L26 : String Algorithm

1-Tut : String Search

Send Feedback

You are given an string S and a test string T. You have to find whether string S is a substring of the string T.

Input Format:

First line of input will contain an integer T, representing the number of test cases
Each test case is as follows:
Line 1: contains the string S.
Line 2: contains the string T.

Constraints:

1 <= T <= 100
1 <= |S|, |T| <= 10^5

Output Format:

For each test case print "Yes" if S is present in T or "No" otherwise.

Sample Input 1:

2
ye
wnpnzijdi
ao
jaoalc

Sample Output 1:

No

Yes

```
#include <bits/stdc++.h>
using namespace std;

int *getLPS(string pattern)
{
    int n = pattern.length();
    int *lps = new int[n];
    lps[0] = 0;
    int i = 1;
    int j = 0;
    while (i < n)
    {
        if (pattern[i] == pattern[j])
        {
            lps[i] = j + 1;
            i++;
```

```
            j++;
        }
        else
        {
            if (j != 0)
            {
                j = lps[j - 1];
            }
            else
            {
                lps[i] = 0;
                i++;
            }
        }
    }
    return lps;
}

int findKMP(string text, string pattern)
{
    int *lps = getLPS(pattern);
    int patternLen = pattern.length();
    int textLen = text.length();
    int i = 0, j = 0;
    while (i < textLen && j < patternLen)
    {
        if (text[i] == pattern[j])
        {
            i++;
            j++;
        }
        else
        {
            if (j != 0)
            {
                j = lps[j - 1];
            }
            else
            {
                i++;
            }
        }
    }
    return j == patternLen ? i - j : -1;
```

```
61. }
62.
63. int main()
64. {
65.
66.     int t;
67.     cin >> t;
68.     while (t--)
69.     {
70.
71.         string S, T;
72.         cin >> S;
73.         cin >> T;
74.
75.         if (findKMP(T, S) == -1)
76.         {
77.             cout << "No" << endl;
78.         }
79.         else
80.         {
81.             cout << "Yes" << endl;
82.         }
83.     }
84.
85.     return 0;
86. }
```

2-Tut : Palindromic Substrings

Send Feedback

Given a string S, count and return the number of substrings of S that are a palindrome.

Single length substrings are also palindromes. You just need to print the count of palindromic substrings, not the actual substrings.

Input Format:

First line of input contains an integer T, representing the number of test cases.
Next T lines contain the string S

Constraints :

1 <= Length of S <= 2000

Output Format :

For each test case, print the count of palindrome substrings in a new line.

Sample Input 1:

1
aba

Sample Output 1: 4

Explanation:

The 4 palindrome substrings are "a", "b", "a" and "aba".

```
#include <bits/stdc++.h>
using namespace std;

int size(char s[])
{
    int n = 0;
    while (s[n] != '\0')
    {
        n++;
    }
    return n;
}
int countPalindromeSubstrings(string s)
{
    int count = 0;
    int n = s.length();
    for (int i = 0; i < n; i++)
    {
        // odd length
        int left = i;
        int right = i;
        while (right < n && left >= 0 && s[left] == s[right])
        {
            count++;
            left--;
            right++;
        }
        // even length
        left = i;
        right = i + 1;
        while (left >= 0 && right < n && s[left] == s[right])
        {
            count++;
            left--;
            right++;
        }
    }
    return count;
}
int main()
{
```

```
42.
43.     // write your code here
44.     int t;
45.     cin >> t;
46.     while (t--)
47.     {
48.
49.         string s;
50.         cin >> s;
51.
52.         cout << countPalindromeSubstrings(s) << endl;
53.     }
54.     return 0;
55. }
```

3-Ass : Longest Palindromic Substring

Send Feedback

You are given a string S .You have to find the length of the longest palindromic substring of S.

Input Format:

First line of input contains the string S.

Constraints:

1 <= |S| <= 4*10^6

Output Format:

You have to print the length of longest palindromic substring

Sample Input 1:

zkpbhxkmauuamkxsyi

Sample Output 1:

10

Explanation:

In the given sample test case, the longest palindromic substring is: xkmauuamkx.

```
1.  #include <bits/stdc++.h>
2.  using namespace std;
3.  int lps(string s)
4.  {
5.      int max = 0;
6.      int n = s.length();
7.      for (int i = 0; i < n; i++)
8.      {
9.          // odd length
10.         int left = i;
11.         int right = i;
12.         while (right < n && left >= 0 && s[left] == s[right])
```

```
        {
            int current_length = right - left + 1;
            if (current_length > max)
            {
                max = current_length;
            }
            left--;
            right++;
        }
        // even length
        left = i;
        right = i + 1;
        while (left >= 0 && right < n && s[left] == s[right])
        {
            int current_length = right - left + 1;
            if (current_length > max)
            {
                max = current_length;
            }
            left--;
            right++;
        }
    }
    return max;
}
int main()
{
    string s;
    cin >> s;
    cout << lps(s) << endl;
    return 0;
}
```

4-Ass : Red Scrabble

Send Feedback

Raymond “Red” Reddington is an international criminal hunted by the police forces of many countries. Recently, a joint Task Force, with the sole purpose of capturing Reddington, is launched, led by Agent Donald Ressler.

