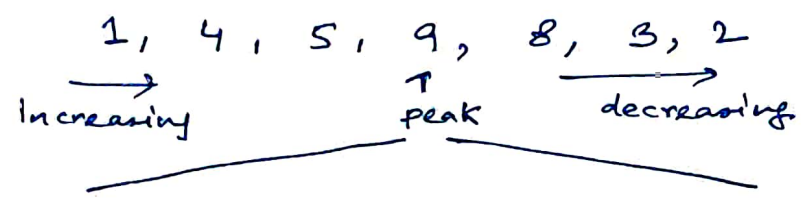
BITONIC ARRAY AND BINARY SEARCH

Bitonic means the array will be in incremental order initially, reach a peak and then elements will decrease.

Example



→ The peak element in the array if at index i will be

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

↑ it will be the ONLY element to satisfy this condition.

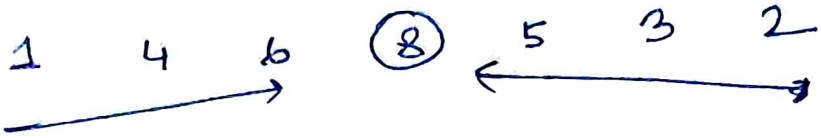
Like rotated arrays bitonic arrays can be thought of having 2 parts: one where elements increase and another where elements will decrease

So in the first half: $A[i+1] > A[i]$

In the second half: $A[i+1] < A[i]$

↑ Using this idea we can do a binary search on a bitonic array

FIND MAX (PEAK) OF BITONIC ARRAY



1. Find middle element
2. check which half does it belong to?
   - ↳ if it belongs to first half then we should move to the right
   - ↳ if it is in the second half then, move left

* how to check which half A[mid] belongs to?

↳ if A[mid+1] > A[mid] then its first half

↳ if A[mid+1] < A[mid] then its second ha

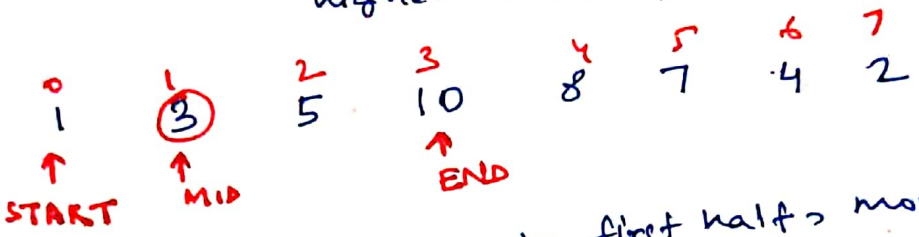
FIND MAX IN BITONIC ARRAY

A[MID] > A[MID+1]

so we are in second half

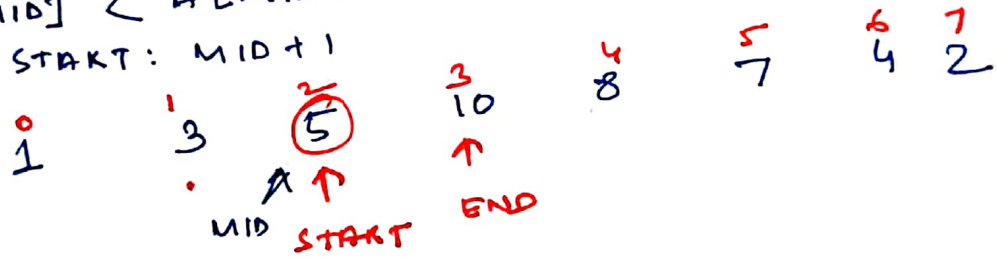
and should move left

END = MID (because MID could be the index of highest number)



A[MID] < A[MID+1], in first half, move right

START: MID + 1



A[MID] < A[MID+1]

in first half, move right

START = MID + 1

START = END, loop ends

return A[START]

SEARCHING IN A BITONIC ARRAY

Like searching in a rotated array we find the max element and then call binary search on 2 halves of the list.

Note: Since the order of elements differs in the 2 halves of the array, so we need to modify binary search function to handle this.

Example: 1 4 5 3 2 key → 3

Find index of max element

0	1	2	3	4
1	4	5	3	2

maxindex : 2

Now call binary search on left half and right half

binary-search(arr[: maxindex], key)

binary-search(arr[maxindex :], key)

How to modify the binary search to work on an ~~bitonic array~~ array that can be either be in ascending/ descending order ?

Example: Search 3 in [1, 4, 5] [1, 4] and [5, 3, 2] with binary search

→ first determine order of elements.

↳ if A[mid+1] > A[mid] :
normal binary search

↳ if A[mid+1] < A[mid]:
if we want to go higher, move to the left instead of moving right.
To go lower, move right-to-left.