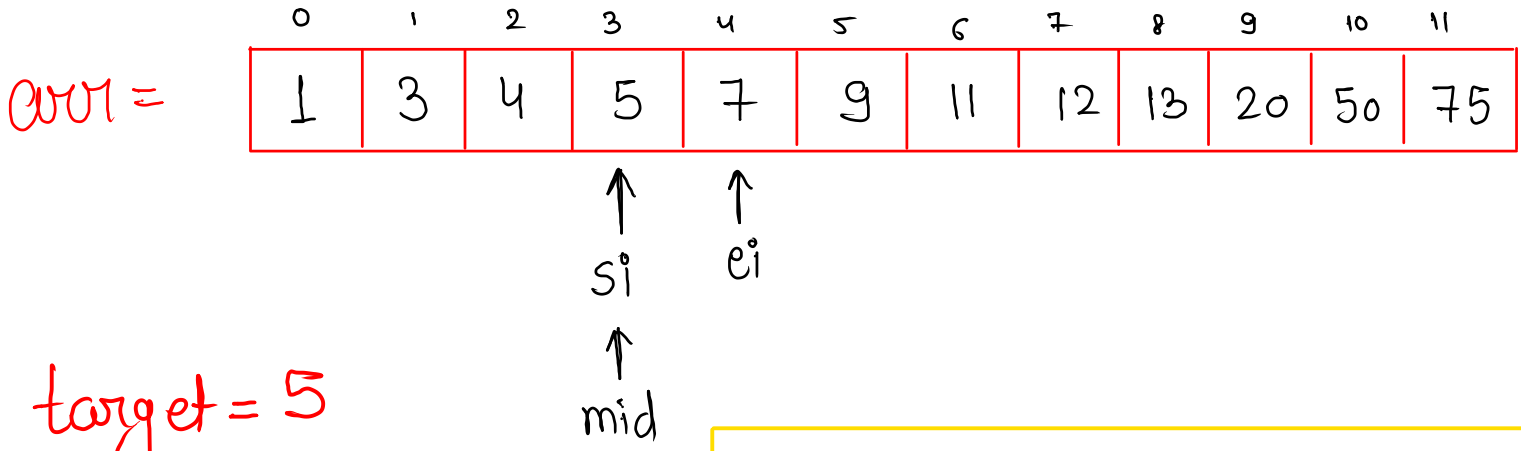


target = 5

psudo code (template) //assume array is sorted

1) declare si = 0, ei = n-1;
2) traverse loop until si <= ei
   2.1) find mid index = (si + ei)/2;
   2.2) check cond of arr[mid] == target
        return mid;
   2.3) check cond. of arr[mid] < target
        si = mid+1
   2.4) check cond. of arr[mid] > target
        ei = mid-1;
3) return -1;

Time Complexity

$$N + \frac{N}{2} + \frac{N}{4} + \frac{N}{8} + \frac{N}{16} + \cdots\cdots + 4 + 2 + 1 = \log(N)$$

---

array N

$N = 10^5$, $\log(N) = 5$

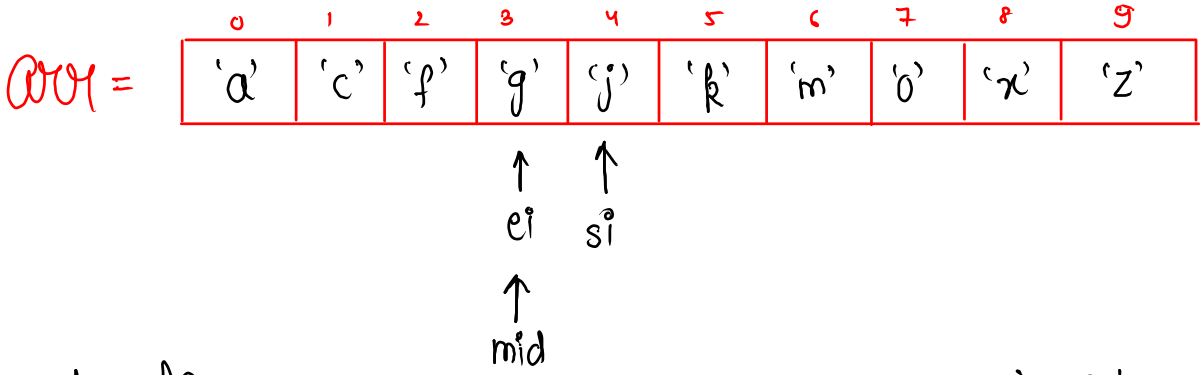
$= 10^{10}$, $\log(N) = 10$

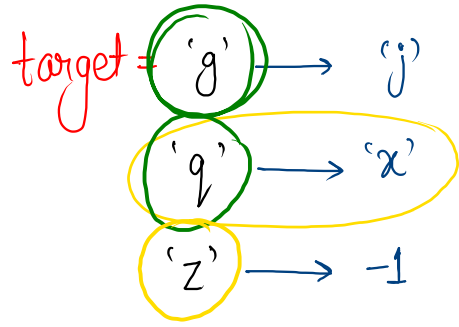
Binary Search in an Array

```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    int[] arr = new int[n];
    for (int i = 0; i < n; i++) {
        arr[i] = scn.nextInt();
    }
    int target = scn.nextInt();
    int ans = BS(arr, n, target);
    System.out.println(ans);
}
public static int BS(int[] arr, int n, int target) {
    int si = 0;
    int ei = n - 1;
    int ans = -1;
    while (si <= ei) {
        int mid = (si + ei) / 2;
        if ( arr[mid] == target ) {
            return mid;
        } else if ( arr[mid] < target ) {
            si = mid + 1;
        } else {
            ei = mid - 1;
        }
    }
    return -1;
}
```

