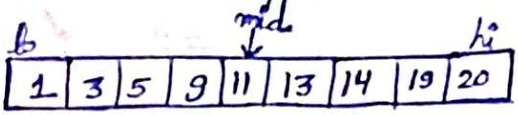
Binary Search Foundations :-



If we is applied on sorted array → Find a num → $O(n \log n)$

Q: [0, 0, 0, 0, 0, 0, 1, 1, 1, 1]
0 1 2 3 4 5 6 7 8 9

[---0---|1---]
Find the pos^n of first 1 ?

① Search space
↳ Ans → index → [0.....n-1]

② Binary Search Logic → lo, hi, ans, mid → $low + \left(\frac{high-low}{2}\right) \simeq \frac{lo+hi}{2}$

[ ] What range to search in

Best ans find till now or default ans

bcoz $\left(\frac{lo+hi}{2}\right)$ → can leads to overflow of int.

lo = 0
hi = n-1
ans = n/-1

Ans	lo	hi	mid $\left(\frac{lo+hi}{2}\right)$
10	0	9	4
10	5	9	7
7	5	6	5
7	6	6	6
6	≥6	≤5	× stop

6 ↳ This is ans

Code :-

```
lo = 0
hi = N-1
ans = N
while (lo ≤ hi)
    mid = (lo+hi)/2
    if (arr[mid]==1) {
        ans = mid;
        hi = mid-1;      → move ^left
    }
    else {
        low = mid+1      → right
    }
```

Lower-bound → First element ≥ x

X = 9

Arr : [1, 2, 4, 8, 8, 8, 20, 25] , N = 8
0 1 2 3 4 5 6 7

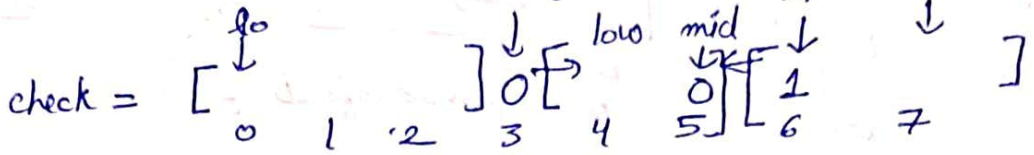
check[i] = arr[i] ≥ x
↓
If yes → 1
else → 0

check = [0, 0, 0, 0, 0, 0, 1, 1]
0 1 2 3 4 5 6 7

So where the pos^n of first 1 will be the ans

↳ This will take O(n) time to create and then O(log n), so instead we do this :

Ans = 6



This We created only the elements we needed. We will not create an array just check it is 0/1 and move forward accordingly.

Code :-

```
int n;
int arr[100100];
int x;
int check(int mid){
  if(arr[mid]>=x) return 1;
  else return 0;
}
void solve(){
   cin >>n;
   for(int i=0; i<n; i++)
      cin >> arr[i];
   cin >>x;
   // lower bound
   int lo = 0;
   int hi = n-1;
   int ans = n;
   while(lo <= hi){
```

```
int mid = (lo+hi)/2; //lo + (hi-lo)/2;
if (check(mid)==1) { check mid = 1
    ans = mid;
    hi = mid-1;
}
else {
    lo = mid+1;
}
}
cout << ans << endl;
```

For upper bound, only slight change in check func^n.

```
int check(int mid){
    if (arr[mid] > x) return 1;
    else return 0;
}
```

Q. Rotated array

How many times original array would be rotated to get this array? Find this in $O(\log(N))$

$[5, 8, 12, 11, 3]$ → Ans = 3

```
   [ 5   8   12   1   2 ]
check->[0   0   0   1   1 ]
```

```
int check(int mid){
  if(arr[mid] <= arr[n-1]) return 1;
  else return 0;
}
```

```
     1   2   3   4   5
check: [1   1   1   1   1 ]

     2   3   4   5   1
check: 0   0   0   0   1
```

Q Peak finding in a Bitonic Array

↳ The array will first strictly inc & then strictly dec.

```
1  2  3 | (4) | 2  1
0  1  2 | 3   | 4  5
0  0  0 | 1   | 1  1
```

```
if (mid == n-1) return -1;
if (arr[mid] > arr[mid+1]) return 1;
else return 0;
```

Time Complexity :-

$O(\log n) \times O(check)$

$O(\log (hi - lo + 1)) * (check))$

How to identify it is a Binary search Problem ?