Binary Search

Time complexity : $O(\log_2 N)$

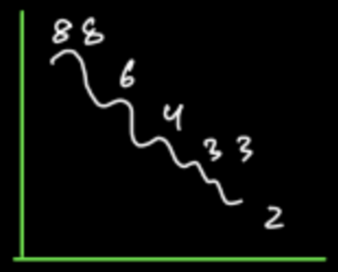
Constraint : Monotonic

- Binary
  - 0000 111
  - 111 000
- Increasing
  - Strictly
  - Non-strictly
- Decreasing
  - Strictly
  - Non strictly

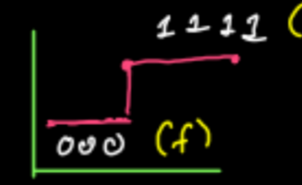
Modified Binary Search

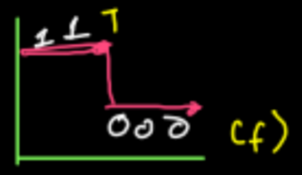
- Matrix
- Infinite (Unbounded)
- Nearly Sorted
- Rotated Sorted
- Bitonic

Non strictly inc



Non Strictly Dec





BS on a binary array is the Binary Search on Answer Concept & it is very very imp.

** The hardest problems on BS are mostly based on this concept.

Apply Binary Search

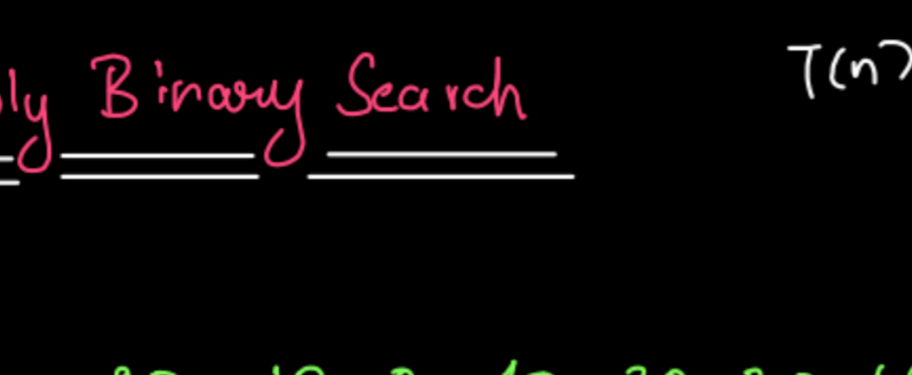
$T(n) = T(n/2) + K$ $O(\log_2 n)$

target => 60

-40	-20	-10	0	10	30	30	60	80	90	100
0	1	2	3	4	5	6	7	8	9	10

low, mid, high

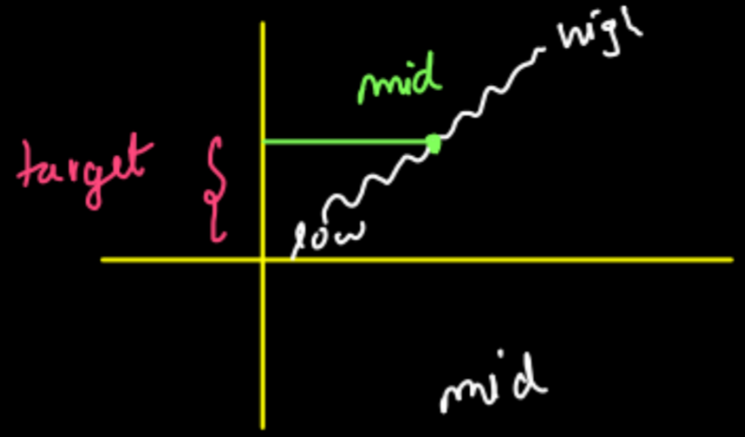
-40	-20	-10	0	10	(30)	30	60	80	90	100
0	1	2	3	4	5	6	7	8	9	10



