Q: Given an Array of size N (distinct elements) find any local minima in the Array.

Google

↳ A no. which is lesser than its available neighbours.

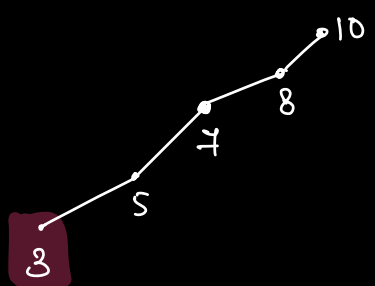
$A[i-1] > A[i] < A[i+1]$

$A[i]$ ⇑ local Minima

0	1	2	3	4	5	6
3	6	1	0	9	15	8
3<6 ✓	✗	✗	✓	✗	✗	✓

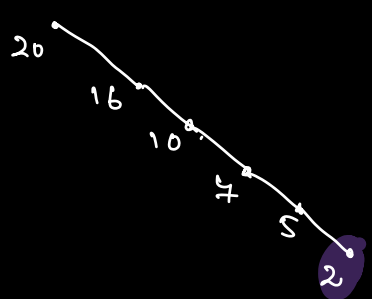
Quiz

0	1	2	3	4
3	5	7	8	10



Quiz

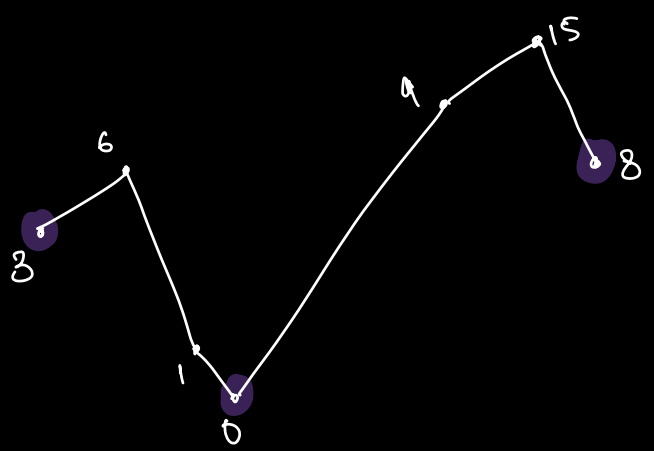
0	1	2	3	4	5
20	16	10	7	5	2



Quiz

	0	1	2	3	4	5	6
A :	24	21	19	14	15	9	7

0	1	2	3	4	5	6
3	6	1	0	9	15	8



NOTE : There will always be a local Minima.

