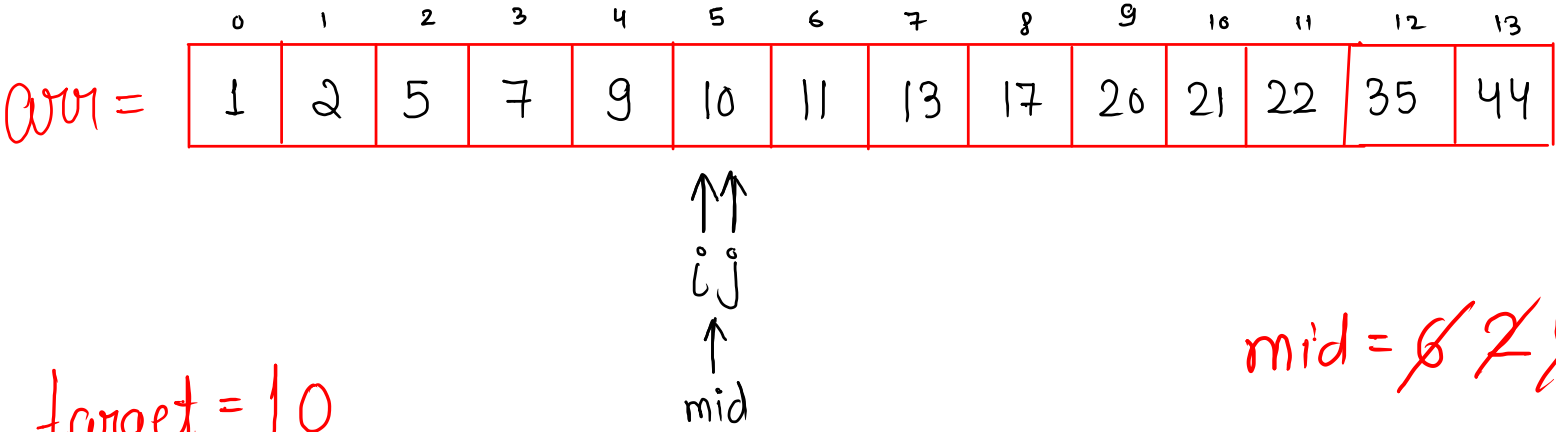
# ⇒ Binary Search (searching algorithm)

(M.M. gmp) ↳ O(log(n))

pre-req. :- array should be sorted




target = 10

mid = ~~6~~ ~~2~~ ~~4~~ 5

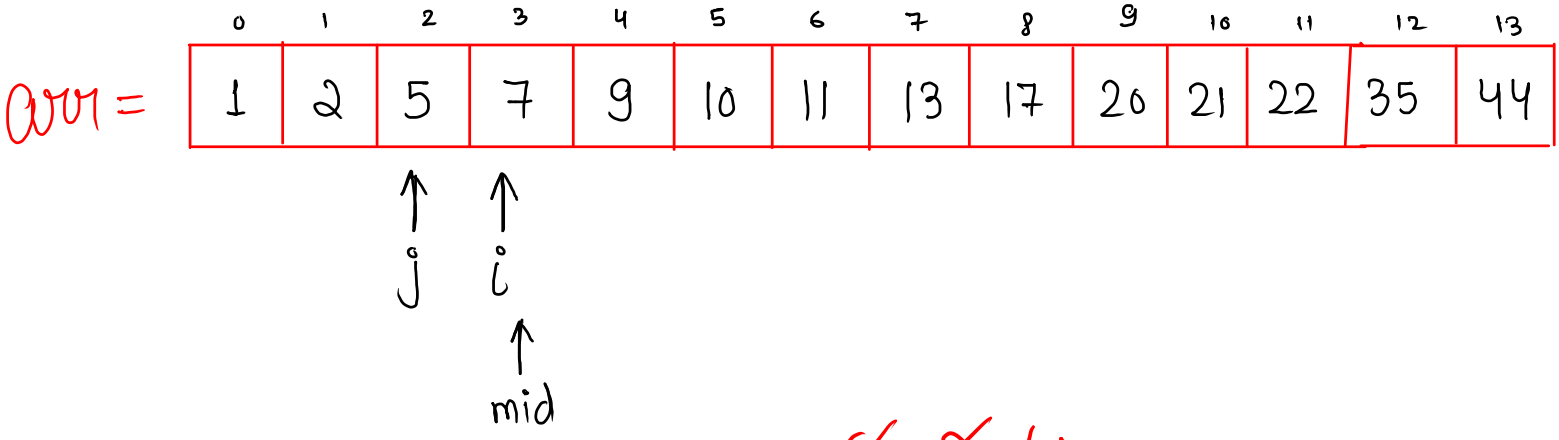
Code

```
int i=0, j= n-1;
while( i <= j) {
    mid= (i+j)/2;
    if (arr[mid] == tar) {
        return true;
    } else if (arr[mid] > tar) {
        j= mid-1;
    } else {
        i= mid+1;
    }
}
return false;
```

