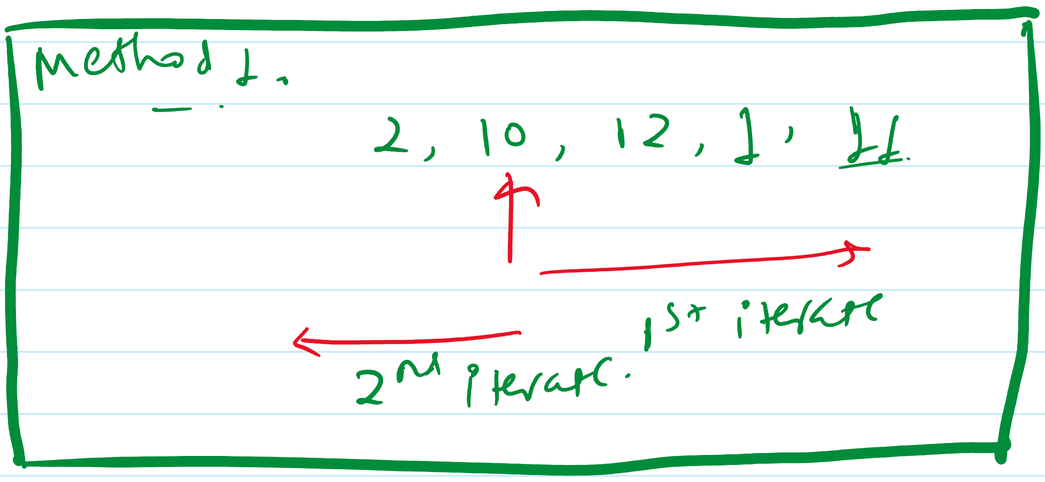
18 February 2025 13:25

# Next Greater element - II^nd

Example

arr = [2 , 10 , 12 , 1 , 11]

10 12 -1 11 12



1st half.
(i+1 → n-1)
2nd half.
(0, → i-1)
then
if found (-1)

method -2 hypothetically Double the array,
How ?

arr = [2 , 10 , 12 , 1 , 11] 2 10 12 1 11

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 2 | 10 | 12 | 1 | 11 | 2 | 10 | 12 | 1 | 11 |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

(Consider as array)

→ Don't Actually double the array but show like that,

How to Iterate for Hypothesis array

$\boxed{(i+1) \% n}$

and iterated for n time.
if don't get

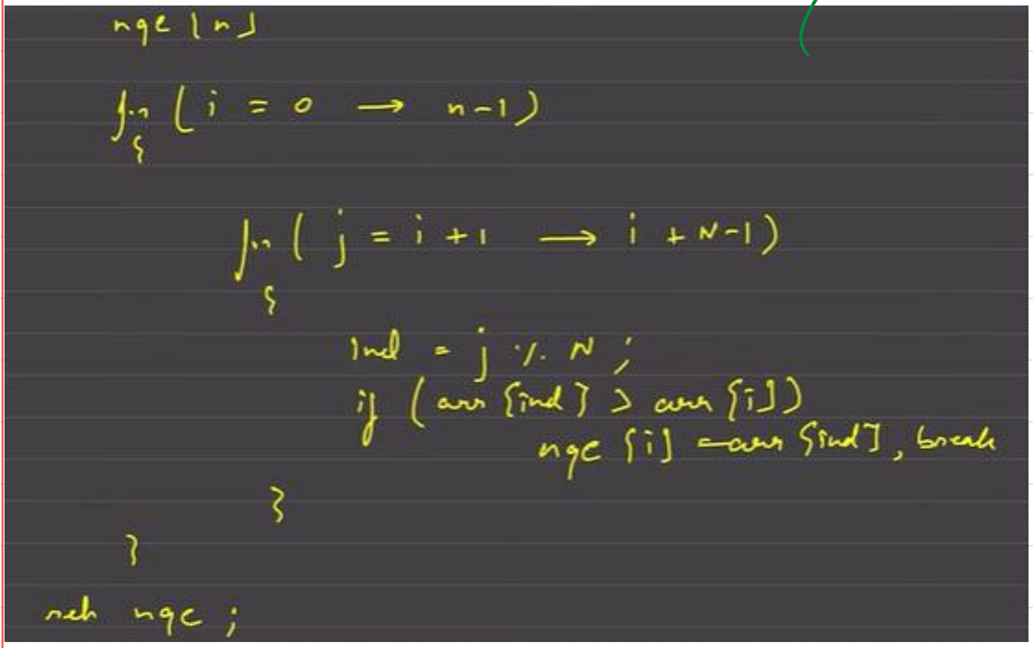
18 February 2025 14:22

```java
public int[] nextGreaterElements(int[] nums) {

    int[] ans=new int[nums.length];
    Arrays.fill(ans,-1);

    for(int i=0;i<nums.length;i++){
        int idx=(i+1)%nums.length;
        for(int j=1;j<nums.length;j++){
            if(nums[idx]>nums[i]){
                ans[i]=nums[idx];
                break;
            }
            idx=(idx+1)%nums.length;
        }
    }
    return ans;
}
```

logic is used to solve it

↳ $\boxed{\text{Concept of Hypothetic Repeating element, with having element size is 'n'}}$

