Date _____ Page _____

Day - 135

Stack-3

* **Next Greater Element:**

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

⇒ We have to write the nearest greater no. of the every element.

⇒ That means, our ans array will look like:

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

⇒ For brute force approach, we will ~~to~~ get the next greater element by iterating every time.

Code

```
vector <int> ans(n,-1);
for( int i= 0; i<n ; i++){
    for (int j= i+1; j<n; j++){
        if (arr[j] > arr[i] ){
            ans[i] = arr[j];
            break;
        }
    }
}
return ans;
```

T.C. → $O(n^2)$ S.C → $O(1)$

Date _____ Page _____

⇒ Our previous approach, will give TLE.

⇒ In our previous approach, we are repeating the elements many times and also coming back again & again.

⇒ So, that's why our TC increases.

⇒ ~~How~~ Now, we will use stack to find the solution.

⇒ We will find the next greatest element of every element ~~of~~ until we ~~found a~~ will find a greatest element.

⇒ So, we will do, first, push the element in the stack and then find the greatest element by traversing the ~~er~~ array.

⇒ If we will check every element with top of stack and ask it if it can be the next greatest element of the top element of stack.

⇒ If yes, then change the element with current element otherwise push the current element into the stack.

⇒ So, stack will store that elements that next greatest doesn't find yet.

⇒ Also, we don't store the element into the stack but we will store their index.

Date _____ Page _____

TLE. ⇒ Otherwise, we will don't know the index of the stack elements if we store elements.

Code

```
stack <int> st;
vector <int> ans(n, -1);
for( i=0; i<n; i++){
    if(st.empty())
        st.push(i);
    else {
        while(!st.empty && arr[st.top()] < arr[i]){
            ans[st.top()] = arr[i];
            st.pop();
        }
        st.push(i);
    }
//After writing this, we can also skip the if part & while can alone done our work.
}
return ans;
```

⇒ We can use another method.

⇒ Here, we are start filling the array from end.

⇒ Now, if there is no elements then write that element.

Date _____ Page _____

⇒ Also, init the array ans by -1 and of size same as 'arr' size.

⇒ If any element is greater than top of stack write that stack top element in the ans array.

⇒ If the element is less than the element then pop the element and again check until the condition matches.

Code

```
stack<int> st;
vector<int> ans(n, -1);
for(i=n-1; i>=0; i++){
    while(!st.empty() && arr[st.top()]<
                                arr[i]){
        st.pop();
    }
    if(!st.empty())
        ans[i] = arr[st.top()];
    st.push(i);
}
return ans;
```

* Next Smaller Element:

Date _____ Page _____

	0	1	2	3	4	5	6	7	8
arr	7	9	12	10	14	8	3	6	9
ans	3	8	10	8	8	3	-1	-1	-1

⇒ The approach will be same as before.

⇒ Here, in the previous question we are finding greatest, here we are finding smallest.

Code

```
stack<int> st;
vector<int> ans(n, -1);
for(i=0; i<n; i++){
    while(!st.empty() && arr[st.top()] > arr[i])
        ans[st.top()] = arr[i];
        st.pop();
    }
    st.push(i);
}
return ans;
```

* Smallest Number on left:

	0	1	2	3	4	5	6	7
arr	4	13	11	15	9	7	8	6
ans	-1	4	4	4	4	9	7	5

⇒ We have to find the min. element that on the left of that element.

Date _____ Page _____

⇒ We will start from the end and check the current element in the arr is less than top of stack.

⇒ If yes, write the element in the top of stack element position in the ~~arr~~ ans.

⇒ Otherwise, push that element.

Code

```
stack<int> st;
vector<int> ans(n,-1);
for(i=n-1; i>=0; i++){
    while(!st.empty() && arr[i]<arr[st.top()]){
        ans[st.top()] = arr[i];
        st.pop();
    }
    st.push();
}
return ans;
```

* Next Greater Element 2

	0	1	2	3	4	5	6
arr	6	10	7	4	8	9	4

ans 10 -1 8 8 9 -1 -1

↓ ↓

10 6

10 -1 8 8 9 10 6

Date _____ Page _____

⇒ Here, we have find the greatest element but the given array is circular array that means for -1 values we can again traverse the array & find greatest element for that.

⇒ We will create a array double size of arr with same elements.

0	1	2	3	4	5	6	7	8	9	10	11	12	13
6	10	7	4	8	9	4	6	10	7	4	8	9	4

⇒ So, how our arr will looks like this.

⇒ We will use the same previous approach to solve this problem now.

⇒ So, how how can be solve this problem without using extra space.

⇒ We can solve it by iterating the loop upto 2n-1.

* Stack Span Problem:

	0	1	2	3	4	5	6
arr	100	80	55	70	60	75	85
ans	1	1	1	2	1	4	6

⇒ Here, we have to return the no. of elements less than ~~the~~ or equal to current element.

Date _____ Page _____

⇒ This question is same as next greatest on left.

⇒ Because, we are finding the greatest element than that element.

⇒ After that count the no. of elements in between including current element.

Code

```
stack<int> st;
vector<int> ans(n, 1);
for(int i= n-1; i>=0; i++){
    while(!st.empty() &&
        price[i] > price[st.top()]){
        ans[st.top()] = st.top()-i;
        st.pop();
    }
    st.push(i);
}
while(!st.empty()){
    ans[st.top()] = st.top() +1; st.pop();
}
return ans;
```