* Where can we apply the Binary Search ?

⇒ Binary Search can be applied when we can come up with a logic of discarding half of the search space in every iteration.

Q. Given an Array of size N of distinct elements find any local minima in Array.

Google

↳ A number smaller than both of its neighbours.

$A[i-1] > A[i] < A[i+1] \Rightarrow$ A[i] is a local minima.

$$A: \{ \overset{0}{3}, \overset{1}{6}, \overset{2}{1}, \overset{3}{0}, \overset{4}{9}, \overset{5}{15}, \overset{6}{8} \} \Rightarrow 3, 0, 8.$$

$3 < 6$ ✓

$3 < 6 > 1$ ✗

$6 > 1 > 0$ ✗

$1 > 0 < 9$ ✓

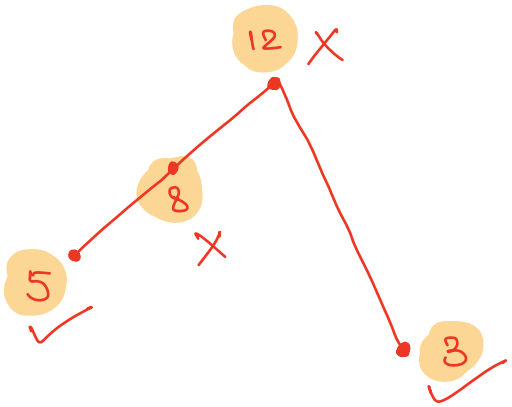
$0 < 9 < 15$ ✗

$9 < 15 > 8$ ✗

$15 > 8$ ✓

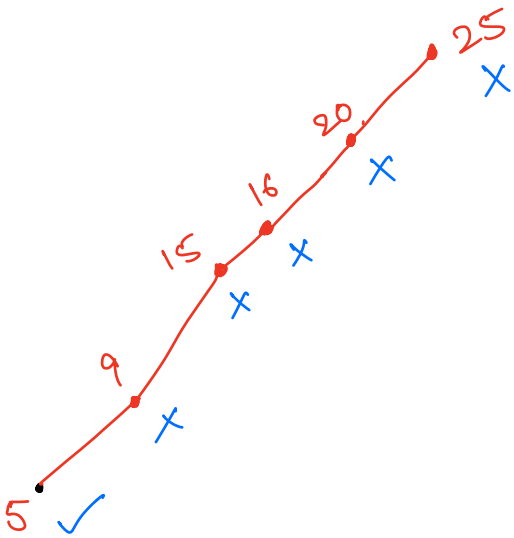
Quiz

A : [5, 8, 12, 3]



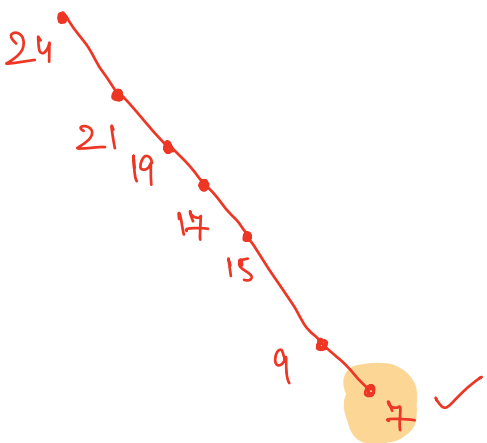
Quiz

A : [5, 9, 15, 16, 20, 25]



Quiz

A : [24, 21, 19, 17, 15, 9, 7]



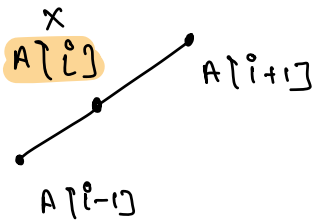
Quiz Will there be always a local minima?

$\Rightarrow$ YES.

For index (i) to be a local minima :-

$$\underset{\text{(left)}}{A[i-1]} > A[i] < \underset{\text{(right)}}{A[i+1]}$$

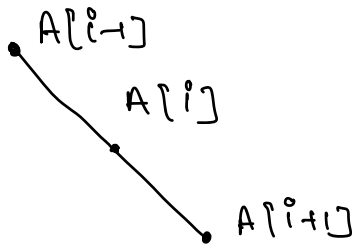
1) $A[i-1] < A[i] < A[i+1]$



$\Rightarrow$ local minima will be present on the left side for sure.

$\Rightarrow$ move to left.

2) $A[i-1] > A[i] > A[i+1]$



$\Rightarrow$ Move to right.

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

$\Rightarrow$ Move to any direction

Moving towards the smaller neighbour gives the local minima.

```
int localMinima( int A[], int N){
    if( A[0] < A[1])
        return A[0];
    if ( A[N-1] < A[N-2])
        return A[N-1];
    l = 1, r = N-2;
    while( l <= r) {
        mid = (l+r)/2;
        if( A[mid] < A[mid+1]
            && A[mid] < A[mid-1])
            return A[mid];
        else if ( A[mid-1] < A[mid]) {
            // move to left.
            r = mid-1;
        }
        else {
            // move to right.
            l = mid+1;
        }
    }
}
```

Ex :-

	0	1	2	3	4	5	6	7
A :	9	8	2	7	6	4	1	5

l	r	mid	A[mid]	A[mid+1]	A[mid-1]
1	6	3	7	6	2
1	2	1	8	2	9
2	2	2	2	7	8

$7 > 2 < 8$

$\Downarrow$

local minima.

