Facebook Run Submit

Description Solutions Submissions Editorial

157. Read N Characters Given Read4 Premium

Easy Topics Companies

Given a `file` and assume that you can only read the file using a given method `read4`, implement a method to read `n` characters.

Method read4:

The API `read4` reads **four consecutive characters** from `file`, then writes those characters into the buffer array `buf4`.

The return value is the number of actual characters read.

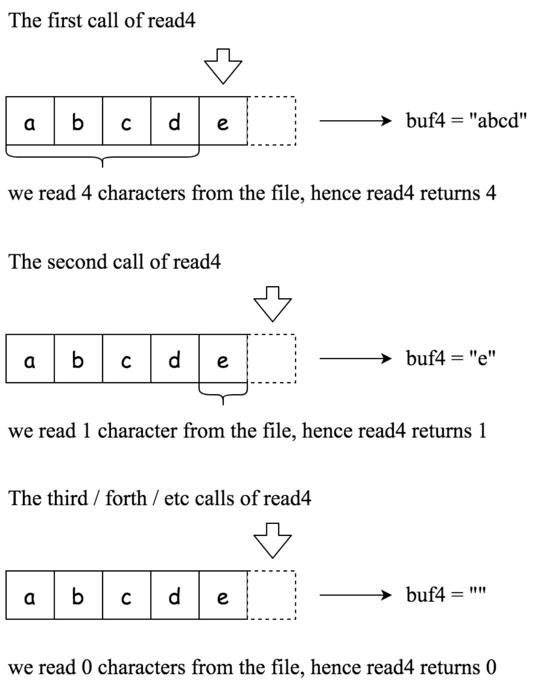
Note that `read4()` has its own file pointer, much like `FILE *fp` in C.

Definition of read4:

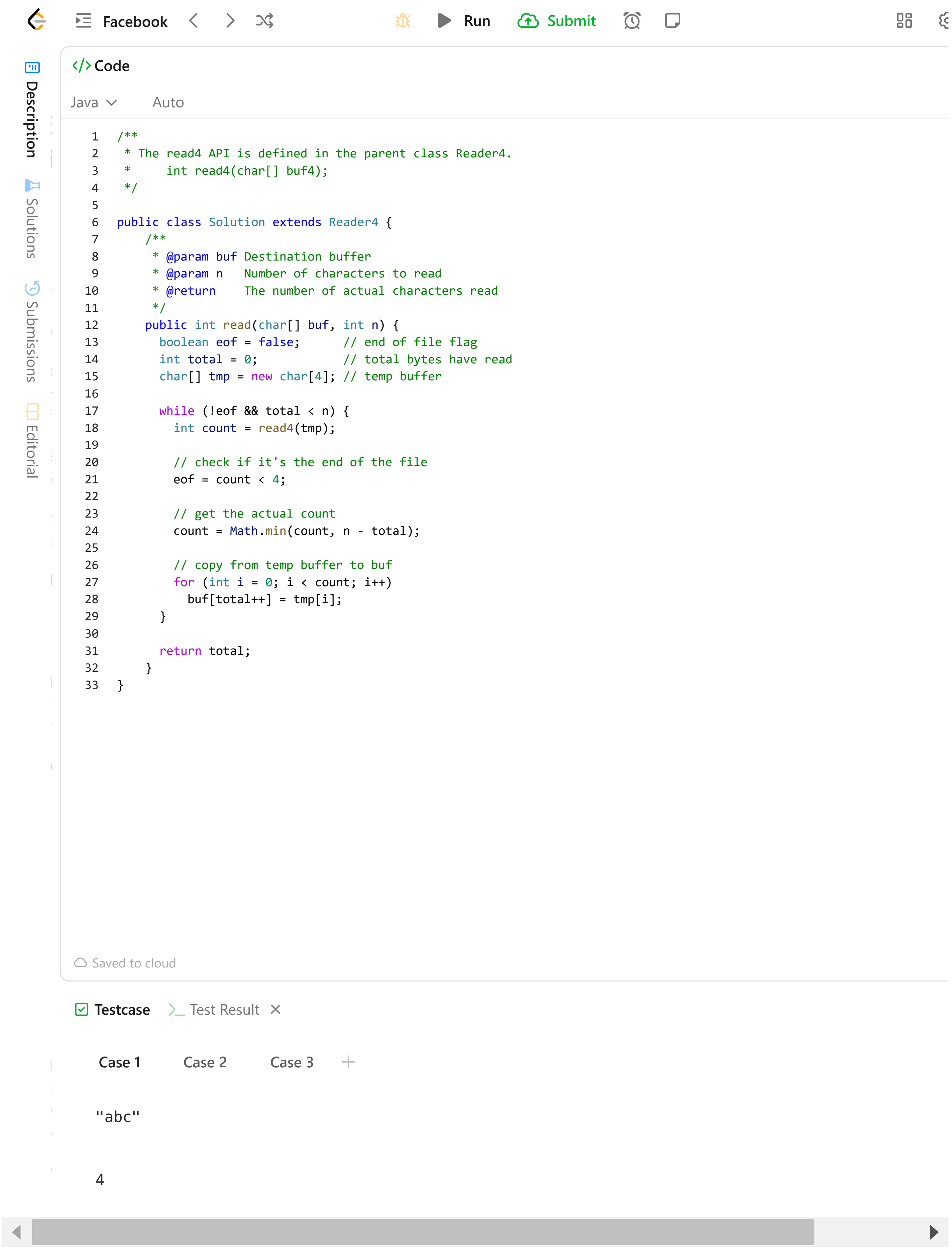
```
    Parameter:  char[] buf4
    Returns:    int

buf4[] is a destination, not a source. The results from read4 will be copied to buf4[].
```

Below is a high-level example of how `read4` works:


The first call of read4
a b c d e
buf4 = "abcd"
we read 4 characters from the file, hence read4 returns 4
The second call of read4
a b c d e
buf4 = "e"
we read 1 character from the file, hence read4 returns 1
The third / forth / etc calls of read4
a b c d e
buf4 = ""
we read 0 characters from the file, hence read4 returns 0

```
File file("abcde"); // File is "abcde", initially file pointer (fp) points to 'a'
char[] buf4 = new char[4]; // Create buffer with enough space to store characters
read4(buf4); // read4 returns 4. Now buf4 = "abcd", fp points to 'e'
read4(buf4); // read4 returns 1. Now buf4 = "e", fp points to end of file
read4(buf4); // read4 returns 0. Now buf4 = "", fp points to end of file
```



```java
/**
 * The read4 API is defined in the parent class Reader4.
 *     int read4(char[] buf4);
 */

public class Solution extends Reader4 {
    /**
     * @param buf Destination buffer
     * @param n   Number of characters to read
     * @return    The number of actual characters read
     */
    public int read(char[] buf, int n) {
      boolean eof = false;      // end of file flag
      int total = 0;            // total bytes have read
      char[] tmp = new char[4]; // temp buffer

      while (!eof && total < n) {
        int count = read4(tmp);

        // check if it's the end of the file
        eof = count < 4;

        // get the actual count
        count = Math.min(count, n - total);

        // copy from temp buffer to buf
        for (int i = 0; i < count; i++)
          buf[total++] = tmp[i];
      }

      return total;
    }
}
```

"abc"

4

158. Read N Characters Given read4 II - Call Multiple Times

Premium

Hard Topics Companies

Given a `file` and assume that you can only read the file using a given method `read4`, implement a method `read` to read `n` characters. Yo
may be **called multiple times**.

Method read4:

The API `read4` reads **four consecutive characters** from `file`, then writes those characters into the buffer array `buf4`.

The return value is the number of actual characters read.

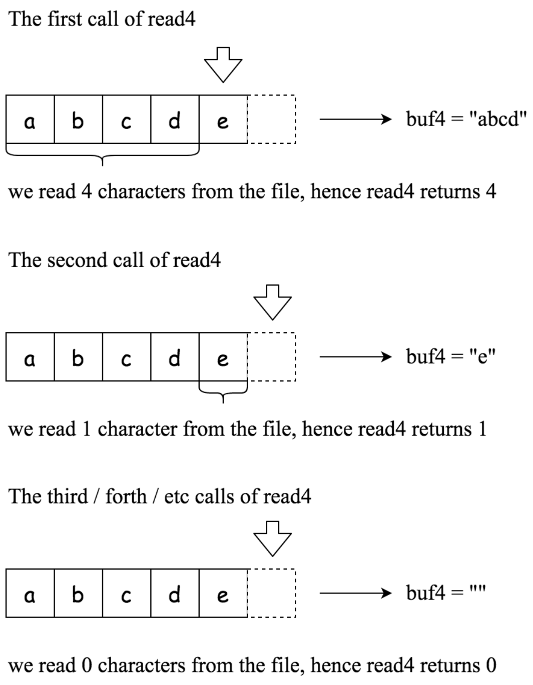
Note that `read4()` has its own file pointer, much like `FILE *fp` in C.

