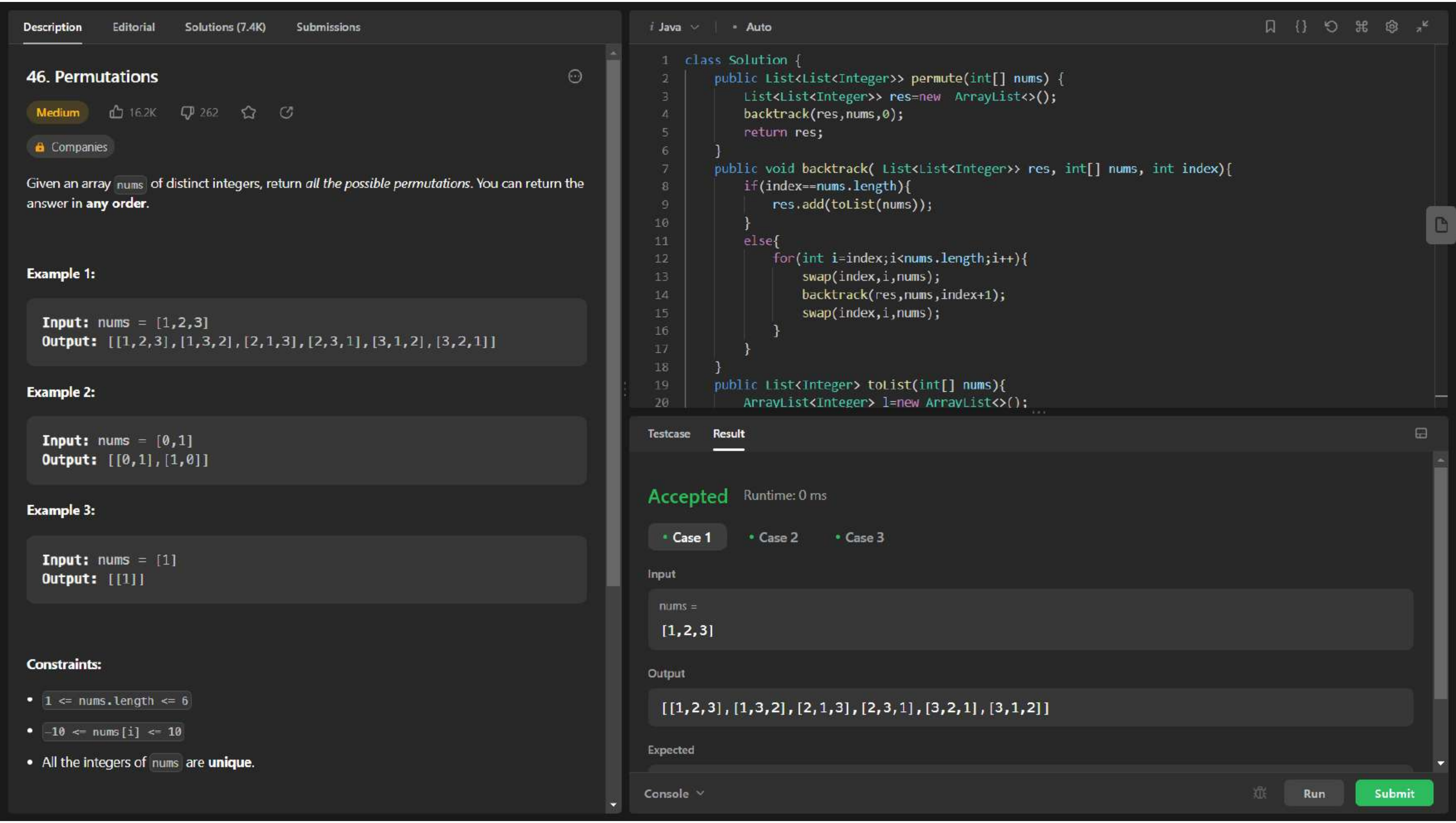


46. Permutations

Given an array `nums` of distinct integers, return *all the possible permutations*. You can return the answer in **any order**.

Example 1:

```
Input: nums = [1,2,3]
Output: [[1,2,3],[1,3,2],[2,1,3],[2,3,1],[3,1,2],[3,2,1]]
```

Example 2:

```
Input: nums = [0,1]
Output: [[0,1],[1,0]]
```

Example 3:

```
Input: nums = [1]
Output: [[1]]
```

Constraints:

- `1 <= nums.length <= 6`
- `-10 <= nums[i] <= 10`
- All the integers of `nums` are **unique**.

```java
class Solution {
    public List<List<Integer>> permute(int[] nums) {
        List<List<Integer>> res=new  ArrayList<>();
        backtrack(res,nums,0);
        return res;
    }
    public void backtrack( List<List<Integer>> res, int[] nums, int index){
        if(index==nums.length){
            res.add(toList(nums));
        }
        else{
            for(int i=index;i<nums.length;i++){
                swap(index,i,nums);
                backtrack(res,nums,index+1);
                swap(index,i,nums);
            }
        }
    }
    public List<Integer> toList(int[] nums){
        ArrayList<Integer> l=new ArrayList<>();
```

