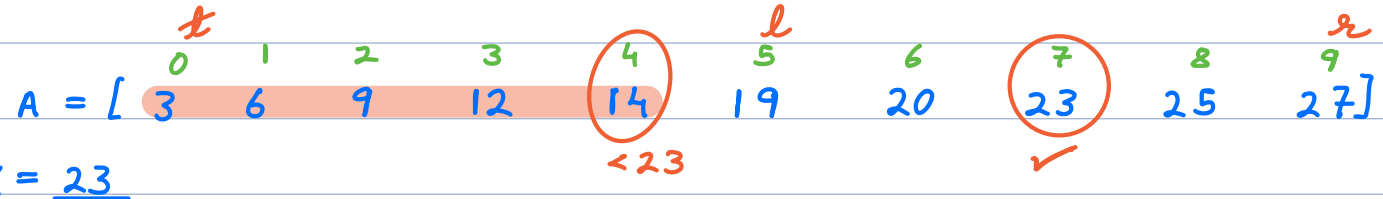
Target + Search Space → Searching

Find book in library ✓
Find word in → Newspaper ✗
→ Dictionary ✓
> organised

---

## Binary Search

Q → Search for element K in a **sorted** array with **distinct elements**. If not present return -1.




K = 23

Mid → checking this will discard half elements.

Linear Search → TC = O(N)

SC = O(1)

Key 3 Steps → 1) Define search space
2) check if mid is target
3) Decide whether to go left or right

```
l = 0        r = N-1
while (l <= r) {
    mid = (l+r)/2   // l + (r-l)/2
    if (A[mid] == target)   return mid
    if (A[mid] > target)    r = mid - 1
    else                    l = mid + 1
```

l = 5  r = 15

(l+r)/2 = 10 ✓

l/2 = 2, r/2 = 7 } 9 ✗

5 + (15-5)/2 = 10 ✓

}

search space → $N \to \frac{N}{2} \to \frac{N}{4} \cdots$

return -1

$TC = O(\log_2(N))$

$SC = O(1)$

---

Q → Find first occurence of K in sorted array.

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| A = [ | 3 | 3 | 6 | 12 | 12 | 12 | 19 | 23 | 23 | 23 ] |

K = 23

Ans

```
// Define search space
l = 0        r = N-1
while (l <= r) {
    mid = (l+r)/2
// check if mid is answer
    if (A[mid] == target &&
        (mid == 0 || A[mid-1] < target))
            return mid
// Define whether to go left or right
    if (A[mid] < target)    l = mid + 1
    else                    r = mid - 1
}
return -1
```

$TC = O(\log(N))$ $SC = O(1)$

---

Q → Given an integer array where every element occurs twice except of 1 element, find that unique element. Duplicate elements are adjacent to each other (may not be sorted).

not equal

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| A = [ | 8 | 8 | 2 | 2 | 6 | 5 | 5 ] |

Ans = 6

Sol → Ans = $\forall i \; \hat{} A[i]$ (XOR of all elements)

TC = O(N) SC = O(1)

optimize → Binary Search

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| A = [ | 8 | 8 | 2 | 2 | 6 | 5 | 5 ] |

ever-odd | ever-odd | odd-ever

(A[mid] == A[mid-1]) && (mid % 2 == 1) ⇒ go right

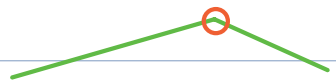
```
// Define search space
l = 0          r = N-1
while (l <= r) {
    mid = (l+r)/2
// check if mid is answer
    if ((mid == 0 || A[mid] != A[mid-1]) &&
        (mid == N-1 || A[mid] != A[mid+1]))
            return A[mid]

// Decide whether to go left or right
    if (mid > 0 && A[mid-1] == A[mid]) {// mid-1, mid
        if (mid % 2 == 1)    l = mid+1
        else                 r = mid-1
    } else {                 // mid, mid+1
```

```
        if (mid % 2 == 0)    l = mid + 1
        else                 r = mid - 1
    }
}
```

