Searching 2 : Binary Search Problems

3 steps of Binary Search

1. Define search space
2. Check if mid is the answer
3. Dceide whether to go left or right

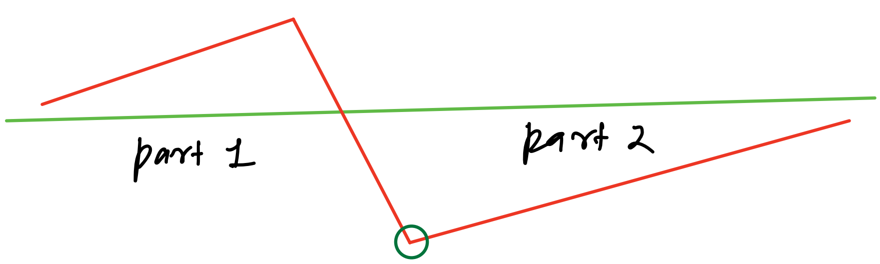
To find mid use,

$$\boxed{l + \frac{(r-l)}{2}} \quad \text{instead of} \quad \frac{(l+r)}{2}$$

Question 1

Given a sorted rotated array of unique elements. find the index of given element K.

	0	1	2	3	4	5	6	7
A = [	6	7	8	1	2	3	4	5]



find the smallest element ?

How to know if A[mid] is in part 1 or part 2 ?

all elements in part 1 > all elements in part 2

A[0] > all elements in Part 2

A[0] < all elements in Part I

Corner case

check for not rotated ? ⟶ A[0] < A[n-1]

TC = O(logN)   SC = O(1)

total 3 binary searches

```
l=0, r=n-1

while ( l<=r) {            <- finding smallest element

    mid = l + (r-l)/2

    if (mid != 0 && A[mid-1] > A[mid])
        return mid

    if( A[mid] < A[0]) { // part 2
        r = mid-1
    }
```

```
    else {
        l = mid+1
    }
}
```

After finding smallest element, do binary search on each part.

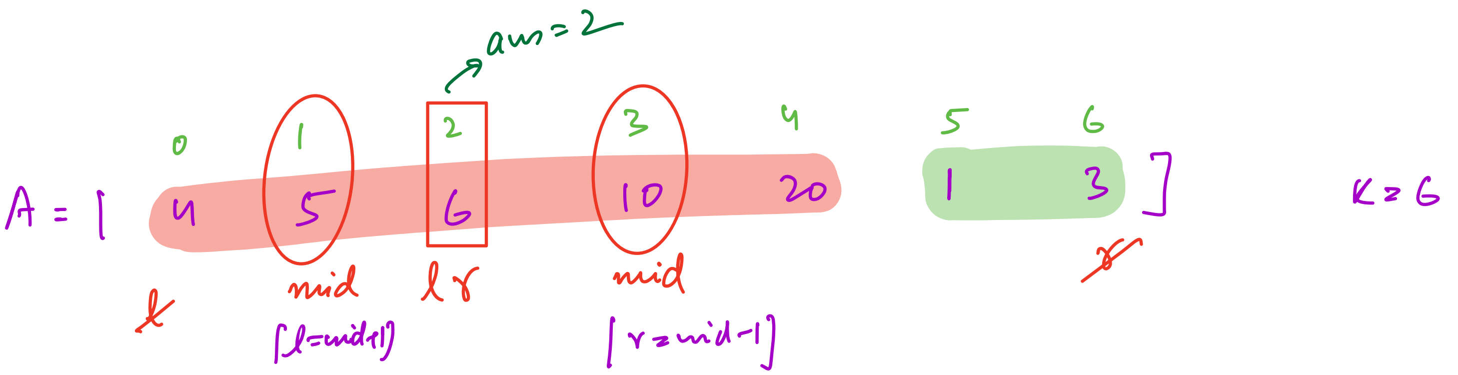
Now let's Solve in 1 binary search

```
// Define search space

l=0 , r=n-1
while ( l <= r) {
    // check if mid is answer or not !
    mid = l + (r-l)/2
    if(A[mid] == K)   return mid

    // decide whether to go left or right

    if ( K < A[0]) { // K is in Part 2
        if( A[mid] < A[0]) { // A[mid] in Part 2
            if( A[mid] < K)  l = mid+1
            else    r = mid-1
```

```
        }
        else {      // A[mid] in Part I
            l = mid+1
        }
    }
    else {      // K is in Part I
        if( A[mid] > A[0]) { // A[mid] in Part I
            if( A[mid] < K)  l = mid+1
            else     r = mid-1
        }
        else {  // A[mid] in Part 2
            r = mid-1
        }
    }
}

return -1
```

> Note : Take care of case when array is not rotated

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



Question 2

find sqrt. of a given perfect sq. number.

$N = 49$ ans $= 7$

$N = 25$ ans $= 5$

$N = 30$ ← invalid input

Bruteforce

min. value of $\sqrt{N} = 1$
max value of $\sqrt{N} = N$] range

```
for ( i=1 to N) {
    if ( i*i == N)
        return i
}
```

$TC = O(\cancel{N})$

$O(\sqrt{N})$

$SC = O(1)$

$N = 25$ $i = 1, 2, 3, 4, 5$

49 $i = 1, 2, 3, 4, 5, 6, 7$

81 $1, 2, 3, 4, 5, 6, 7, 8, 9$

always stops after $\sqrt{N}$ iterations

Use binary search

```
// Define search space

l=1 , r= N

while ( l <=r) {
    // check if mid is answer
    mid = l +(r-l)/2
    if( mid * mid == N) return mid

    // decide whether to go left or right
    if( mid * mid < N) l =mid+1    // go right
    else r = mid-1
}
```

$TC = O(\log N)$

$SC = O(1)$

$N=36$

$l = \cancel{1}\ 5$

$r = \cancel{36}\ \cancel{17}\ 8$

$mid = \cancel{18}\ \cancel{9}\ \cancel{4}\ 6$

$18^2 > 36$

$9^2 > 36$

$4^2 < 36$

$6^2 = 36$

ans

If input is not perfect sq. number,

find the nearest sqrt.

$N = 30$ ans $= 5$

$N = 37$ ans $= 6$

Doubt

$A = 10^9$ $\Rightarrow$ max 2^{32} $\Rightarrow$ convert to binary

11000 1010

$N = 10^5$

$B(i) = 10^9$ weight 2^i