Searching → Linear Search TC = O(N) ✓

→ Organised → Binary Search

Search Space
Target
} ✓

Binary Search

search space

	0	1	2	3	4	5	6	7	8	9
A = [	3	6	9	12	14	19	20	23	25	27]

9 ← x (20 > 12)

(sorted)

target = 20

start from mid ⇒ search space is half.

	0	1	2	3	4	5	6	7	8	9
A = [	3	6	9	12	14	19	20	23	25	27]

l, m, r

target = 20 20 > 14 20 > 19 20 < 23

```
l = 0        r = N-1
while (l <= r) {
    m = (l+r)/2    // l + (r-l)/2
    if (A[m] == target) return m
    if (A[m] < target)  l = m+1
    else                r = m-1
}
return -1  // Not found
```

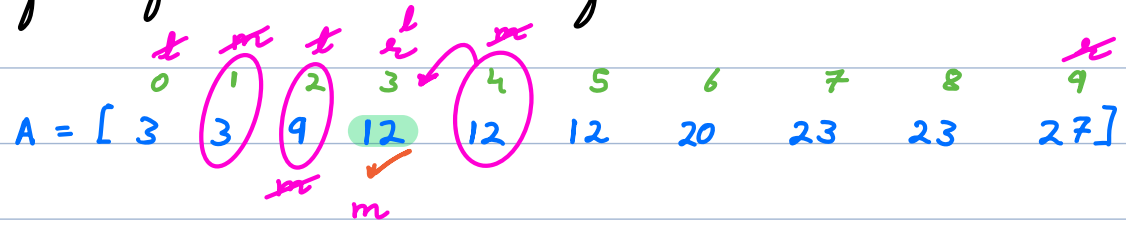
$N \to \frac{N}{2} \to \frac{N}{4} \dots \frac{N}{2^K} = 1$

$\Rightarrow K = \log_2(N)$

TC = O(log(N))

SC = O(1)

Q → (Find first mail of particular date)
Given a sorted integer array (with duplicates), find first occurrence of a number.



target = 12

Ans = 3

3 steps →
1) Define search space
2) Check if mid is answer
3) Decide going left/right

HW → Find last occurrence.

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

TC = O(log(N))

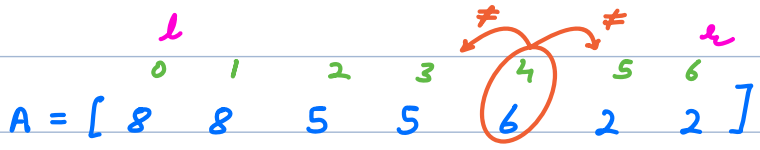
SC = O(1)

Q → Given an integer array where every element occurs twice except for 1 element, find that unique element. Duplicate elements are adjacent to each other.

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

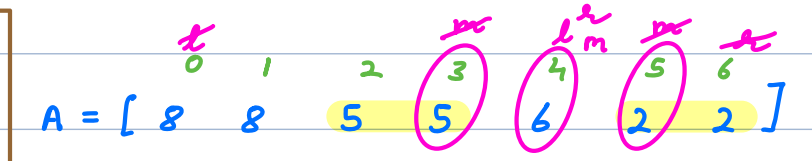
unsorted
target → condition

Ans = $\forall i \; \hat{} A[i]$ (XOR) TC = O(N) SC = O(1)



3 steps →

1) Define search space ✓
2) Check if mid is answer ✓
3) Decide going left/right ←



(m-1, m) / (m, m+1)

