Sherlock and Queries

Authored by darkshadows on *Apr 28 2014*

Statistics

Difficulty: 0.32401490947816824
Time Complexity: O(N*log(N))
Required Knowledge: Complexity Analysis
Publish Date: Jun 26 2014

Originally featured in 101 Hack June'14

Editorial by darkshadows

For given arrays A of size N, and B and C of size M, we have to do.

```
for i = 1 to M do
    for j = 1 to N do
        if j % B[i] == 0 then
            A[j] = A[j] * C[i]
        endif
    end do
end do
```

For each multiple (say x) of each B[i], we have to mutiply A[x] with C[i]. If we brute force it, it will time out.
We can store for each B[i], we will store how many times B[i] (the value) has occured. Like we know 1 came p1 times, 2 came p2 times, 3 came p3 times and so on.
For each multiple x of B[1] in x = 1 to N, multiply A[x] by C[1] p1 times. For each multiple y of B[2] in y = 1 to N, multiply A[x] by C[2] p1 times. .
.
and so on.

```
for each i=1 to N:
    for j=1 to (n/i):
         multiply A[i*j] with C[i] pi times
```

If we analyse complexity it will be in worst case, \` $\frac{N}{1}+\frac{N}{2}+\frac{N}{3}+\frac{N}{4}+....+\frac{N}{N}\approx O(N\times log(n))$\`.

Set by darkshadows

Problem Setter's code :

```
#include<bits/stdc++.h>
using namespace std;
#define MOD 1000000007
typedef long long LL;
#define assn(n,a,b) assert(n>=a && n<=b)
LL a[100006],b[100006],c[100006],counti[100006]={};
int main()
{
    memset(counti,-1,sizeof(counti));
    LL n,m,i,j;
    cin >> n >> m;
    assn(n,1,100000);
    assn(m,1,100000);
    for(i=1; i<=n; i++)
    {
        cin >> a[i];
        assn(a[i],1,100000);
    }
    for(i=1; i<=m; i++)
    {
        cin >> b[i];
        assn(b[i],1,n);
    }
    for(i=1; i<=m; i++)
    {
        cin >> c[i];
        assn(c[i],1,100000);
    }
    for(i=1; i<=m; i++)
    {
        if(counti[b[i]]==-1)
            counti[b[i]]=c[i];
        else counti[b[i]]=(counti[b[i]]*c[i])%MOD;
    }
    for(i=1; i<=n; i++)
    {
        for(j=1; (j*i)<=n; j++)
        {
            if(counti[i]!=-1)
                a[j*i]=(a[j*i]*counti[i])%MOD;
        }
    }
    for(i=1; i<n; i++)
        cout << a[i] << " " ;
    cout << a[n] << endl;
    return 0;
}
```

