Next Greater Element

Given an array, print the Next Greater Element (NGE) for every element.

> The **Next greater Element** for an element x is the first greater element on the right side of x in the array. Elements for which no greater element exist, consider the next greater element as -1.

Example:

```
Input: arr[] = [ 4 , 5 , 2 , 25 ]
Output:  4    -->  5
         5    -->  25
         2    -->  25
         25   -->  -1
```
Explanation: *except **25** every element has an element greater than them present on the right side*

```
Input: arr[] = [ 13 , 7, 6 , 12 ]
Output:  13    -->   -1
         7     -->   12
         6     -->   12
         12    -->   -1
```
***Explanation: 13** and **12** don't have any element greater than them present on the right side*

Dash
All
Articles
Videos
Problems
Quiz

Find Next Greater Element using Nested Loops:

> The idea is to use two loops , The outer loop picks all the elements one by one. The inner loop looks for the first greater element for the element picked by the outer loop. If a greater element is found then that element is printed as next, otherwise, -1 is printed.

Follow the steps mentioned below to implement the idea:

- Traverse The array from **index 0** to **end**.
- For each element start another loop from **index i+1** to **end.**
- If a greater element is found in the second loop then print it and **break** the loop, else print **-1.**

Below is the implementation of the above approach:

C++ Java

```
 // Simple Java program to print next
// greater elements in a given array

class Main {
    /* prints element and NGE pair for
     all elements of arr[] of size n */
    static void printNGE(int arr[], int n)
    {
        int next, i, j;
        for (i = 0; i < n; i++) {
```

Dash

All

Articles

Videos

Problems

Quiz

```
            next = -1;
            for (j = i + 1; j < n; j++) {
                if (arr[i] < arr[j]) {
                    next = arr[j];
                    break;
                }
            }
            System.out.println(arr[i] + " -- " + next);
        }
    }

    public static void main(String args[])
    {
        int arr[] = { 11, 13, 21, 3 };
        int n = arr.length;
        printNGE(arr, n);
    }
}
```

Time Complexity: O(N^2)
Auxiliary Space: O(1)

Find Next Greater Element using Stack:

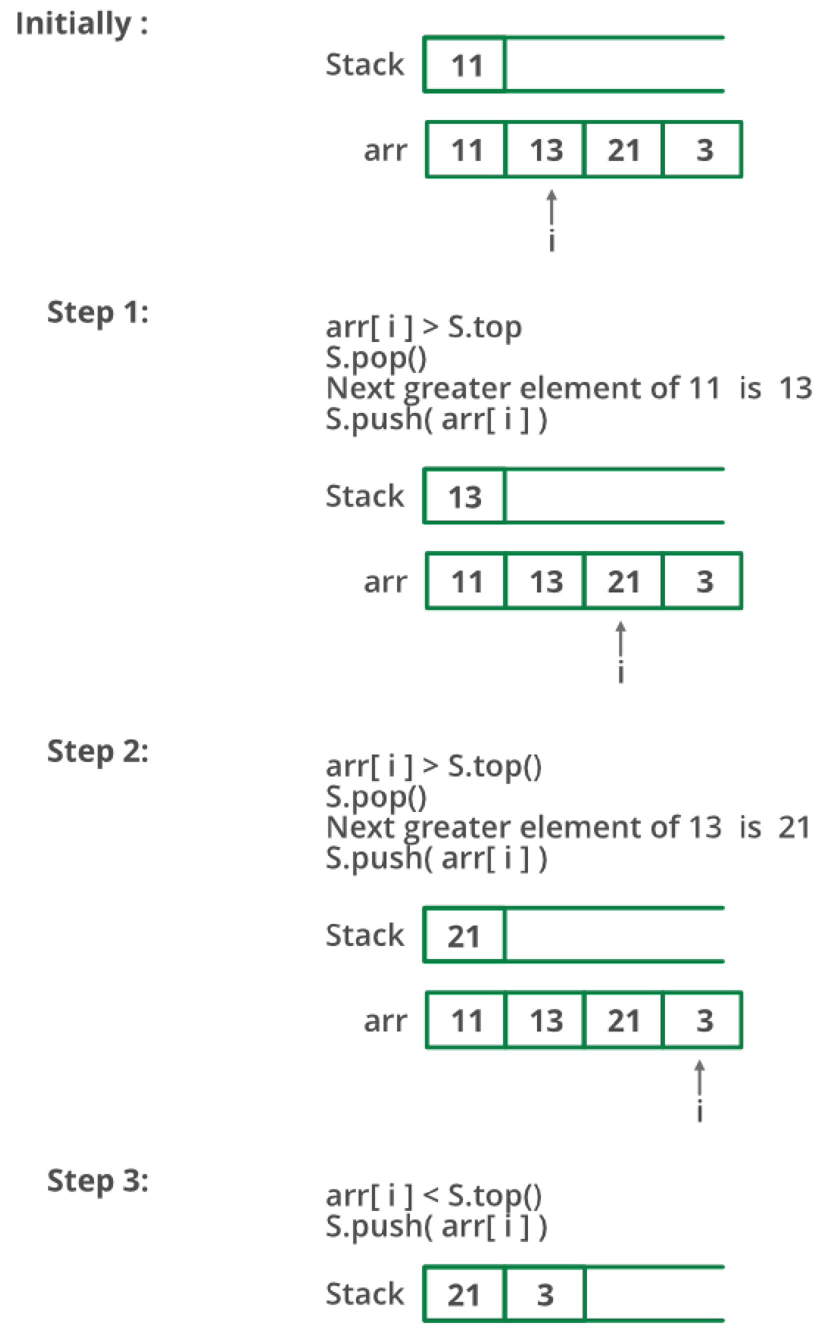
> The idea is to store the elements for which we have to find the next greater element in a stack and while traversing the array, if we find a greater element, we will pair it with the elements from the stack