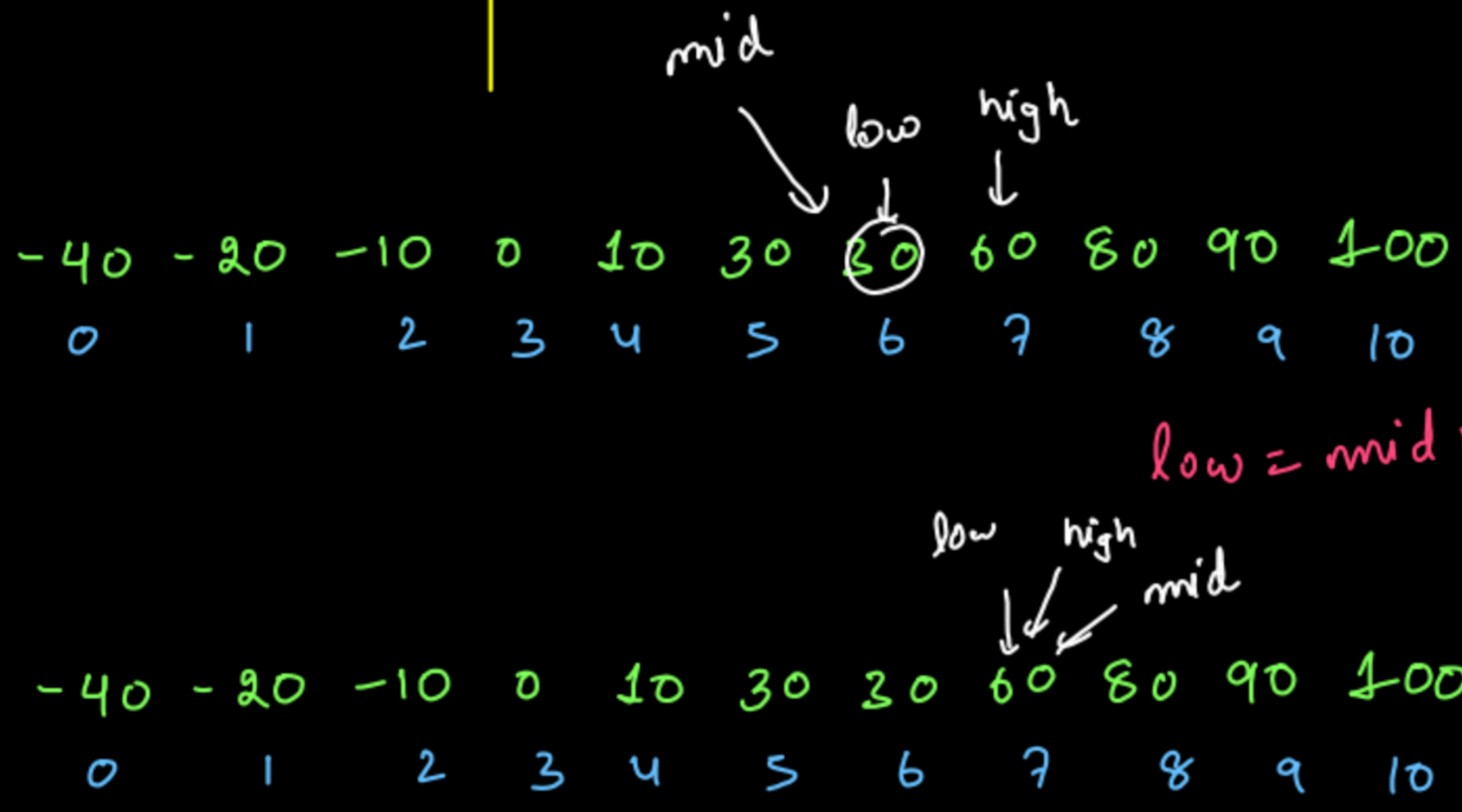
So, low = mid + 1

low, mid, high

-40	-20	-10	0	10	30	30	60	(80)	90	100
0	1	2	3	4	5	6	7	8	9	10



So, high = mid - 1



low = mid + 1

in case of unsuccessful search low > high & element is not found.

