



ALL TRACKS > ALGORITHMS > STRING ALGORITHMS > > PROBLEM

Little bear and strings

Attempted by: 294 / Accuracy: 88% / ★★★★★

Tag(s): Algorithms, Hard



PROBLEM

EDITORIAL

MY SUBMISSIONS

ANALYTICS

The problem "Little bear and strings" doesn't have any editorial. You can contribute it by sending editorial in markdown format to moderator@hackerearth.com.

Author Solution by [Ankit Srivastava](#)

```
1. import java.io.*;
2. import java.util.*;
3.
4. class ANKSTR01 {
5.     static int[] occ;
6.     static String s;
7.     static String[] strs = new String[2];
8.
9.     static ArrayList<Integer> soccs;
10.    static BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
11.
12.    public static void main(String[] args) throws IOException {
13.        try{
14.            while (true) go();
15.        } catch (Exception e) {
16.
17.        }
18.    }
19.
20.    static void go() throws IOException {
21.        //System.out.println("called");
22.
23.        s = br.readLine();
24.        strs[0] = br.readLine();
25.        strs[1] = br.readLine();
26.        //System.out.println(s);
27.
28.        int[] sa = suffixArray(s);
29.        int[] lcp = lcp(sa, s);
30.
31.        soccs = new ArrayList<Integer>();
32.        occ = new int[s.length()];
33.
34.        markOccurrences();
```

```

35.
36.     long ans = 0;
37.
38.     /*System.out.println(soccs);
39.     System.out.println(Arrays.toString(sa));
40.     System.out.println(Arrays.toString(lcp));*/
41.
42.     int m = Math.max(strs[1].length() - 1, strs[0].length() - 1);
43.
44.     for (int i = 0; i < sa.length; i++) {
45.         int least = sa[i] + m;
46.         if(i > 0) least = Math.max(sa[i] + lcp[i - 1], least);
47.         if(occ[sa[i]] == 1) {
48.             int ind = Collections.binarySearch(soccs, least);
49.             if(ind < 0) ind = -ind - 1;
50.             ans += soccs.size() - ind;
51.         }
52.     }
53.     System.out.println(ans);
54. }
55.
56. public static int[] suffixArray(final CharSequence str) {
57.     int n = str.length();
58.     Integer[] order = new Integer[n];
59.     for (int i = 0; i < n; i++)
60.         order[i] = n - 1 - i;
61.
62.     Arrays.sort(order, new Comparator<Integer>() {
63.
64.         public int compare(Integer o1, Integer o2) {
65.             return str.charAt(o1) - str.charAt(o2);
66.         }
67.     });
68.
69.     // sa[i] - suffix on i'th position after sorting by first len char
70.     // rank[i] - position of the i'th suffix after sorting by first len
71.     int[] sa = new int[n];
72.     int[] rank = new int[n];
73.     for (int i = 0; i < n; i++) {
74.         sa[i] = order[i];
75.         rank[i] = str.charAt(i);
76.     }
77.
78.     for (int len = 1; len < n; len *= 2) {
79.         int[] r = rank.clone();
80.         for (int i = 0; i < n; i++) {
81.             // condition s1 + len < n simulates 0-symbol at the end of
82.             // a separate class is created for each suffix followed by
83.             rank[sa[i]] = i > 0 && r[sa[i] - 1] == r[sa[i]] && sa[i] -
84.         }
85.         // Suffixes are already sorted by first len characters
86.         // Now sort suffixes by first len * 2 characters
87.         int[] cnt = new int[n];
88.         for (int i = 0; i < n; i++)

