

Nearest Smaller To Left(NSL)

For every integer in an array , you have to find the nearest smallest element to the left of the array .

Let's take an example ,

1 3 0 0 1 2 4 → -1 1 -1 -1 0 1 2

If in any case , you come across any element for which there is no smaller element on its right , then print -1.

ALGORITHM:

1. We will traverse this array from first to last element.
2. Create a stack and a vector .
3. If the stack is empty , then push -1 , in the vector and push a[i] in the stack.
4. Else if a[i] > st.top() , then push st.top() in vector.
5. If a[i] <= st.top() , while(a[i] <= st.top()) then st.pop() . if(st.size() == 0) , push -1 in vector , else push st.top() in vector .
6. Return vector.



CODE :

```
1  class
2  {
3      public:
4      vector<long long> nextLargerElement(vector<long long> arr, int n){
5          vector<long long> v; // creating a vector for storing result
6          stack<long long> s; // creating a stack for temp. hold the values from array
7          for (int i=0;i<n;i++){
8              if(s.size() ==0) // when stack size is empty there is no element in stack return output as -1;
9                  v.push_back(-1);
10             else if (s.size()>0 && s.top()<arr[i]) // when there is element in stack and stack top is smaller then array element
11             {
12                 v.push_back(s.top()); // take stack top in the result vector
13             }
14             else if (s.size()>0 && s.top()>=arr[i]) // when there is element in stack and that element is greater then equal to array element
15             {
16                 while(s.size()>0 && s.top()>=arr[i]) // upto when there is element and stack top is greater then equal to array's element delete the element from stack
17                 {
18                     s.pop(); // delete the element from stack
19                 }
20                 if (s.size()==0) // when stack became empty return -1
21                 {
22                     v.push_back(-1);
23                 }
24             }
25             else
26             {
27                 v.push_back(s.top()); // else push stack top in the vector
28             }
29             s.push(arr[i]); // push array in stack
30         }
31         return v;
32     }
33 };
34
35
36 // Time Complexity: O(N)
37 // Auxiliary Space: O(N)
```