↗ This is leetcode code

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

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

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

   return -1; //search unsuccessful
}
```

mid = (left + right) / 2 → can cause overflow

⇒ instead use

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

Binary Search Recurrence Relation

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

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

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

   return -1; //search unsuccessful
}
```

Recursive code ⟶

```
if(l>r) return -1;
int mid = lo+(hi-lo)/2;
if(arr[mid] == target)
    return mid;
else if (arr[mid] < target)
  return bs(arr, mid+1, hi);
else return bs(arr, lo, mid-1);
```

$T(n) = k + T(n/2)$

$T(n) = T(n/2) + k$

$T(n/2) = T(n/4) + k$

$\vdots$

$T(n/2^{x-1}) = T(n/2^x) + k$

$T(n) = T(n/2^x) + x * k$

for $n = 1$ $T(n) = 1$

$T(\frac{n}{2^x}) = T(1) \Rightarrow n = 2^x$

$\Rightarrow x = \log_2 n$

$T(n) = T(n/2^{\log_2 n}) + \log_2 n \, k$

$\boxed{T(n) = O(\log_2 N)}$

Avoiding overflow

$left = 10^9$

$right = 10^9$

$mid = \frac{left + right}{2} = (2 * 10^9) / 2$

↓

might cause overflow (exceed Integer range)

$\frac{l+r}{2} \Longleftrightarrow l + \frac{(r-l)}{2}$ ⤳ using this will not cause overflow

Transition Point (GfG)

Given a sorted array containing only 0s and 1s, find the transition point.

Example 1:

```
Input:
N = 5
arr[] = {0,0,0,1,1}
Output: 3
Explanation: index 3 is the transition
point where 1 begins.
```

If: first occurence of 1.

low mid high

{ 0, 0, 0, 0, 0, 1, 1, 1, 1 }
0 1 2 3 4 5 6 7 8

pa = -1

low mid high

{ 0, 0, 0, 0, 0, 1, 1, 1, 1 }
0 1 2 3 4 5 6 7 8

pa = -1 6

low high
mid

{ 0, 0, 0, 0, 0, 1, 1, 1, 1 }
0 1 2 3 4 5 6 7 8

pa = -1 5

high low

since low > high, STOP.

{ 0, 0, 0, 0, 0, 1, 1, 1, 1 }
0 1 2 3 4 5 6 7 8

pa = -1 5

Transition Point Code

```
int transitionPoint(int arr[], int n) {
    // code here
    int lo = 0;
    int hi = n-1;

    int pa = -1;

    while(lo <= hi) {
        int mid = lo + (hi-lo)/2;

        if(arr[mid] == 0) {
            lo = mid + 1;
        } else {
            pa = mid;
            hi = mid -1;
        }
    }

    return pa;
}
```

$TC = O(\log_2 N)$

First Bad Version (Leetcode) (Asked in Google)