Red has managed to elude all the forces because he is always one step ahead. Before Ressler can catch him, Red manages to capture Ressler for interrogation.

Red is a huge fan of the game scrabble, and has created many modified versions of the game. He will let Ressler go, if he is able to solve one such version of the game.

The premise is quite simple. Ressler is given a string S which contains only digits. He needs to count the number of substrings of S, which are palindromes. There are some additional rules to be followed. Red explains them as follows -

Ressler needs to count the number of substrings of S, which are palindromes without leading zeros and can be divided by 3 without a remainder.

A string is a palindrome without leading zeros if it reads the same backward as forward and doesn't start with the symbol '0'.

Ressler should consider string "0" as a palindrome without leading zeros.

You need to help Agent Ressler.

Input Format:

The first and only line of input contains string S.

Constraints:

1 <= |S| <= 10^5

0 <= S[i] <= 9

Output Format:

Print the answer obtained.

Sample Input 1:

10686

Sample Output 1:

3

Explanation:

The three palindromic substrings are: 0, 6, 6.

```
#include <bits/stdc++.h>
using namespace std;

#define ll long long

int cnt[2000005][3];
int sum[2000005];
string s;
ll answer = 0;

string daniTrick(string s)
{
    string dani = "";
    int n = s.length();
    for (int i = 0; i < n; ++i)
    {
        dani += '@';
        dani += s[i];
    }
```

```
    return dani + '@';
}

int cal(int l, int r, int y)
{
    if (l > r)
        return 0;
    if (l > 0)
    {
        int total = (sum[l - 1] + y) % 3;
        return cnt[r][total] - cnt[l - 1][total];
    }
    return cnt[r][y];
}

void prepare(string s)
{
    memset(cnt, 0, sizeof(cnt));
    memset(sum, 0, sizeof(sum));
    for (int i = 1; i < s.length(); ++i)
    {
        sum[i] = sum[i - 1];
        if (s[i] != '@')
        {
            sum[i] += (s[i] - '0');
            sum[i] %= 3;
        }
        for (int j = 0; j < 3; ++j)
        {
            cnt[i][j] = cnt[i - 1][j];
        }
        if (s[i] != '@' && s[i] != '0')
            cnt[i][sum[i]]++;
    }
}

int main()
{

    cin >> s;
    s = daniTrick(s);
    int n = s.length();
    vector<int> dp(n, 0);
    prepare(s);
```

```
64.
65.    int l = 0, r = -1;
66.
67.    for (int i = 0; i < n; ++i)
68.    {
69.       int k;
70.       if (i > r)
71.          k = 1;
72.       else
73.          k = min(r - i, dp[r - i + l]);
74.       while (i - k >= 0 && i + k < n && s[i - k] == s[i + k])
75.          k++;
76.       k--;
77.       dp[i] = k;
78.       if (i + k > r)
79.       {
80.          r = i + k;
81.          l = i - k;
82.       }
83.       if (s[i] != '@')
84.       {
85.          answer += cal(i + 1, i + k, (s[i] - '0') % 3);
86.          if ((s[i] - '0') % 3 == 0)
87.             ++answer;
88.       }
89.       else
90.       {
91.          answer += cal(i + 1, i + k, 0);
92.       }
93.    }
94.    cout << answer << endl;
95.
96.    return 0;
97. }
```

5-Ass : Ready Player S?

Send Feedback

OASIS is a virtual reality created by the legendary James Halliday. After Halliday's death, a pre-recorded message left by him announced a game, which would grant the ownership and control of the OASIS to the first player who finds the Golden Easter Egg within it.

Shivali, an OASIS player, is obsessed with the game and finding the Easter Egg. But she has to fight the IOI clan, who wants to control OASIS for evil purposes. Both of them gather troops of different types and form a big army to fight each other.

IOI has N troops of lowercase letters forming a huge army. Shivali has an army of length M.

She will win, if the first k troops she takes from her army, can kill any of the k consecutive troops of the IOI army.

Remember a troop of type 'a' can only kill a troop of type 'a'.

You have to find how many times she can win.

Input Format:

The first line of input contains N, M and k, space separated.
Next two lines contains the string of troops of length N and M respectively in lowercase letters.

Constraints:

1 <= N, M <= 10^6
1 <= K <= M

Output Format:

Output the number of wins she is going to take at the end of the day. Print -1 if she can't win.

