① Sqrt(x) –

→ x = 4
2
→ x = 8
2
→ x = 10
3

→ x < 2 ; return x
→ left = 2 ; right = x/2

```
int sqrtFn(int x) {

        if (x < 2) {
            return x;
        }
        int left=2, right= x/2;   long num;
        while (left <= right) {

                int mid= left +( right - left)/2;

                num = (long) mid* mid;

                if( num > x) {
                        right = mid-1;
                } else if( num < pivot) {
                        left = mid+1;
                } else {
                    return mid;
                }

        }

        return right;
}
```

$T = O(\log x)$
$S = O(1)$

10

2 . 3 . 4 . 5
l m r
l r

② → Guess number higher of lower -

1 — n

Every time you guess wrong, I will tell you whether the number you picked is higher or lower than your guess

guess (num) =

0 : correct
1 : higher
-1 : lower ∈ {guessed no. is longer than than required one}

```
int guessNumber(int n) {

    int left = 1,
        right = n;

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

        int res = guess(mid);
```

```
if(res == 0) {
    return mid;
} else if(res < 0) {
    right = mid-1;
} else {
    left = mid+1;
}
}

return -1;
}
```

$T = O(\log n)$

$S = O(1)$

③ Search in rotated sorted array-

→ 4 5 6 7 0 1 2

0

4

→ 4 5 6 7 0 1 2

3

-1

→ 1

0

-1

Distinct values

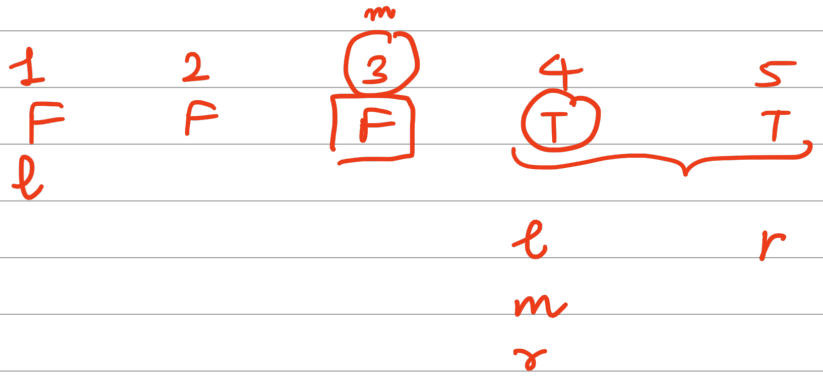
```
int search(int[] nums, int target) {
    int left=0,
        right= nums.length-1;

    while( left <= right ){

        int mid = left + (right - left)/2;

        if( nums[mid] == target) {
            return mid;
        } else if( nums[mid] >= nums[left]){

            if( target >= nums[left] &&
                target < nums[mid]){

                right= mid-1;
            } else {
                left = mid+1;
            }
        } else {
            if( target <= nums[right] &&
                target > nums[mid]) {
                left := mid+1;
            } else {
                right= mid-1;
            }
        }
    }
    return -1;
}
```

T = O(log n)
S = O(1)

II Advanced form of binary search-

① → first bad version-

[1, 2, ..., n]




```
int firstBadVersion(int n) {
    int left = 1,
    right = n;
    while (left < right) {
        int mid = left + (right - left)/2;
        if (isBadVersion(mid)) {
            right = mid;
        } else {
            left = mid+1;
        }
    }
    return left;
}
```

T = O(log n)
S = O(1)

② → find peak element -

→ $-\infty$ [1, 2, 3, 1] $-\infty$

2

→ $-\infty$ [1, 2, 1, 3, 5, 6, 4] $-\infty$

0 1 2 3 4 5 6

1 or 5

```
int findPeakElement(int[] nums) {
    int left = 0,
        right = nums.length - 1;
    while (left < right) {
        int mid = left + (right - left)/2;
        if (nums[mid] > nums[mid+1]) {
            right = mid;
        } else {
            left = mid+1;
        }
    }
    return left;
}
```

T = O(log n)
S = O(1)

$-\infty$ [1 2] $-\infty$

l r

l

m

Find minimum in rotated sorted array

① → Search for a range -

[5, 7, 7, 8, 8, 10] 8
 0 1 2 3 4 5

→ [3, 4]