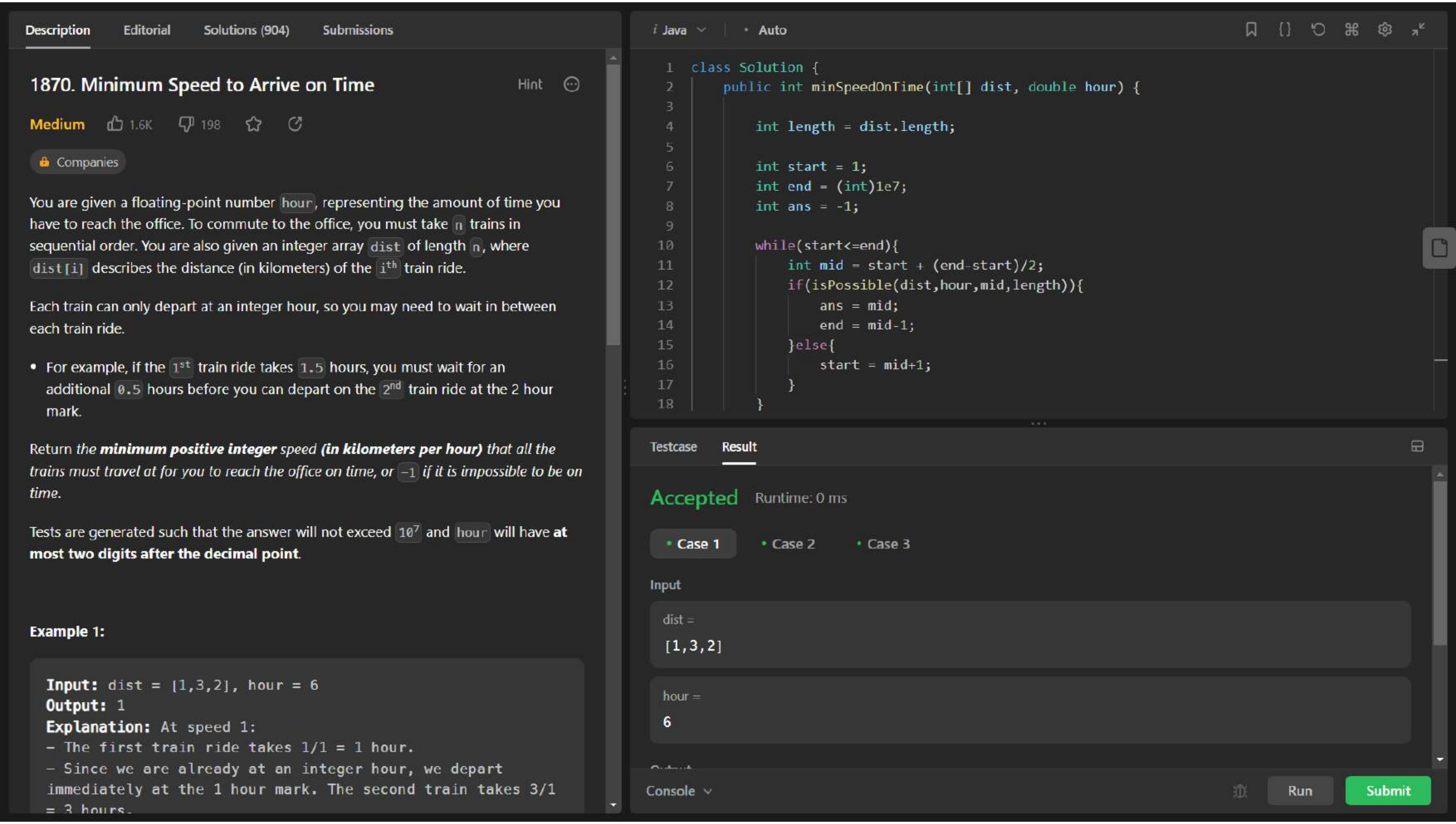


1870. Minimum Speed to Arrive on Time

You are given a floating-point number `hour`, representing the amount of time you have to reach the office. To commute to the office, you must take `n` trains in sequential order. You are also given an integer array `dist` of length `n`, where `dist[i]` describes the distance (in kilometers) of the i^{th} train ride.

Each train can only depart at an integer hour, so you may need to wait in between each train ride.

- For example, if the 1^{st} train ride takes `1.5` hours, you must wait for an additional `0.5` hours before you can depart on the 2^{nd} train ride at the 2 hour mark.

Return *the* ***minimum positive integer*** *speed* ***(in kilometers per hour)*** *that all the trains must travel at for you to reach the office on time, or* `-1` *if it is impossible to be on time.*

Tests are generated such that the answer will not exceed 10^7 and `hour` will have **at most two digits after the decimal point.**

Example 1:

```
Input: dist = [1,3,2], hour = 6
Output: 1
Explanation: At speed 1:
- The first train ride takes 1/1 = 1 hour.
- Since we are already at an integer hour, we depart
immediately at the 1 hour mark. The second train takes 3/1
= 3 hours.
```

```java
class Solution {
    public int minSpeedOnTime(int[] dist, double hour) {

        int length = dist.length;

        int start = 1;
        int end = (int)1e7;
        int ans = -1;

        while(start<=end){
            int mid = start + (end-start)/2;
            if(isPossible(dist,hour,mid,length)){
                ans = mid;
                end = mid-1;
            }else{
                start = mid+1;
            }
        }
```

