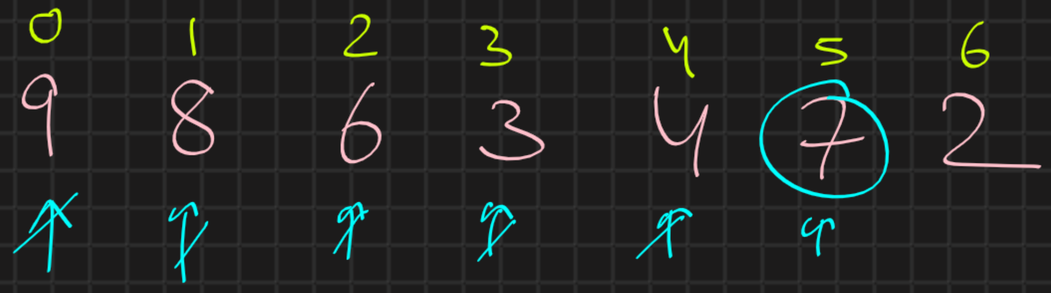
Linear Search (Searching Algo)

target = 7

Case: 1

Successful search

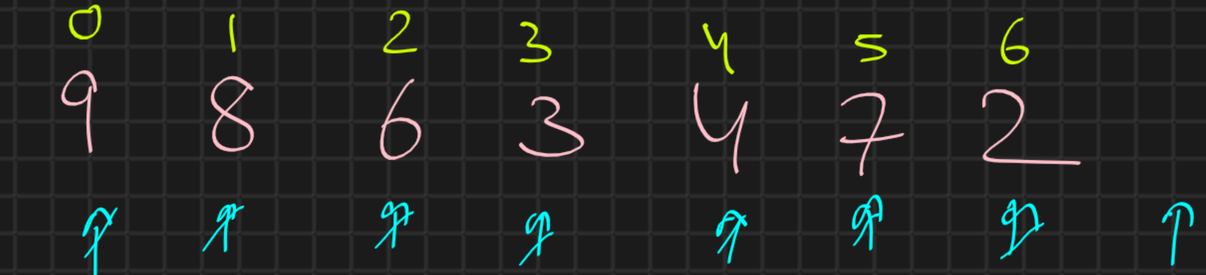


TC = O(N)

target = 10

Case: 2

Unsuccessful Search



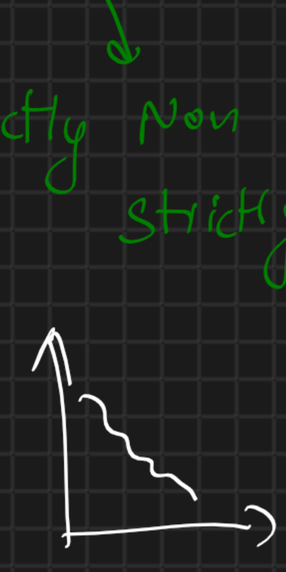
In linear search

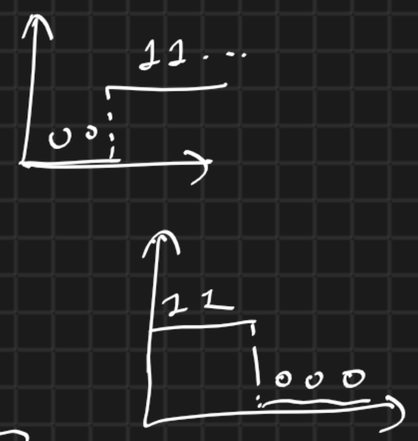
Avg case, Worst case = O(N)

Best case = O(1)

