# Program to print the next greater element in the array

Posted on August 17, 2019 | by Prashant Yadav

Posted in Algorithms, Arrays | Tagged Easy

An algorithm to print the next greater element for each element in the array.

A next greater element for a given element in the array is the first greater element on the right-hand side of the array. If no greater elements are present then print -1.

## Example

```
Input:
[4, 5, 2, 25]

Output:
4 ---> 5
5 ---> 25
2 ---> 25
25 ----> -1
```

In the above example the next greater element after `4` is `5` and for `5` is `25` and for `2` is also `25`. As there are no greater element after `25` we return `-1`;

## Method 1: Using Brute Force approach

### Implementation

- We are going to use two nested loops.
- The outer loops picks all the elements one by one and the inner loops looks for the first greater element after the element.
- If no greater elements are found then return `-1`.

```
let nextGreater = (arr, n = arr.length) => {
  for(let i = 0; i < n; i++){
    let next = -1;

    //Find the next greater element
    for(let j = i; j < n; j++){
      if(arr[i] < arr[j]){
        next = arr[j];
        break;
      }
    }

    console.log(next);
  }
}
```

```
Input:
nextGreater([11, 13, 21, 3]);

Output:
13
21
-1
-1
```

Time complexity: O(n ^ 2);

Space complexity: O(n);

The worst-case occurs when all the elements are sorted in descending order.



## Method 2: Using stack for optimization

### Implementation

- Add the first element to the stack on initialization.
- Now loop all the other elements with following steps.
    1. Mark the current element as **next**.
    2. If stack is not empty then check its top element with the **next**.

3. If next is greater than the top element, remove element from stack. **next** is the next greater element for the popped element.
4. Keep remove elements from till while elements are smaller than **next**, This way **next** becomes the next greater element for all those elements.

- At the end push the `next` in the stack.
- After the loop in step 2 is over, pop all the elements from stack and print -1 as next element for them.

Copy

```
let nextGreaterWithStack = (arr, n = arr.length) => {
  let stack = new stackUsingLL();
  let element, next;

  //push the first element in the stack
  stack.push(arr[0]);

  for(let i = 0; i < n; i++){
    next = arr[i];

    if(!stack.isEmpty()){
      element = stack.pop();

      //Print the next greater element
      while(element < next){
        console.log(next);
        if(stack.isEmpty()){
          break;
        }

        element = stack.pop();
      }

      //If next element is smaller then add it to the stack
      if(element > next){
        stack.push(element)
      }
    }

    stack.push(next);
  }

  //Print the remaining next greaters
  while(!stack.isEmpty()){
    element = stack.pop();
    next = -1;
    console.log(next);
  }
}
```

```
Input:
nextGreater([11, 13, 21, 3]);

Output:
13
21
-1
-1
```

Time complexity: O(n).

Space complexity: O(n).

## Time complexity to find the next greater element in the array using stack.

The worst-case occurs when all the elements are sorted in decreasing order.

In this case, every element is processed at most 4 times.

1. Initially pushed to the stack.
2. Removed from the stack when next element is being processed.
3. Pushed back to the stack because next element is smaller.
4. Popped from the stack in step 3 of algorithm.



## Recommended Posts:

Find the most frequent element in an array

Find height and width of binary tree

Program to print the Collatz sequence in javascript.

Longest repeated subsequence

Longest common subsequence | Print all LCS

Recursive Insertion Sort Algorithm

Reverse a string using stack

Longest Consecutive Sequence

Find the longest common prefix

Difference between square of sum of numbers and sum of square of numbers.

Prev

Next

Handcrafted with ♥somewhere in **Mumbai**