Sample Input 1:

3 2 1
bbb
bb

Sample Output 1:

3

```
#include <bits/stdc++.h>
using namespace std;

void computeLPSArray(string pat, int M,
                     int lps[])
{

    int len = 0;
    int i = 1;
    lps[0] = 0;
    while (i < M)
    {
        if (pat[i] == pat[len])
        {
            len++;
            lps[i] = len;
            i++;
        }
        else
        {
            if (len != 0)
            {
```

```
            len = lps[len - 1];
        }
        else
        {
            lps[i] = len;
            i++;
        }
    }
}
}

int KMPSearch(string pat, string txt)
{
    int M = pat.length();
    int N = txt.length();

    int lps[M];
    int j = 0;

    computeLPSArray(pat, M, lps);

    int i = 0;
    int res = 0;
    int next_i = 0;

    while (i < N)
    {
        if (pat[j] == txt[i])
        {
            j++;
            i++;
        }
        if (j == M)
        {
            j = lps[j - 1];
            res++;
        }

        // Mismatch after j matches
        else if (i < N && pat[j] != txt[i])
        {

            if (j != 0)
                j = lps[j - 1];
```

```
            else
                i = i + 1;
        }
    }
    return res;
}

int main()
{
    int N, M, k;
    cin >> N >> M >> k;
    string txt;
    string pat;
    cin >> txt;
    cin >> pat;
    string pat2 = pat.substr(0, k);
    int ans = KMPSearch(pat2, txt);
    if (ans == 0)
    {
        ans = -1;
    }

    cout << ans;

    return 0;
}
```

6-Ass : Red Riding Hood's Adventure

Send Feedback

Little Red Riding Hood has to go to her grandmother's house. Her grandmother is very sick. To reach her house, Red Riding Hood has to cross a very long dark forest. She is also carrying a basket full of food for her journey. The forest was much longer than anticipated, and Riding Hood decided to take rest. She finds a cave and lights a campfire inside it to keep herself warm for the night. But alas! The campfire attracts the Big Bad Wolf.

Luckily the Big Bad Wolf has recently turned vegetarian, and will not eat Red Riding Hood. He takes all the food that she was carrying instead, but decided to keep her as a minion. She cries a lot and begs the wolf to let her go. Big Bad Wolf used to be a problem solver in the past, and decides to let her go if and only if she is able to solve a problem for him. He gives the Red Riding Hood a long sentence "X" and a small word "W" . She has to find how many times word "W" occurs as a substring of "X" (spaces in the sentence are not to be considered). Red Riding Hood is just a kid, you have to help her solve the problem and escape the Big Bad Wolf.

Input Format :

First line of input will contain T(number of test cases). Each test case follows.
Line 1: contains the sentence X
Line 2: contains the string W

Constraints:

1 <= T <= 100
1 <= |S| ,|s| <= 10^5

Output Format:

For each test case print the answer in a new line.

Sample Input 1:

```
4
axb ycz d
abc
ab  cab cabc abc
abc
aab acbaa bbaaz
aab
aaaaaa
aa
```

Sample Output 1:

```
0
4
2

5
```

```
#include <bits/stdc++.h>
using namespace std;

#define MAXN 100001

void computeLPSArray(string pat, int M,
                     int lps[])
{

    int len = 0;
    int i = 1;
    lps[0] = 0;
    while (i < M)
    {
        if (pat[i] == pat[len])
        {
            len++;
            lps[i] = len;
            i++;
        }
```

```
21.         else
22.         {
23.             if (len != 0)
24.             {
25.                 len = lps[len - 1];
26.             }
27.             else
28.             {
29.                 lps[i] = len;
30.                 i++;
31.             }
32.         }
33.     }
34. }
35.
36. int KMPSearch(string pat, string txt)
37. {
38.     int M = pat.length();
39.     int N = txt.length();
40.
41.     int lps[M];
42.     int j = 0;
43.
44.     computeLPSArray(pat, M, lps);
45.
46.     int i = 0;
47.     int res = 0;
48.     int next_i = 0;
49.
50.     while (i < N)
51.     {
52.         if (pat[j] == txt[i])
53.         {
54.             j++;
55.             i++;
56.         }
57.         if (j == M)
58.         {
59.             j = lps[j - 1];
60.             res++;
61.         }
62.
63.         // Mismatch after j matches
64.         else if (i < N && pat[j] != txt[i])
```

```
65.         {
66.
67.             if (j != 0)
68.                 j = lps[j - 1];
69.             else
70.                 i = i + 1;
71.         }
72.     }
73.     return res;
74. }
75.
76. int main()
77. {
78.
79.     // write your code here
80.     int t;
81.     cin >> t;
82.     cin.clear();
83.
84.     while (t--)
85.     {
86.
87.         cin.ignore();
88.
89.         char X[MAXN];
90.         string W;
91.
92.         string X2;
93.         string W2;
94.
95.         cin.getline(X, MAXN);
96.         cin >> W;
97.
98.         int j = 0;
99.         for (int i = 0; X[i] != '\0'; i++)
100.             {
101.                 if (X[i] != ' ')
102.                     X2 += X[i];
103.                 else
104.                     continue;
105.             }
106.             X2 += '\0';
107.
108.             int ans = KMPSearch(W, X2);
```

```
109.
110.            cout << ans << endl;
111.        }
112.        return 0;
113.    }
```