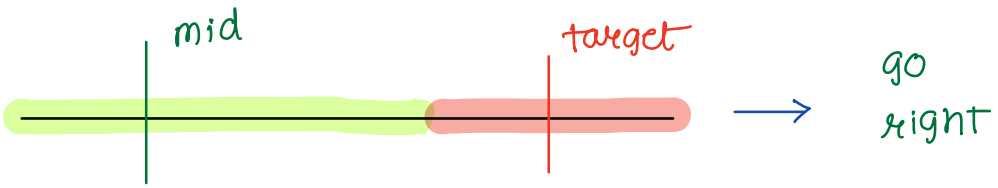
found 20 at index 1

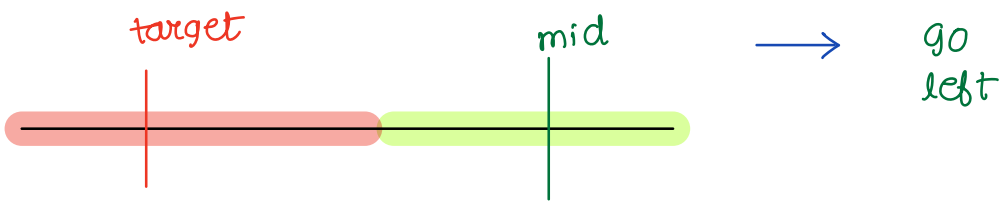
K=60

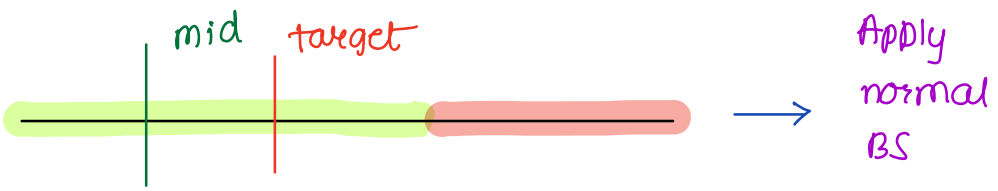
0	1	2	3	4	5	6
70	80	90	100	40	50	60

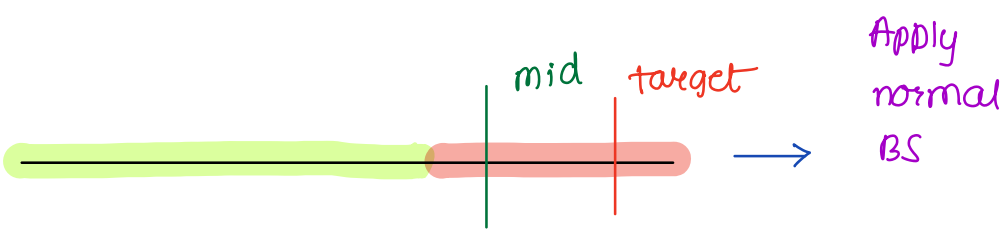
left	right	mid	mid Region	target Region	
0	6	3	1	2	go right
4	6	5	2	2	go right
6	6	6	2	2	

Found value at index 6










```
int getRegion (A[], X) {
    if (X >= A[0]) return 1
    return 2
}
```

Pseudo code

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

```
int rotatedSearch ( A[N] , target) {
    left = 0
    right = n - 1
    targetRegion = getRegion(target)

    while ( left <= right ) {
        mid = left + ( right - left) / 2
        if ( A[mid] == target ) return mid
    3> Decide whether to go left or right
        midRegion = getRegion(A[mid])
        if ( midRegion == 1 &&
             targetRegion == 2 ) {  // go right
            left = mid + 1
        }
        else if ( midRegion == 2 &&
                  targetRegion == 1 ) {  // go left
            right = mid - 1
        }
        else { // same region
            // Normal Binary search
            if (A[mid] < target ) {
                left = mid + 1
            }
            else { right = mid - 1 }
        }
    }
    return -1
}
```

Square Root of N

Q> Given +ve N, find sqrt(N) ⟶ floor sqrt(N)

sqrt(25) = 5
sqrt(20) = 4
sqrt(10) = 3

Bruteforce : Linear Search.

```
i = 1   ans = 0
while (i*i <= N)
    ans = i
    i++
print(ans)
```

TC : ~~O(N)~~ O(√N)

√64 vs $\log_2 64$
8 6

Idea : Binary Search

1> Search space ⟶ 1 — N
2> Target ⟶ if (mid * mid == N) return mid.