Definition of read4:

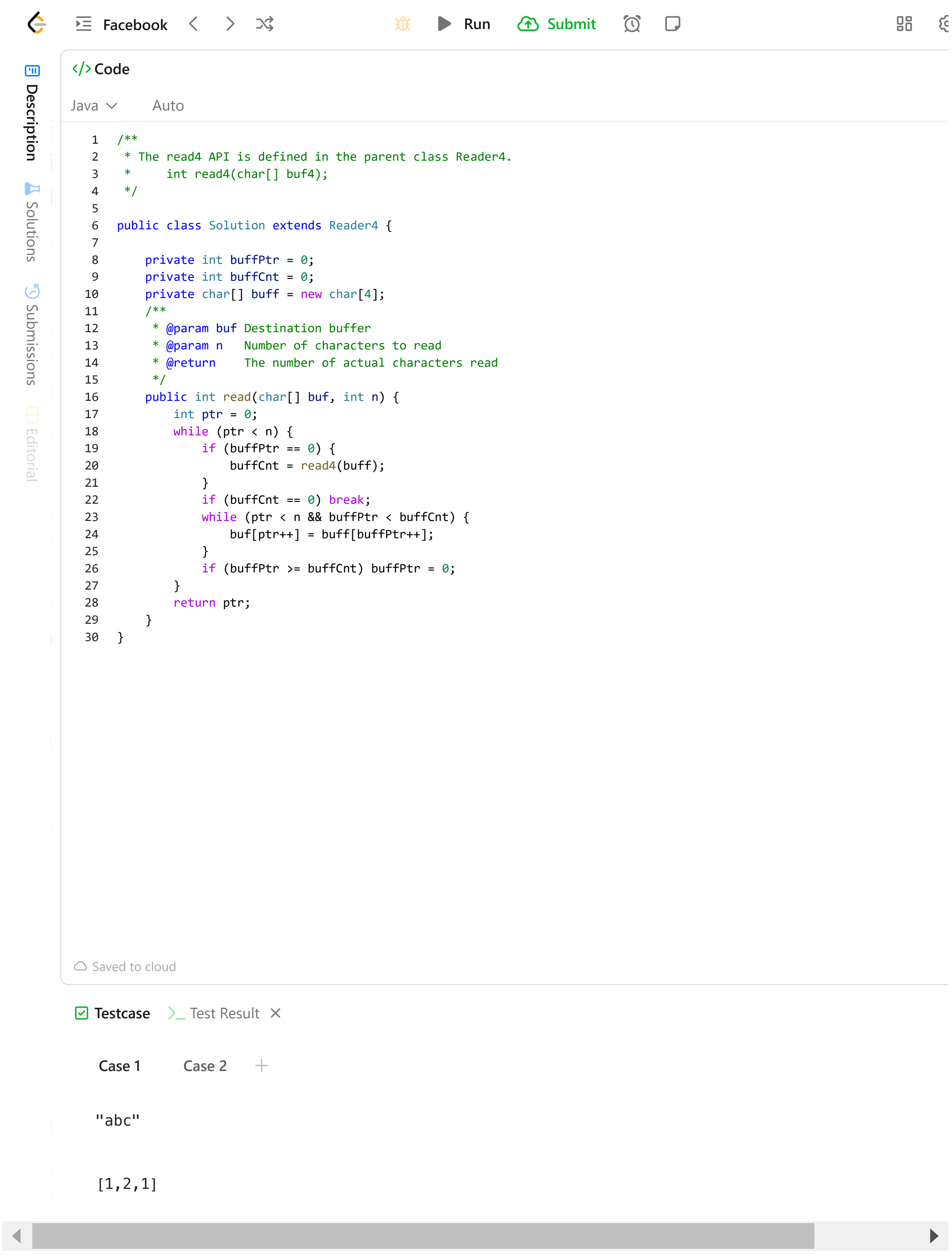
```
    Parameter:  char[] buf4
    Returns:    int

buf4[] is a destination, not a source. The results from read4 will be copied to buf4[].
```

Below is a high-level example of how `read4` works:


The first call of read4
a
b
c
d
e
buf4 = "abcd"
we read 4 characters from the file, hence read4 returns 4
The second call of read4
buf4 = "e"
we read 1 character from the file, hence read4 returns 1
The third / forth / etc calls of read4
buf4 = ""
we read 0 characters from the file, hence read4 returns 0

```
File file("abcde"); // File is "abcde", initially file pointer (fp) points to 'a'
char[] buf4 = new char[4]; // Create buffer with enough space to store characters
read4(buf4); // read4 returns 4. Now buf4 = "abcd", fp points to 'e'
read4(buf4); // read4 returns 1. Now buf4 = "e", fp points to end of file
read4(buf4); // read4 returns 0. Now buf4 = "", fp points to end of file
```



```java
/**
 * The read4 API is defined in the parent class Reader4.
 *     int read4(char[] buf4);
 */

public class Solution extends Reader4 {

    private int buffPtr = 0;
    private int buffCnt = 0;
    private char[] buff = new char[4];
    /**
     * @param buf Destination buffer
     * @param n   Number of characters to read
     * @return    The number of actual characters read
     */
    public int read(char[] buf, int n) {
        int ptr = 0;
        while (ptr < n) {
            if (buffPtr == 0) {
                buffCnt = read4(buff);
            }
            if (buffCnt == 0) break;
            while (ptr < n && buffPtr < buffCnt) {
                buf[ptr++] = buff[buffPtr++];
            }
            if (buffPtr >= buffCnt) buffPtr = 0;
        }
        return ptr;
    }
}
```

```
"abc"

[1,2,1]
```

Facebook Run Submit

Description Solutions Submissions Editorial

159. Longest Substring with At Most Two Distinct Characters Premium

Medium Topics Companies

Given a string `s`, return *the length of the longest substring that contains at most* ***two distinct characters***.

Example 1:

```
Input: s = "eceba"
Output: 3
Explanation: The substring is "ece" which its length is 3.
```

Example 2:

```
Input: s = "ccaabbb"
Output: 5
Explanation: The substring is "aabbb" which its length is 5.
```

Constraints:

- `1 <= s.length <=` 10^5
- `s` consists of English letters.

Seen this question in a real interview before? 1/4

Yes No

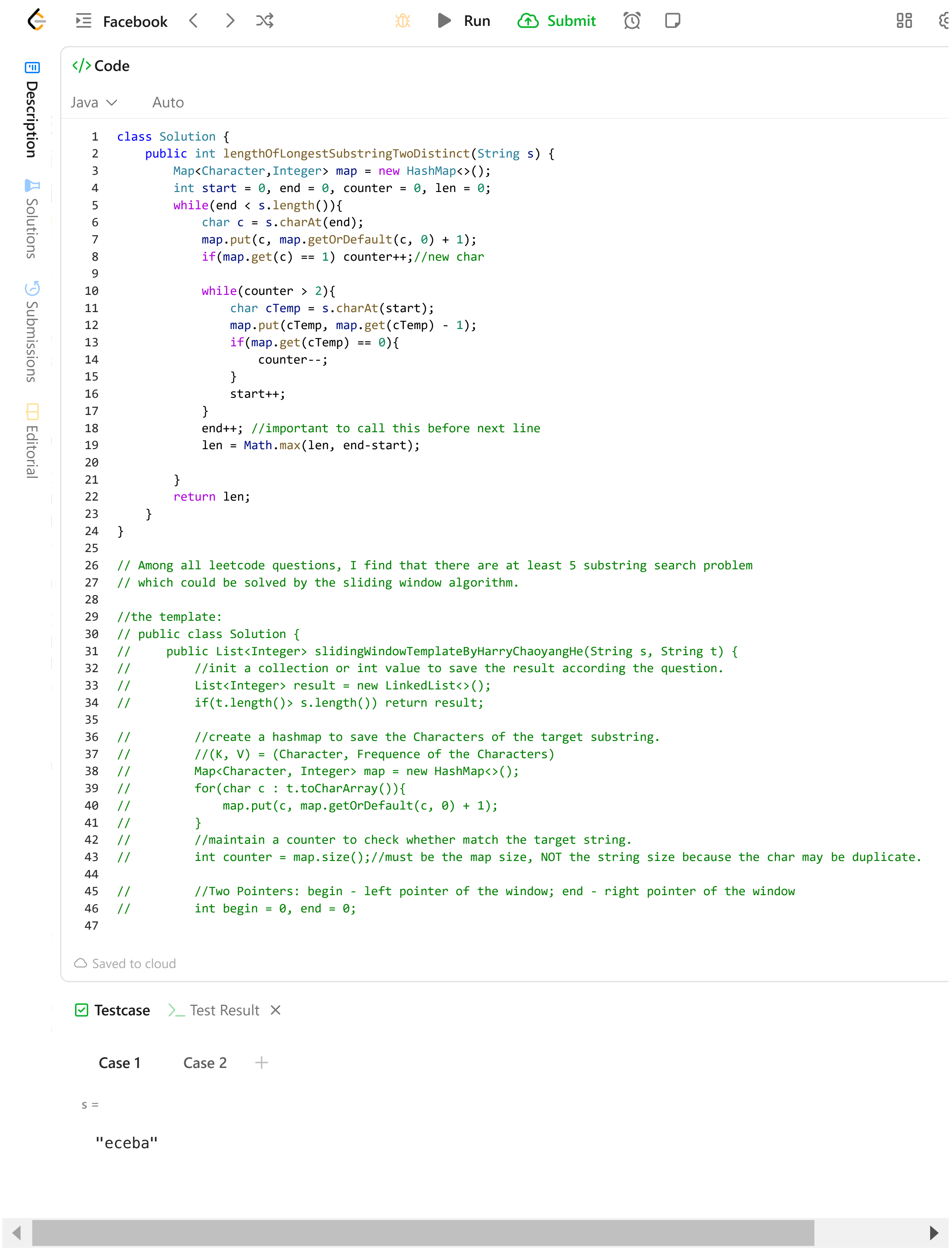
Accepted 247.5K Submissions 452.1K Acceptance Rate 54.7%

Topics

Companies

Similar Questions

Discussion (7)


Facebook   Run   Submit

Description   Solutions   Submissions   Editorial

</> Code

Java   Auto

```java
class Solution {
    public int lengthOfLongestSubstringTwoDistinct(String s) {
        Map<Character,Integer> map = new HashMap<>();
        int start = 0, end = 0, counter = 0, len = 0;
        while(end < s.length()){
            char c = s.charAt(end);
            map.put(c, map.getOrDefault(c, 0) + 1);
            if(map.get(c) == 1) counter++;//new char

            while(counter > 2){
                char cTemp = s.charAt(start);
                map.put(cTemp, map.get(cTemp) - 1);
                if(map.get(cTemp) == 0){
                    counter--;
                }
                start++;
            }
            end++; //important to call this before next line
            len = Math.max(len, end-start);

        }
        return len;
    }
}

// Among all leetcode questions, I find that there are at least 5 substring search problem
// which could be solved by the sliding window algorithm.

//the template:
// public class Solution {
//     public List<Integer> slidingWindowTemplateByHarryChaoyangHe(String s, String t) {
//         //init a collection or int value to save the result according the question.
//         List<Integer> result = new LinkedList<>();
//         if(t.length()> s.length()) return result;

//         //create a hashmap to save the Characters of the target substring.
//         //(K, V) = (Character, Frequence of the Characters)
//         Map<Character, Integer> map = new HashMap<>();
//         for(char c : t.toCharArray()){
//             map.put(c, map.getOrDefault(c, 0) + 1);
//         }
//         //maintain a counter to check whether match the target string.
//         int counter = map.size();//must be the map size, NOT the string size because the char may be duplicate.

//         //Two Pointers: begin - left pointer of the window; end - right pointer of the window
//         int begin = 0, end = 0;