$$T(N) = T(N/2) + T(N/2) + 1$$

$$\boxed{T(N) = 2T(N/2) + 1} \Rightarrow O(N)$$

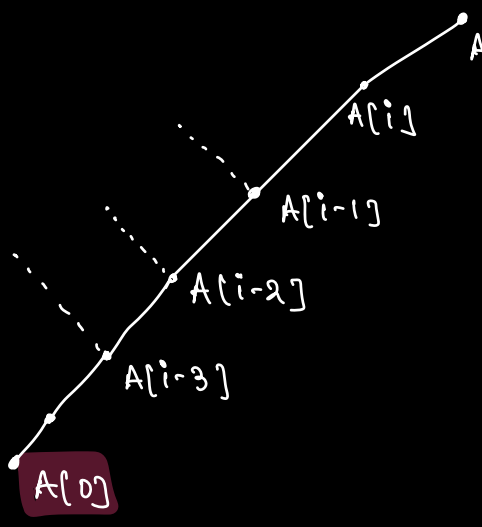
*

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

$\Downarrow$

minima

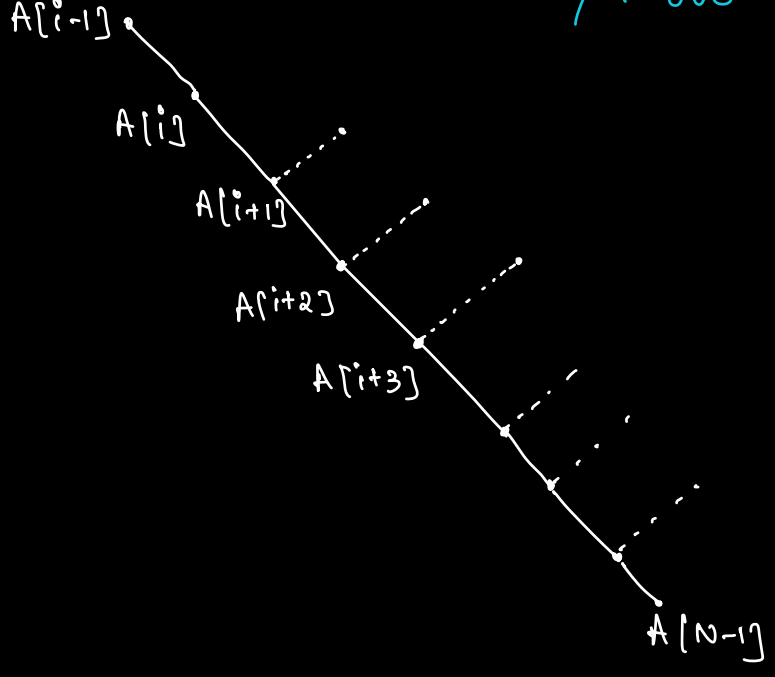
$A[i-1] < A[i] < A[i+1]$



⇒ Move to left

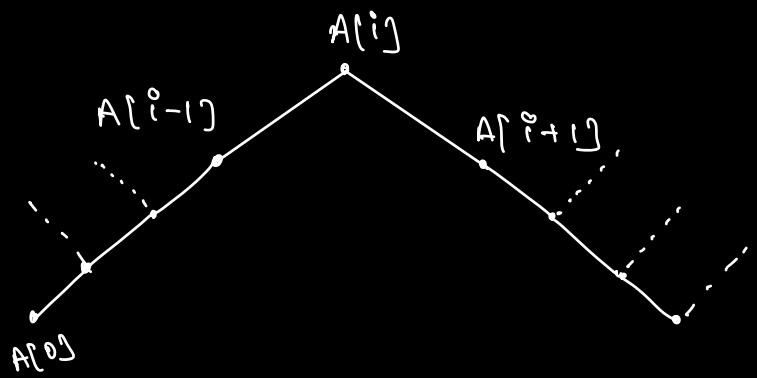
$A[i-1] > A[i] > A[i+1]$

⇒ Move to right.



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

Move to any side.




```
int localMinima( A[], 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] > A[mid-1])
            r = mid-1;
        else
            l = mid+1;
    }
}
```

TC : O(log N)

SC : O(1)

What is the condition to apply Binary Search ?

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

0	1	2	3	4	5	6	7
9	8	2	7	6	4	1	5
		↑ m					

l	r	A[mid-1]	A[mid]	A[mid+1]	Move to
1	6	2	7	6	Left
1	2	9	8	2	right
2	2	8	2	7	

↓↓

Local Minima.

Q. Given a **sorted** but **rotated** array, find the given element in it. (**Distinct elements**).

Google
Meta
Microsoft
Paytm
Atlassian
Amazon
GS | Zeta | Rapyd.

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

$k = 1 \Rightarrow 5$

$k = 7 \Rightarrow 3$

$k = 10 \Rightarrow -1$

Quiz

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

$k = 2$

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

↓

	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.

Observations.

1) Sorted & Rotated Array $\Rightarrow$ Concatenation of 2 sorted arrays.

2) All elements of 1st part > All elements of 2nd part.

3) All elements of 2nd Part < A[0]

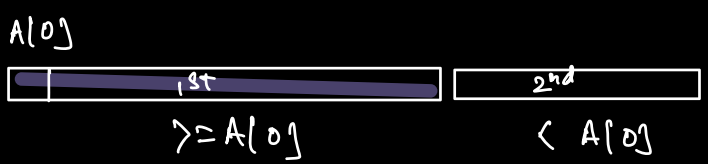
All elements of 1st Part >= A[0].

Quiz

A[0] = 8
A[i] = 4

} ⇒ 2nd Part.

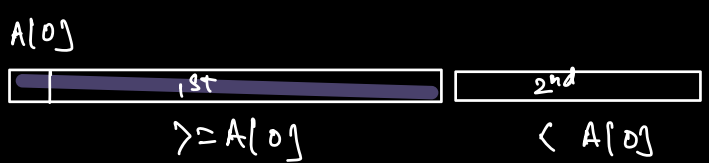
A[i] < A[0]



Quiz

A[0] = 8
A[i] = 15

} 1st Part.