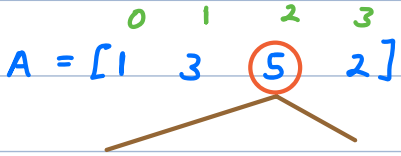
0,1 2,3 4,5

l = 0 r = N-1

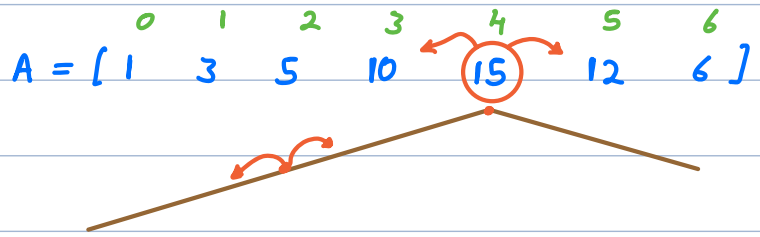
```
while (l <= r) {
    m = (l + r) / 2
    if ((m == 0 || A[m] != A[m-1]) &&
        (m == N-1 || A[m] != A[m+1]))   ✓
            return A[m]
    if (m != 0 && A[m] == A[m-1]) {   // m-1, m   ✓
        if (m % 2 == 0)   r = m-2   // odd, even
        else              l = m+1   // even, odd
    } else {   // m, m+1   ✓
        if (m % 2 == 1)   r = m-1   // odd, even
        else              l = m+2   // even, odd
    }
}
```

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

Q → Given an increasing decreasing array with distinct elements, find max element.



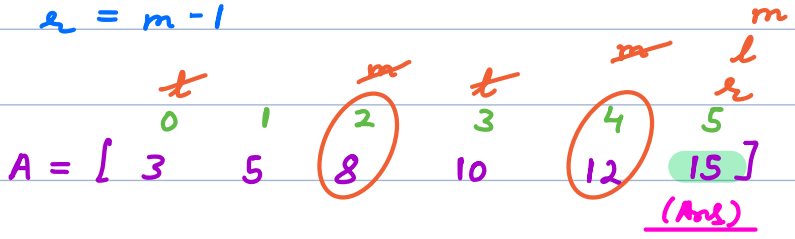
Ans = 5



Ans = 15

```
l = 0        r = N-1
while (l <= r) {
    m = (l+r) / 2
    if ((m == 0 || A[m] > A[m-1]) &&
        (m == N-1 || A[m] > A[m+1]))
            return A[m]
    if (m != 0 && A[m] > A[m-1])
            l = m+1
    else    r = m-1
}
```

(m-1) ← m → go right



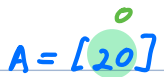
TC = O(log (N)) SC = O(1)

Q → (Google) Given an array with distinct elements.
Find any one local minima.

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

Ans = 0 / 8

Ans = 5



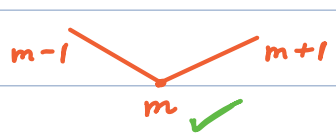
Ans = 20

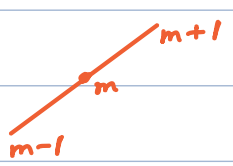
Bruteforce → TC = O(N) SC = O(1)

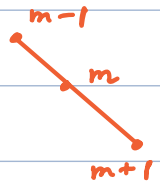
Binary Search

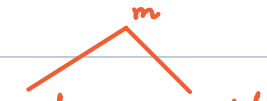
3 steps →

1) Define search space
2) check if mid is answer
3) Decide going left/right

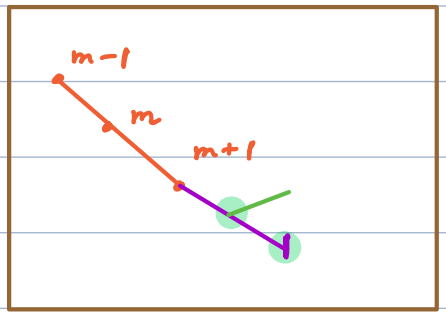




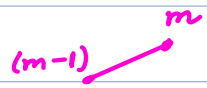




both side




```
l = 0        r = N-1
while ( l <= r ) {
    m = (l + r) / 2
    if ((m == 0 || A[m] < A[m-1]) &&
        (m == N-1 || A[m] < A[m+1]))
            return A[m]
    if ( m != 0 && A[m] > A[m-1] )
            r = m - 1
    else    l = m + 1
}
```



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