```

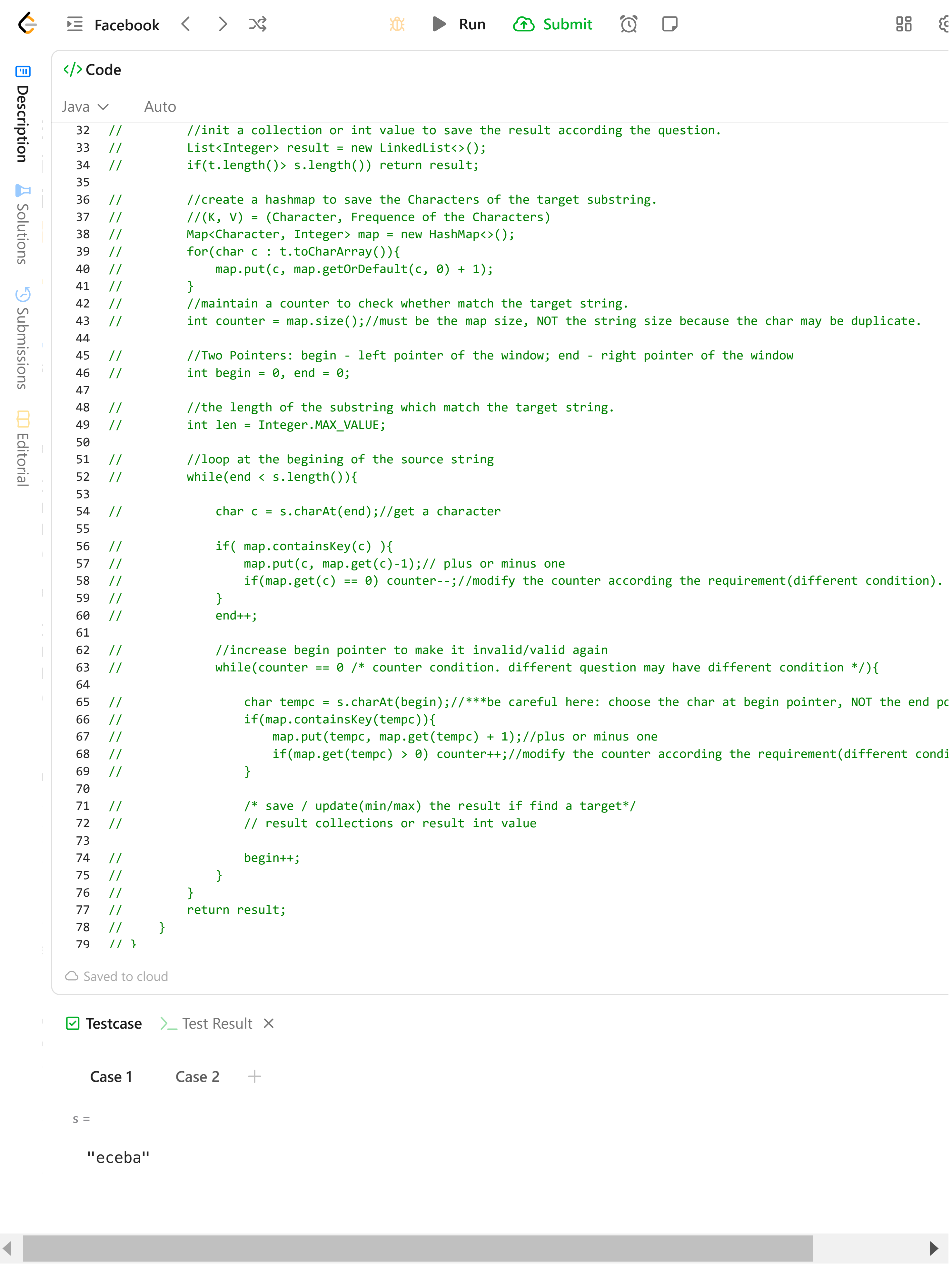
Saved to cloud

Testcase   Test Result

Case 1   Case 2

s =

"eceba"



```
//         //init a collection or int value to save the result according the question.
//         List<Integer> result = new LinkedList<>();
//         if(t.length()> s.length()) return result;

//         //create a hashmap to save the Characters of the target substring.
//         //(K, V) = (Character, Frequence of the Characters)
//         Map<Character, Integer> map = new HashMap<>();
//         for(char c : t.toCharArray()){
//             map.put(c, map.getOrDefault(c, 0) + 1);
//         }
//         //maintain a counter to check whether match the target string.
//         int counter = map.size();//must be the map size, NOT the string size because the char may be duplicate.

//         //Two Pointers: begin - left pointer of the window; end - right pointer of the window
//         int begin = 0, end = 0;

//         //the length of the substring which match the target string.
//         int len = Integer.MAX_VALUE;

//         //loop at the begining of the source string
//         while(end < s.length()){

//             char c = s.charAt(end);//get a character

//             if( map.containsKey(c) ){
//                 map.put(c, map.get(c)-1);// plus or minus one
//                 if(map.get(c) == 0) counter--;//modify the counter according the requirement(different condition).
//             }
//             end++;

//             //increase begin pointer to make it invalid/valid again
//             while(counter == 0 /* counter condition. different question may have different condition */){

//                 char tempc = s.charAt(begin);//***be careful here: choose the char at begin pointer, NOT the end pc
//                 if(map.containsKey(tempc)){
//                     map.put(tempc, map.get(tempc) + 1);//plus or minus one
//                     if(map.get(tempc) > 0) counter++;//modify the counter according the requirement(different condi
//                 }

//                 /* save / update(min/max) the result if find a target*/
//                 // result collections or result int value

//                 begin++;
//             }
//         }
//         return result;
//     }
// }
```

s =

"eceba"

Facebook   Run   Submit

Description   Solutions   Submissions   Editorial

161. One Edit Distance Premium

Medium   Topics   Companies

Given two strings `s` and `t`, return `true` if they are both one edit distance apart, otherwise return `false`.

A string `s` is said to be one distance apart from a string `t` if you can:

- Insert **exactly one** character into `s` to get `t`.
- Delete **exactly one** character from `s` to get `t`.
- Replace **exactly one** character of `s` with **a different character** to get `t`.

Example 1:

```
Input: s = "ab", t = "acb"
Output: true
Explanation: We can insert 'c' into s to get t.
```

Example 2:

```
Input: s = "", t = ""
Output: false
Explanation: We cannot get t from s by only one step.
```

Constraints:

- `0 <= s.length, t.length <=` 10^4
- `s` and `t` consist of lowercase letters, uppercase letters, and digits.

Seen this question in a real interview before? 1/4

Yes No

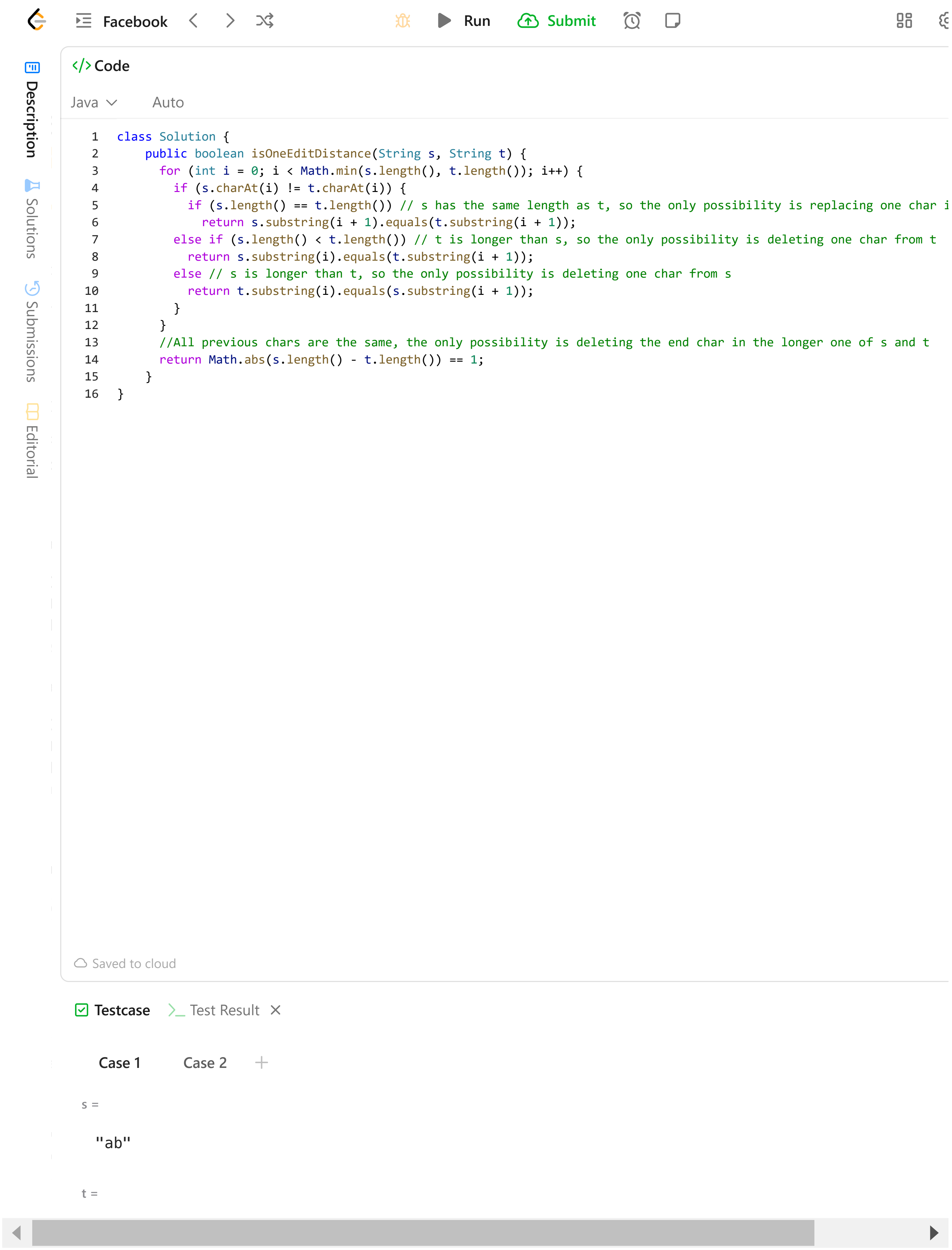
Accepted 202.3K   Submissions 590.4K   Acceptance Rate 34.3%

Topics

Companies

Similar Questions

Discussion (7)



Code

Java   Auto

```java
class Solution {
    public boolean isOneEditDistance(String s, String t) {
      for (int i = 0; i < Math.min(s.length(), t.length()); i++) {
        if (s.charAt(i) != t.charAt(i)) {
          if (s.length() == t.length()) // s has the same length as t, so the only possibility is replacing one char i
            return s.substring(i + 1).equals(t.substring(i + 1));
        else if (s.length() < t.length()) // t is longer than s, so the only possibility is deleting one char from t
          return s.substring(i).equals(t.substring(i + 1));
        else // s is longer than t, so the only possibility is deleting one char from s
          return t.substring(i).equals(s.substring(i + 1));
        }
      }
      //All previous chars are the same, the only possibility is deleting the end char in the longer one of s and t
      return Math.abs(s.length() - t.length()) == 1;
    }
}
```

Saved to cloud

Testcase   Test Result

Case 1   Case 2

s =

"ab"

t =

163. Missing Ranges Premium

Easy Topics Companies

You are given an inclusive range `[lower, upper]` and a **sorted unique** integer array `nums`, where all elements are within the inclusive rang

A number `x` is considered **missing** if `x` is in the range `[lower, upper]` and `x` is not in `nums`.