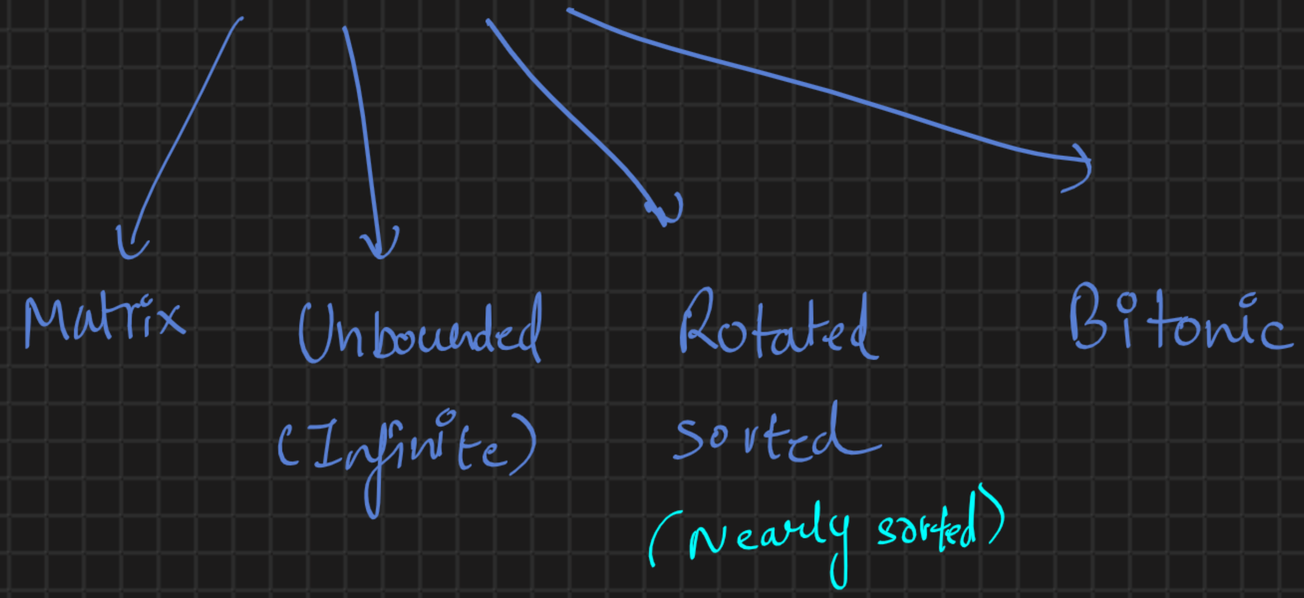
Binary Search

- Monotonic
  - Increasing
    - Strictly
    - Non-strictly
  - Decreasing
    - Strictly
    - Non
  - Binary





Modified BS



+ Binary Search on Answer

↳ Toughest questions on BS are from this category. Here, BS is not applied on the array but on the answer that we can get.

Iterative Binary Search

while (lo ≤ hi)

target = 60

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

lo = 0, mid = 5, hi = 10

$$mid = \frac{0+10}{2} = 5$$

if (arr[mid] < target)

lo = mid + 1

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

lo = 6, mid = 8, hi = 10

$$mid = \frac{6+10}{2} = 8$$

if (arr[mid] > target)

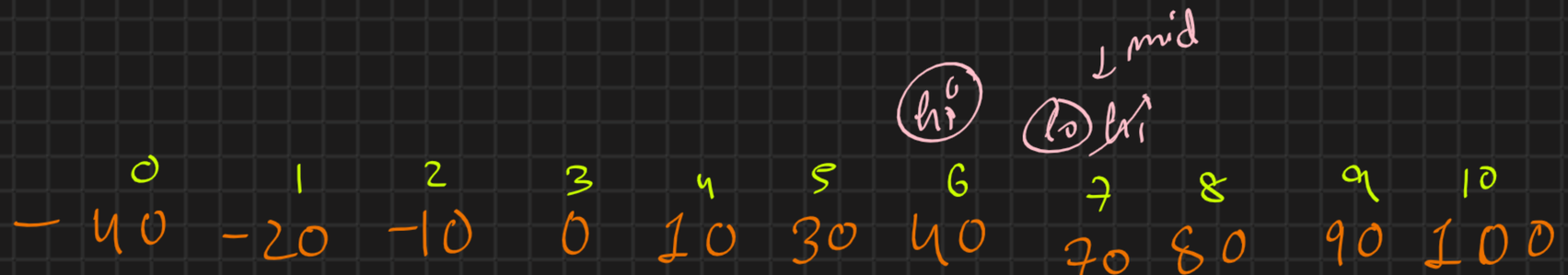
hi = mid - 1

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

lo = 6, mid = 6, hi = 7

$$mid = \frac{6+7}{2} = \frac{13}{2} = 6$$

arr[mid] = target → successful search



Unsuccessful Search

```java
class Solution {
    public int search(int[] nums, int target) {
        int lo = 0;
        int hi = nums.length -1;

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

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

        return -1;
    }
}
```

