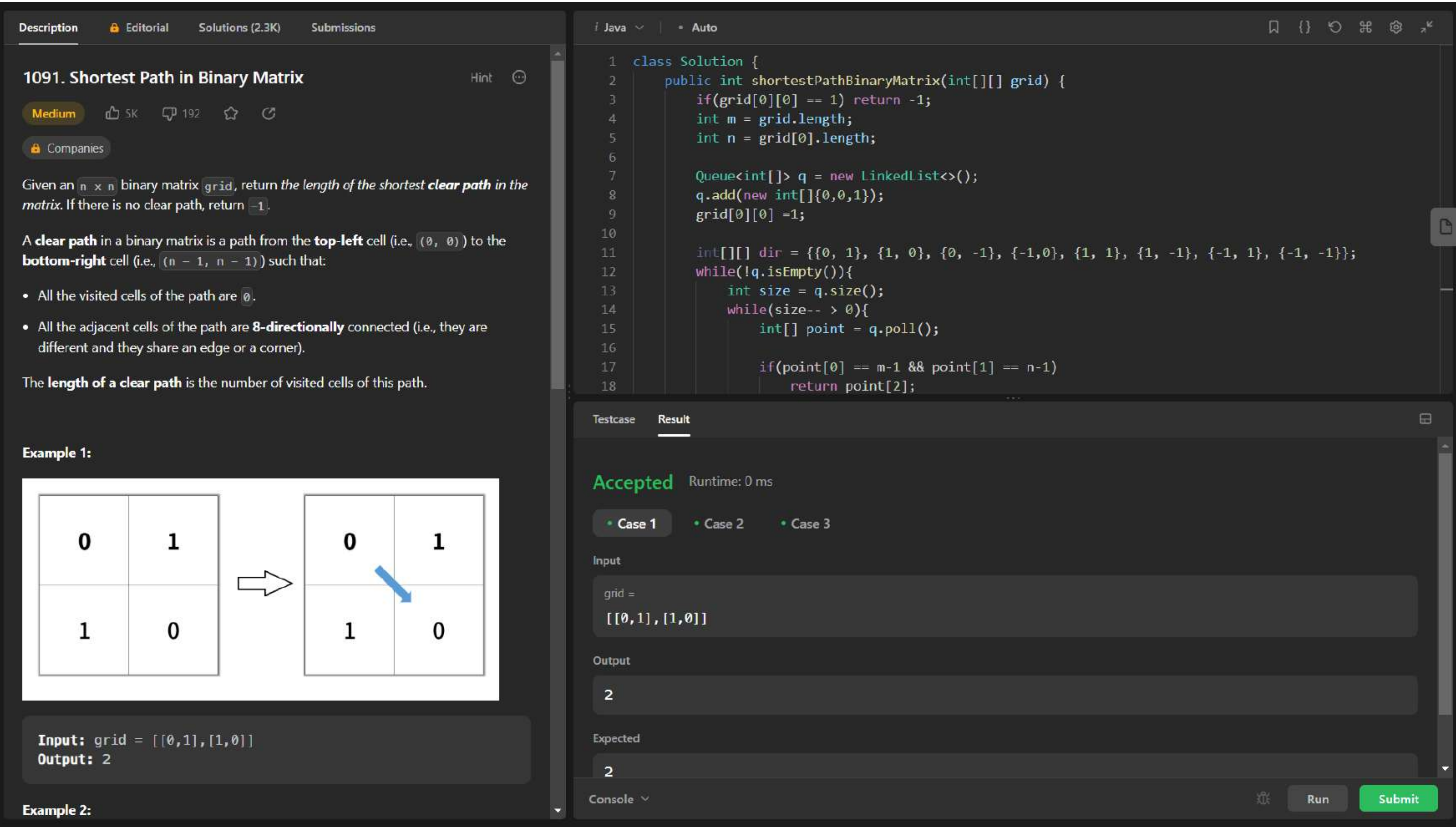


1091. Shortest Path in Binary Matrix

Given an `n x n` binary matrix `grid`, return *the length of the shortest **clear path** in the matrix*. If there is no clear path, return `-1`.

A **clear path** in a binary matrix is a path from the **top-left** cell (i.e., `(0, 0)`) to the **bottom-right** cell (i.e., `(n - 1, n - 1)`) such that:

- All the visited cells of the path are `0`.
- All the adjacent cells of the path are **8-directionally** connected (i.e., they are different and they share an edge or a corner).

The **length of a clear path** is the number of visited cells of this path.

Example 1:

```
Input: grid = [[0,1],[1,0]]
Output: 2
```

Example 2:

```java
class Solution {
    public int shortestPathBinaryMatrix(int[][] grid) {
        if(grid[0][0] == 1) return -1;
        int m = grid.length;
        int n = grid[0].length;

        Queue<int[]> q = new LinkedList<>();
        q.add(new int[]{0,0,1});
        grid[0][0] =1;

        int[][] dir = {{0, 1}, {1, 0}, {0, -1}, {-1,0}, {1, 1}, {1, -1}, {-1, 1}, {-1, -1}};
        while(!q.isEmpty()){
            int size = q.size();
            while(size-- > 0){
                int[] point = q.poll();

                if(point[0] == m-1 && point[1] == n-1)
                    return point[2];
```