Return *the **shortest sorted** list of ranges that **exactly covers all the missing numbers***. That is, no element of `nums` is included in any of the missing number is covered by one of the ranges.

Example 1:

```
Input: nums = [0,1,3,50,75], lower = 0, upper = 99
Output: [[2,2],[4,49],[51,74],[76,99]]
Explanation: The ranges are:
[2,2]
[4,49]
[51,74]
[76,99]
```

Example 2:

```
Input: nums = [-1], lower = -1, upper = -1
Output: []
Explanation: There are no missing ranges since there are no missing numbers.
```

Constraints:

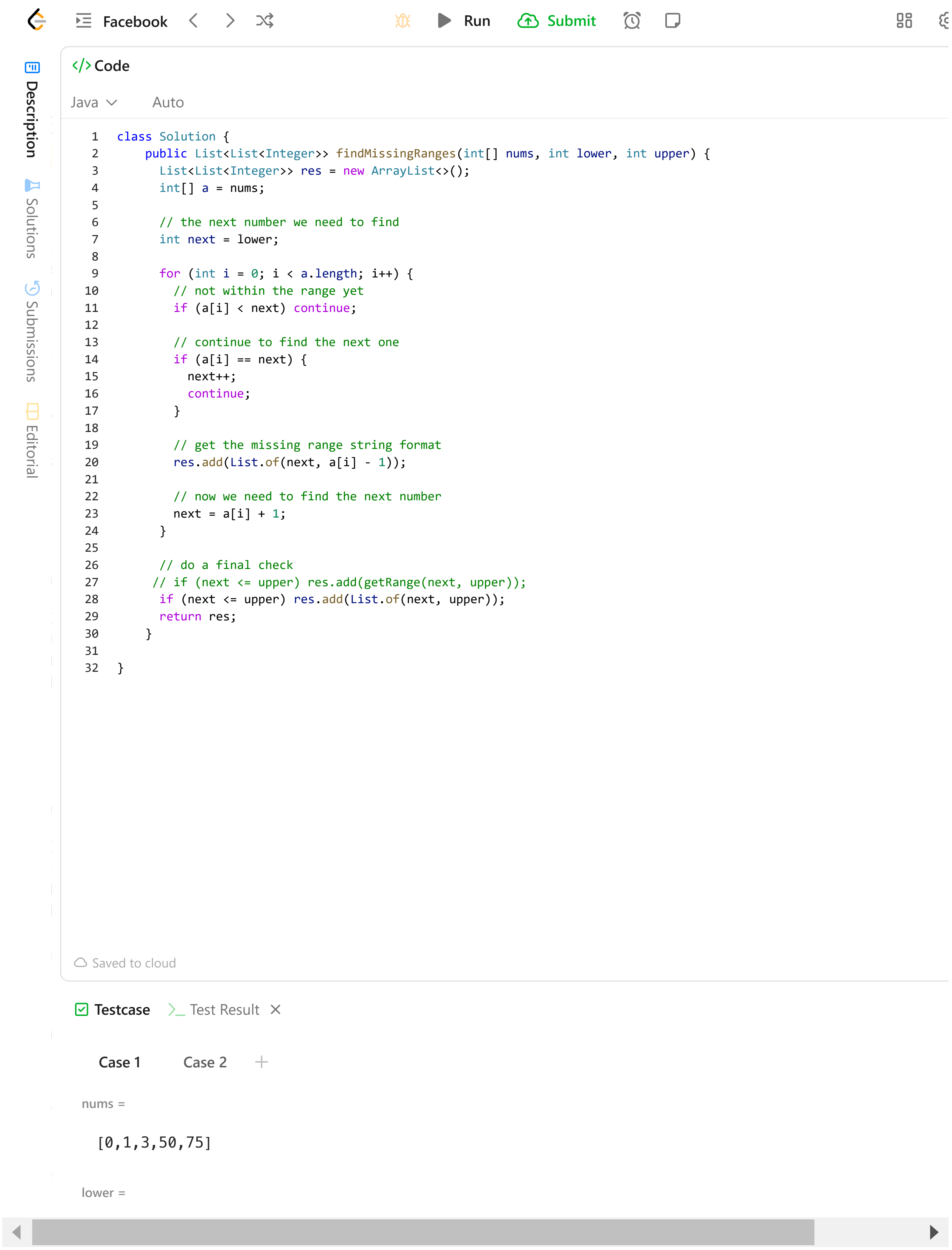
- -10^{9} <= lower <= upper <= 10^{9}
- `0 <= nums.length <= 100`
- `lower <= nums[i] <= upper`
- All the values of `nums` are **unique**.

Seen this question in a real interview before? 1/4

Yes No

Accepted 229.9K Submissions 689.8K Acceptance Rate 33.3%

Topics



```java
class Solution {
    public List<List<Integer>> findMissingRanges(int[] nums, int lower, int upper) {
      List<List<Integer>> res = new ArrayList<>();
      int[] a = nums;

      // the next number we need to find
      int next = lower;

      for (int i = 0; i < a.length; i++) {
        // not within the range yet
        if (a[i] < next) continue;

        // continue to find the next one
        if (a[i] == next) {
          next++;
          continue;
        }

        // get the missing range string format
        res.add(List.of(next, a[i] - 1));

        // now we need to find the next number
        next = a[i] + 1;
      }

      // do a final check
     // if (next <= upper) res.add(getRange(next, upper));
      if (next <= upper) res.add(List.of(next, upper));
      return res;
    }

}
```

nums =

```
[0,1,3,50,75]
```

lower =

246. Strobogrammatic Number Premium

Easy Topics Companies

Given a string `num` which represents an integer, return `true` *if* `num` *is a* ***strobogrammatic number***.

A **strobogrammatic number** is a number that looks the same when rotated `180` degrees (looked at upside down).

Example 1:

```
Input: num = "69"
Output: true
```

Example 2:

```
Input: num = "88"
Output: true
```

Example 3:

```
Input: num = "962"
Output: false
```

Constraints:

- `1 <= num.length <= 50`
- `num` consists of only digits.
- `num` does not contain any leading zeros except for zero itself.

Seen this question in a real interview before? 1/4

Yes No

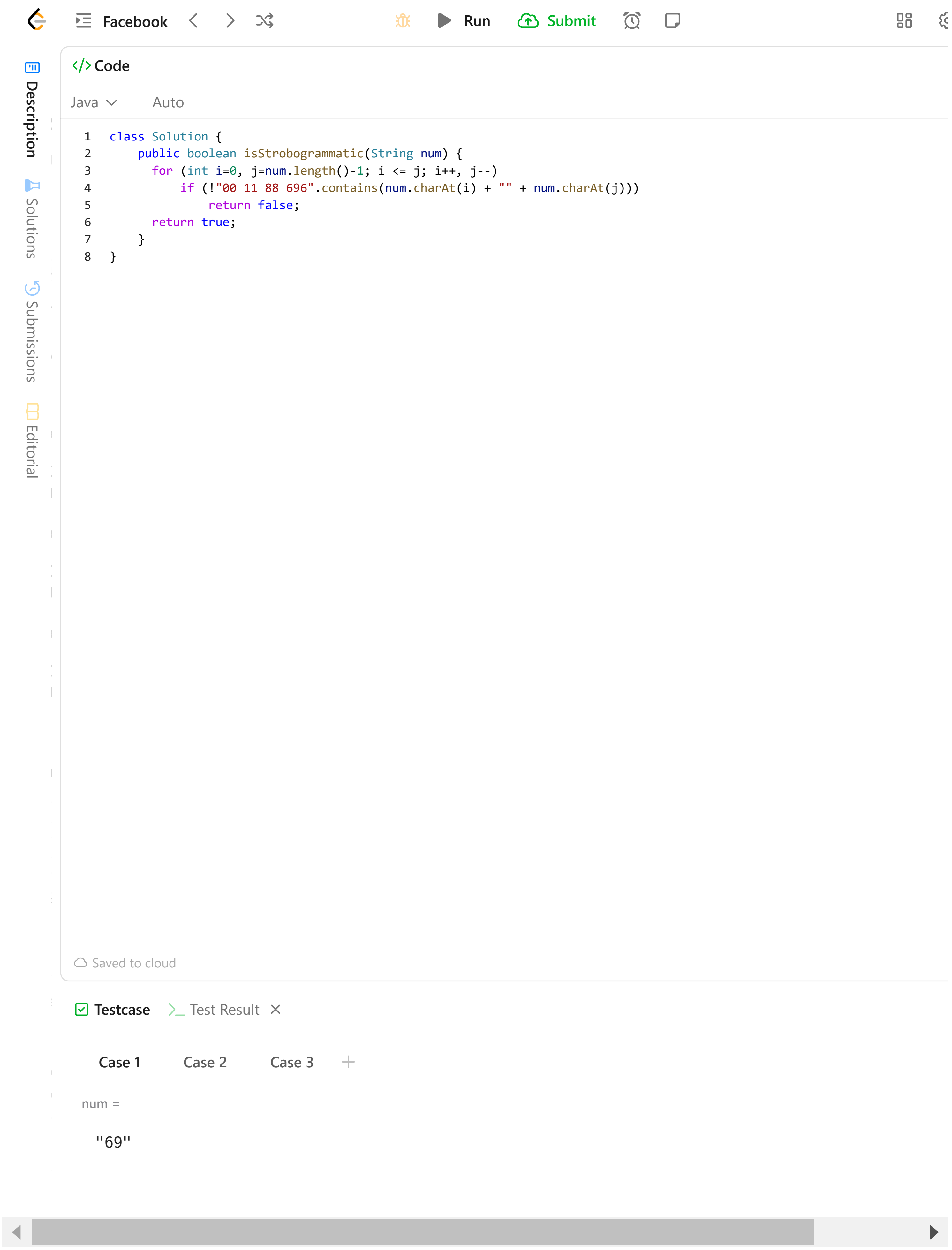
Accepted 170.3K Submissions 357.2K Acceptance Rate 47.7%

Topics

Companies

Similar Questions

Discussion (9)



```java
class Solution {
    public boolean isStrobogrammatic(String num) {
      for (int i=0, j=num.length()-1; i <= j; i++, j--)
          if (!"00 11 88 696".contains(num.charAt(i) + "" + num.charAt(j)))
              return false;
      return true;
    }
}
```

num =

"69"

Facebook Run Submit

Description Solutions Submissions Editorial

247. Strobogrammatic Number II

Premium

Medium Topics Companies Hint

Given an integer `n`, return all the **strobogrammatic numbers** that are of length `n`. You may return the answer in **any order**.

