Geometric Trick

locked

by philipsweng

Editorial by philipsweng

First we have to note that once i and k are fixed,j is fixed too. So we come up with an approach whose times complexity is $O(n^2)$ that we simply iterate every pair of (i,k) and check whether $i*k$ is a square number. If it is, let $j=\sqrt{ik}$ and check if $s[i]=a, s[j]=b, s[k]=c$. This approach is too slow to pass the strict time limit, however, if we write a brute-force problem to check how many pairs of (i,k) satisfies that $i*k$ is a square-number, it's surprising to find that the number is nearly O(n), which is very small. So our goal is to iterate every legal pair of (i,k) efficiently.

For number i, assumed that we have c prime numbers less than n, we can construct a string $s[i]$ for i such that if the jth prime occurs in the factorization of i for odd times,$s[i,j]=1$, otherwise it's 0. Then if $i*k$ is a square number, s[i] should be equal to s[k]. We don't need to know the exact $s[i]$ but to know whether two s[i] and s[k] are the same so that we could use hash. In order to get the hash for every s[i] we have to use prime seive. As for the detail you can infer to the code written by me. For every k, we simply iterate every i whose s[i] is equal to s[k], which we can apply by using map in c++.

So the overall time complexity is $O(n\log n)$. Of course, the algorithm can be improved to $O(n)$, but I think it's a little bit trivial and meaningless.

Set by philipsweng

Problem Setter's code :

```
#include <bits/stdc++.h>

using namespace std;

typedef unsigned long long ull;

const int maxn = 500005;

map<ull,vector<int> > sav;
ull f[maxn],v[maxn];
char s[maxn];
int pri[maxn],n;

void pre_treat()
{
        static bool f[maxn];
        static int stk[maxn];
        for(int i = 2,top = 0;i <= n;i ++)
        {
                if (!f[i]) stk[++ top] = i,pri[i] = i;
                for(int j = 1;i * stk[j] <= n && j <= top;j ++)
                {
                        f[i * stk[j]] = 1;
                        pri[i * stk[j]] = stk[j];
                        if (i % stk[j] == 0) break;
                }
        }
}
```

Statistics

Difficulty: Hard
Time Complexity: O(nlogn)
Required Knowledge: math
Publish Date: Apr 07 2017

```
int main()
{
        scanf("%d", &n);
        f[0] = 1;
        for(int i = 1;i <= n;i ++) f[i] = ((f[i - 1] * 3711 + 123847) << 31) + rand();
        char c;
        for(int i = 1;i <= n;i ++)
        {
                while (c = getchar(),c < 'a' || c > 'z');
                s[i] = c;
        }
        pre_treat();
        ull ans = 0;
        for(int i = 1;i <= n;i ++)
        {
                for(int q = i;q > 1;q /= pri[q]) v[i] ^= f[pri[q]];
                if (s[i] != 'b' && sav.count(v[i]))
                {
                        for(auto p : sav[v[i]])
                        {
                                int mid = sqrt(i * 1ll * p);
                                if (s[mid] == 'b' && (s[i] == 'a' && s[p] == 'c' || s[i]
== 'c' && s[p] == 'a'))
                                        ++ ans;
                        }
                }
                sav[v[i]].push_back(i);
        }
        cout << ans << endl;
        return 0;
}
```