467. Unique Substrings in Wraparound String

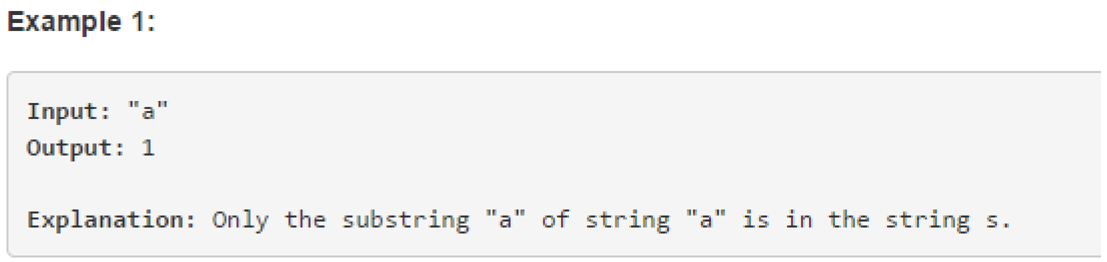
Add to List
QuestionEditorial Solution
My Submissions

- Total Accepted: **2082**
- Total Submissions: **7116**
- Difficulty: **Medium**
- Contributors: **ramanjot**

Consider the string s to be the infinite wraparound string of "abcdefghijklmnopqrstuvwxyz", so s will look like this: "...zabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcd....".

Now we have another string p. Your job is to find out how many unique non-empty substrings of p are present in s. In particular, your input is the string p and you need to output the number of different non-empty substrings of p in the string s.

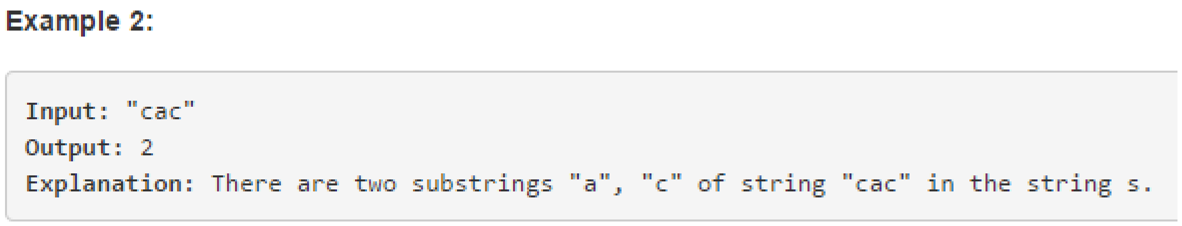
Note: p consists of only lowercase English letters and the size of p might be over 10000.



Example 1:

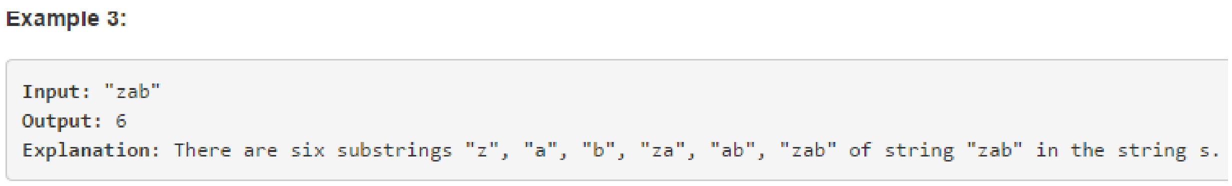
```
Input: "a"
Output: 1

Explanation: Only the substring "a" of string "a" is in the string s.
```



Example 2:

```
Input: "cac"
Output: 2
Explanation: There are two substrings "a", "c" of string "cac" in the string s.
```



Example 3:

```
Input: "zab"
Output: 6
Explanation: There are six substrings "z", "a", "b", "za", "ab", "zab" of string "zab" in the string s.
```

After failed with pure math solution and time out with DFS solution, I finally realized that this is a DP problem...
The idea is, if we know the max number of unique substrings in p ends with 'a', 'b', ..., 'z', then the summary of them is the answer. Why is that?

1. The max number of unique substring ends with a letter equals to the length of max contiguous substring ends with that letter. Example "abcd", the max number of unique substring ends with 'd' is 4, apparently they are "abcd", "bcd", "cd" and "d".
2. If there are overlapping, we only need to consider the longest one because it covers all the possible substrings. Example: "abcdbcd", the max number of unique substring ends with 'd' is 4 and all substrings formed by the 2nd "bcd" part are covered in the 4 substrings already.
3. No matter how long is a contiguous substring in p, it is in s since s has infinite length.
4. Now we know the max number of unique substrings in p ends with 'a', 'b', ..., 'z' and those substrings are all in s. Summary is the answer, according to the question.

```
//Java
public class Solution {
    public int findSubstringInWraproundString(String p) {
        // count[i] is the maximum unique substring end with ith letter.
        // 0 - 'a', 1 - 'b', ..., 25 - 'z'.
        int[] count = new int[26];

        // store longest contiguous substring ends at current position.
        int maxLengthCur = 0;

        for (int i = 0; i < p.length(); i++) {
            if (i > 0 && (p.charAt(i) - p.charAt(i - 1) == 1 || (p.charAt(i - 1) -
p.charAt(i) == 25))) {
                maxLengthCur++;
            }
            else {
                maxLengthCur = 1;
            }

            int index = p.charAt(i) - 'a';
            count[index] = Math.max(count[index], maxLengthCur);
        }

        // Sum to get result
        int sum = 0;
        for (int i = 0; i < 26; i++) {
            sum += count[i];
        }
        return sum;
    }
}

//C++
class Solution {
public:
    int findSubstringInWraproundString(string p) {
        //a 0 b 1 ... z 25
        vector<int> count(26,0);
        int maxLengthCur = 0;int ans=0;
        for(int i=0;i<p.size();++i)
        {
            if(i>0 && (p[i]-p[i-1]==1 || p[i-1]-p[i]==25))
            {
                maxLengthCur++;
            }else
            {
                maxLengthCur=1;
            }
            int index  = p[i]-'a';
            count[index] = max(count[index],maxLengthCur);
        }
        //sum
        for(int k=0;k<count.size();k++)
        ans+=count[k];
        return ans;
    }
};
```