A **strobogrammatic number** is a number that looks the same when rotated `180` degrees (looked at upside down).

Example 1:

```
Input: n = 2
Output: ["11","69","88","96"]
```

Example 2:

```
Input: n = 1
Output: ["0","1","8"]
```

Constraints:

- `1 <= n <= 14`

Seen this question in a real interview before? 1/4

Yes No

Accepted 136K Submissions 261.3K Acceptance Rate 52.1%

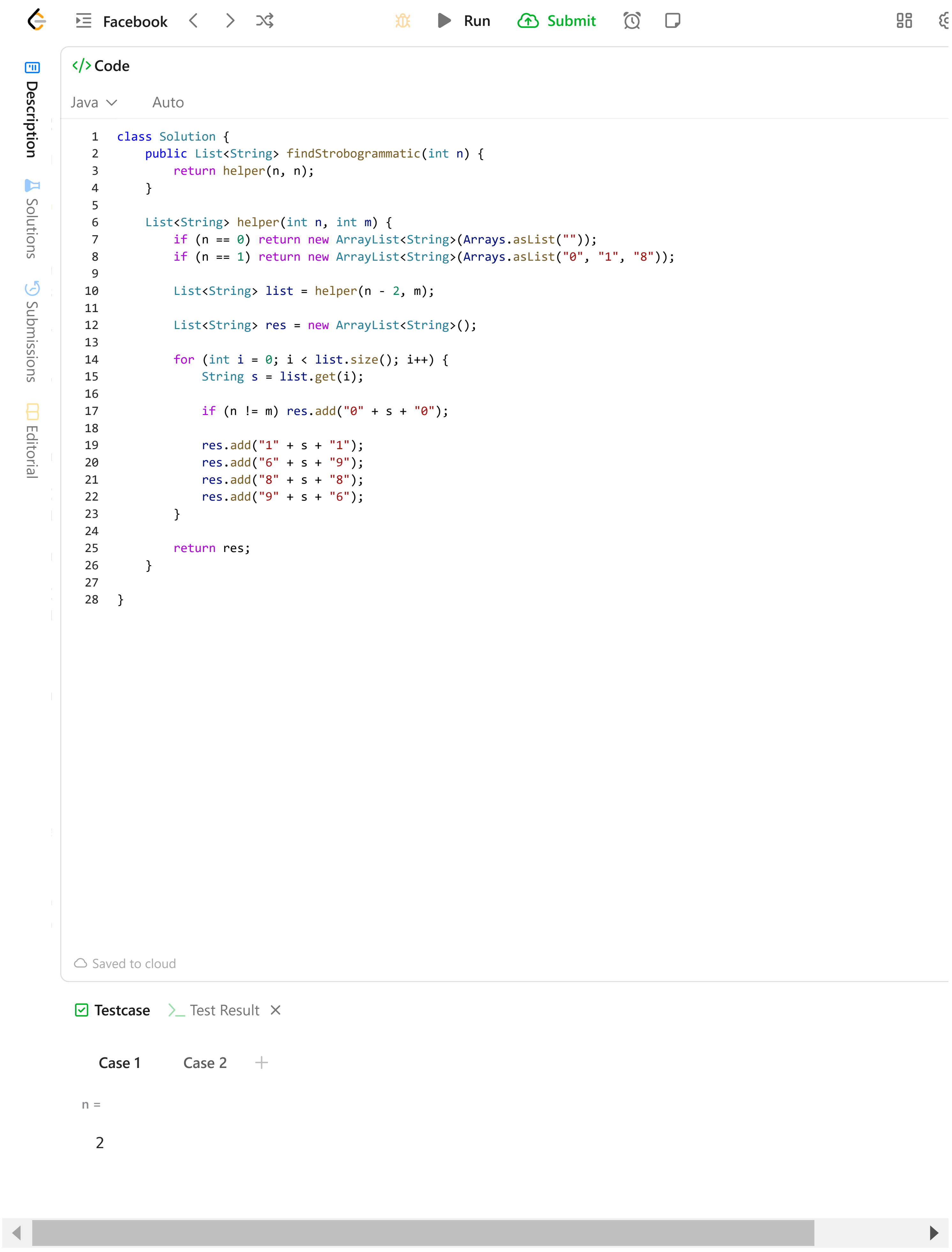
Topics

Companies

Hint 1

Similar Questions

Discussion (8)



```java
class Solution {
    public List<String> findStrobogrammatic(int n) {
        return helper(n, n);
    }

    List<String> helper(int n, int m) {
        if (n == 0) return new ArrayList<String>(Arrays.asList(""));
        if (n == 1) return new ArrayList<String>(Arrays.asList("0", "1", "8"));

        List<String> list = helper(n - 2, m);

        List<String> res = new ArrayList<String>();

        for (int i = 0; i < list.size(); i++) {
            String s = list.get(i);

            if (n != m) res.add("0" + s + "0");

            res.add("1" + s + "1");
            res.add("6" + s + "9");
            res.add("8" + s + "8");
            res.add("9" + s + "6");
        }

        return res;
    }

}
```

n =

2

Facebook Run Submit

Description Solutions Submissions Editorial

249. Group Shifted Strings Premium

Medium Topics Companies

We can shift a string by shifting each of its letters to its successive letter.

- For example, `"abc"` can be shifted to be `"bcd"`.

We can keep shifting the string to form a sequence.

- For example, we can keep shifting `"abc"` to form the sequence: `"abc" -> "bcd" -> ... -> "xyz"`.

Given an array of strings `strings`, group all `strings[i]` that belong to the same shifting sequence. You may return the answer in **any ord**

Example 1:

```
Input: strings = ["abc","bcd","acef","xyz","az","ba","a","z"]
Output: [["acef"],["a","z"],["abc","bcd","xyz"],["az","ba"]]
```

Example 2:

```
Input: strings = ["a"]
Output: [["a"]]
```

Constraints:

- `1 <= strings.length <= 200`
- `1 <= strings[i].length <= 50`
- `strings[i]` consists of lowercase English letters.

Seen this question in a real interview before? 1/4

Yes No

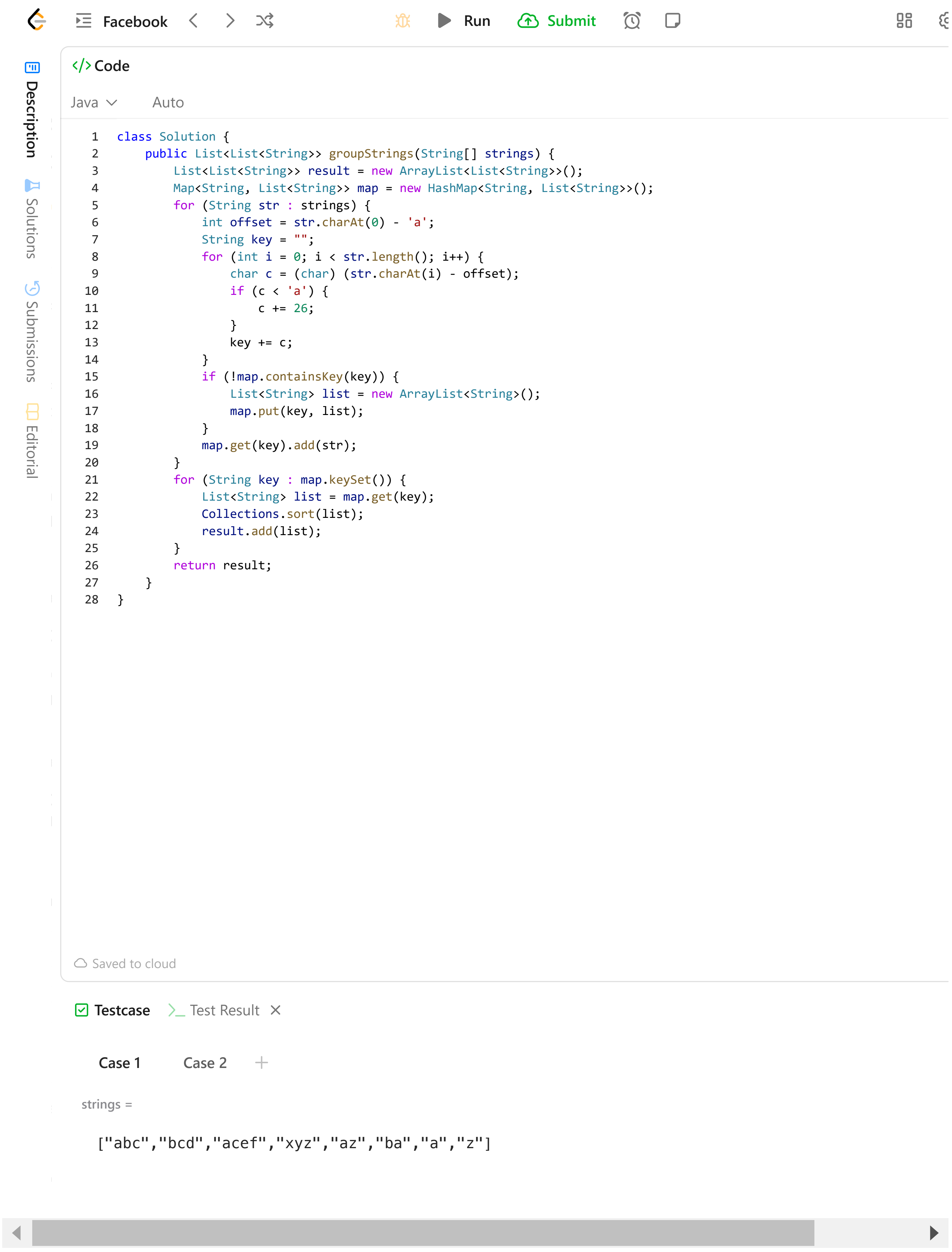
Accepted 225.4K Submissions 345.2K Acceptance Rate 65.3%