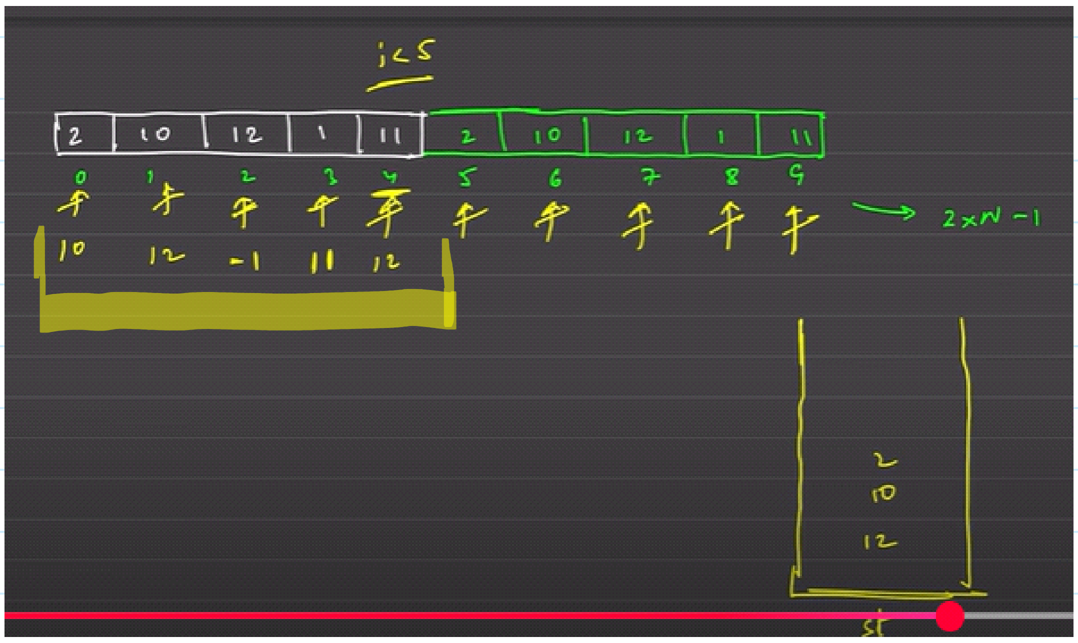
Difference only in the Balancing of idx



```
nge [n]
for (i = 0 → n-1)
{
    for ( j = i+1 → i + N-1)
    {
        Ind = j % N ;
        if (arr [ind] > arr [i])
            nge [i] = arr [ind] , break
    }
}
ret nge ;
```