Case 1 : mid * mid == N ⟶ return mid

Case 2 : mid * mid < N ⟶ go right
ans = mid

Case 3 : mid * mid > N ⟶ go left.

N = 50

ans = ~~-1~~ ~~8~~ 7

left	right	mid		
1	50	25	25*25 > 50	go left
1	24	12	12*12 > 50	go left
1	11	6	6*6 < 50	go right
7	11	9	9*9 > 50	go left
7	8	7	7*7 < 50	go right
8	8	8	8*8 > 50	go left
8	7			

Pseudocode

TC : O(log(N))

SC : O(1)

```
int sqrt(N) {
    left = 1
    right = N                    } use long for l, r, mid
    ans = 0
    while (left <= right) {
        mid = left + (right - left) / 2

        if (mid*mid == N) return mid
        if (mid*mid < N) {
            ans = mid
            left = mid+1
        } else { right = mid-1 }
    }
    return ans
}
```

Break — 22:26

A^{th} Magical Number

Pre-requisites — gcd

greatest common divisor

$$lcm(X, Y) = \frac{X * Y}{gcd(X, Y)}$$

Multiples of 1	$\longrightarrow$	[1, 100]	100	100/1
Multiples of 3	$\longrightarrow$	[1, 100]	33	100/3
Multiples of 4	$\longrightarrow$	[1, 100]	25	100/4
Multiples of 6	$\longrightarrow$	[1, 100]	16	100/6

Multiples of 4 or 6 from [1, 100] — double counted

12 24 36 .....

$$\frac{100}{4} + \frac{100}{6} = 25 + 16 = 41$$

$$- \frac{100}{12} = -8$$

$$= 33$$

No. of multiples of B or C from [1 $\longrightarrow$ X] =

$$\frac{X}{B} + \frac{X}{C} - \frac{X}{lcm(B,C)}$$

Q) Given A, B, C, find A^{th} magical no.

Number is magical if it is divisible by B or C

Eg: B = 2, C = 3, A = 8 ans = 12

multiples of B = 2 4 6 8 10 12 14 16

multiples of C = 3 6 9 12 15 18 21 24

1 2 3 4 5 6 7 8 9 10 11 12

1 2 3 4 5 6 7 8

Eg B = 4, C = 6, A = 5

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

1 2 3 4 5

* Brute force

TC: O(min(B,C) * A)

```
cnt = 0
for i ⟶ 1 to ∞
    if (i % B || i % C)
        cnt++
    if(cnt == A) return i
```

Search Space ⟶ min(B,C) —— min(B,C) * A

Eg:

B	C	A
4	6	10

guess
Is 36 my 10th magical no. ?

No. of multiples till 36 for 4 or 6

$= \frac{36}{4} + \frac{36}{6} - \frac{36}{12} = 9 + 6 - 3$

$= 12$

36 is my 12th magical no. dec the guess go left.

guess
Is 24 my 10th magical no. ?

$\frac{24}{4} + \frac{24}{6} - \frac{24}{12} = 6 + 4 - 2 = 8$

increase the guess ⟶ go right

Is 30 my 10th magical no. ?

$\frac{30}{4} + \frac{30}{6} - \frac{30}{12} = 7 + 5 - 2$

$= 10$

Dry Run

B	C	A
5	7	4

left	right	mid	
5	20	12	$\frac{12}{5} + \frac{12}{7} - \frac{12}{35} = 3$
13	20	16	$\frac{16}{5} + \frac{16}{7} - \frac{16}{35} = 5$
13	15	14	$\frac{14}{5} + \frac{14}{7} - \frac{14}{35} = 4$

B	C	A
5	7	3

Is 12 my 3rd magical no ?

$$\frac{12}{5} + \frac{12}{7} - \frac{12}{35} = 2 + 1 = 3$$

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
0	0	0	0	1	1	2	2	2	3	3	3	3	4	5

First occurrence of 3

Even if you get value == A go left to find first occurrence of A.