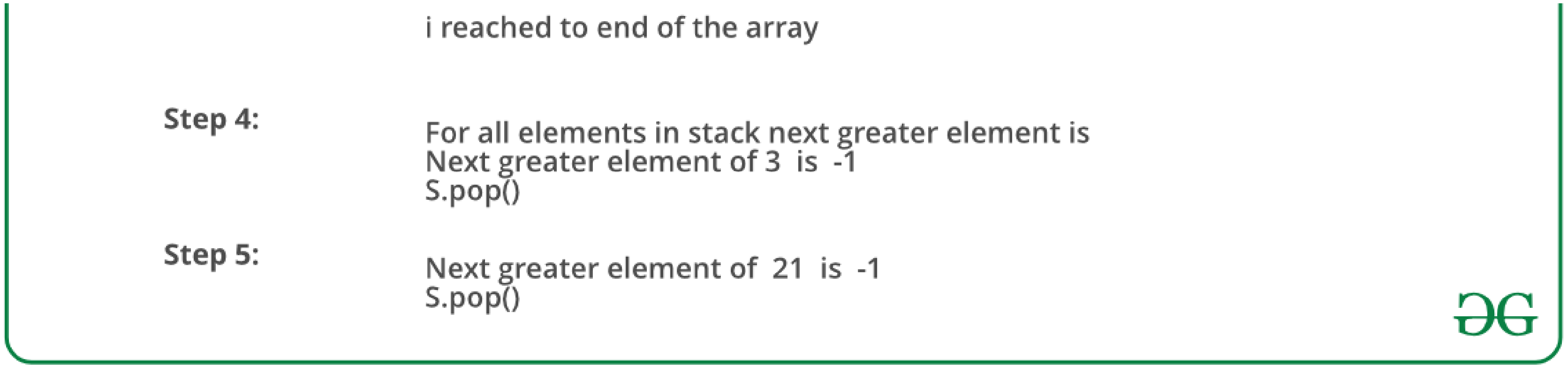
> till the top element of the stack is less than the current element.

illustration:

Below is the illustration of the above approach:

Dash
All
Articles
Videos
Problems
Quiz




i reached to end of the array

Step 4: For all elements in stack next greater element is
Next greater element of 3 is -1
S.pop()

Step 5: Next greater element of 21 is -1
S.pop()

Follow the steps mentioned below to implement the idea:

- Push the first element to stack.
- Pick the rest of the elements one by one and follow the following steps in the loop.
  - Mark the current element as **next.**
  - If the stack is not empty, compare top most element of stack with **next**.
  - If **next** is greater than the top element, Pop element from the stack. **next** is the next greater element for the popped element.
  - Keep popping from the stack while the popped element is smaller than **next. next** becomes the next greater element for all such popped elements.
- Finally, push the **next** in the stack.
- After the loop in step 2 is over, pop all the elements from the stack and print **-1** as the next element for them.