Int. Max_Value Int. Max_Value

$mid = \frac{lo + hi}{2}$ ✓ ✓ out of range

$mid = lo + \frac{(hi - lo)}{2}$

↓

To avoid overflow

$TC = O(\log_2 N)$

Recursive Binary Search

mid lo hi target = 60

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

BS(arr, 0, 10, target) (lo, hi)

mid = 5

$TC = (calls)^{h} + (pre + post) * h$

$TC = (1)^{\log_2 N} + (K) * \log_2 N$

$TC = O(\log_2 N)$

BS(arr, 6, 10, target) (mid+1)

mid = 8

BS(arr, 6, 7, target) (mid-1)

mid = 6

BS(arr, 7, 7, target)

mid = 7

return 7

R Relation

$$T(N) = T(N/2) + K$$

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

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

$$T(n/4) = T(n/8) + k$$

$\}$ x times

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

$$T(n) = T(n/2^x) + xk$$

ignore constant

for $n = 1$ $\quad T(1) = 1 \Rightarrow T(n=1)$

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

$$\Leftrightarrow \quad n/2^x = 1$$

$$x = \log_2 n$$

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

$$x = \log_2 N$$

$$T(N) = T(N/2^{\log_2 N}) + \log_2 N * k$$

$2^{\log_2 N} = N$

$$T(N) = \underbrace{T(1)}_{\text{constant}} + k \log_2 N$$

$$T(N) = k \log_2 N \approx O(\log_2 N)$$

```java
public int binarySearch(int[] nums, int lo, int hi, int target) {

    if(lo > hi) return -1;

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

    if(nums[mid] == target) return mid;
    else if(nums[mid] < target) return binarySearch(nums,mid + 1,hi,target);

    return binarySearch(nums,lo,mid -1,target);
}

public int search(int[] nums, int target) {
    return binarySearch(nums,0,nums.length-1,target);
}
```

Transition Point

Find Transition Point

Easy Accuracy: **53.15%** Submissions: **100k+** Points: **2**

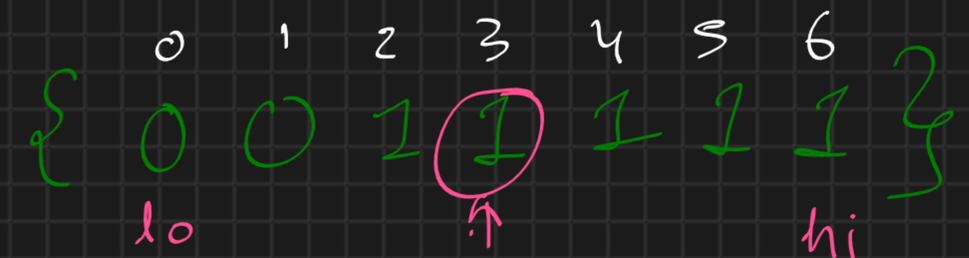
Given a sorted array containing only 0s and 1s, find the transition point.

Example 1:

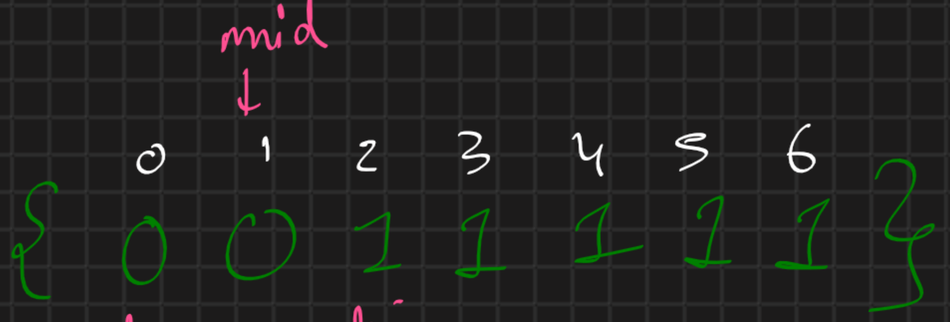
Input:
N = 5
arr[] = {0,0,0,1,1}
Output: 3
Explanation: index 3 is the transition point where 1 begins.