Pseudo code

H.W
Complete the
Pseudocode

```
int   magical ( A, B, C ) {

    left  =
    right =


while ( left <= right ) {
        mid =  left + ( right - left ) / 2
```

Median of Array

middle value in sorted form of array

Median ⟶ 1, 2, 3, 4, 5 ans = 3

Median ⟶ 5, 4, 1, 3, 2

sort the array 1 2 3 4 5 ans = 3

Median ⟶ 1, 2, 3, 4, 5, 6

$\frac{3+4}{2}$ ans = 3.5

Given two sorted arrays A & B. Find median of combined A+B array

Eg. A[] = 1, 4, 5

B[] = 2, 3 ans = 3

Eg. A[] = 1, 2, 3 A+B = 1 2 3 4

B[] = 4 ans = 2.5

Eg. A[] = 1, 3, 4, 7, 10, 12

B[] = 2, 3, 6, 15

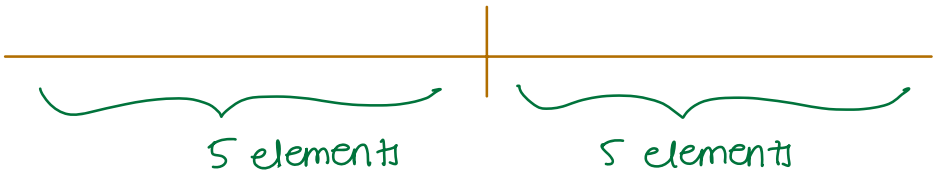
	0	1	2	3	4	5	6	7	8	9
A+B =	1	2	3	3	4	6	7	10	12	15

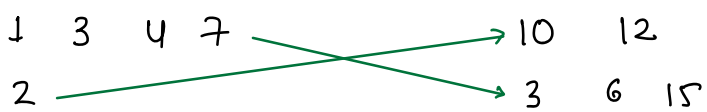
ans = $\frac{4+6}{2}$ = 5

Brute force merge A & B then find median

Case I Assume 4 elements from A

A[] = 1, 3, 4, 7, 10, 12 } 10 elements
B[] = 2, 3, 6, 15





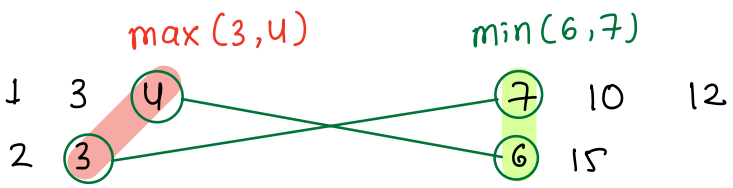
go left.

$2 < 10$ but $7 \nless 3$

Case II Assume 2 elements from A

$6 \nless 4$ $3 < 15$ go right.

Case III Assume 3 elements from A



$3 < 7$ && $4 < 6$

NOTE — We are binary searching on no. of element to pick from A

A = 7 12 14 15

B = 1 2 3 4 9 11

} 10 elements

5 elements 5 elements

0 len(A)