Tested by gera1d

Problem Tester's code :

```
#ifdef ssu1
#define _GLIBCXX_DEBUG
#endif
#undef NDEBUG

#include <algorithm>
#include <functional>
#include <numeric>
#include <iostream>
#include <cstdio>
#include <cmath>
#include <cstdlib>
#include <ctime>
#include <cstring>
#include <cassert>
#include <vector>
#include <list>
#include <map>
#include <set>
#include <deque>
#include <queue>
#include <bitset>
#include <sstream>

using namespace std;

#define fore(i, l, r) for(int i = (l); i < (r); ++i)
#define forn(i, n) fore(i, 0, n)
#define fori(i, l, r) fore(i, l, (r) + 1)
#define sz(v) int((v).size())
#define all(v) (v).begin(), (v).end()
#define pb push_back
#define mp make_pair
#define X first
#define Y second

#if ( _WIN32 || __WIN32__ )
    #define LLD "%I64d"
#else
    #define LLD "%lld"
#endif

typedef long long li;
typedef long double ld;
typedef pair<int, int> pt;

template<typename T> T abs(T a) { return a < 0 ? -a : a; }
template<typename T> T sqr(T a) { return a*a; }

const int INF = (int)1e9;
const ld EPS = 1e-9;
const ld PI = 3.1415926535897932384626433832795;
```

```
/*

    This is just to check correctness of input

*/
void ensure(bool value){
    if(!value){
        fprintf(stderr, "Assertion failed");
        throw;
    }
}
void ensure(bool value, string message){
    if(!value){
        fprintf(stderr, "Assertion failed. Message = %s", message.c_str());
        throw;
    }
}
int readInt(int l, int r){
    int x;
    if(scanf("%d", &x) != 1){
        fprintf(stderr, "Expected int in range [%d, %d], but haven't found!", l,
r);
        throw;
    }
    if(!(l <= x && x <= r)){
        fprintf(stderr, "Expected int in range [%d, %d], but found %d!", l, r, x)
;
        throw;
    }
    return x;
}
int readInt(int l, int r, string name){
    int x;
    if(scanf("%d", &x) != 1){
        fprintf(stderr, "Expected int %s in range [%d, %d], but haven't found!",
name.c_str(), l, r);
        throw;
    }
    if(!(l <= x && x <= r)){
        fprintf(stderr, "Expected int %s in range [%d, %d], but found %d!", name.
c_str(), l, r, x);
        throw;
    }
    return x;
}
li readLong(li l, li r){
    li x;
    if(scanf(LLD, &x) != 1){
        fprintf(stderr, "Expected long long in range ["LLD", "LLD"], but haven't
found!", l, r);
        throw;
    }
    if(!(l <= x && x <= r)){
        fprintf(stderr, "Expected long long in range ["LLD", "LLD"], but found "L
LD"!", l, r, x);
        throw;
    }
    return x;
}
li readLong(li l, li r, string name){
    li x;
    if(scanf(LLD, &x) != 1){
        fprintf(stderr, "Expected long long %s in range ["LLD", "LLD"], but haven
't found!", name.c_str(), l, r);
        throw;
    }
    if(!(l <= x && x <= r)){
        fprintf(stderr, "Expected long long %s in range ["LLD", "LLD"], but found
 "LLD"!", name.c_str(), l, r, x);
        throw;
    }
    return x;
}
const ld __EPS__ = 1e-15;
ld readDouble(double l, double r){
    double x;
    if(scanf("%lf", &x) != 1){
        fprintf(stderr, "Expected double in range [%lf, %lf], but haven't found!"
, l, r);
        throw;
    }
    if(!(l <= x + __EPS__ && x <= r + __EPS__)){
        fprintf(stderr, "Expected double in range [%lf, %lf], but found %lf!", l,
 r, x);
        throw;
    }
    return x;
}
ld readDouble(double l, double r, string name){
    double x;
    if(scanf("%lf", &x) != 1){
        fprintf(stderr, "Expected double %s in range [%lf, %lf], but haven't foun
d!", name.c_str(), l, r);
        throw;
    }
    if(!(l <= x + __EPS__ && x <= r + __EPS__)){
        fprintf(stderr, "Expected double %s in range [%lf, %lf], but found %lf!",
 name.c_str(), l, r, x);
        throw;
    }
    return x;
}
const int __MAXBUF__ = (int)1e7;
char __buf__[__MAXBUF__];
string readString(char lfc, char rgc, int lfn, int rgn){
    ensure(scanf(" %s ", __buf__) == 1, "Expected string, haven't found");
    string ans = __buf__;
    ensure(lfn <= sz(ans) && sz(ans) <= rgn, "String have incorrect length");
    forn(i, sz(ans))
        ensure(lfc <= ans[i] && ans[i] <= rgc, "String contains incorrect charact
ers");
    return ans;
}
string readString(string pat, int lfn, int rgn){
    ensure(scanf(" %s ", __buf__) == 1, "Expected string, haven't found");
    string ans = __buf__;
    ensure(lfn <= sz(ans) && sz(ans) <= rgn, "String have incorrect length");
    forn(i, sz(ans))
        ensure(find(all(pat), ans[i]) != pat.end(), "String contains incorrect ch
aracters");
    return ans;
}
string readString(char lfc, char rgc, int lfn, int rgn, string name){
    ensure(scanf(" %s ", __buf__) == 1, "Expected string " + name + ", haven't fo
und");
    string ans = __buf__;
    ensure(lfn <= sz(ans) && sz(ans) <= rgn, "String " + name + " have incorrect
length");
    forn(i, sz(ans))
        ensure(lfc <= ans[i] && ans[i] <= rgc, "String " + name + " contains inco
rrect characters");
    return ans;
}
string readString(string pat, int lfn, int rgn, string name){
    ensure(scanf(" %s ", __buf__) == 1, "Expected string " + name + ", haven't fo
und");
    string ans = __buf__;
    ensure(lfn <= sz(ans) && sz(ans) <= rgn, "String " + name + " have incorrect
length");
    forn(i, sz(ans))
        ensure(find(all(pat), ans[i]) != pat.end(), "String " + name + " contains
 incorrect characters");
    return ans;
}
string readLine(char lfc, char rgc, int lfn, int rgn){
    ensure(gets(__buf__) != NULL, "Line expected, haven't found");
    string ans = __buf__;
    ensure(lfn <= sz(ans) && sz(ans) <= rgn, "Line have incorrect length");
    forn(i, sz(ans))
        ensure(lfc <= ans[i] && ans[i] <= rgc, "Line contains incorrect character
s");
    return ans;
}
string readLine(string pat, int lfn, int rgn){
    ensure(gets(__buf__) != NULL, "Line expected, haven't found");
    string ans = __buf__;
    ensure(lfn <= sz(ans) && sz(ans) <= rgn, "Line have incorrect length");
    forn(i, sz(ans))
        ensure(find(all(pat), ans[i]) != pat.end(), "Line contains incorrect char
acters");
    return ans;
```

```
}
string readLine(){
    ensure(gets(__buf__) != NULL, "Line expected, haven't found");
    string ans = __buf__;
    return ans;
}
char readChar(){
    char c;
    ensure(scanf(" %c ", &c) == 1, "Non-whitespace character expected");
    return c;
}
char readWChar(){
    int ans = getchar();
    ensure(ans != EOF, "Character expected");
    return (char)ans;
}

const int NMAX = 100000;
const int mod = 1000000000 + 7;

int n, m, a[NMAX], b[NMAX], c[NMAX];

int ml[NMAX + 1];

int main(){
#ifdef ssu1
    assert(freopen("input.txt", "rt", stdin));
    //assert(freopen("output.txt", "wt", stdout));
#endif

    n = readInt(1, 100000), m = readInt(1, 100000);
    forn(i, n)
        a[i] = readInt(1, 100000);
    forn(i, m)
        b[i] = readInt(1, n);
    forn(i, m)
        c[i] = readInt(1, 100000);

    forn(i, NMAX + 1)
        ml[i] = 1;

    forn(i, m){
        ml[b[i]] = (ml[b[i]] * 1LL * c[i]) % mod;
    }

    fori(step, 1, NMAX){
        for(int j = step; j <= n; j += step){
            a[j - 1] = (a[j - 1] * 1LL * ml[step]) % mod;
        }
    }

    forn(i, n){
        printf("%d ", a[i]);
    }
    puts("");
    return 0;
}
```