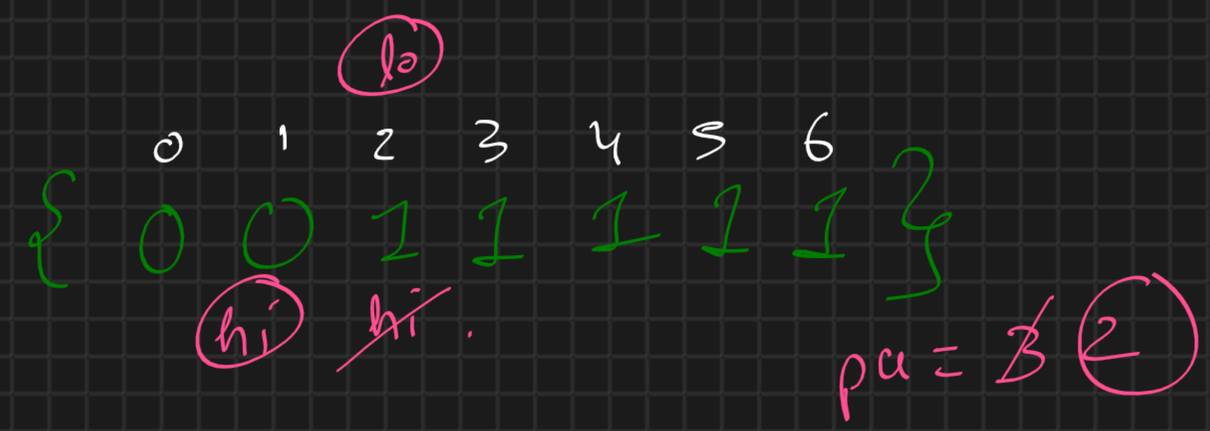
Transition Point : first occ of 1

pa = -1



pa = ~~-1~~ 3





pa = 3 2

```
if (arr[i] == 1)
    pa = i;
    hi = mid - 1;

else
    lo = mid + 1
```

```java
class GfG {
    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] == 1) {
                pa = mid;
                hi = mid-1;
            } else {
                lo = mid + 1;
            }
        }

        return pa;
    }
}
```

find Transition Point (GfG)

$TC = O(\log_2 N)$

$SC = O(1)$

First Bad Version (Asked in Google)

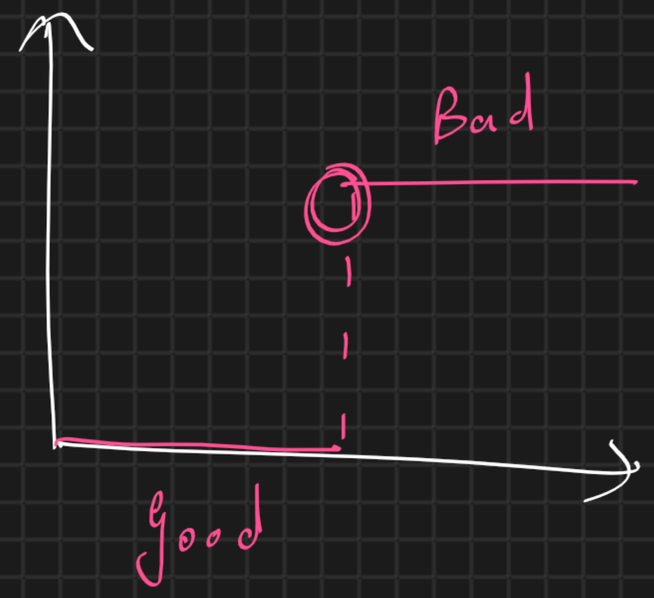
278. First Bad Version

Easy 5558 2092 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.



