Stringonomics

locked

by abizerl123

Editorial by abizerl123

Let's analyze this problem method by method.

Brute logic

After each query, make the change in the string S, and check if the string P still exists in the string S using an $O(N^2)$ brute logic.
If the string P does not exist in string S, print the index number of that query (1-indexed).

Time complexity: $O(Q * N^2)$ per test case.

Slightly efficient brute logic :P

Instead of using an $O(N^2)$ logic for string matching, use algorithms such as KMP or Rabin Karp etc, that takes only $O(N)$ for string matching.

Time complexity: $O(Q * N)$

But, we still won't be able to pass all the testcases within the time limit as N and Q can go upto 10^5.

Efficient logic

Uptil some number of queries, the string P will exist in the string S, and after a particular query, the string P will cease to exist in the string S. It is also guaranteed that once the string P ceases to exist, it would never emerge in string S again.

This tells us that, the string P always exists in S before our answer query index, and does not exit after our answer query index.

Due to this feature of our queries, we can use binary search on the queries array, as explained below.

Binary search on queries array

1. Initialise l=0, r=q-1, qmid=(l+r)/2.
2. When binary seaching on the array of queries, we select a mid value (say qmid) and check whether the string P exists when computed uptil query qmid or not.
3. This can be done by executing all queries from 0 to qmid on a temporary string that is the same as S, an then search for P in O(n) time using KMP.
4. If it exists, then make l=qmid+1 and continue the binary search, if it doesn't exist, make r=qmid-1 and continue.
5. Therefore, once r>l, l gives you the index of the query where string P ceases to exist, which is your answer.

Final Complexity

$O(N * logQ)$ per testcase where N is the length of S.

Statistics

Difficulty:	Hard
Time Complexity:	O(NlogQ) Required
Knowledge:	Binary Search, String Matching Algorithms
Publish Date:	Jun 16 2018

Set by abizerl123

Problem Setter's code :

```
#include<bits/stdc++.h>
#define pb push_back
#define mp make_pair
#define fi first
#define se second
#define MOD 1000000007
#define MOD9 1000000009
#define gc getchar_unlocked
#define ms(s, n) memset(s, n, sizeof(s))
#define prec(n) fixed<<setprecision(n)
#define eps 0.000001
#define all(v) v.begin(), v.end()
#define bolt ios::sync_with_stdio(0)
#define forr(i,p,n) for(ll i=p;i<n;i++)
typedef long long ll;
using namespace std;
ll mult(ll a,ll b, ll p=MOD){return ((a%p)*(b%p))%p;}
ll add(ll a, ll b, ll p=MOD){return (a%p + b%p)%p;}
ll fpow(ll n, ll k, ll p = MOD) {ll r = 1; for (; k; k >>= 1) {if (k & 1) r
 = r * n%p; n = n * n%p;} return r;}
ll inv(ll a, ll p = MOD) {return fpow(a, p - 2, p);}

string s,p;
ll ind[200003];
char c[200003];

void compute(string pat, int lps[]){
        int len=0,m=pat.length();
        lps[0]=0;
        int i=1;
        while(i<m){
                if(pat[i]==pat[len]){
                        len++;
                        lps[i]=len;
                        i++;
                }else{
                        if(len!=0){
                                len=lps[len-1];
                        }else{
                                lps[i]=0;
                                i++;
                        }
                }
        }
}

ll kmp(string text, string pat){
    int lps[200003];
        compute(pat,lps);
        int i=0,j=0;
        while(i<text.length()){
                if(pat[j]==text[i]){
                        i++;
                        j++;
                }
                if(j==pat.length()){
                        return 1;
                }else if(pat[j]!=text[i] && i<text.length()){
                        if(j!=0){
                                j=lps[j-1];
                        }else{
                                i++;
                        }
                }
        }
    return 0;
}

ll ch(ll mid){
    string temp=s;
    for(ll i=0;i<=mid;i++){
        temp[ind[i]]=c[i];
    }
    return kmp(temp,p);
}

int main(){
        bolt;
    ll t;
    cin>>t;
        ll finlen=0;
    while(t--){
        ms(ind,0);
        ms(c,'\0');
        cin>>s>>p;
        ll q;
        cin>>q;
        for(ll i=0;i<q;i++){
            cin>>ind[i]>>c[i];
        }
        ll l=0;
        ll r=q-1;
        if(!ch(-1)){
            cout<<0<<"\n";
            continue;
        }
        while(l<=r){
            ll mid=(l+r)/2;
            if(ch(mid)){
                l=mid+1;
            }else{
                r=mid-1;
            }
        }
        if(l>=q){
            cout<<-1<<"\n";
        }else cout<<l+1<<"\n";
    }
}
```

Tested by pkacprzak

Problem Tester's code :

```
//intended solution: binary search over queries with linear pattern matchin
g
#include <iostream>
#include <vector>
#include <set>
using namespace std;

class Query {
public:
    Query() {}
    Query(int position, char replacement) : position_(position), replacemen
t_(replacement) {}
    void perform(string& s) const {
        s[position_] = replacement_;
    }
    int get_position() const {
        return position_;
    }
private:
    int position_;
    char replacement_;
};

class Kmp {
public:
    Kmp() {}
    Kmp(const string& pattern) : pattern_(pattern) {
        _compute_pi();
    }
    bool occurs_in(const string& text) {
        int matched = -1;
        int i = 0;
        while (i < text.size()) {
            if (text[i] == pattern_[matched+1]) {
                ++matched;
                if (matched == pattern_.size()-1) {
                    return true;
                }
                ++i;
            } else {
                if (matched == -1) {
                    ++i;
                } else {
                    matched = pi_[matched];
                }
            }
        }
        return false;
    }
private:
    void _compute_pi() {
        pi_ = vector<int>(pattern_.size());
        pi_[0] = -1;
        for (int i = 1; i < pattern_.size(); ++i) {
            int j = i-1;
            while (j >= 0 and pattern_[pi_[j]+1] != pattern_[i]) {
                --j;
            }
            if (j == -1) {
                pi_[i] = -1;
            } else {
                pi_[i] = pi_[j]+1;
            }
        }
    }

    string pattern_;
    vector<int> pi_;
};

Kmp kmp;

void perform_first_k_queries(string& s, const vector<Query>& queries, int k
) {
    for (int i = 0; i < k; ++i) {
        queries[i].perform(s);
    }
}

bool is_substring(const string& p, const string& s) {
    return kmp.occurs_in(s);
}

int solve_binary_search(const string& s, const string& p, const vector<Quer
y>& queries) {
    int res = -1;
    int lo = 0;
    int hi = queries.size();
    while (lo <= hi) {
        int k = lo + (hi-lo)/2;
        string text = s;
        perform_first_k_queries(text, queries, k);
        if (!is_substring(p, text)) {
            res = k;
            hi = k-1;
        } else {
            lo = k+1;
        }
    }
    return res;
}
```

```
int main() {
    ios_base::sync_with_stdio(0);
    int t;
    cin >> t;
    for (int tid = 1; tid <= t; ++tid) {
        string s;
        cin >> s;
        int n = s.size();
        string p;
        cin >> p;
        int q;
        cin >> q;
        vector<Query> queries(q);
        for (int qid = 1; qid <= q; ++qid) {
            int x;
            char c;
            cin >> x >> c;
            queries[qid-1] = (Query(x, c));
        }
        kmp = Kmp(p);
        int res = solve_binary_search(s, p, queries);
        cout << res << endl;
    }
    return 0;
}
```