Q.2 Every element appears twice except one element. Find the single element.

* All the pairs of duplicates will always be adjacent to each other.

	0	1	2	3	4	5	6	7	8	9	10	11	12
A :	3	3	1	1	8	8	10	10	6	2	2	4	4

* Before the single element.
  - ⇒ 1st occurrence of all the pairs is present at even index.
  - ⇒ 2nd occurrence of all the pairs is present at odd index.

* After the single element.
  - ⇒ 1st occurrence of all the pairs is present at odd index.
  - ⇒ 2nd occurrence of all the pairs is present at even index.

$$\boxed{\begin{array}{l} TC : O(\log N) \\ SC : O(1) \end{array}}$$

Q.3 Given a number N, find $\underbrace{sqrt(N)}$

floor(sqrt(N))
⇒ integer part.

sqrt(10) = 3
sqrt(25) = 5
sqrt(20) = 4

```
int sqrt(N) {
    i = 1;
    ans = 0;
    while( i * i <= N) {
        ans = i;
        i++;
    }
    return ans;
}
```

$$i * i <= N$$

$$i^2 <= N$$

$$i <= \sqrt{N}$$

TC: O(sqrt(N))

find max value of i such that

$i * i <= N$

min ans $\Rightarrow 1$

max ans $\Rightarrow N$

Search space : $[1, N]$

Search for ans in the range $[1, N]$
(Using Binary Search)

* $[1, N]$, $l = 1$, $r = N$

if mid * mid > N $\Rightarrow$ move to left
$\Rightarrow r = mid - 1;$

if mid * mid < N $\Rightarrow$ move to right
$\Rightarrow$ ans = mid.
$\Rightarrow l = mid + 1$

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

Q.4 Search in sorted & rotated Array.

Google, FB, MS, Apple, LinkedIn, Flipkart, Snapdeal, Paytm, Adobe ...

A : 1, 2, 3, 4, 5, 8, 10

rotate

0	1	2	3	4	5	6
4,	5,	8,	10,	1,	2,	3

K = 2 ⇒ 5

K = 8 ⇒ 2

K = 14 ⇒ -1

Quiz

0	1	2	3	4	5	6
4,	5	6,	8,	1,	2,	3

0	1	2	3	4	5	6	7	8	9	10	11	12	13
1,	2,	3,	4,	5,	7,	8,	10,	12,	15,	17,	21,	32,	40

rotate

A[0]

0	1	2	3	4	5	6	7	8	9	10	11	12	13
4,	5,	7,	8,	10,	12,	15,	17,	21,	32,	40,	1,	2,	3

1st part | 2nd part.

* Sorted & rotated Array ⇒ Concatenation of 2 sorted arrays.

* All the elements in the 2nd part are less than the elements in 1st part
⇒ A[0] will be greater than all the elements of 2nd part.

Quiz $A[0] = 8$ & $A[i] = 4$

$A[i] < A[0]$

$\Rightarrow$ $A[i]$ will be present in 2nd part.

Quiz $A[0] = 8$ & $A[i] = 15$

$A[i] > A[0]$

$\Rightarrow$ $A[i]$ will be present in 1st part.

Generalisation :-

1) If $A[i] > A[0] \Rightarrow 1^{st}$ part

2) If $A[i] < A[0] \Rightarrow 2^{nd}$ part.

* find pivot index

0	1	2	3	4	5	6	7	8	9	10	11
10,	20,	30,	1,	2,	3,	4,	5,	6,	7,	8,	9

end index of 1st part $\leftarrow x$ $\quad y \rightarrow$ Start index of 2nd half

pivot index

A[mid] ≥ A[0]

⇒ mid is present in 1st part.

⇒ move to right side

l = mid+1

A[mid] < A[0]

⇒ mid is present in 2nd part.

⇒ pivot = mid.

⇒ move to left

⇒ r = mid−1

0	1	2	3	4	5	6	7	8	9	10	11
10,	20,	30,	1,	2,	3,	4,	5,	6	7	8	9

l = 0, r = 11, p = 3, p−1 = 2

l	r	mid	A[mid]	Pivot index	move to
0	11	5	3	5	left
0	4	2	30	5	right
3	4	3	1	3	left
3	2				

Pivot index

l > r

⇒ break

A : 5, 8, 10, 14, 18, 22, -15, -9, -6, 0, 2

(indexes: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10)

l	r	mid	A[mid]	Pivot index	move to
0	10	5	22	-1	right
6	10	8	-6	8	left
6	7	6	-15	6	left
6	5				

l > r

Break.

A : 1, 2, 3, 4. Pivot = -1

(indexes: 0, 1, 2, 3)

l	r	mid	A[mid]	Pivot index	move to
0	3	1	2	-1	right
2	3	2	3	-1	right
3	3	3	4	-1	right
4	3				

Break

* If NO updates in pivot index.
  ⇒ NO rotations.

Steps :-

1) find $\underbrace{\text{pivot index}}_{P}$ (Binary Search) ⇒ $O(\log N)$

2) if (K > A[0]) ⇒ Search in 1st part
   ⇒ BS (A, 0, P-1);

   if (K <= A[0]) ⇒ Search in 2nd part
   ⇒ BS(A, P, n-1);

$O(\log N)$

$$\boxed{\begin{array}{l} TC : O(\log N) \\ SC : O(1) \end{array}}$$

HW
* Implement this with One Binary Search call.

———— * ————