A[i] < A[0] ⇒ 2nd Part

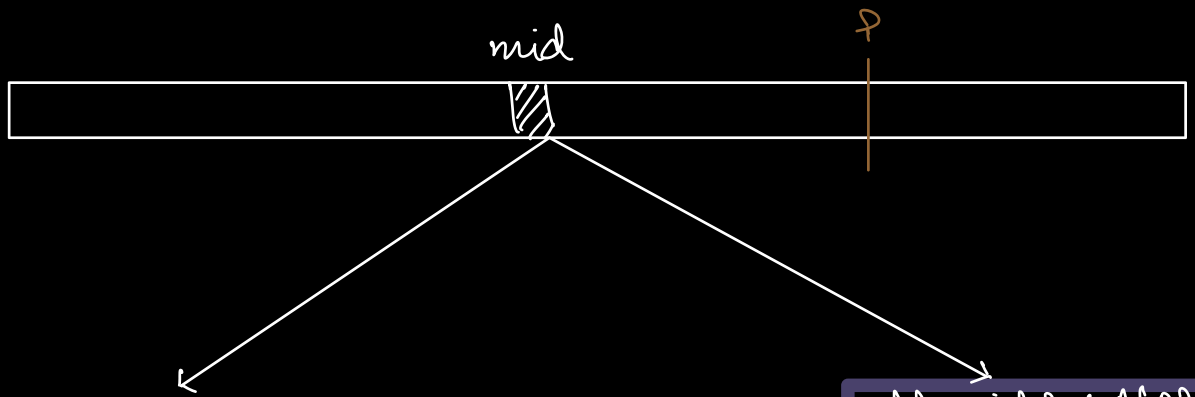
A[i] >= A[0] ⇒ 1st Part

Pivot

→ last index of the 1st Part

? → first index of the 2nd Part.

Starting index of 2nd Part



A[mid] >= A[0]

⇒ mid is present in 1st part

⇒ Move to right.

⇒ l = mid + 1

A[mid] < A[0]

⇒ mid is present in 2nd part

⇒ mid can be the pivot

Pivot = mid

⇒ Move to left

⇒ r = mid - 1

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
											l		r

l	r	mid	Pivot	Move to
0	13	6 (A[6] >= 4)	-1	Right
7	13	10 (40 >= 4)	-1	Right
11	13	12 (2 < 4) 2nd part	12	left
11	11	11 (1 < 4) 2nd part	11	left
11	10			

l > r ⇒ Break

K

if (K >= A[0])

↳ K is present in 1st part

⇒ BS(A, K, 0, Pivot-1);

else

↳ K is present in 2nd part

⇒ BS(A, K, Pivot, N-1);

TC: O(log N)

SC: O(1)

A: 1 (0) 2 (1) 3 (2) 4 (3)

l	r	mid	Pivot
0	3	1	-1
2	3	2	-1
3	3	3	-1
4	3		

4 > 3 → Break

if(Pivot == -1) ⇒ No rotation
↳ BS(A, K, 0, N-1)

HW Try to implement this in 1 iteration of B.S.

Q. Given N, find Sqrt(N).

$N = 25 \Rightarrow \sqrt{N} = 5$

$N = 15 \Rightarrow \sqrt{N} = 3$

$N = 10 \Rightarrow \sqrt{10} = 3$

$N = 20 \Rightarrow \sqrt{20} = 4$

```
i = 1
while ( i * i <= N ) {
        ans = i
        i++
}
```