Accepted Runtime: 0 ms

• Case 1 • Case 2 • Case 3

Input

grid =

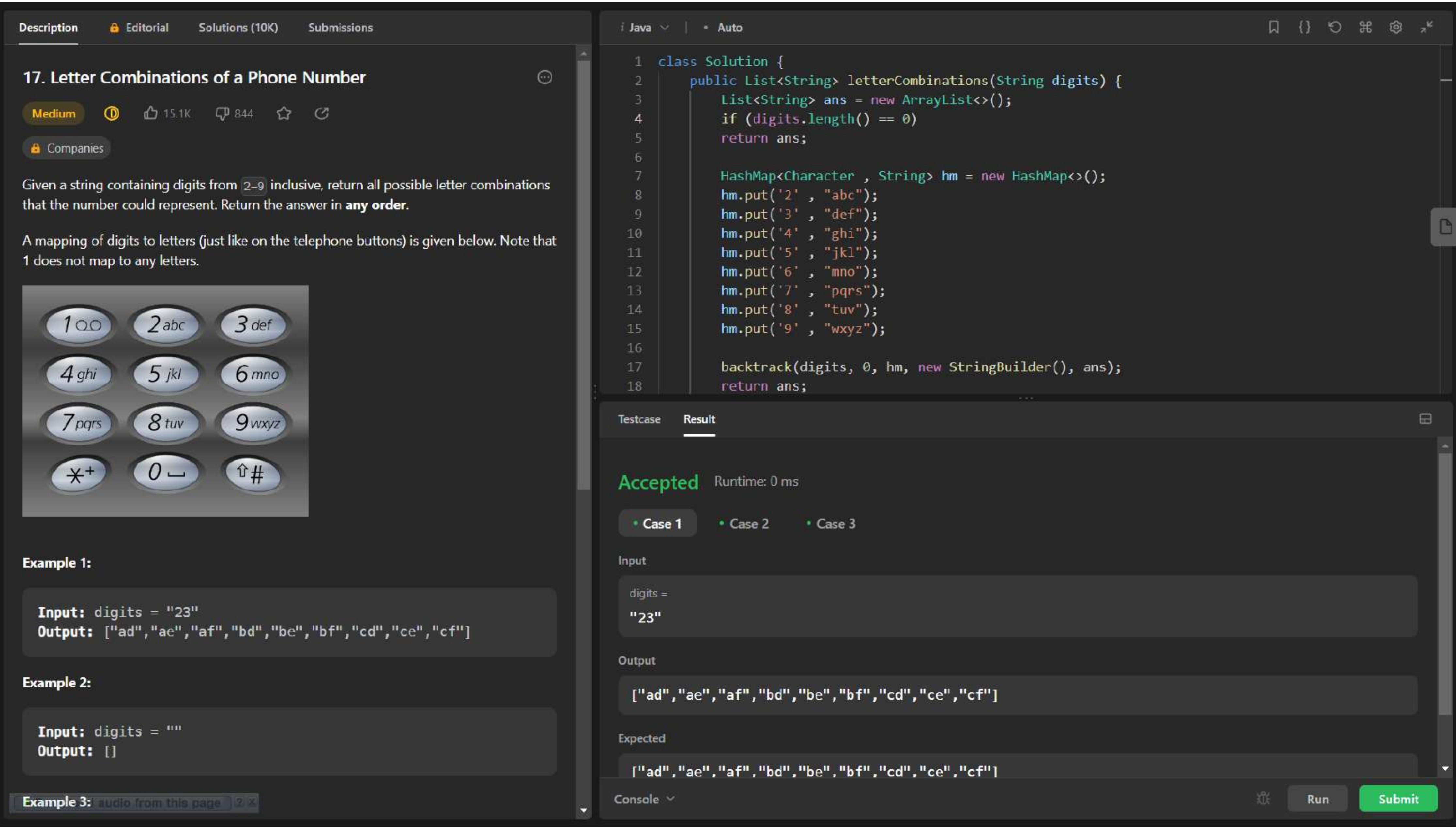
```
[[0,1],[1,0]]
```

Output

```
2
```

Expected

```
2
```



17. Letter Combinations of a Phone Number

Given a string containing digits from `2-9` inclusive, return all possible letter combinations that the number could represent. Return the answer in **any order**.

A mapping of digits to letters (just like on the telephone buttons) is given below. Note that 1 does not map to any letters.

Example 1:

```
Input: digits = "23"
Output: ["ad","ae","af","bd","be","bf","cd","ce","cf"]
```

Example 2:

```
Input: digits = ""
Output: []
```

Example 3:

```java
class Solution {
    public List<String> letterCombinations(String digits) {
        List<String> ans = new ArrayList<>();
        if (digits.length() == 0)
        return ans;

        HashMap<Character , String> hm = new HashMap<>();
        hm.put('2' , "abc");
        hm.put('3' , "def");
        hm.put('4' , "ghi");
        hm.put('5' , "jkl");
        hm.put('6' , "mno");
        hm.put('7' , "pqrs");
        hm.put('8' , "tuv");
        hm.put('9' , "wxyz");

        backtrack(digits, 0, hm, new StringBuilder(), ans);
        return ans;
```

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

digits =

"23"

Output

["ad","ae","af","bd","be","bf","cd","ce","cf"]

Expected

["ad","ae","af","bd","be","bf","cd","ce","cf"]