Topics

Companies

Similar Questions

Discussion (8)



```
class Solution {
    public List<List<String>> groupStrings(String[] strings) {
        List<List<String>> result = new ArrayList<List<String>>();
        Map<String, List<String>> map = new HashMap<String, List<String>>();
        for (String str : strings) {
            int offset = str.charAt(0) - 'a';
            String key = "";
            for (int i = 0; i < str.length(); i++) {
                char c = (char) (str.charAt(i) - offset);
                if (c < 'a') {
                    c += 26;
                }
                key += c;
            }
            if (!map.containsKey(key)) {
                List<String> list = new ArrayList<String>();
                map.put(key, list);
            }
            map.get(key).add(str);
        }
        for (String key : map.keySet()) {
            List<String> list = map.get(key);
            Collections.sort(list);
            result.add(list);
        }
        return result;
    }
}
```

strings =

```
["abc","bcd","acef","xyz","az","ba","a","z"]
```

Facebook   Run   Submit

Description   Solutions   Submissions   Editorial

252. Meeting Rooms Premium

Easy   Topics   Companies

Given an array of meeting time `intervals` where `intervals[i] = [start_i, end_i]`, determine if a person could attend all meetings.

Example 1:

```
Input: intervals = [[0,30],[5,10],[15,20]]
Output: false
```

Example 2:

```
Input: intervals = [[7,10],[2,4]]
Output: true
```

Constraints:

- $0 <= intervals.length <= 10^4$
- `intervals[i].length == 2`
- $0 <= start_i < end_i <= 10^6$

Seen this question in a real interview before? 1/4

Yes   No

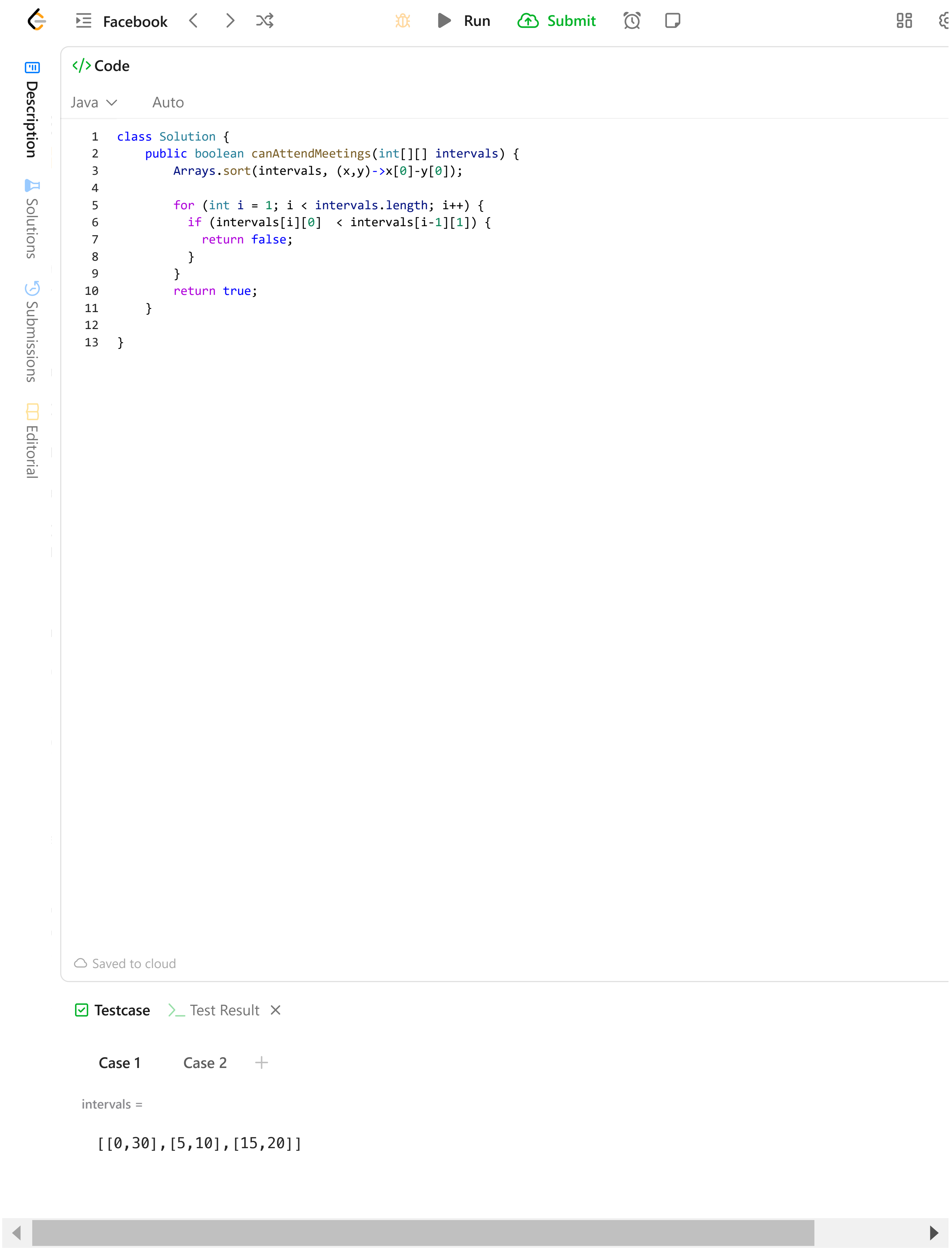
Accepted 385.6K   Submissions 666K   Acceptance Rate 57.9%

Topics

Companies

Similar Questions

Discussion (7)



```java
class Solution {
    public boolean canAttendMeetings(int[][] intervals) {
        Arrays.sort(intervals, (x,y)->x[0]-y[0]);

        for (int i = 1; i < intervals.length; i++) {
          if (intervals[i][0]  < intervals[i-1][1]) {
            return false;
          }
        }
        return true;
    }

}
```

intervals =

```
[[0,30],[5,10],[15,20]]
```

Facebook Run Submit

Description Solutions Submissions Editorial

253. Meeting Rooms II Premium

Medium Topics Companies Hint

Given an array of meeting time intervals `intervals` where `intervals[i] = [start_i, end_i]`, return *the minimum number of conference rc*

Example 1:

```
Input: intervals = [[0,30],[5,10],[15,20]]
Output: 2
```

Example 2:

```
Input: intervals = [[7,10],[2,4]]
Output: 1
```

Constraints:

- `1 <= intervals.length <=` 10^4
- $0 <= start_i < end_i <= 10^6$

Seen this question in a real interview before? 1/4

Yes No

Accepted 861.8K Submissions 1.7M Acceptance Rate 51.0%

Topics

Companies

Hint 1

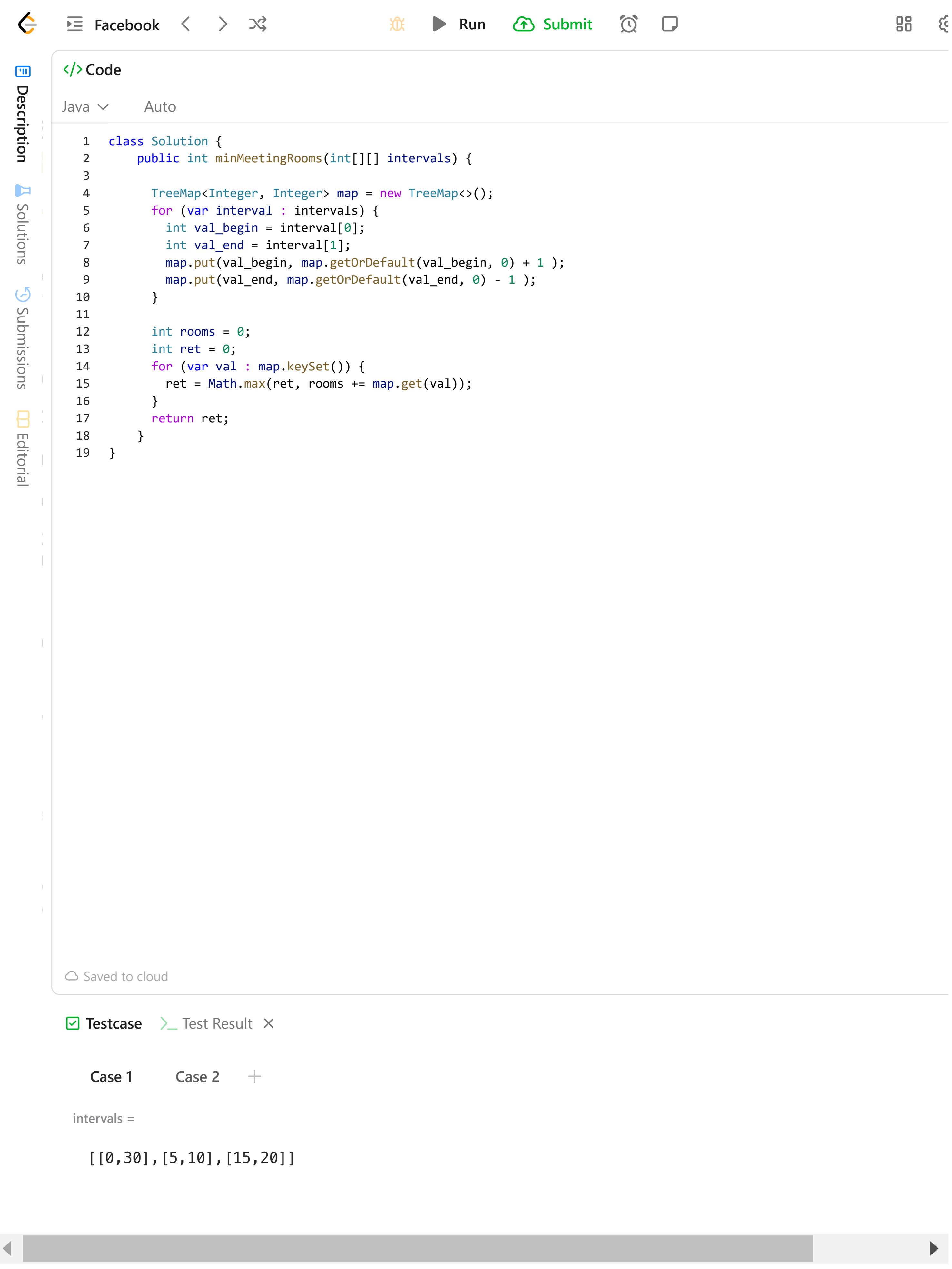
Hint 2

Hint 3

Hint 4

Similar Questions

Discussion (27)



```java
class Solution {
    public int minMeetingRooms(int[][] intervals) {

      TreeMap<Integer, Integer> map = new TreeMap<>();
      for (var interval : intervals) {
        int val_begin = interval[0];
        int val_end = interval[1];
        map.put(val_begin, map.getOrDefault(val_begin, 0) + 1 );
        map.put(val_end, map.getOrDefault(val_end, 0) - 1 );
      }

      int rooms = 0;
      int ret = 0;
      for (var val : map.keySet()) {
        ret = Math.max(ret, rooms += map.get(val));
      }
      return ret;
    }
}
```