Below is the implementation of the above approach:

C++ Java

```
 // Java program to print next
// greater element using stack
```

Dash
All
Articles
Videos
Problems
Quiz

```
public class NGE {
    static class stack {
        int top;
        int items[] = new int[100];

        // Stack functions to be used by printNGE
        void push(int x)
        {
            if (top == 99) {
                System.out.println("Stack full");
            }
            else {
                items[++top] = x;
            }
        }

        int pop()
        {
            if (top == -1) {
                System.out.println("Underflow error");
                return -1;
            }
            else {
                int element = items[top];
                top--;
                return element;
            }
        }
```

Dash
All
Articles
Videos
Problems
Quiz

```
        boolean isEmpty()
        {
            return (top == -1) ? true : false;
        }
    }

    /* prints element and NGE pair for
       all elements of arr[] of size n */
    static void printNGE(int arr[], int n)
    {
        int i = 0;
        stack s = new stack();
        s.top = -1;
        int element, next;

        /* push the first element to stack */
        s.push(arr[0]);

        // iterate for rest of the elements
        for (i = 1; i < n; i++) {
            next = arr[i];

            if (s.isEmpty() == false) {

                // if stack is not empty, then
                // pop an element from stack
                element = s.pop();

                /* If the popped element is smaller than
```

Dash

All

Articles

Videos

Problems

Quiz

```
                   next, then a) print the pair b) keep
                   popping while elements are smaller and
                   stack is not empty */
                while (element < next) {
                    System.out.println(element + " --> "
                                       + next);
                    if (s.isEmpty() == true)
                        break;
                    element = s.pop();
                }

                /* If element is greater than next, then
                   push the element back */
                if (element > next)
                    s.push(element);
            }

            /* push next to stack so that we can find next
               greater for it */
            s.push(next);
        }

        /* After iterating over the loop, the remaining
           elements in stack do not have the next greater
           element, so print -1 for them */
        while (s.isEmpty() == false) {
            element = s.pop();
            next = -1;
            System.out.println(element + " -- " + next);
        }
    }

    // Driver Code
    public static void main(String[] args)
    {
        int arr[] = { 11, 13, 21, 3 };
        int n = arr.length;
        printNGE(arr, n);
    }
}
```

Time Complexity: O(N)
Auxiliary Space: O(N)

Find Next Greater Element using Map:

In this particular approach we are using the map as our main stack

> - This is same as above method but the elements are pushed and popped only once into the stack. The array is changed in place. The array elements are pushed into the stack until it finds a greatest element in the right of array. In other words the elements are popped from stack when top of the stack value is smaller in the current array element.

- Once all the elements are processed in the array but stack is not empty. The left out elements in the stack doesn't encounter any greatest element . So pop the element from stack and change it's index value as -1 in the array.

C++ Java

```
 // Java code to implement the approach

import java.util.*;

class GFG
{
    static void nextLargerElement(int[] arr, int n)
    {
        ArrayList<HashMap<String, Integer> > s = new  ArrayList<HashMap<String, Integer> >();

        // iterating over the array
        for (int i = 0; i < n; i++) {
            while (s.size() > 0
                   && s.get(s.size() - 1).get("value") < arr[i]) {
                // updating the array as per the stack top
                HashMap<String, Integer> d = s.get(s.size() - 1);
                s.remove(s.size() - 1);
                arr[d.get("ind")] = arr[i];
            }
            // pushing values to stack
            HashMap<String, Integer> e = new HashMap<String, Integer>();
```

```
                e.put("value", arr[i]);
                e.put("ind", i);
                s.add(e);
            }

            // updating the array as per the stack top
            while (s.size() > 0) {
                HashMap<String, Integer> d = s.get(s.size() - 1);
                s.remove(s.size() - 1);
                arr[d.get("ind")] = -1;
            }
        }

        // Driver Code

        public static void main(String[] args)
        {
            int[] arr = { 6, 8, 0, 1, 3 };
            int n = 5;

            // Function call
            nextLargerElement(arr, n);
            for (int i = 0; i < n; i++)
                System.out.print(arr[i] + " ");
        }
    }

// This code is contributed by phasing17
```

Dash

All

Articles

Videos

Problems

Quiz

Output

```
8 -1 1 3 -1
```

Time Complexity: O(N)
Auxiliary Space: O(N)

Mark as Read