Same as Transition Point

```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) == true) {
                pa = mid;
                hi = mid -1;
            } else {
                lo = mid + 1;
            }
        }

        return pa;
    }
}
```

$TC = O(\log_2 N)$

$SC = O(1)$

374. Guess Number Higher or Lower

Easy 1329 212 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 three possible results:

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

Return *the number that I picked.*

```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; //this statement will never execute
    }
}
```

target

mid = lo + (hi-lo)/2

guess(mid) = -1

↳ mid > target

hi = mid - 1

guess(mid) = 1

↳ mid < target

lo = mid + 1

TC = O(log₂N)

SC = O(1)

Broken Economy {ceil and floor}

ceil → Bado me sabse chota (Min) (Initial value = ∞)

floor → Chota me sabse bada (Max) (Initial value = -∞)

mid

l r

0 1 2 3 4 5 6 7

{ 5, 10, 15, 22, 33, 40, 42, 55 }

f

target = 42

```
if (arr[mid] < target) {
    floor = arr[mid];
    lo = mid + 1;
}
```

lo mid r

0 1 2 3 4 5 6 7

{ 5, 10, 15, 22, 33, 40, 42, 55 }

f

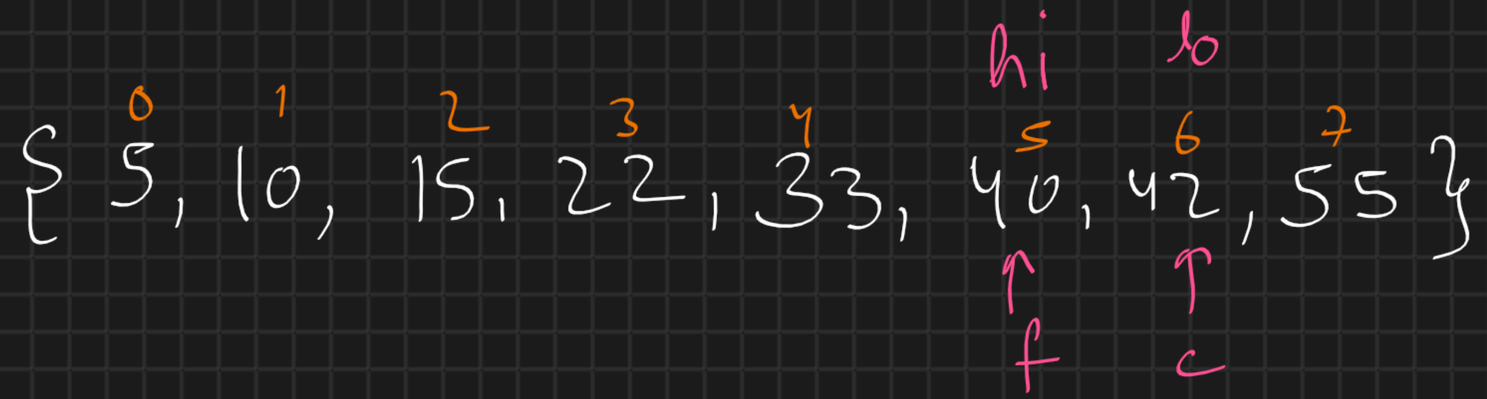
```
if (arr[mid] > target)
{ ceil = arr[mid];
    hi = mid - 1;
}
```

hi lo

0 1 2 3 4 5 6 7

{ 5, 10, 15, 22, 33, 40, 42, 55 }

f c



floor = arr [hi]

ceil = arr [lo]

```java
public static void solve(int[] arr,int target) {

    int ceil = Integer.MAX_VALUE;
    int floor = Integer.MIN_VALUE;

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

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

        if(arr[mid] == target) {
            System.out.println(arr[mid]);
            return;
        } else if(arr[mid] < target) {
            if(arr[mid] > floor) {
                floor = mid;
            }
            lo = mid + 1;
        } else {
            if(arr[mid] < ceil) {
                ceil = mid;
            }
            hi = mid -1;
        }
    }

    //ceil
    System.out.println(arr[lo]);
    //floor
    System.out.println(arr[hi]);
}
```

$TC = O(\log_2 N)$

$SC = O(1)$