intervals =

```
[[0,30],[5,10],[15,20]]
```

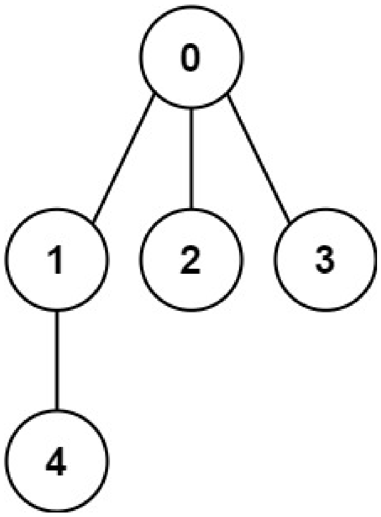
261. Graph Valid Tree Premium

Medium Topics Companies Hint

You have a graph of `n` nodes labeled from `0` to `n - 1`. You are given an integer n and a list of `edges` where `edges[i] = [a_i, b_i]` indica
undirected edge between nodes `a_i` and `b_i` in the graph.

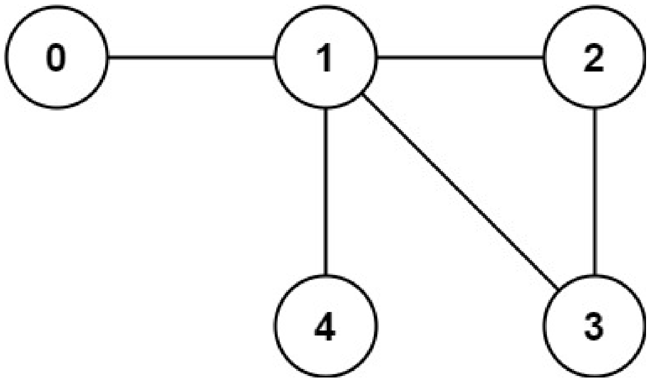
Return `true` *if the edges of the given graph make up a valid tree, and* `false` *otherwise.*

Example 1:



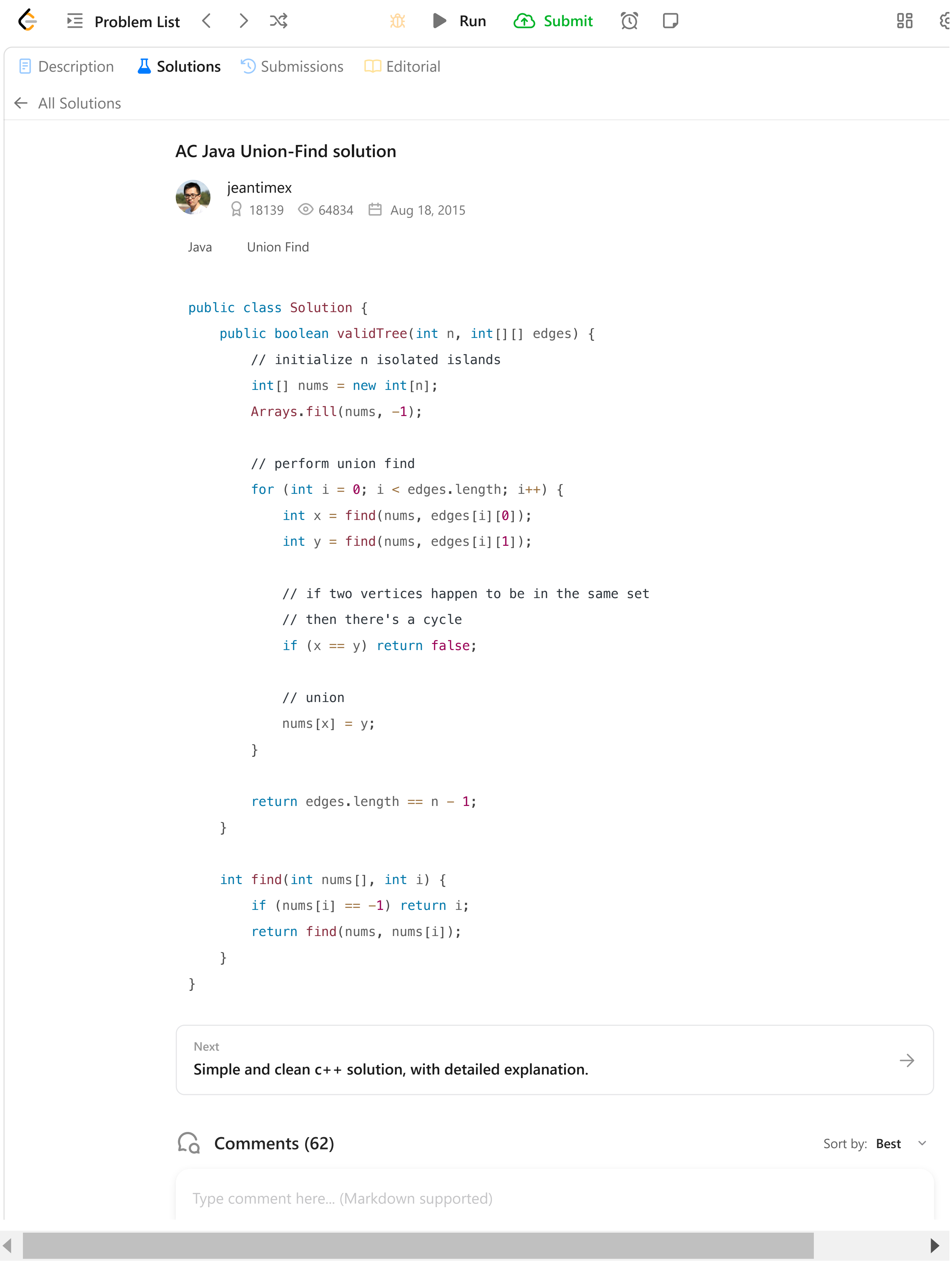
```
Input: n = 5, edges = [[0,1],[0,2],[0,3],[1,4]]
Output: true
```

Example 2:



```
Input: n = 5, edges = [[0,1],[1,2],[2,3],[1,3],[1,4]]
Output: false
```

Constraints:


Problem List

Run Submit

Description Solutions Submissions Editorial

All Solutions

AC Java Union-Find solution

jeantimex
18139 64834 Aug 18, 2015

Java Union Find

```
public class Solution {
    public boolean validTree(int n, int[][] edges) {
        // initialize n isolated islands
        int[] nums = new int[n];
        Arrays.fill(nums, -1);

        // perform union find
        for (int i = 0; i < edges.length; i++) {
            int x = find(nums, edges[i][0]);
            int y = find(nums, edges[i][1]);

            // if two vertices happen to be in the same set
            // then there's a cycle
            if (x == y) return false;

            // union
            nums[x] = y;
        }

        return edges.length == n - 1;
    }

    int find(int nums[], int i) {
        if (nums[i] == -1) return i;
        return find(nums, nums[i]);
    }
}
```

Next
Simple and clean c++ solution, with detailed explanation.

Comments (62) Sort by: Best

Type comment here... (Markdown supported)

Problem List Run Submit

Description Solutions Submissions Editorial

265. Paint House II Premium

Hard Topics Companies

There are a row of `n` houses, each house can be painted with one of the `k` colors. The cost of painting each house with a certain color is di
paint all the houses such that no two adjacent houses have the same color.

The cost of painting each house with a certain color is represented by an `n x k` cost matrix costs.

- For example, `costs[0][0]` is the cost of painting house `0` with color `0`; `costs[1][2]` is the cost of painting house `1` with color `2`, anc

Return *the minimum cost to paint all houses*.

Example 1:

```
Input: costs = [[1,5,3],[2,9,4]]
Output: 5
Explanation:
Paint house 0 into color 0, paint house 1 into color 2. Minimum cost: 1 + 4 = 5;
Or paint house 0 into color 2, paint house 1 into color 0. Minimum cost: 3 + 2 = 5.
```

Example 2:

```
Input: costs = [[1,3],[2,4]]
Output: 5
```

Constraints:

- `costs.length == n`
- `costs[i].length == k`
- `1 <= n <= 100`
- `2 <= k <= 20`
- `1 <= costs[i][j] <= 20`

Follow up: Could you solve it in `O(nk)` runtime?

Seen this question in a real interview before? 1/4

Yes No

Accepted 120.8K Submissions 221.7K Acceptance Rate 54.5%

The idea is similar to the problem Paint House I, for each house and each color, the minimum cost of painting the house with that color should be the minimum cost of painting previous houses, and make sure the previous house doesn't paint with the same color.

We can use min1 and min2 to track the indices of the 1st and 2nd smallest cost till previous house, if the current color's index is same as min1, then we have to go with min2, otherwise we can safely go with min1.