left right mid

0 4 2

7 12 14 15

1 2 3 4 9 11

$3 < 14$ $12 \nless 4$

Take len values from A

go left

---

0 1 0

$-\infty$

7 12 14 15

1 2 3 4 9 11

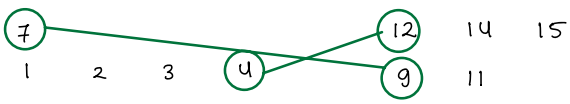
$9 \nless 7$

Take more values from A

go right

---

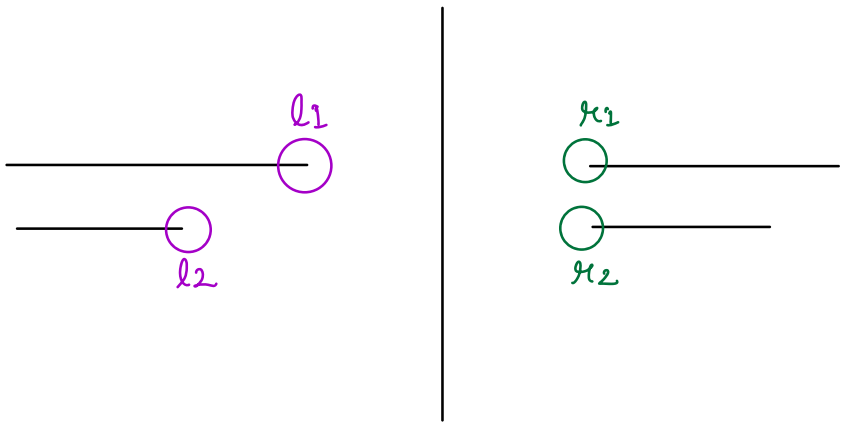
1 1 1



We found out the right partitioning or split

$7 < 9$ && $4 < 12$

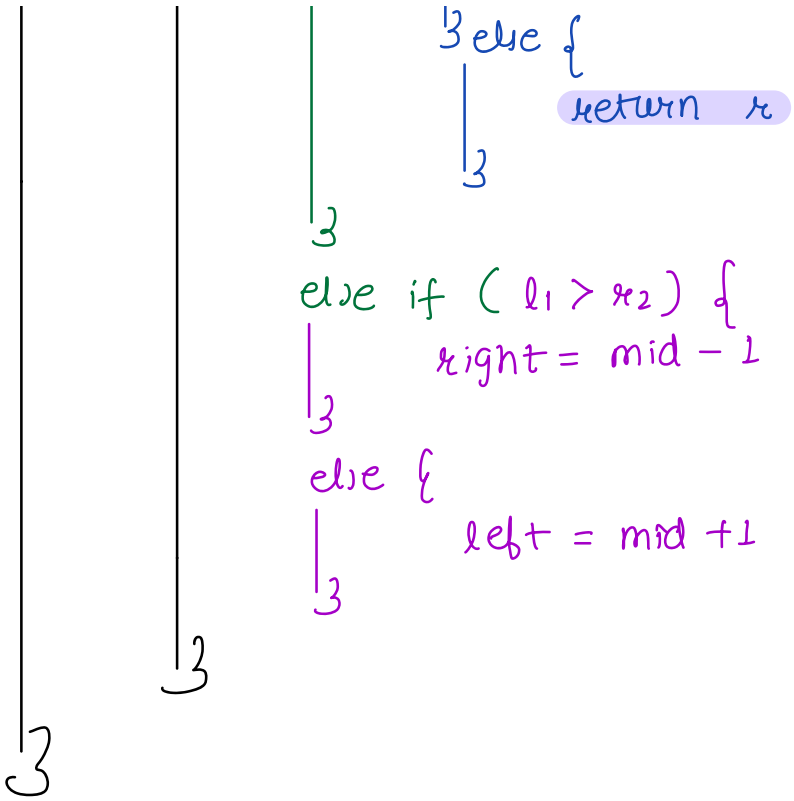
$$\text{median} = \frac{\max(7,4) + \min(12,9)}{2} = 8$$



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

Pseudo code

```
double findMedian(A[], B[]) {                    TC: O(log(lenA))
    lengthA = A.length                           SC: O(1)
    lengthB = B.length

    totalLength = lengthA + lengthB
    halfLength = totalLength / 2
    left = 0
    right = lengthA   // min(lengthA, halfLength)

    while (left <= right) {
        mid = left + (right - left) / 2
        // mid = no. of elements to be taken from A
        takeB = halfLength - mid

        l1 = A[mid-1]
        l2 = B[takeB-1]                          } handle edge cases using safeGet

        r1 = A[mid]
        r2 = B[takeB]

        if (l1 <= r2 && l2 <= r1) {
            // Partition is correct
            l = max(l1, l2)
            r = min(r1, r2)
            if (totalLength % 2 == 0) {
                return (l+r)/2
```


} else {
return r
}
}
else if (l1 > r2) {
right = mid - 1
}
else {
left = mid + 1
}
}
}

Doubt Session

Odd

$\min(7, 9) = 7$

9 elements

$\frac{9}{2} = 4$