Q1 Square root of +ve integer biggest x st $x*x \leq N$

25 → 5 47 → 6

Idea Binary search l = 1 h = N

mid * mid $\leq$ N ans = mid, l = mid + 1

$>$ N h = mid − 1

Dry run

N = 47

mid $\leq \frac{N}{mid}$

l	h	m		
1	47	24	24*24 > 47	
1	23	12	12*12 > 47	
1	11	6	6*6 ≤ 47	ans = 6
7	11	9	9*9 > 47	
7	8	7	7*7 > 47	
7	6	STOP!!!		

```
int sqrt (int N) {
    l = 1      h = n
    while (l ≤ h) {
        mid = (l+h)/2
        if (mid ≤ N/mid) {
            ans = mid
            l = mid+1
        }
        else
            h = mid-1
    }
    return ans
}
```

TC : $O(\log n)$
SC : $O(1)$

4 5 6 1 2 3

02 Search in Rotated sorted array

unique values

```
      0  1  2  3  4  5  6
Eg    4  5  6  7  0  1  2      k = 0
                              ans = 4

      0  1  2  3  4  5
Eg2   4  5  6  7  0  1        k = 100
                              ans = -1
```

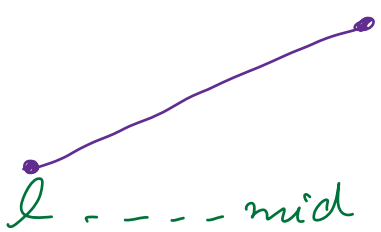
Idea Modify BS algo. But how?

Need to know whether to go left or right

```
      0  1  2  3  4  5  6
      4  5  6  7  0  1  2
```

all yellow > all green

$\Rightarrow$ 2 sorted parts.



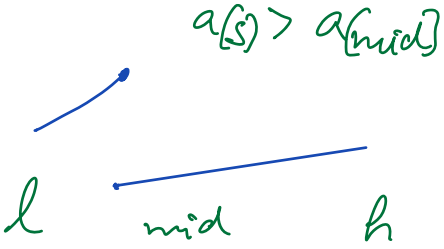
If a[mid] > a[l]

if a[l] < target < a[mid]

h = mid - 1

else

l = mid + 1



else // mid to h sorted

if a[mid] < target < a[h]
l = mid + 1

else

h = mid - 1

Code

```
int search (int arr[], int k) {
  l = 0    h = n-1
  while (l <= h) {
    m = (l+h)/2
    if (arr[m] == k)
        return m
    else if (arr[l] < arr[mid]) {
        if (arr[l] <= k && k <= arr[mid])
            h = mid-1
        else
            l = mid+1
    }
    else {          // mid to h sorted
        if (arr[mid] <= k && k <= arr[h])
            l = mid+1
        else
            h = mid-1
    }
```

```
}
return -1
}
```

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

	l	m	h					h
	0	1	2	3	4	5	6	7
	70	0	10	20	30	40	50	60

k = 21

l	h	m
0	7	3
4	7	5
6	7	6

k = 10

l	h	m
0	7	3
0	2	1
2	2	2

✓

$\frac{7+1}{2} = 4$

0 1 2 3 4 5 6

Tough ques.

03 Median of 2 sorted arrays

Given 2 sorted array, find median of the merged array. If merged array is even length, then return average of 2 middle elements

Eg 1 A = [1, 3] merged = 1, 2, 3 (0 1 2)
B = [2]

ans = 2

Eg 2 A = [1, 2] merged = 1, 2, 3, 4 (0 1 2 3)
B = [3, 4]

ans = 2.5

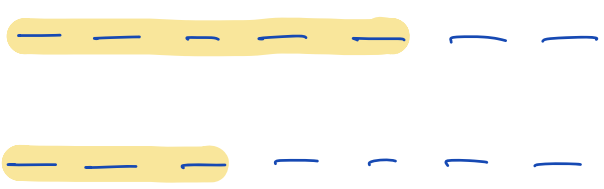
25 ⇒ 10
15

Brute: Merge the 2 sorted arrays & get answer.

TC: $O(n+m)$

We want answer in logarithmic TC

Idea



Some elem of A & some of B will be part of first half of merged array

Now total size = $n_1 + n_2$

$\Rightarrow$ people in first half = $\frac{n_1+n_2+1}{2}$

1 2 3 4 5 6

So, if x elem from A, $\frac{n_1+n_2+1}{2} - x$ elem from B

1 2 3 4 5 6 7

- A = 3 4 5 6 7
- B = 2 5 7 8 10

min_1 max_1 min_2 max_2

How to verify this

$min_1 \leq max_2$ && $min_2 \leq max_1$

Now if even total size

$(max(min_1, min_2) + min(max_1, max_2))/2$

else $max(min_1, min_2)$

else if $min_1 > max_2$

$h = mid - 1$

else

$l = mid + 1$

Dry run

0	1	2	3	4
3	4	5	6	7

0	1	2	3	4
2	5	7	8	10

l	h	m
0	5	2
3	5	4

2 elem A

3 elem B

4 elem A

1 elem B

3 3 3

$ans = \frac{5+6}{2} = 5.5$

merged 2 3 4 5 5 6 7 7 8 10

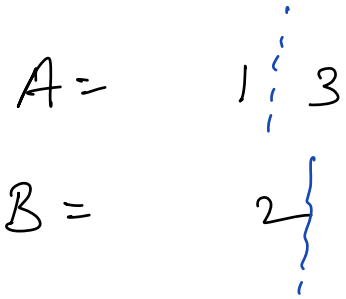
Code

```
double median (int A[], int B[]) {
    n1 = A.size()        n2 = B.size()

    l = 0                h = n1

    while (l <= h) {
        mid = (l+h)/2
        count1 = mid
        count2 = (n1+n2+1)/2 - count1

        int min1, max1, min2, max2

        if (count1 == 0)
            min1 = INT_MIN
        else
            min1 = A[count1-1]

        if (count2 == 0)
            min2 = INT_MIN
        else
            min2 = B[count2-1]
```

```
{
    if (count1 == n1)
        max1 = INT_MAX
    else
        max1 = A[count1]

    if (count2 == n2)
        max2 = INT_MAX
    else
        max2 = B[count2]

    if(min1 <= max2 && min2 <= max1){
        if((n1+n2)%2 == 0)
            return [max(min1, min2) +
                    min(max1, max2)] / 2.0
        else
            return max(min1, min2)
    }

    else if (min1 > max2)
        h = mid - 1
    else
        l = mid + 1
}
}
```

TC: $\log(n_1)$

SC: $O(1)$



done

l	h	m
0	2	1

1 from A

$\frac{2+1+1}{2} - 1 = 1$ from B

ans = 2