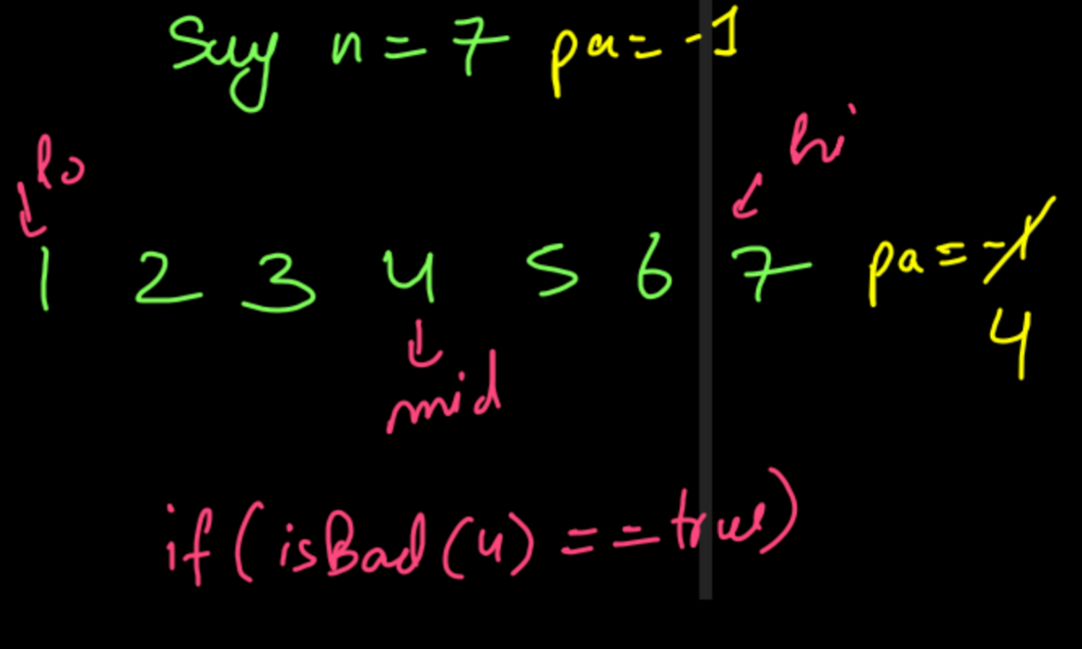
278. First Bad Version

Easy 3961 1444 Add to List Share

You are a product manager and currently leading a team to develop a new product. Unfortunately, the latest version of your product fails the quality check. Since each version is developed based on the previous version, all the versions after a bad version are also bad.

Suppose you have `n` versions `[1, 2, ..., n]` and you want to find out the first bad one, which causes all the following ones to be bad.

You are given an API `bool isBadVersion(version)` which returns whether `version` is bad. Implement a function to find the first bad version. You should minimize the number of calls to the API.



Say n = 7 pa = -1

lo, hi

1 2 3 4 5 6 7 pa = ~~-1~~ 4

mid

if (isBad(4) == true)

mid

lo, hi

1 2 3 4 5 6 7

isBadVersion = false pa = 4

lo, hi

1 2 3 4 5 6 7

isBadVersion = true pa = ~~4~~ 3

hi, lo

since lo > hi = STOP

1 2 3 4 5 6 7

pa = 3 ✓

first Bad Version Code

```java
public class Solution extends VersionControl {
    public int firstBadVersion(int n) {

        int lo = 1;
        int hi = n;

        int pa = -1;
        while(lo <= hi) {
            int mid = lo + (hi-lo)/2;

            if(isBadVersion(mid)) {
                pa = mid;
                hi = mid -1;
            } else {
                lo = mid + 1;
            }
        }

        return pa;
    }
}
```

Guess Number Higher or Lower (leetcode)

374. Guess Number Higher or Lower

Easy 541 92 Add to List Share

We are playing the Guess Game. The game is as follows:

I pick a number from `1` to `n`. You have to guess which number I picked.

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

You call a pre-defined API `int guess(int num)`, which returns 3 possible results:

- `-1`: The number I picked is lower than your guess (i.e. `pick < num`).
- `1`: The number I picked is higher than your guess (i.e. `pick > num`).
- `0`: The number I picked is equal to your guess (i.e. `pick == num`).

Return *the number that I picked*.

