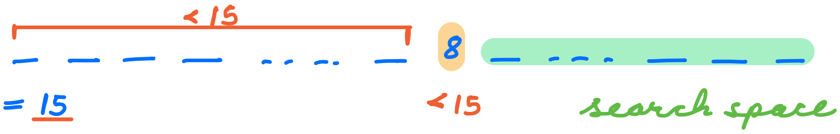
Ascending Order

target = 15



Binary Search

Q→ Given a sorted array of distinct elements, find index of a given target.

	~~l~~							~~l~~		l r				~~r~~
	0	1	2	3	4	5	6	7	8	9	10	11	12	13
A = [	1	3	5	7	9	10	11	13	15	17	19	30	35	40]

target = 17

N → N/2 → N/4 . . .

$\# steps = \log_2(N)$

l	r	mid	
0	13	6	A[6] < 17
7	13	10	A[10] > 17
7	9	8	A[8] < 17
9	9	9	A[9] == 17 ✓

l —— r

Ans = 9 (index)

```
l = 0    r = N-1          // Define Search Space
while (l <= r) {
    mid = (l+r)/2         // l + (r-l)/2
    if (A[mid] == target) // check if mid is answer
        return mid
    if (A[mid] < target)  // Decide when to go left/right
        l = mid + 1
    else
        r = mid - 1
}
return -1
```

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

$SC = O(1)$

Q→ Given a sorted array of elements, find first index of a given target.

	0	1	2	3	4	5	6	7	8	9	10	11
A = [	2	2	5	5	5	5	8	10	10	13	13	13]

target = 5   target = 8

Sol 1 → Find any location of target & iterate on left side to find first index. ✓

[2 2 2 2 . . . . 2 . . . 2 . . . 2 . . . 2 2 2]

target = 2

TC = O(N)   SC = O(1)   [L R] → R-L+1 ✓

Sol 2 →

	l				r							r
	0	1	2	3	4	5	6	7	8	9	10	11
A = [	2	2	5	5	5	5	8	10	10	13	13	13]

target = 5

l	r	mid	
0	11	5	A[mid] == 5 but A[mid] == A[mid-1]
0	4	2 ✓	A[mid] == 5 & A[mid] != A[mid-1]

```
l = 0      r = N-1                       // Define search space
while (l <= r) {
    mid = (l+r) / 2
    if (A[mid] == target &&              // check if mid is answer
        (mid == 0 || A[mid] != A[mid-1]))
            return mid
    if (A[mid] < target)                 // Decide when to go left/right
            l = mid + 1
    else // >=                           TC = O(log_2 (N))
            r = mid - 1                  SC = O(1)
}
```

H.W → last index of target.   _F_ && _X_   _T_ || _X_

Q→ Given a sorted integer array where every element appears twice except for one element, find that unique element. (No target)

A = [2 2 5 5 8 10 10 13 13 18 18]

Ans

Sol1 → $Ans = \forall_{i} \wedge A[i]$ ✓

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

Sol2 → Linear Search ✓

∀i check if A[i] != A[i-1] && A[i] != A[i+1]

```
l = 0          r = N-1          // Define Search Space
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]))      A = [2✓ 5 5 10 10]
            return A[mid]                        A = [2 2 5 5 10✓]
    // Decide when to go left/right              A = [5] → Ans = 5
✓→  if ( mid != 0 && A[mid] == A[mid-1]) {
    ✗→ if( mid % 2 == 0)                         mid-1 , mid
            r = mid - 1                          odd    even
    ✓ else                                       even   odd
            l = mid + 1
    } else {                                     mid , mid+1
        if ( mid % 2 == 0)                       even  odd
            l = mid + 1 ✓
        else                                     odd   even
            r = mid - 1
    }
}
return -1
```

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

SC = O(1)

	0	1	2	3	4 ✓	5	6	7	8	9	10
A =	[2	2	5	5	8	10	10	13	13	18	18]

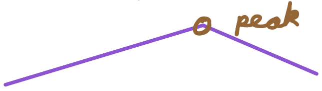
even - odd index →   ← odd - even index

	0	1	2	3	4	5	6	7	8	9	10
	l			l	r						r
A =	[2	2	5	5	8	10	10	13	13	18	18]
				mid							

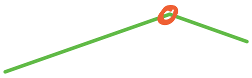
l	r	mid	
0	10	5	odd - even
0	4	2	even-odd
3	4	3	even-odd
4	4	4	✓

10:40 PM

Q → Given an increasing - decreasing array with distinct elements. Find max element.

