491. Non-decreasing Subsequences

Code:

```java
public static List<List<Integer>> findSubsequences(int[] nums) {

    Set<List<Integer>> result = new LinkedHashSet<>();
    List<Integer> current = new ArrayList<>();

    generate(nums, 0, current, result);

    return new ArrayList<>(result);

}

public static void generate(int[] nums, int index, List<Integer> currentSubsequence,
Set<List<Integer>> result) {

    // Save two elements in array
    if (currentSubsequence.size() >= 2) {
        result.add(new ArrayList<>(currentSubsequence));
    }

    // For non decreasing subsequences
    for (int i = index; i < nums.length; i++) {
        if (currentSubsequence.isEmpty() || nums[i] >=
currentSubsequence.get(currentSubsequence.size() - 1)) {
            currentSubsequence.add(nums[i]);
            generate(nums, i + 1, currentSubsequence, result);
            currentSubsequence.remove(currentSubsequence.size() - 1); // remove 1 size
after recursive is finish
        }
    }

}
```

The idea:

1. Rules to follow:
    - Subsequences must contain at least two elements (numbers)
    - Subsequences must be ascending (each element is greater than or equal to the preceding element).
    - Non-duplicate subsequences
2. Use recursive to generate all possible subsequence according to the rules
3. Use HashSet to automatically handle duplicate subsequences elimination

Code Summary:

1. findSubsequences method
    - Purpose:
        - This function finds all unique non-decreasing subsequences of length 2 or more from the input array **nums**
    - Parameters:
        - **nums**: The input array of integers from which subsequences are generated.

2. generate method
    - Purpose:
        - This recursive function explores all possible non-decreasing subsequences of length 2 or more starting from a specified index in the input array **nums**.
    - Parameters:
        - **nums**: The input array of integers.
        - **index**: The current index in the input array where subsequences are being generated.
        - **currentSubsequence**: The current subsequence being constructed during recursion.
        - **result**: The set used to store unique subsequences.
    - Code Flow:
        - The function recursively explores subsequences starting from **index** in **nums**.
        - For each element **nums[i]** at **index**, if it can extend the **currentSubsequence** while maintaining non-decreasing order, it's added to **currentSubsequence**.
        - After recursive is finish, the last added element is removed (bracktacking) to explore other possibilities.
        - Valid subsequences (length ≥ 2) are added to **result** as they are discovered.