TC — $O(N^2)$
SC — $O(N)$

→ $O(n^2) \longrightarrow O(N)$

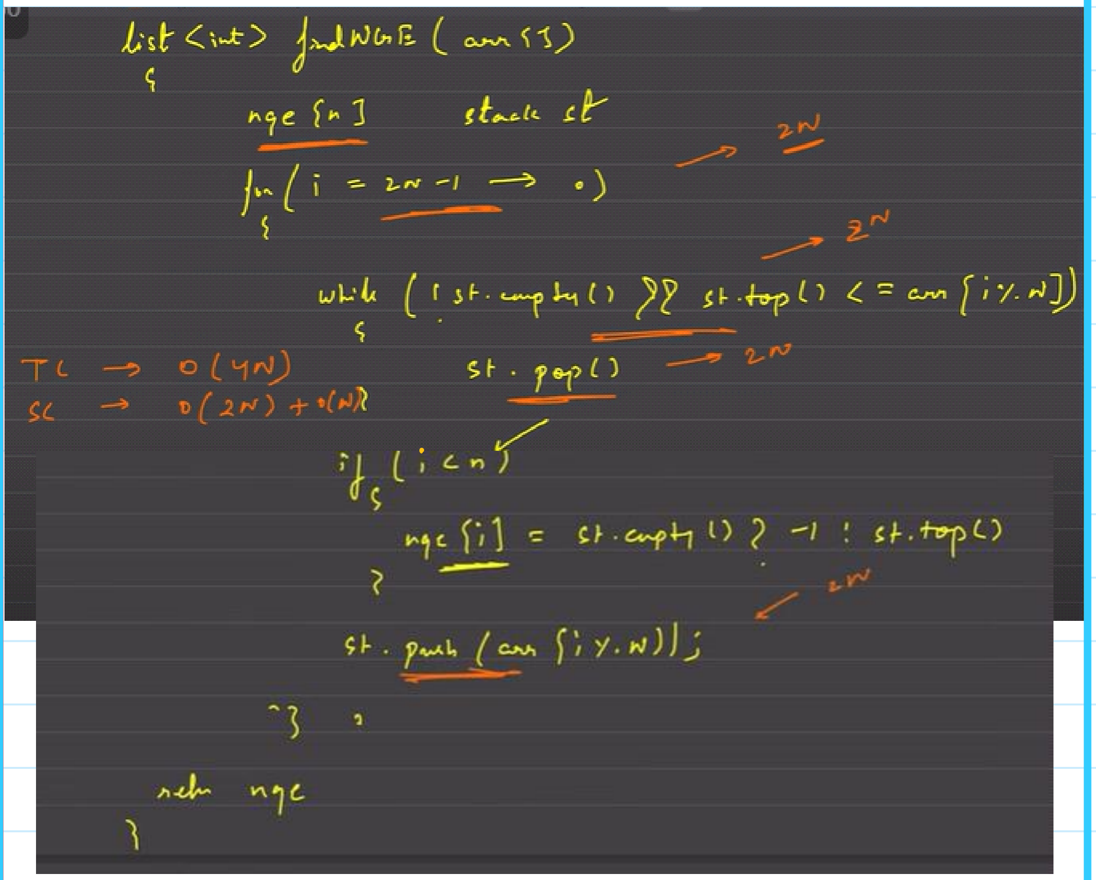


⇒ take Hypothesis element
⇒ until the element Not inside the Array, Not store the element...

→

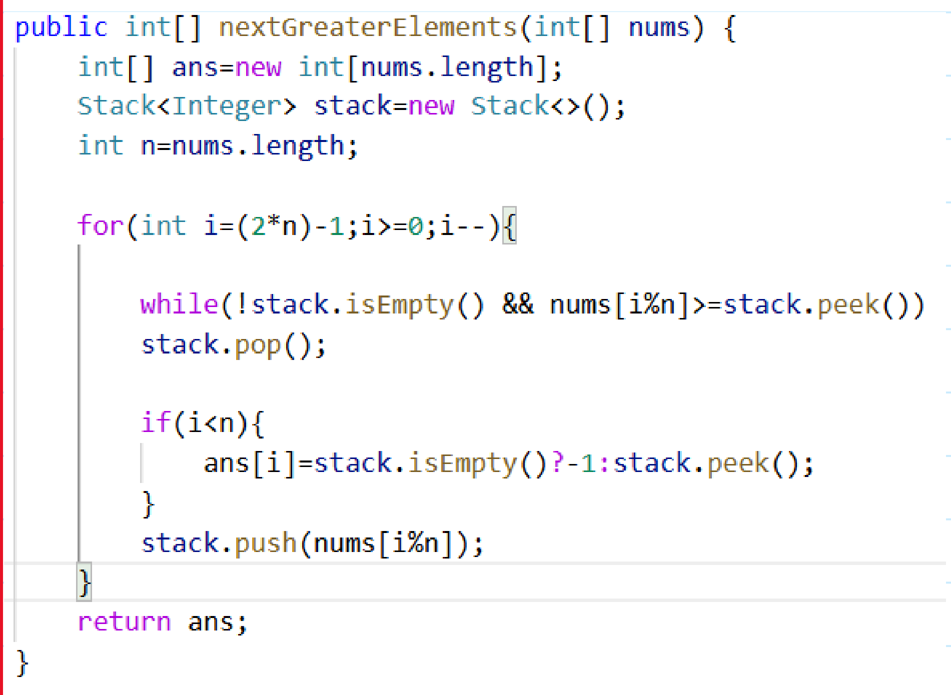
# code

18 February 2025 15:12



```
list <int> findNGE ( arr [] )
{
        nge [n]        stack st                     → 2N
        for ( i = 2N -1 → 0 )
        {                                                → 2N
            while ( ! st.empty() && st.top() <= arr [i % N] )
            {
TC → O(4N)          st.pop()    → 2N
SC → O(2N) + O(N)
            if (i < n)
            {
                nge [i] = st.empty() ? -1 : st.top()
            }                                        2N
            st.push ( arr [i % N] );
        }
    return nge
}
```

TC – O(4N)
SC – O(2N) + O(N)



```java
public int[] nextGreaterElements(int[] nums) {
    int[] ans=new int[nums.length];
    Stack<Integer> stack=new Stack<>();
    int n=nums.length;

    for(int i=(2*n)-1;i>=0;i--){

        while(!stack.isEmpty() && nums[i%n]>=stack.peek())
        stack.pop();

        if(i<n){
            ans[i]=stack.isEmpty()?-1:stack.peek();
        }
        stack.push(nums[i%n]);
    }
    return ans;
}
```