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

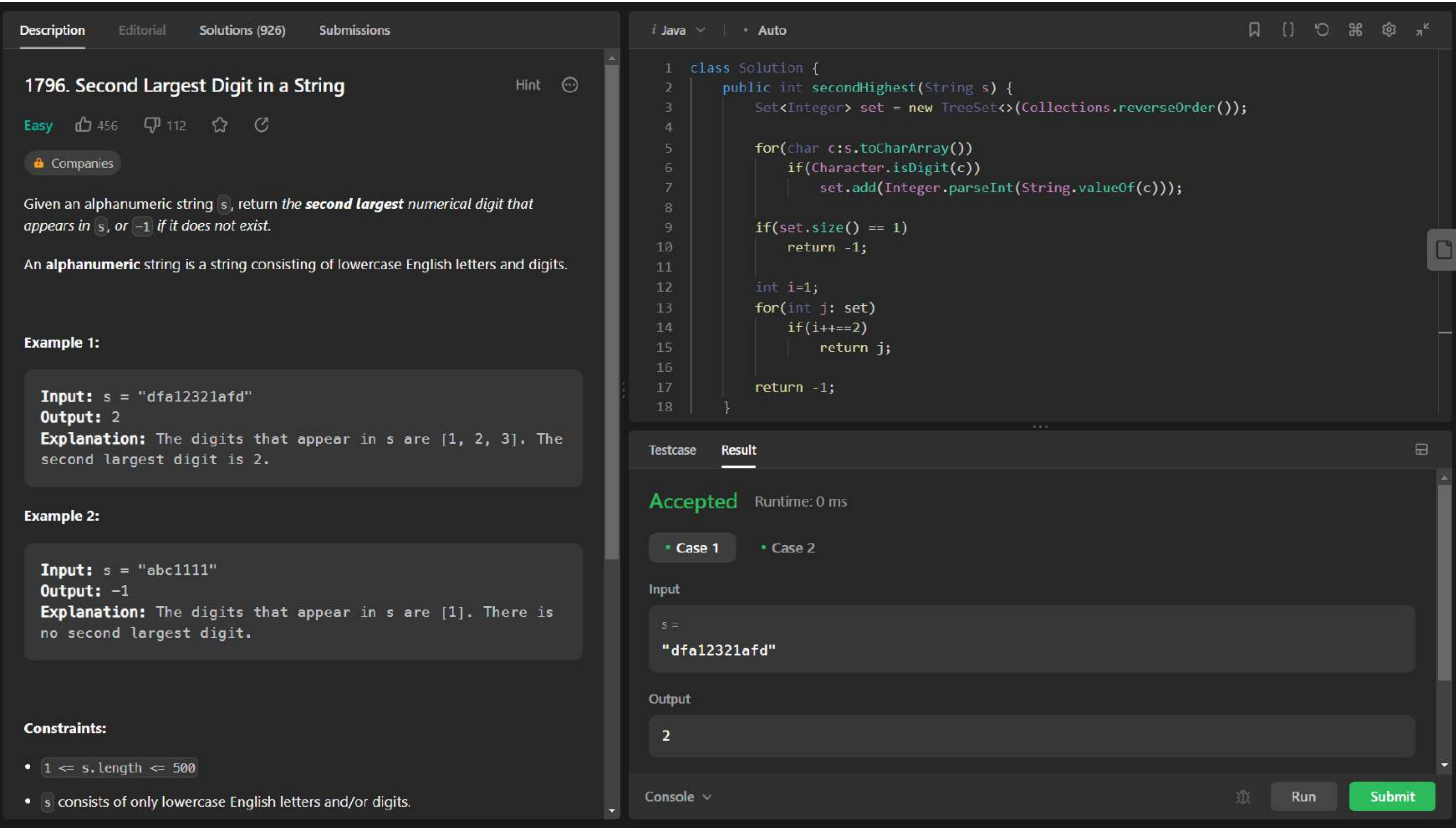
Input

dist =

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

hour =

```
6
```



1796. Second Largest Digit in a String

Given an alphanumeric string `s`, return *the* ***second largest*** *numerical digit that appears in* `s`*, or* `-1` *if it does not exist.*

An **alphanumeric** string is a string consisting of lowercase English letters and digits.

Example 1:

```
Input: s = "dfa12321afd"
Output: 2
Explanation: The digits that appear in s are [1, 2, 3]. The
second largest digit is 2.
```

Example 2:

```
Input: s = "abc1111"
Output: -1
Explanation: The digits that appear in s are [1]. There is
no second largest digit.
```

Constraints:

- `1 <= s.length <= 500`
- `s` consists of only lowercase English letters and/or digits.

```java
class Solution {
    public int secondHighest(String s) {
        Set<Integer> set = new TreeSet<>(Collections.reverseOrder());

        for(char c:s.toCharArray())
            if(Character.isDigit(c))
                set.add(Integer.parseInt(String.valueOf(c)));

        if(set.size() == 1)
            return -1;

        int i=1;
        for(int j: set)
            if(i++==2)
                return j;

        return -1;
    }
```

Accepted Runtime: 0 ms

Case 1 Case 2

Input

s =

```
"dfa12321afd"
```

Output

```
2
```