NEXT GREATER ELEMENT

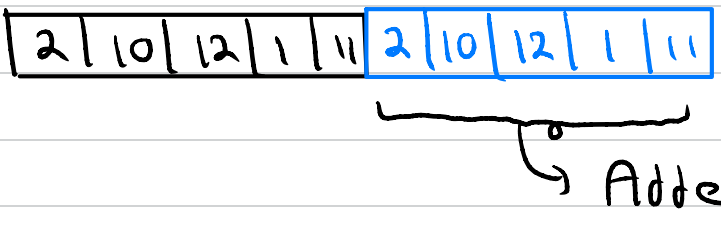
Same problem as NGE - I but has a small twist, if a greater element is not found towards the right, then we look in the circular direction (i.e look across entire array) and only return -1 if there also it is not found.

Brute force solution is to obviously have two iterations and look through the entire array for each element. But time complexity is $O(N^2)$.

Better Solution involves use of circular array as we are moving in that direction. but is still $O(N^2)$ in time complexity.

Optimal Solution involves the use of monotonic stack on circular array. We hypothetically double the array and then apply NGE process

Eg :



Time Complexity is O(2N).

Pseudocode :

```
nextGreaterElement II ( arr , N ) {
    int ans[N] ;
    stack <int> st ;
    for (int i = 2 * N - 1 ; i >= 0 ; i--) {
        if (i > N - 1) {
            int ind = i % N ;
            if (st.empty ()) {
                st.push (arr[ind]) ;
            } else {
                if (st.top() > arr[ind]) {
                    st.push (arr[ind]) ;
                } else {
                    while (!st.empty() &&
                                st.top() <= arr[ind])
                        st.pop() ;
                    }
                    st.push (arr[ind]) ;
                }
            }
        } else {
            if (st.top() > arr[ind]) {
                ans[ind] = st.top() ;
                st.push (arr[ind]) ;
            } else {
                while(!st.empty() && st.top()
                                <= arr[ind]) {
                    st.pop() ;
                }
                if(st.empty()) {
```

```
                ans[ind] = -1;
            } else {
                ans[ind] = st.top();
            }
            st.push(arr[ind]);
        }
    }
}
return ans
}
```