```

```

89.         cnt[i] = i;
90.         int[] s = sa.clone();
91.         for (int i = 0; i < n; i++) {
92.             // s[i] - order of suffixes sorted by first len characters
93.             // (s[i] - len) - order of suffixes sorted only by second
94.             int s1 = s[i] - len;
95.             // sort only suffixes of length > len, others are already
96.             if (s1 >= 0)
97.                 sa[cnt[rank[s1]]++] = s1;
98.         }
99.     }
100.    return sa;
101. }
102.
103. // longest common prefixes array in O(n)
104. public static int[] lcp(int[] sa, CharSequence s) {
105.     int n = sa.length;
106.     int[] rank = new int[n];
107.     for (int i = 0; i < n; i++)
108.         rank[sa[i]] = i;
109.     int[] lcp = new int[n - 1];
110.     for (int i = 0, h = 0; i < n; i++) {
111.         if (rank[i] < n - 1) {
112.             int j = sa[rank[i] + 1];
113.             while (Math.max(i, j) + h < s.length() && s.charAt(i + h)
114.                 == s.charAt(j + h))
115.                 ++h;
116.             lcp[rank[i]] = h;
117.             if (h > 0)
118.                 --h;
119.         }
120.     }
121.     return lcp;
122. }
123.
124. static void markOccurrences() {
125.     StringBuilder builder = new StringBuilder();
126.     builder.append(strs[0]).append('#').append(s);
127.     int[] z1 = getZfunc(builder.toString());
128.     builder = new StringBuilder();
129.     builder.append(strs[1]).append('#').append(s);
130.     int[] z2 = getZfunc(builder.toString());
131.     int l1 = strs[0].length(), l2 = strs[1].length();
132.     for (int i = l1 + 1; i < z1.length; i++) {
133.         if (z1[i] == l1) occ[i - l1 - 1] = 1;
134.     }
135.     for (int i = l2 + 1; i < z2.length; i++) {
136.         if (z2[i] == l2) soccs.add(i - l2 - 1 + l2 - 1);
137.     }
138. }
139.
140.
141. public static int[] getZfunc(String str) {
142.     int L = 0, R = 0;

```

```
143.     char[] s = str.toCharArray();
144.     int n = s.length;
145.     int[] z = new int[n];
146.     z[0] = n;
147.     for (int i = 1; i < n; i++) {
148.         if (i > R) {
149.             L = R = i;
150.             while (R < n && s[R - L] == s[R]) R++;
151.             z[i] = R - L;
152.             R--;
153.         } else {
154.             int k = i - L;
155.             if (z[k] < R - i + 1) z[i] = z[k];
156.             else {
157.                 L = i;
158.                 while (R < n && s[R - L] == s[R]) R++;
159.                 z[i] = R - L;
160.                 R--;
161.             }
162.         }
163.     }
164.     return z;
165. }
166. }
```

COMMENTS (0)



Start Discussion...

Cancel

Post

ABOUT US

[Blog](#)

[Engineering Blog](#)

[Updates & Releases](#)

[Team](#)

[Careers](#)

HACKEREARTH

[API](#)

[Chrome Extension](#)

[CodeTable](#)

[HackerEarth Academy](#)

[Developer Profile](#)

DEVELOPERS

[AMA](#)

[Code Monk](#)

[Judge Environment](#)

[Solution Guide](#)

[Problem Setter Guide](#)

[In the Press](#)[Resume](#)[Practice Problems](#)[Get Badges](#)[HackerEarth Challenges](#)[Campus Ambassadors](#)[College Challenges](#)[Get Me Hired](#)[College Ranking](#)[Privacy](#)[Organise Hackathon](#)[Terms of Service](#)[Hackathon Handbook](#)[Competitive Programming](#)[Open Source](#)

EMPLOYERS

[Developer Sourcing](#)[Lateral Hiring](#)[Campus Hiring](#)[Hackathons](#)[FAQs](#)[Customers](#)

REACH US



IIIrd Floor, Salarpuria Business Center,
4th B Cross Road, 5th A Block,
Koramangala Industrial Layout,
Bangalore, Karnataka 560095, India.

✉ contact@hackerearth.com

☎ +91-80-4155-4695

☎ +1-650-461-4192



© 2016 HackerEarth