Give an array, print the Next Smaller Element (NSE) for every element. The NSE for an element x is the first smaller element on the right side x in the array for elements for which no smaller element exists (on the right side) then consider NSE as -1

Input: [4,8,5,2,25]
Output: [2,5,2,-1,-1]

understanding the problem
we have an array & for each element in the array we need to find the NSE

what is NSE ??
The NSE for an element x in the array is the first element that is smaller than x & appears to its right in the array.

If no such element exists, we return -1?

Code

```
def next_small_element(arr):
    n = len(arr)
    result = [-1]*n
    stack = []

    for i in range(n-1, -1, -1):
        while stack and stack[-1] >=
            arr[i]:
            stack.pop()

        if stack:
            result[i] = stack[-1]

        stack.append(arr[i])
    return result

# Inputing.
arr = [4,8,5,2,25]
print(next_small_element(arr))
```

Execution with stack

① n = len(arr) = 5
result = [-1,-1,-1,-1,-1]
↳ initial result why?

② If an element does not have a smaller element to its right its NSE should be -1

③ Ensure correct output length

stack = []

) Initially stack is empty because to avoid unnecessary comparisons. stack helps us store only relevant elements.

If stack is not empty?!

If we initialized stack with some values, we would have to clear it first, which adds extra complexity.

for i in range(n-1, -1, -1):

traversing form right to left?

we need to find the NSE on its right instead of storing all elements we keep only the useful ones on the stack

i.e

we process elements from right to left, so that the stack always contains elements to the right of arr[i]

Before adding arr[i] to the stack, we remove elements greatest than or equal to it because they will never be useful fo NSE

what happens if we traverse left to right?

we would need to scan all remaining elements on the right every time

Ex:

At 4 we'd have to scan [8, 5, 2, 25] to find '2'

while stack and stack[-1] >= arr[i]:

    stack.pop()

Remove useless elements from stack.

stack[-1] → this checks the top element of the stack.

stack[-1] >= arr[i] → if the top element is greater than or equal to arr[i], it can not be the NSE for arr[i] or any future element so we remove (pop) it.

we keep removing elements until

stack becomes empty → NO SM - aller element found

stack top is smaller than arr[i] → this element is the NSE

Example : [4, 8, 5, 2, 25]
traversal direction : Right to left

Step i	arr[i]	Stack Before	Action taken	Stack After	NSE
1	25	[ ]	NO NSE push 25	[25]	-1
2	2	[25]	2 < 25 → pop push 2	[2]	-1
3	5	[2]	5 > 2 → NSE for 5 is 2, push 5	[2,5]	2
4	8	[2,5]	8 > 5 → NSE for 8 is 5 push 8	[2,5,8]	5
5	4	[2,5,8]	4 < 8 → pop 8, 4 < 5 → pop 5, NSE for 4 is 2 push 4	[2,4]	2

pf stack :

result[i] = stack[-1] # set NSE as the top of the stack
stack.append(arr[i]) # push the current element onto the stack

result = [-1, -1, -1, -1, -1]
Stack = [ ]

① result[4] = stack[-1]
i = 4 Stack is empty no explain
, push 25 on to stack
result = [-1, -1, -1, -1, -1]
stack: [25]

② i = 3 arr[3] = 2
result[3] = stack[-1]
while cond → 25

which is ≥ 2 so pop it
stack is now empty
so result[3] remain -1
push 2 on to stack
stack [2]
res [-1, -1, -1, -1, -1]

DOMS

i = 2

stack = [2, 5]

res = [-1, -1, 2, -1, -1]

i = 1

stack [2, 5, 8]

res = [-1, 5, 2, -1, -1]

i = 0

stack [2, 4]

res = [2, 5, 2, -1, -1]