**1. Problem Statement:**

1. You are given a number n, representing the size of array a. 2. You are given n numbers, representing elements of array a. 3. You are required to "next greater element on the right" for all elements of array 4. Input and output is handled for you. "Next greater element on the right" of an element x is defined as the first element to the right of x having value greater than x. Note -> If an element does not have any element on it's right side greater than it, consider -1 as it's "next greater element on right" e.g. for the array [2 5 9 3 1 12 6 8 7] Next greater for 2 is 5 Next greater for 5 is 9 Next greater for 9 is 12 Next greater for 3 is 12 Next greater for 1 is 12 Next greater for 12 is -1 Next greater for 6 is 8 Next greater for 8 is -1 Next greater for 7 is -1

Example:

Input: 5 5 3 8 -2 7 Output: 8 8 -1 7 -1

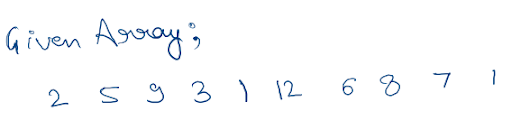
Note: If you have not tried enough to come up with logic, then we recommend you to first spend an hour or so doing it, else read only the logic used, take it as a hint and try the problem again with the same logic.

**2. Solution:**

We need to find the next greater element in the right. First of all let’s discuss the brute force approach.

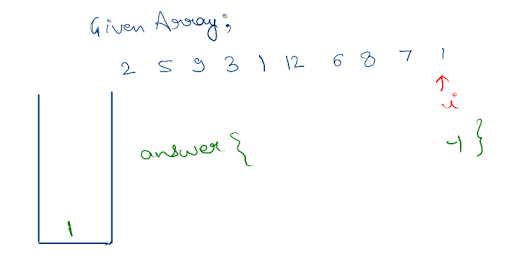
● We will iterate from the end and also we need to create an answer array. ● If we are not able to find the greater element than our current element then we will simply store -1 at that position, otherwise we will store the greater element at that place.

Now as I said earlier we are discussing the brute force approach but we need to do this in O(n) time. And for that we need to optimize our approach. And according to the above approach we will be checking every element in the right so it will take so much of time.

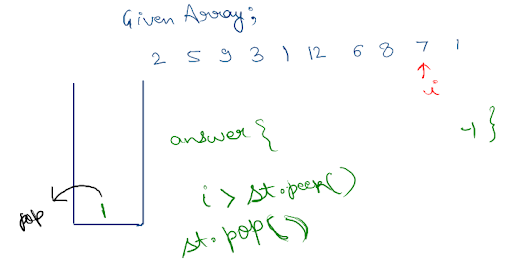
****

Let’s discuss the optimized approach:

● We will take a stack. ● We will take an answer array. ● We need to iterate from the end.

****

● Now we will check if the element present at the top of the stack is less than or equal to the current element then we will pop the element present at the top of the stack.

****

● If the stack is empty then put -1 in the answer array otherwise put the current element to the answer array.

● In the end we will simply push the current element.

Now in the end we will return the answer array and that’s it for this question.

**3. Code Implementation:**

Note: Before reading the Code, we recommend that you must try to come up with the solution on your own. Now, hoping that you have tried by yourself, here is the Java code.

ConsoleCpp

#include <bits/stdc++.h>

using namespace std;

void display(vector<int>a){

for(int i=0;i<a.size();i++)

{

cout<<a[i]<<endl;

}

}

vector<int> solve(vector<int>arr)

{

int n = arr.size();

vector<int> ngr(n,0);

ngr[n-1]=-1;

stack<int> st;

st.push(arr[n-1]);

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

while(st.size()>0&& st.top()<=arr[i]){

st.pop();

}

if(st.size()==0){

ngr[i]=-1;

}

else{

ngr[i]= st.top();

}

st.push(arr[i]);

}

return ngr;

}

int main(int argc, char \*\*argv)

{

int n;

cin>>n;

vector<int>arr(n,0);

for(int i=0;i<n;i++)

{

cin>>arr[i];

}

vector<int>nge(n,0);

nge=solve(arr);

display(nge);

return 0;

}

**4. Time Complexity:**

O(n)

**5. Space Complexity:**

O(n)

Hope that you liked the article on Next Greater Element To The Right using Stack. Subscribe to Pepcoding’s youtube channel for more such amazing content on Data Structures & Algorithms and follow the resources available for all students in the resources section of Pepcoding’s website. You can suggest any improvements to the article on our telegram channel, or on the youtube channel’s comment section.