



# Stringonomics

by [abizer123](#)

Problem

Submissions

Leaderboard

Discussions

You are given a string  $S$  consisting of lowercase English letters.

You are given another string  $P$ , that may or may not exist as a substring in  $S$ .

Given  $Q$  queries, where each query is of the form  $X$  and  $C$ , where  $X$  is a non-negative integer and  $C$  a character. For every query, you are supposed to change the index  $X$  (Assume 0 based indexing) of the string  $S$  to the character  $C$ .

You are supposed to find the minimum number of queries, when sequentially executed from the start, after which the string  $P$  no longer exists as a substring in  $S$ .

If the string  $P$  exists in  $S$  even after executing all the queries, print -1.

It is guaranteed that each index is only changed once, and once the string  $P$  ceases to exist in  $S$ , it would not reappear again later (If  $P$  never existed in  $S$ , it would not appear again later).

Your device is connected.

## Input Format

First line contains  $T$  number of testcases. For each testcase:

- The first line contains the string  $S$ .
- Next line contains the string  $P$ .
- Next line contains number of queries  $Q$ .
- Following  $Q$  lines contains  $X$  and  $C$ , space separated.

## Constraints

- $1 \leq T \leq 50$
- $1 \leq |S| \leq 2 \cdot 10^5$
- $1 \leq |P| \leq |S|$
- $1 \leq Q \leq |S|$
- $0 \leq X < |S|$
- $C$  is a lowercase English letter
- Sum of  $|S|$  and  $|P|$  over all  $T \leq 7 \cdot 10^5$
- Sum of  $Q$  over all  $T \leq 7 \cdot 10^5$

## Output Format

For each testcase  $T$ , output a single integer denoting the minimum number of queries after which the string  $P$  ceases to exist in string  $S$ .

## Sample Input 0

```
2
abcde
bc
3
0 p
1 q
2 w
abcde
cde
2
0 t
1 z
```

### Sample Output 0

```
2
-1
```

### Explanation 0

#### First test case

- Initial - *a[bc]de*
- First update - *p[bc]de*
- Second update - *pqcde*  
So after second update the string P is not in string S , so the answer is 2.

#### Second test case

- Initial - *ab[cde]*
- First update - *tb[cde]* Your device is connected.
- Second update - *tz[cde]*  
Even, after all the updates the string P is in the string S , so the answer is -1.

[f](#) [t](#) [in](#)

Contest ends in 4 hours

Submissions: 479



Max Score: 60

Difficulty: Hard

Rate This Challenge:

☆☆☆☆☆

[More](#)

Current Buffer (saved locally, editable)  

C++14



```
1▼ #include <bits/stdc++.h>
2
3 using namespace std;
4
5 string ltrim(const string &);
6 string rtrim(const string &);
7 vector<string> split(const string &);
8
9 bool precomputed = false;
10 vector<int> preprocessed_info;
11 set<int> indexes;
12 void precompute(string &pattern)
13▼ {
14     int i=1, len = 0;
15▼     preprocessed_info[0] = 0;
16     while(i<pattern.size())
17▼     {
```

```

18▼         if(pattern[i]==pattern[len])
19▼         {
20▼             preprocessed_info[i++] = len++;
21         }
22         else
23▼         {
24             if(len!=0)
25▼                 len = preprocessed_info[len-1];
26             else
27▼                 preprocessed_info[i++]=0;
28         }
29     }
30
31 }
32
33 void KMP(string &txt,string &pat)
34▼{
35     preprocessed_info.resize(pat.length(),0);
36     precomputed = true;
37     precompute(pat);
38     indexes.clear();
39     int i=0,j=0;
40     while(i<txt.size())
41▼     {
42▼         if(pat[j]==txt[i])
43             i++,j++;
44         if(j==pat.length())
45▼         {
46             indexes.insert(i-1);
47▼             j = preprocessed_info[j-1];
48         }
49         else if(i<txt.length() && pat[j]!=txt[i])
50▼         {
51             if(j!=0)
52▼                 : Your device is connected.
53             else
54                 i++;
55         }
56     }
57 }
58
59 int main()
60▼{
61     string t_temp;
62     getline(cin, t_temp);
63
64     int t = stoi(ltrim(rtrim(t_temp)));
65
66▼    for (int t_itr = 0; t_itr < t; t_itr++) {
67        string s;
68        precomputed = false;
69        getline(cin, s);
70
71        string p;
72        getline(cin, p);
73
74        string q_temp;
75        getline(cin, q_temp);
76
77        KMP(s,p);
78
79        int q = stoi(ltrim(rtrim(q_temp)));
80        bool isPresent = true;
81        int count = 0;
82
83▼        for (int q_itr = 0; q_itr < q; q_itr++) {
84
85            string first_multiple_input_temp;
86            getline(cin, first_multiple_input_temp);
87            if(!isPresent)
88▼            {
89                count++;
90                continue;

```

```

91     }
92     vector<string> first_multiple_input = split(rtrim(first_multiple_input_temp));
93
94     int x = stoi(first_multiple_input[0]);
95
96     char c = first_multiple_input[1][0];
97
98     if(s[x]==c)
99         continue;
100    else
101    {
102        s[x] = c;
103        auto it = indexes.lower_bound(x);
104        if(it==indexes.end() || (*it)-p.length()+1>x)
105            continue;
106        auto it2 = it;
107        while((*it2)-p.length()+1<=x && it2!=indexes.end())
108        {
109            it2++;
110        }
111        indexes.erase(it,it2);
112        s[x] = c;
113    }
114
115    if(indexes.size()==0)
116    {
117        KMP(s,p);
118        if(indexes.size()==0)
119            isPresent = false;
120    }
121    if(!isPresent)
122    {
123        count++;
124        continue;
125    }
126    Your device is connected.
127 }
128 cout<<((count>0)?count:-1)<<endl;
129 }
130
131 return 0;
132 }
133
134 string ltrim(const string &str) {
135     string s(str);
136
137     s.erase(
138         s.begin(),
139         find_if(s.begin(), s.end(), not1(ptr_fun<int, int>(isspace)))
140     );
141
142     return s;
143 }
144
145 string rtrim(const string &str) {
146     string s(str);
147
148     s.erase(
149         find_if(s.rbegin(), s.rend(), not1(ptr_fun<int, int>(isspace))).base(),
150         s.end()
151     );
152
153     return s;
154 }
155
156 vector<string> split(const string &str) {
157     vector<string> tokens;
158
159     string::size_type start = 0;
160     string::size_type end = 0;
161
162     while ((end = str.find(" ", start)) != string::npos) {
163         tokens.push_back(str.substr(start, end - start));

```

```
164
165     start = end + 1;
166 }
167
168 tokens.push_back(str.substr(start));
169
170 return tokens;
171 }
172
```

Line: 58 Col: 1

 [Upload Code as File](#) ☐ [Test against custom input](#)

Run Code

Submit Code

---

[Contest Calendar](#) | [Interview Prep](#) | [Blog](#) | [Scoring](#) | [Environment](#) | [FAQ](#) | [About Us](#) | [Support](#) | [Careers](#) | [Terms Of Service](#) | [Privacy Policy](#) | [Request a Feature](#)