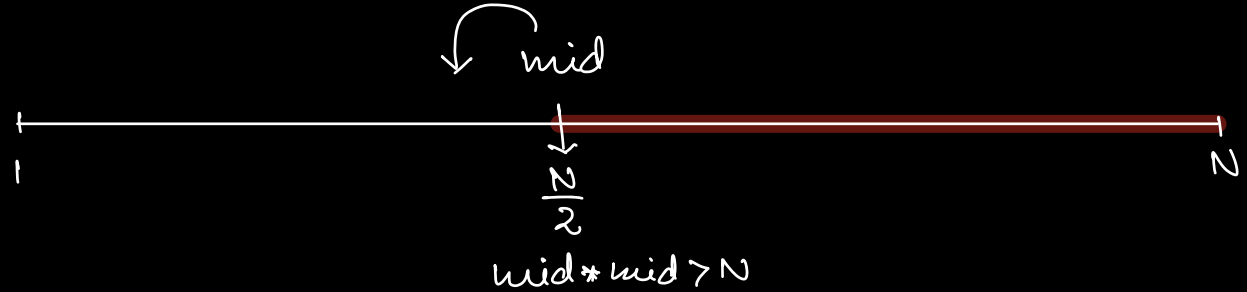
N = 36

i	i*i	ans
1	1	1
2	4	2
3	9	3
4	16	4
5	25	5
6	36	6
7	X	

TC : Sqrt(N)

$ans_{min} \longrightarrow 1$

$ans_{max} \longrightarrow N$



mid * mid <= N ⇒ ans = mid
Move to right.

Q. Given an Array of +ve no's, find the MAX length K, such that there exists NO subarray of length K with sum >= B.

Google

⇒ All subarrays of length K have sum < B.

	0	1	2	3	4	5	6	7	
A:	3	2	5	4	6	3	7	2	, B = 20

K = 5 ✗

K = 4 ✗

K = 3 ✓

⇒ All subarrays of length 3 have sum < 20.

Kmin ⟶ 0

Kmax ⟶ N

Brute force

Iterate from K = N to K = 0 & check if Ki is satisfying the **condition.**

All subarrays of length Ki have sum < B.

$O(N^2)$

linearly →

```
for (K = N; K >= 0; K--) {
    if (check(A, K, B)) {
        return K;
    }
}
```

* Write a function to check if All subarrays of length K have sum < B.

	0	1	2	3	4	5	6	7	
A:	3	2	5	4	6	3	7	2	B = 20

K = 4

PS: O(N) + O(N)

PS

SC: O(N)

Sliding Window

TC: O(N)

SC: O(1)

	0	1	2	3	4	5	6	7	
A:	3	2	5	4	6	3	7	2	N=8

1st Window

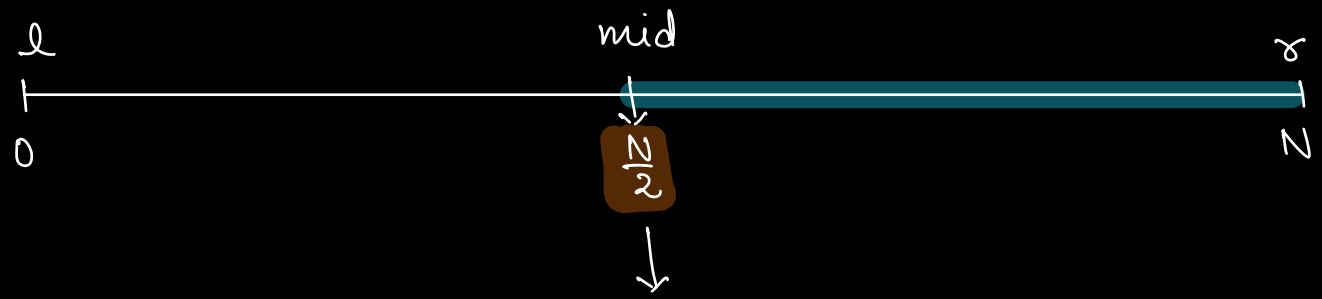
[0-3]	Sum
[1-4]	Sum - A[0] + A[4] (4 → K)
[2-5]	Sum - A[1] + A[5]
[3-6]	Sum - A[2] + A[6]
[4-7]	Sum - A[3] + A[4] (4 → N-1)

TC : O(N)

```
bool Check(A[], K, B) {
    Sum → [0, K-1]
    if(sum >= B) return false;
    // Sliding Window.
    for(i=K; i<N; i++)
        Sum += A[i]
        Sum -= A[i-K]
        if(sum >= B) return false;
    }
    return true;
}
```

Search Space ⇒ [0, N]

Target ⇒ Kmax



Check(A, mid, B)

false

⇒ Move to left

⇒ r = mid - 1

True.

→ Mid can be the ans

→ Store mid as ans, & move to right

→ ans = mid

l = mid + 1

TC: O(NLogN)

SC: O(1)

———*———