Summary

Time Complexity: O(n) where n is the number of elements in the array since we traverse through the array only once
Space Complexity: O(n) since we may have to store all n elements on the stack, in case of a descending array.

Problem Pattern

For any array, the rightmost element always has the next greater element as -1, because there is no element after it.
For an array which is sorted in decreasing order, all elements have the next greater element as -1.
For the input array {4, 5, 2, 25}, the next greater elements for each element are as follows:
4 → 5,
5 → 25
2 → 25
25 → -1
The most intuitive approach is using 2 nested loops. But is that approach optimal? Think about the time complexity of that approach. How can we do better?
Can we use a data structure for this? Consider using a stack.

Problem Approach

At any point the stack will only contain elements for which we have not found the next greater element yet. This way, after we finish traversing the array, all the elements left in the stack would have -1 as their next greater element.
Following are the steps to use stack and solve this problem:

1. If the Stack is empty (which it is initially), then push the element in it.
2. Then we take the next element from the array, compare it with the top element from the stack. If the incoming element is greater, then it is the next greater element for the top stack element. We pop this element from the stack.
3. Similarly compare the incoming element with the top element in stack and keep popping till the incoming element is greater than top element of stack.
4. When we find that the element on the stack is greater than the incoming element, we will not pop that element. We will push the incoming element onto the stack. This is because the next greater element for this element still needs to be found.

5. Then pick another element from the array and repeat the above procedure. At the end, when all the array elements have been traversed, for the remaining elements in the Stack, we can simply print -1.

Problem Pseudocode

```
void nextGreaterElement(int[] arr) {

    new Stack s

    for (i = 0 to len(arr)) {

        while (!s.isEmpty() && arr[i] > s.peek()) {

            int popped = s.pop()

            print(popped + ' → ' + arr[i])

        }

        s.push(arr[i]);

    }


    while (!s.isEmpty()) {

        int popped = s.pop()

        print(popped + ' → ' + -1)

    }


}
```

Alternate Approaches

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. The Time Complexity of this solution is higher.