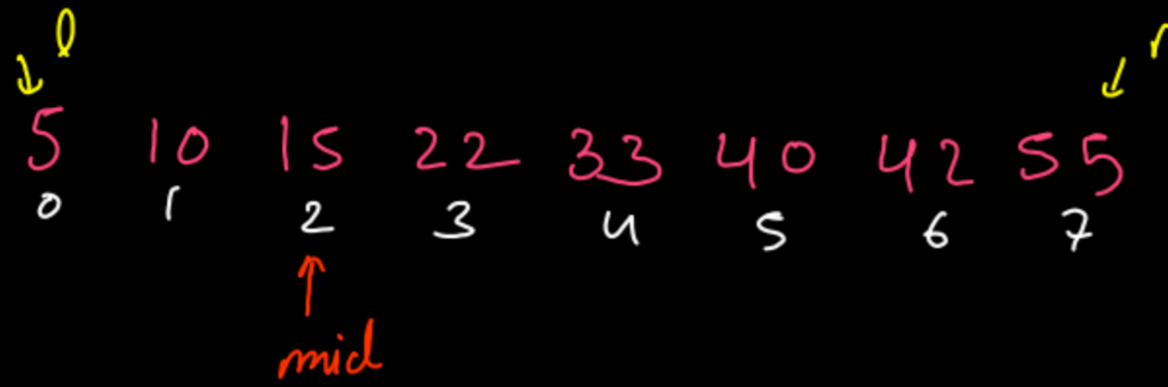
Almost same as prev question writing code directly -

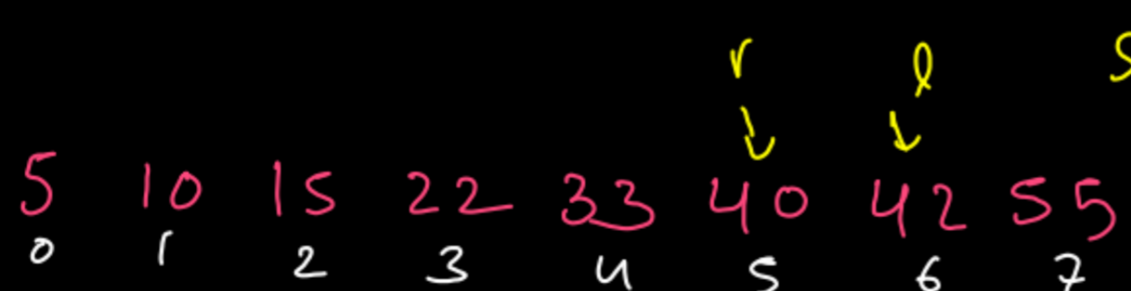
Guess No Higher or Lower Code

```java
public class Solution extends GuessGame {
    public int guessNumber(int n) {

        int lo = 1;
        int hi = n;

        while(lo <= hi) {

            int mid = lo + (hi-lo)/2;

            if(guess(mid) == 0) {
                return mid;
            } else if(guess(mid) == -1) {
                hi = mid - 1;
            } else {
                lo = mid + 1;
            }
        }

        return -1; //unreacheble statement
    }
}
```

Broken Economy (Ceil & Floor) (Pepcoding)

target = 41





Since $l > r$

STOP

arr[r] = floor

arr[l] = ceil

Broken Economy Code

```java
public static void ceilAndFloor(int[] arr, int target) {
    int ceil = Integer.MAX_VALUE; //bado me sabse chota
    int floor = Integer.MIN_VALUE; //choto me sabse bada

    int lo = 0;
    int hi = arr.length-1;

    while(lo <= hi) {
       int mid = lo + (hi-lo)/2;

       if(arr[mid] == target) {
           ceil = arr[mid];
           floor = arr[mid];
           break;
       } else if(arr[mid] < target) {
           if(arr[mid] > floor) {
               floor = arr[mid];
           }
           lo = mid + 1;
       } else {
           if(arr[mid] < ceil) {
               ceil = arr[mid];
           }
           hi = mid - 1;
       }
    }

    System.out.println(ceil);
    System.out.println(floor);
}
```