$T.C = O(\log(N))$

Search Character





psudo code

```
si = 0,  ei = n-1
while ( si <= ei) {
    mid = (si + ei) /2;
    if ( arr[mid] == target){
a       si = mid+1;
    } else if ( arr[mid] < target){
b       si = mid+1;
    } else if ( arr[mid] > target){
c       ei = mid-1;
    }
}
return arr[si];
```

Imp

- if we are looking for ans == target, then ans = mid
- if we are looking for ans > target, then ans = si
- if we are looking for ans < target, then ans = ei

code

```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    char target = scn.next().charAt(0);
    int n = scn.nextInt();
    char[] arr = new char[n];
    for (int i = 0; i < n; i++) {
        arr[i] = scn.next().charAt(0);
    }
    justGreaterTarget(arr, n, target);
}
public static void justGreaterTarget(char[] arr, int n, char target) {
    int si = 0;
    int ei = n - 1;
    while (si <= ei) {
        int mid = (si + ei) / 2;
        if ( arr[mid] <= target ) {
            si = mid + 1;
        } else if ( arr[mid] > target ) {
            ei = mid - 1;
        }
    }
    if ( si == n ) {
        System.out.println(-1);
    } else {
        System.out.println(arr[si]);
    }
}
```

$T.C = O(\log(N))$

⇒ Variation of Binary Search

arr =

0	1	2	3	4	5	6	7	8	9	10
1	2	3	3	3	3	3	4	5	6	7

target = 3

↳ find first occ. of target = BSLB
(Binary Search lower bound)

↳ find last occ. of target = BSUB
(Binary Search upper bound)

target = 2

arr =

0	1	2	3	4	5	6	7	8	9	10
1	2	2	2	2	2	2	3	3	3	7

si ei mid

```
si = 0, ei = n-1
while ( si <= ei) {
    mid = (si + ei) /2;
    if ( arr[mid] == target) {
        if ( arr [mid] == arr [mid+1] ){
            si = mid+1;
        } else{
            return mid;
        }
    } else if ( arr[mid] > target){
        ei = mid-1;
    } else if ( arr[mid] < target)
        si = mid+1;
    }
}
```

a

b

c

(BSUB)

target = 2

arr =

0	1	2	3	4	5	6	7	8	9	10
1	2	2	2	2	2	2	3	3	3	7

si, ei

mid

si = 0, ei = n-1

```
while ( si <= ei) {
    mid = (si + ei) /2;
    if ( arr[mid] == target) {
        if ( arr [mid] == arr[mid-1] ){
            ei = mid-1;
        }else{
            return mid;
        }
    } else if ( arr[mid] > target){
        ei = mid-1;
    } else if ( arr[mid] < target)
        si = mid+1;
    }
}
```

a

b

c

(BSLB)

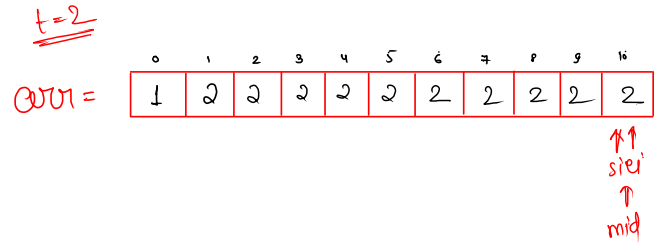
Note:-

BSLB:- if mid element is equal to (mid-1) element then mid is not the first occ. of target so move to left side

BSUB:- if mid element is equal to (mid+1) element then mid is not the last occ. of target so move to right side

Find Last Occurrence

```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    int[] arr = new int[n];
    for (int i = 0; i < n; i++) {
        arr[i] = scn.nextInt();
    }
    int target = scn.nextInt();
    int ans = BSUB(arr, n, target);
    System.out.println(ans);
}
public static int BSUB(int[] arr, int n, int target) {
    int si = 0;
    int ei = n - 1;
    while (si <= ei) {
        int mid = (si + ei) / 2;
        if ( arr[mid] == target ) {
            if ( mid + 1 < n && arr[mid] == arr[mid + 1] ) {
                si = mid + 1;
            } else {
                return mid;
            }
        } else if ( arr[mid] > target ) {
            ei = mid - 1;
        } else if ( arr[mid] < target ) {
            si = mid + 1;
        }
    }
    return -1;
}
```



BSUB

Code for BSLB

```java
public static int BSLB(int[] arr, int n, int target) {
    int si = 0;
    int ei = n - 1;
    while (si <= ei) {
        int mid = (si + ei) / 2;
        if ( arr[mid] == target ) {
            if ( mid - 1 >= 0 && arr[mid] == arr[mid - 1] ) {
                ei = mid - 1;
            } else {
                return mid;
            }
        } else if ( arr[mid] > target ) {
            ei = mid - 1;
        } else if ( arr[mid] < target ) {
            si = mid + 1;
        }
    }
    return -1;
}
```