beecrowd

EN VIVO
Mira los envíos en tiempo real.

LISTAR
Listar todos tus envíos.

INTENTADO
Problemas aún no resueltos.

PREGUNTAS FRECUENTES
¿Necesitas ayuda?

RESPUESTAS
¿Qué significa esto?

FORO
Obtén ayuda para resolver.

CÓDIGO FUENTE

EDITAR Y ENVIAR

MIRA EL CÓDIGO FUENTE DE TU ENVÍO, MÁS ALGUNOS DETALLES ADICIONALES.

ENVÍO # 25675295

PROBLEMA:	1530 - How Many Substrings?
RESPUESTA:	**Time limit exceeded**
LENGUAJE:	C++17 (g++ 7.3.0, -std=c++17 -O2 -lm) [+0s]
TIEMPO:	2.000s
TAMAÑO DE ARCHIVO:	2,65 KB
MEMORIA:	-
ENVIO:	15/12/21 3:47:33

CÓDIGO FUENTE

```
1  //https://www.beecrowd.com.br/judge/es/problems/view/1530
2  #include <bits/stdc++.h>
3  using namespace std;
4   #define MAX_CHARS 200000
5  #define ALPHABET_SIZE 26
6
7  struct State
8  {
9      int len;
10     int link;
11     int next[ALPHABET_SIZE];
12 };
13
14 class SuffixAutomata
15 {
16     State *m_state;
17     int m_stateCount;
18     int m_last;
19
20 public:
21     SuffixAutomata(unsigned int maxStates);
22     void extend(int sym);
23     void reset();
24     int64_t substringCount() const;
25 };
26
27
28 SuffixAutomata::SuffixAutomata(unsigned int maxStates)
29 {
30     m_state = new State[maxStates];
31     reset();
32 }
33
34 void SuffixAutomata::extend(int sym)
35 {
36     int cur = m_stateCount++;
37     m_state[cur].len = m_state[m_last].len+1;
38     memset(m_state[cur].next,0,sizeof(int)*ALPHABET_SIZE);
39
40     int p;
41     for(p = m_last; p != -1 && m_state[p].next[sym] == 0; p = m_state[p].link) {
42         m_state[p].next[sym] = cur;
43     }
44
45     if(p == -1) {
46         m_state[cur].link = 0;
47     }
48     else {
49         int q = m_state[p].next[sym];
50         if(m_state[p].len + 1 == m_state[q].len)
51             m_state[cur].link = q;
52         else {
53             int clone = m_stateCount++;
54             m_state[clone].len = m_state[p].len + 1;
                emcpy(m_state[clone].next,m_state[q].next,sizeof(int)*ALPHABET_SIZE);
               state[clone].link = m_state[q].link;

               r(; p != -1 && m_state[p].next[sym] == q; p = m_state[p].link) {
                   m_state[p].next[sym] = clone;


62             m_state[q].link = m_state[cur].link = clone;
```

```
65
66      m_last = cur;
67  }
68
69
70  int64_t SuffixAutomata::substringCount() const
71  {
72      int64_t count = 0;
73      int stateCount = m_stateCount;
74      for(int i = 1; i < stateCount; ++i) {
75          count += m_state[i].len - m_state[m_state[i].link].len;
76      }
77      return count;
78  }
79
80  void SuffixAutomata::reset()
81  {
82      m_state[0].len = 0;
83      m_state[0].link = -1;
84      memset(m_state[0].next,0,sizeof(int)*ALPHABET_SIZE);
85      m_stateCount = 1;
86      m_last = 0;
87  }
88
89  int main(int argc, char *argv[]) {
90
91      char input[MAX_CHARS+1];
92      int64_t substrCount;
93
94      SuffixAutomata sa(MAX_CHARS*2);
95
96      while(scanf("%s",input) != EOF)
97      {
98          getchar();
99          const char *p = input;
100         substrCount = 0;
101
102         while(*p) {
103             if(*p >= 'a' && *p <= 'z') {
104                 int sym = (int)(*p - 'a');
105                 sa.extend(sym);
106                 substrCount += sa.substringCount();
107             }
108             else if(*p == '?') {
109                 printf("%llu\n",substrCount);
110             }
111             ++p;
112         }
113
114         sa.reset();
115     }
116     return 0;
117 }
```