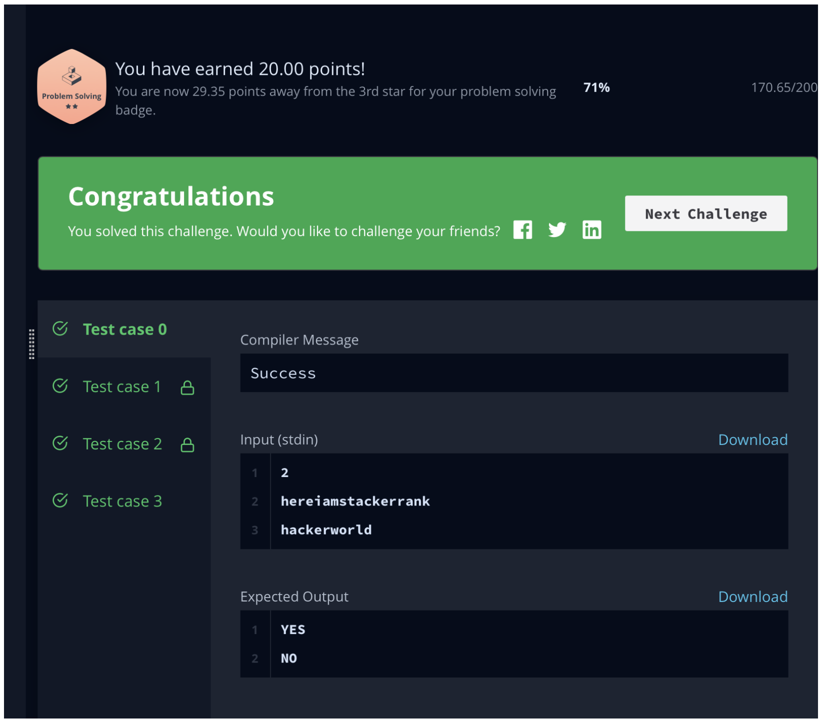


You have earned 20.00 points!

You are now 29.35 points away from the 3rd star for your problem solving badge.

71%

170.65/200

Congratulations

You solved this challenge. Would you like to challenge your friends?

Next Challenge

Test case 0

Test case 1

Test case 2

Test case 3

Compiler Message

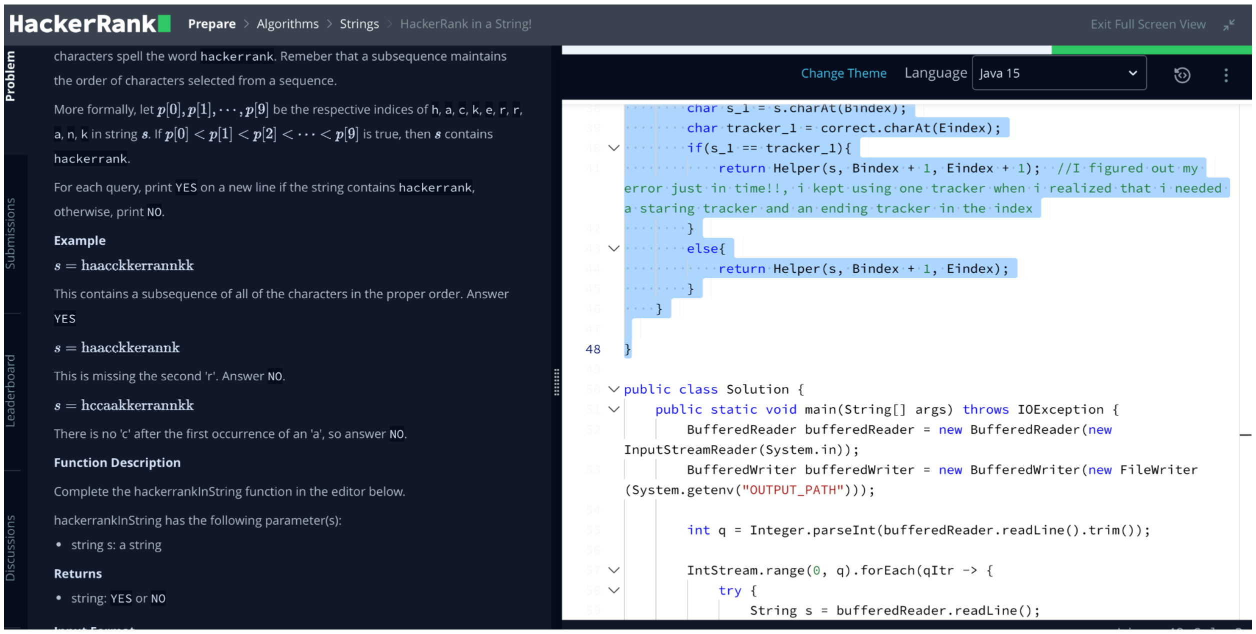
```
Success
```

Input (stdin) Download

```
2
hereiamstackerrank
hackerworld
```

Expected Output Download

```
YES
NO
```



HackerRank Prepare > Algorithms > Strings > HackerRank in a String!

Exit Full Screen View

characters spell the word `hackerrank`. Remeber that a subsequence maintains the order of characters selected from a sequence.

More formally, let $p[0], p[1], \cdots, p[9]$ be the respective indices of h, a, c, k, e, r, r, a, n, k in string s. If $p[0] < p[1] < p[2] < \cdots < p[9]$ is true, then s contains `hackerrank`.

For each query, print `YES` on a new line if the string contains `hackerrank`, otherwise, print `NO`.

Example

$s = \text{haacckkerrannkk}$

This contains a subsequence of all of the characters in the proper order. Answer `YES`

$s = \text{haacckkerannk}$

This is missing the second 'r'. Answer `NO`.

$s = \text{hccaakkerrannkk}$

There is no 'c' after the first occurrence of an 'a', so answer `NO`.

Function Description

Complete the hackerrankInString function in the editor below.

hackerrankInString has the following parameter(s):

- string s: a string

Returns

- string: `YES` or `NO`

Change Theme Language Java 15

```java
        char s_1 = s.charAt(Bindex);
        char tracker_1 = correct.charAt(Eindex);
        if(s_1 == tracker_1){
            return Helper(s, Bindex + 1, Eindex + 1);  //I figured out my error just in time!!, i kept using one tracker when i realized that i needed a staring tracker and an ending tracker in the index
        }
        else{
            return Helper(s, Bindex + 1, Eindex);
        }
    }

}

public class Solution {
    public static void main(String[] args) throws IOException {
        BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(System.in));
        BufferedWriter bufferedWriter = new BufferedWriter(new FileWriter(System.getenv("OUTPUT_PATH")));

        int q = Integer.parseInt(bufferedReader.readLine().trim());

        IntStream.range(0, q).forEach(qItr -> {
            try {
                String s = bufferedReader.readLine();
```

Time and space complexity

The time complexity in this code depends on “n” (I.e the number of inputs that the string s has), so it cannot be constant and there are no other dominating values that can overtake the linearithmic complexity.

Space complexity can be determined by the tracker parameters, since we are trying to find the String “hackerrank” then we are always tying to find that set amount of strings no matter what the parameters are in B and E, since it only wants to track the string Hackerrank. It is O(1)