Accepted Runtime: 0 ms

• Case 1 • Case 2 • Case 3

Input

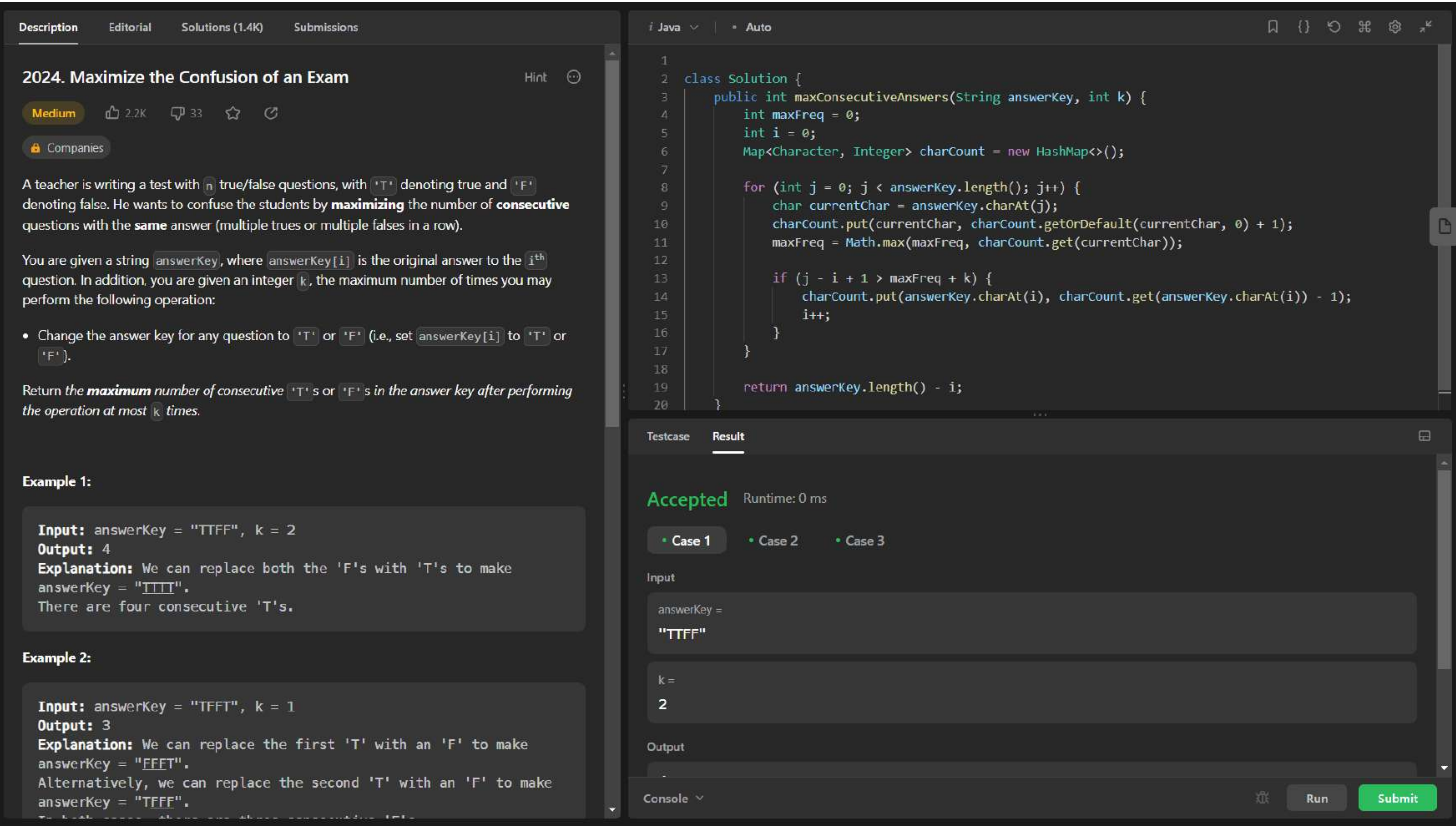
nums =

```
[1,2,3]
```

Output

```
[[1,2,3],[1,3,2],[2,1,3],[2,3,1],[3,2,1],[3,1,2]]
```

Expected



2024. Maximize the Confusion of an Exam

A teacher is writing a test with `n` true/false questions, with `'T'` denoting true and `'F'` denoting false. He wants to confuse the students by **maximizing** the number of **consecutive** questions with the **same** answer (multiple trues or multiple falses in a row).

You are given a string `answerKey`, where `answerKey[i]` is the original answer to the `i`th question. In addition, you are given an integer `k`, the maximum number of times you may perform the following operation:

- Change the answer key for any question to `'T'` or `'F'` (i.e., set `answerKey[i]` to `'T'` or `'F'`).

Return *the* ***maximum*** *number of consecutive* `'T'`*s or* `'F'`*s in the answer key after performing the operation at most* `k` *times.*

Example 1:

```
Input: answerKey = "TTFF", k = 2
Output: 4
Explanation: We can replace both the 'F's with 'T's to make
answerKey = "TTTT".
There are four consecutive 'T's.
```

Example 2:

```
Input: answerKey = "TFFT", k = 1
Output: 3
Explanation: We can replace the first 'T' with an 'F' to make
answerKey = "FFFT".
Alternatively, we can replace the second 'T' with an 'F' to make
answerKey = "TFFF".
In both cases, there are three consecutive 'F's.
```

```java

class Solution {
    public int maxConsecutiveAnswers(String answerKey, int k) {
        int maxFreq = 0;
        int i = 0;
        Map<Character, Integer> charCount = new HashMap<>();

        for (int j = 0; j < answerKey.length(); j++) {
            char currentChar = answerKey.charAt(j);
            charCount.put(currentChar, charCount.getOrDefault(currentChar, 0) + 1);
            maxFreq = Math.max(maxFreq, charCount.get(currentChar));

            if (j - i + 1 > maxFreq + k) {
                charCount.put(answerKey.charAt(i), charCount.get(answerKey.charAt(i)) - 1);
                i++;
            }
        }

        return answerKey.length() - i;
    }
```

Accepted Runtime: 0 ms

• Case 1 • Case 2 • Case 3

Input

answerKey =

"TTFF"

k =

2

Output