a

b

c




target = 6

mid = ~~6~~ ~~2~~ 4

return false

# Time Complexity

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

---

$\log_{10}(10) = 1$

$\log_{10}(100) = 2$

$\log_{10}(1000000) = 6$

| linear | binary |
|---|---|
| 10 | 1 |
| 100 | 2 |
| $10^6$ | 6 |

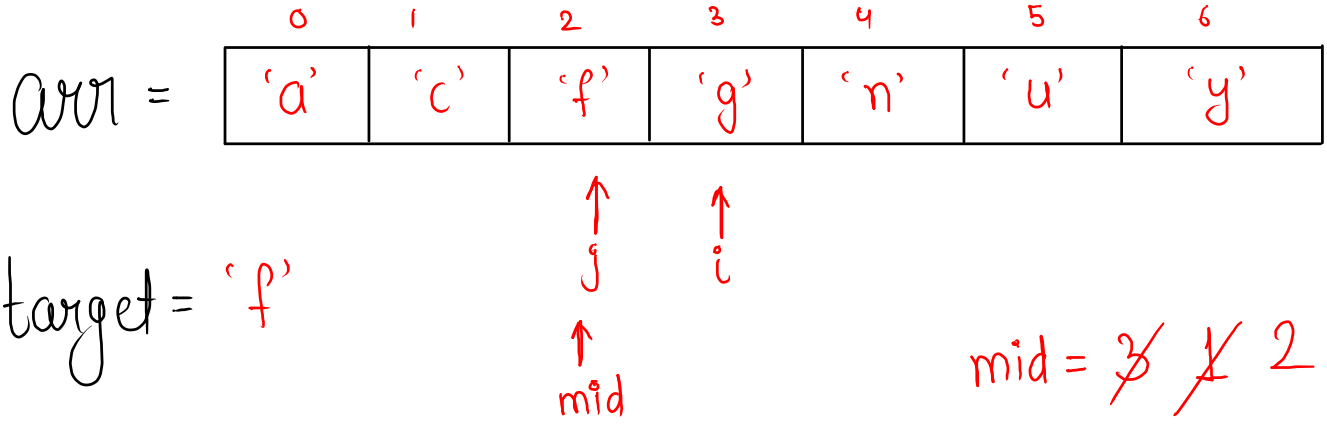
# Binary Search in an Array

```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();
    System.out.println(binarySearch(arr, n, target));
}
public static int binarySearch(int[] arr, int n, int target) {
    int i = 0;
    int j = n - 1;
    while ( i <= j ) {
        int mid = (i + j) / 2;
        if ( arr[mid] == target ) {
            return mid;
        } else if ( arr[mid] < target ) {
            i = mid + 1;
        } else if ( arr[mid] > target ) {
            j = mid - 1;
        }
    }
    return -1;
}
```

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

where, n is size of array

# Search Character




psudo code

```
mid = (i+j)/2;
if ( mid ele == target){
        i = mid+1;
} else if (mid ele < target) {
        i = mid+1;
} else {
        j = mid-1;
}
```

# Imp note

if required arr[mid] == target, ans = mid

if reqd. ans just greater than target, ans = i

if reqd. ans just smaller than target, ans = j

Code

```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    char ch = 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);
    }
    greaterChar(arr, n, ch);
}
public static void greaterChar(char[] arr, int n, char ch) {
    int i = 0;
    int j = n - 1;
    while ( i <= j ) {
        int mid = (i + j) / 2;
        if ( arr[mid] == ch ) {
            i = mid + 1;
        } else if ( arr[mid] < ch ) {
            i = mid + 1;
        } else if ( arr[mid] > ch ) {
            j = mid - 1;
        }
    }
    if ( i == n ) {
        System.out.println("-1");
    } else {
        System.out.println(arr[i]);
    }
}
```

T.C = log(n)

S.C = O(1)

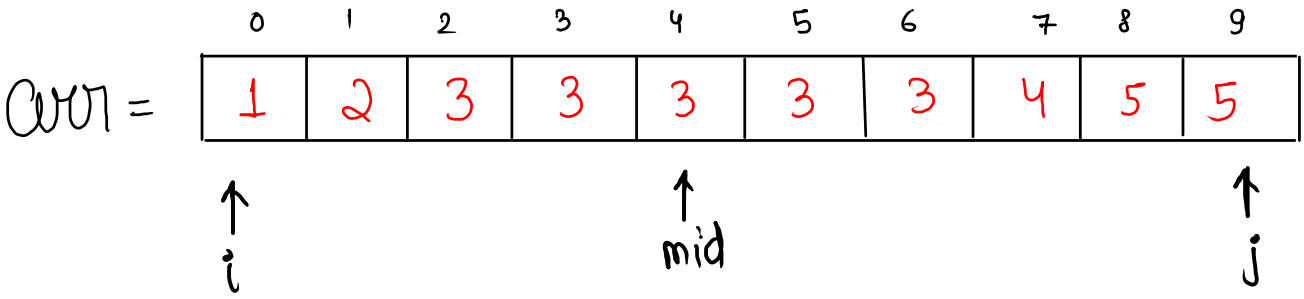
# ⇒ Variation of Binary Search

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| arr = | 1 | 2 | 3 | 3 | 3 | 3 | 3 | 4 | 5 | 5 |

target = 3

↳ first occ. of target :- BSLB
(Binary Search Lower Bound)

↳ last occ. of target :- BSUB
(Binary Search Upper Bound)




```
int i=0, j = n-1;
while( i <= j ) {
    mid = (i+j)/2;
    if ( arr[mid] == tar ) {
a       [                ]
    } else if ( arr[mid] > tar ) {
b       j = mid-1;
    } else {
c       i = mid+1;
    }
}
return false;
```