First And Last Position of Element in Sorted Array

{Leetcode}

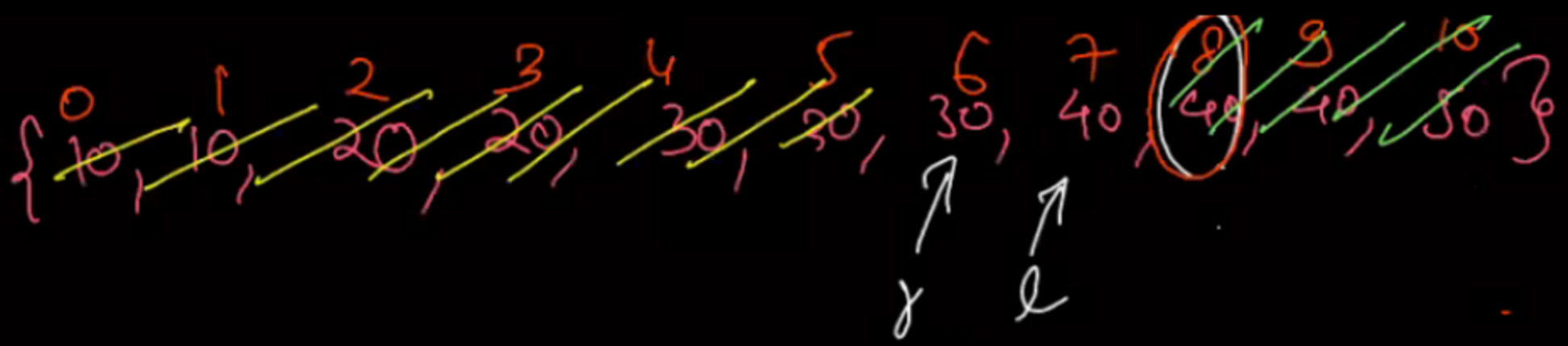
34. Find First and Last Position of Element in Sorted Array

Medium 8848 267 Add to List Share

Given an array of integers `nums` sorted in non-decreasing order, find the starting and ending position of a given `target` value.

If `target` is not found in the array, return `[-1, -1]`.

You must write an algorithm with `O(log n)` runtime complexity.



First & Last Position Code

```java
public int[] searchRange(int[] nums, int target) {

    int[] ans = new int[2];
    ans[0] = firstIndex(nums,target);
    ans[1] = lastIndex(nums,target);

    return ans;
}
```

```java
private int firstIndex(int[] arr,int target) {
    int lo = 0;
    int hi = arr.length-1;

    int pans = -1;
    while(lo <= hi) {
        int mid = lo+ (hi-lo)/2;
        if(arr[mid] == target) {
            pans = mid;
            hi = mid - 1;
        } else if(arr[mid] < target) {
            lo = mid + 1;
        } else {
            hi = mid - 1;
        }
    }

    return pans;
}
```

```java
private int lastIndex(int[] arr, int target) {
    int lo = 0;
    int hi = arr.length-1;

    int pans = -1;
    while(lo <= hi) {
        int mid = lo+ (hi-lo)/2;
        if(arr[mid] == target) {
            pans = mid;
            lo = mid + 1;
        } else if(arr[mid] < target) {
            lo = mid + 1;
        } else {
            hi = mid - 1;
        }
    }

    return pans;
}
```

Search Insert Position

35. Search Insert Position

Easy 6404 365 Add to List Share

Given a sorted array of distinct integers and a target value, return the index if the target is found. If not, return the index where it would be if it were inserted in order.

You must write an algorithm with `O(log n)` runtime complexity.

if element is found then return mid
else return left.

Search Insert Position Code

```java
public int searchInsert(int[] nums, int target) {

    int lo = 0;
    int hi = nums.length-1;

    while(lo <= hi) {
        int mid = lo + (hi-lo)/2;
        if(nums[mid] == target) {
            return mid;
        } else if(nums[mid] < target) {
            lo = mid + 1;
        } else {
            hi = mid - 1;
        }
    }

    return lo;
}
```