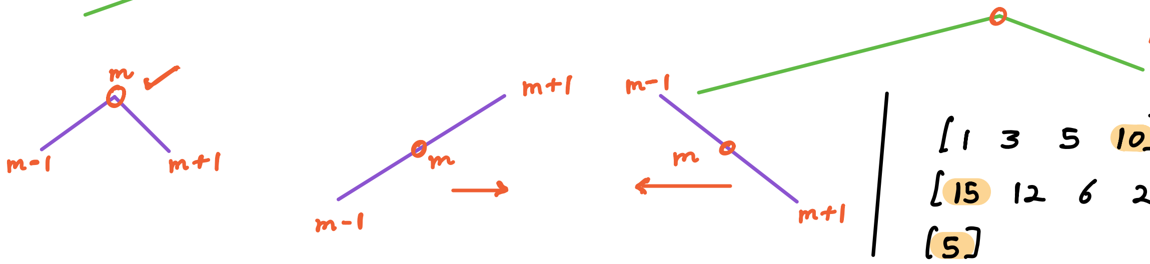


A = [1 3 5 2]



Ans = 5

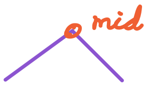
A = [1 3 5 10 15 12 6]



Ans = 15

[1 3 5 10]
[15 12 6 2]
[5]

```
l = 0          r = N - 1                  // Define Search Space
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 when to go left/right
    if ( mid == 0 || A[mid] > A[mid-1]) {
```

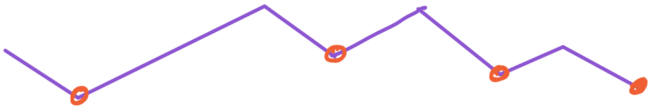


```
            l = mid + 1
    } else {
            r = mid - 1
    }
}
return -1
```

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

$SC = O(1)$

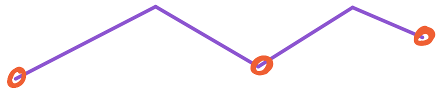
Q → Given a random array with distinct elements,
Google   find any one local minima in the array.



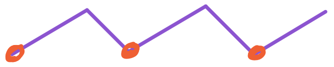
$A[i-1] > A[i] < A[i+1]$

$$A = \begin{array}{ccccc} & 0 & 1 & 2 & 3 \\ [& 5 & 2 & 1 & 6\,] \end{array}$$

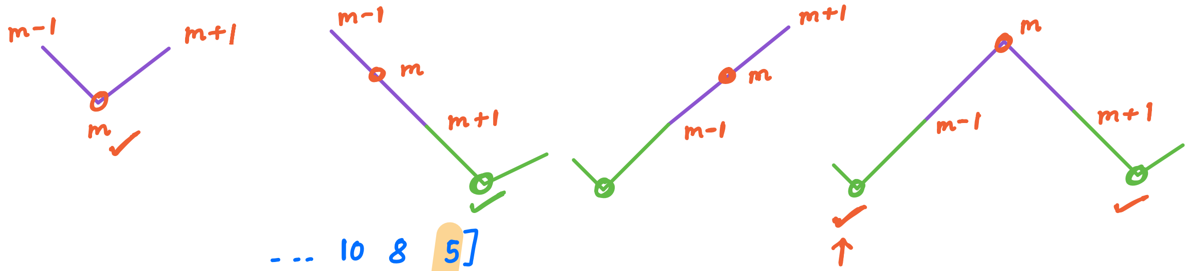
$$A = \begin{array}{cccccccc} & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ [& 1 & 3 & 8 & 5 & 2 & 6 & 4\,] \end{array}$$



$$A = \begin{array}{ccccccc} & 0 & 1 & 2 & 3 & 4 & 5 \\ [& 1 & 2 & 1 & 2 & 1 & 2\,] \end{array}$$



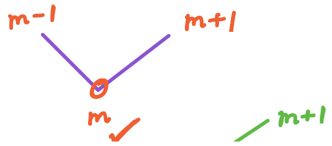
Sol 1 → Ans = smallest element → $TC = O(N)$ $SC = O(1)$



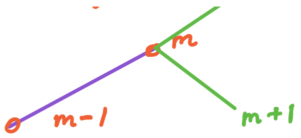
... 10 8 5]

... 10 8 5 7 ..]

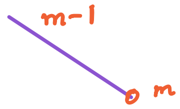
```
l = 0        r = N-1                    // Define Search Space
while (l <= r) {
    mid = (l + r) / 2                   // check if mid is answer
    if ((mid == 0 || A[mid] < A[mid-1]) &&
        (mid == N-1 || A[mid] < A[mid+1]))
```



```
            return A[mid]
    if ( mid != 0 && A[mid] > A[mid-1]) {
            r = mid - 1
    } else {
            l = mid + 1
    }
}
return -1
```



// Decide when to go left/right



[1 3 6 . . .]

[10 6 4 . . .]

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

SC = $O(1)$

$$l + \frac{r-l}{2} = \frac{2l}{2} + \frac{r-l}{2} = \frac{2l+r-l}{2} = \frac{l+r}{2}$$