**problem:**

> Given an array **Arr[]** of size **N**. For every element in the array, the task is to find the index of the farthest element in the array to the right which is smaller than the current element. If no such number exists then print **-1**.
> Note: 0 based indexing.

**Example:**

**Input:**

```
N=5
Arr[] = {3, 1, 5, 2, 4}
```

**Output:**

```
3 -1 4 -1 -1
```

**Explanation:**

```
Arr[3] is the farthest smallest element to the right of Arr[0].
Arr[4] is the farthest smallest element to the right of Arr[2].
And for the rest of the elements, there is no smaller element to their
right.
```

**Approach:**

⟶ First we will find prefix Minimum element For all element i.e we will store the minimum element for all `i` which will represent smallest element after `Arr[i]` on right side

For example we have array:

```
3 1 5 2 4
```

⟶ the `prefix_min` array will be

```
1 1 2 2 4
```

we will start traversing from right side and then do like this:

```
vector<int>prefix_min(N);
// For last element, the minimum after i will be the last element
itself
prefix_min[N-1] = Arr[N-1];
for(int i=N-2;i>=0;i--){
        prefix_min[i] = min(prefix_min[i+1],Arr[i]);
}
```

⟶ Now we will do binary search in the `prefix_min` array with these conditions.

1. If prefix_min[mid] is greater than Arr[i] then we will Goto Left side to find element which will be lesser than Arr[i] and if we goto left side then we will find smaller elements.
2. If prefix_min[mid] is less than Arr[i] then we will Goto right side to get more greater index because we want farthest element so we will goto right side.

⟶ The above 2 conditions will look like this in code:

```
int low = i+1,high = N-1,ans = -1;
while(low <= high){
        // Finding Mid for binary search
        int mid = (low+high)/2;
        if(prefix_min[mid] < Arr[i]){
                // Store the ans
                ans = mid;
                // Goto right side
                low = mid+1;
        }else{
```

```
                // Goto left side
                high = mid-1;
        }
}
```

⟶ After exiting from this while loop, we will enter our `ans` variable in final vector which we will return.

So the final code will look like this:

```
vector<int> farNumber(int N,vector<int> Arr){
        vector<int>prefix_min(N);
        prefix_min[N-1] = Arr[N-1];
        for(int i=N-2;i>=0;i--){
            prefix_min[i] = min(prefix_min[i+1],Arr[i]);
        }
                // This vector will store our ans
        vector<int>far;
        for(int i=0;i<N;i++){
            int low = i+1,high = N-1,ans=-1;
            while(low <= high){
                int mid = (low+high)/2;
                if(prefix_min[mid] < Arr[i]){
                    ans = mid;
                    low = mid+1;
                }else{
                    high = mid-1;
                }
            }
            far.push_back(ans);
        }
        return far;
    }
```