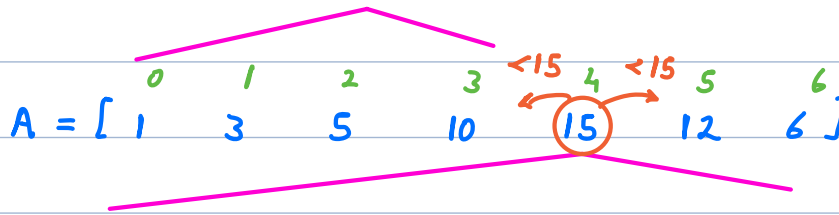
TC = O(log (N))  SC = O(1)

---

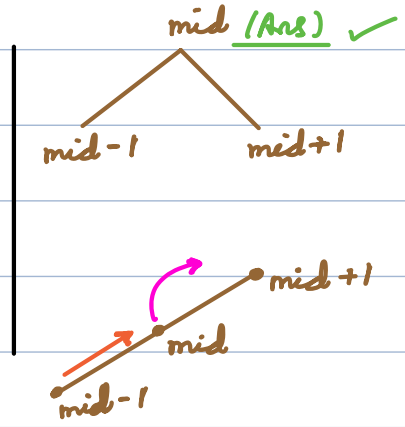
Q → Given an increasing-decreasing array.
Find **max element** in the array.



A = [ 1 (0) 3 (1) 5 (2) 2 (3) ]  Ans = 5




A = [ 1 3 5 10 15 12 6 ]  Ans = 15




```
// Search Space
l = 0          r = N-1
while (l <= r) {
    mid = (l + r) / 2
    // check mid
    if ((mid == 0 || A[mid] > A[mid-1]) &&
        (mid == N-1 || A[mid] > A[mid+1]))
            return A[mid]
    // Go left / right
    if (mid > 0 && A[mid] > A[mid-1])
            l = mid + 1
    else    r = mid - 1
}
```

TC = O(log (N))  SC = O(1)

---

Q → Given an integer array with distinct elements, find any one local minima in the array.

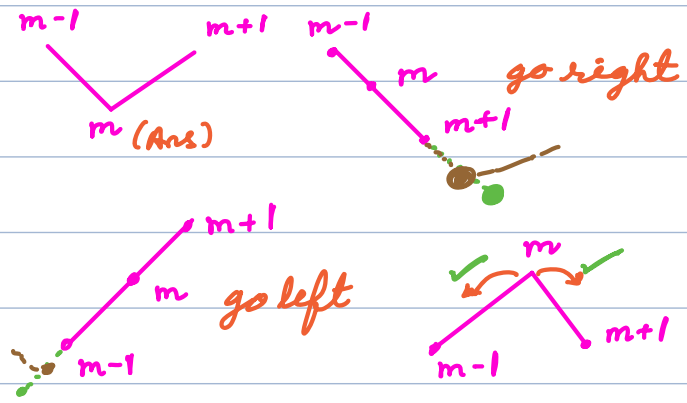
$A[i-1] > A[i] < A[i+1]$

$A = [\underset{0}{3} \quad \underset{1}{6} \quad \underset{2}{1} \quad \underset{3}{0} \quad \underset{4}{9}]$

Linear Search → TC = O(N)

SC = O(1)

$A = [\underset{0}{10} \quad \underset{1}{8} \quad \underset{2}{5} \quad \underset{3}{1}]$




```
// Search Space
l = 0          r = N-1
while (l <= r) {
    mid = (l + r) / 2
    // check mid
    if ((mid == 0 || A[mid] < A[mid-1]) &&
        (mid == N-1 || A[mid] < A[mid+1]))
            return A[mid]
    // Decide go left / right
    if (mid > 0 && A[mid] < A[mid-1])
            l = mid + 1
    else    r = mid - 1
}
```


TC = O(log(N))  SC = O(1)