The code below modifies the value of costs[][] so we don't need extra space.

```
public int minCostII(int[][] costs) {
    if (costs == null || costs.length == 0) return 0;

    int n = costs.length, k = costs[0].length;
    // min1 is the index of the 1st-smallest cost till previous house
    // min2 is the index of the 2nd-smallest cost till previous house
    int min1 = -1, min2 = -1;

    for (int i = 0; i < n; i++) {
        int last1 = min1, last2 = min2;
        min1 = -1; min2 = -1;

        for (int j = 0; j < k; j++) {
            if (j != last1) {
                // current color j is different to last min1
                costs[i][j] += last1 < 0 ? 0 : costs[i - 1][last1];
            } else {
                costs[i][j] += last2 < 0 ? 0 : costs[i - 1][last2];
            }

            // find the indices of 1st and 2nd smallest cost of painting current house i
            if (min1 < 0 || costs[i][j] < costs[i][min1]) {
                min2 = min1; min1 = j;
            } else if (min2 < 0 || costs[i][j] < costs[i][min2]) {
                min2 = j;
            }
        }
    }

    return costs[n - 1][min1];
}
```

266. Palindrome Permutation Premium

Easy Topics Companies Hint

Given a string `s`, return `true` *if a permutation of the string could form a* ***palindrome*** *and* `false` *otherwise.*

Example 1:

```
Input: s = "code"
Output: false
```

Example 2:

```
Input: s = "aab"
Output: true
```

Example 3:

```
Input: s = "carerac"
Output: true
```

Constraints:

- `1 <= s.length <= 5000`
- `s` consists of only lowercase English letters.

Seen this question in a real interview before? 1/4

Yes No

Accepted 197K Submissions 294.2K Acceptance Rate 67.0%

Topics

Companies

Hint 1

Hint 2

Hint 3

Similar Questions

Java solution w/Set, one pass, without counters.

ammv
285 21895 Aug 21, 2015

The idea is to iterate over string, adding current character to `set` if `set` doesn't contain that character, or removing current character from `set` if `set` contains it.
When the iteration is finished, just return `set.size()==0 || set.size()==1`.

`set.size()==0` corresponds to the situation when there are even number of any character in the string, and
`set.size()==1` correspondsds to the fact that there are even number of any character except one.

```java
public class Solution {
    public boolean canPermutePalindrome(String s) {
        Set<Character> set=new HashSet<Character>();
        for(int i=0; i<s.length(); ++i){
            if (!set.contains(s.charAt(i)))
                set.add(s.charAt(i));
            else
                set.remove(s.charAt(i));
        }
        return set.size()==0 || set.size()==1;
    }
}
```

Next
1-4 lines Python, Ruby, C++, C, Java

Comments (30) Sort by: Best

Type comment here... (Markdown supported)

Preview Comment

StefanPochmann Aug 21, 2015

Just a shorter alternative:

Facebook Run Submit

Description Solutions Submissions Editorial

267. Palindrome Permutation II Premium

Medium Topics Companies Hint

Given a string s, return *all the palindromic permutations (without duplicates) of it*.

You may return the answer in **any order**. If `s` has no palindromic permutation, return an empty list.

Example 1:

```
Input: s = "aabb"
Output: ["abba","baab"]
```

Example 2:

```
Input: s = "abc"
Output: []
```

Constraints:

- `1 <= s.length <= 16`
- `s` consists of only lowercase English letters.

Seen this question in a real interview before? 1/4

Yes No

Accepted 65.2K Submissions 158.9K Acceptance Rate 41.1%

Topics

Companies

Hint 1

Hint 2

Similar Questions

Discussion (1)

Backtrack Summary: General Solution for 10 Questions!!!!!!!! Python (Combination Sum, Subsets, Permu...

dichen001
909 10844 Dec 24, 2016

For Java version, please refer to isssac3's answer.

If you find it helpful, please vote to let more people see this post. Besides, it would be great if you find other questions could be solved use this general solution. Please make a comment below.

39. Combination Sum
https://leetcode.com/problems/combination-sum/

```
def combinationSum(self, candidates, target):
    def backtrack(tmp, start, end, target):
        if target == 0:
            ans.append(tmp[:])
        elif target > 0:
            for i in range(start, end):
                tmp.append(candidates[i])
                backtrack(tmp, i, end, target - candidates[i])
                tmp.pop()
    ans = []
    candidates.sort(reverse= True)
    backtrack([], 0, len(candidates), target)
    return ans
```

40. Combination Sum II
https://leetcode.com/problems/combination-sum-ii/

```
def combinationSum2(self, candidates, target):
    def backtrack(start, end, tmp, target):
        if target == 0:
            ans.append(tmp[:])
        elif target > 0:
            for i in range(start, end):
                if i > start and candidates[i] == candidates[i-1]:
                    continue
                tmp.append(candidates[i])
                backtrack(i+1, end, tmp, target - candidates[i])
                tmp.pop()
    ans = []
```

```
                tmp.pop()
    ans = []
    candidates.sort(reverse= True)
    backtrack([], 0, len(candidates), target)
    return ans
```

40. Combination Sum II

https://leetcode.com/problems/combination-sum-ii/

```
def combinationSum2(self, candidates, target):
    def backtrack(start, end, tmp, target):
        if target == 0:
            ans.append(tmp[:])
        elif target > 0:
            for i in range(start, end):
                if i > start and candidates[i] == candidates[i-1]:
                    continue
                tmp.append(candidates[i])
                backtrack(i+1, end, tmp, target - candidates[i])
                tmp.pop()
    ans = []
    candidates.sort(reverse= True)
    backtrack(0, len(candidates), [], target)
    return ans
```

78. Subsets

https://leetcode.com/problems/subsets/

```
def subsets(self, nums):
    def backtrack(start, end, tmp):
        ans.append(tmp[:])
        for i in range(start, end):
            tmp.append(nums[i])
            backtrack(i+1, end, tmp)
            tmp.pop()
    ans = []
    backtrack(0, len(nums), [])
    return ans
```

90. Subsets II

https://leetcode.com/problems/subsets-ii/

```
            tmp.append(nums[i])
            backtrack(i+1, end, tmp)
            tmp.pop()
    ans = []
    backtrack(0, len(nums), [])
    return ans
```

90. Subsets II
https://leetcode.com/problems/subsets-ii/

```
def subsetsWithDup(self, nums):
    def backtrack(start, end, tmp):
        ans.append(tmp[:])
        for i in range(start, end):
            if i > start and nums[i] == nums[i-1]:
                continue
            tmp.append(nums[i])
            backtrack(i+1, end, tmp)
            tmp.pop()
    ans = []
    nums.sort()
    backtrack(0, len(nums), [])
    return ans
```

46. Permutations
https://leetcode.com/problems/permutations/

```
def permute(self, nums):
    def backtrack(start, end):
        if start == end:
            ans.append(nums[:])
        for i in range(start, end):
            nums[start], nums[i] = nums[i], nums[start]
            backtrack(start+1, end)
            nums[start], nums[i] = nums[i], nums[start]

    ans = []
    backtrack(0, len(nums))
    return ans
```

47. Permutations II

Facebook Run Submit

Description Solutions Submissions Editorial

All Solutions

47. Permutations II

https://leetcode.com/problems/permutations-ii/

```
def permuteUnique(self, nums):
    def backtrack(tmp, size):
        if len(tmp) == size:
            ans.append(tmp[:])
        else:
            for i in range(size):
                if visited[i] or (i > 0 and nums[i-1] == nums[i] and not visited[i-1]):
                    continue
                visited[i] = True
                tmp.append(nums[i])
                backtrack(tmp, size)
                tmp.pop()
                visited[i] = False
    ans = []
    visited = [False] * len(nums)
    nums.sort()
    backtrack([], len(nums))
    return ans
```

60. Permutation Sequence

https://leetcode.com/problems/permutation-sequence/

```
def getPermutation(self, n, k):
    nums = [str(i) for i in range(1, n+1)]
    fact = [1] * n
    for i in range(1,n):
        fact[i] = i*fact[i-1]
    k -= 1
    ans = []
    for i in range(n, 0, -1):
        id = k / fact[i-1]
        k %= fact[i-1]
        ans.append(nums[id])
        nums.pop(id)
    return ''.join(ans)
```

131. Palindrome Partitioning

131. Palindrome Partitioning
https://leetcode.com/problems/palindrome-partitioning/

```
def partition(self, s):
    def backtrack(start, end, tmp):
        if start == end:
            ans.append(tmp[:])
        for i in range(start, end):
            cur = s[start:i+1]
            if cur == cur[::-1]:
                tmp.append(cur)
                backtrack(i+1, end, tmp)
                tmp.pop()
    ans = []
    backtrack(0, len(s), [])
    return ans
```

---

267. Palindrome Permutation II
https://leetcode.com/problems/palindrome-permutation-ii/
Related to this two:
`31. Next Permutation` : https://leetcode.com/problems/next-permutation/
`266. Palindrome Permutation` : https://leetcode.com/problems/palindrome-permutation/

```
def generatePalindromes(self, s):
    kv = collections.Counter(s)
    mid = [k for k, v in kv.iteritems() if v%2]
    if len(mid) > 1:
        return []
    mid = '' if mid == [] else mid[0]
    half = ''.join([k * (v/2) for k, v in kv.iteritems()])
    half = [c for c in half]

    def backtrack(end, tmp):
        if len(tmp) == end:
            cur = ''.join(tmp)
            ans.append(cur + mid + cur[::-1])
        else:
            for i in range(end):
                if visited[i] or (i>0 and half[i] == half[i-1] and not visited[i-1]):
                    continue
```

Facebook Run Submit

Description Solutions Submissions Editorial

All Solutions

267. Palindrome Permutation II
https://leetcode.com/problems/palindrome-permutation-ii/
Related to this two:
`31. Next Permutation` : https://leetcode.com/problems/next-permutation/
`266. Palindrome Permutation` : https://leetcode.com/problems/palindrome-permutation/

```
    def generatePalindromes(self, s):
        kv = collections.Counter(s)
        mid = [k for k, v in kv.iteritems() if v%2]
        if len(mid) > 1:
            return []
        mid = '' if mid == [] else mid[0]
        half = ''.join([k * (v/2) for k, v in kv.iteritems()])
        half = [c for c in half]

        def backtrack(end, tmp):
            if len(tmp) == end:
                cur = ''.join(tmp)
                ans.append(cur + mid + cur[::-1])
            else:
                for i in range(end):
                    if visited[i] or (i>0 and half[i] == half[i-1] and not visited[i-1]):
                        continue
                    visited[i] = True
                    tmp.append(half[i])
                    backtrack(end, tmp)
                    visited[i] = False
                    tmp.pop()

        ans = []
        visited = [False] * len(half)
        backtrack(len(half), [])
        return ans
```

Next
AC Java solution with explanation

Comments (6)   Sort by: Best