解题思路

首先，都可以意识到这道题的数据是相当庞大的，m ≤ 1000000。

如此一来，一道简单的解方程的题便不能用枚举法一一带入了，这种方法可谓是简单粗暴，然而并没有什么卵用，这样涉及到高精度乘高精度，高精度乘单精度，高精度加高精度和高精度减高精度。复杂度 n\*m\*len\*len ，显然只能过30%的数据，故舍去。

所以在这种方法的基础上要考虑优化——

 耗时大的方面主要是：带入数个x并且求值。还可以在开头加上一个艾森斯坦定理的判根法（不知道的自己百度）

 （1）对于求值麻烦的问题：这里可以考虑秦九韶算法，将方程左边改写成如下形式：

      这样一来便省去了高精×高精这一计算过程，复杂度大大下降，然而还是太复杂（复杂度为 n\*len），即使也只能得到50分。

解题思路：使用秦九韶算法然后将【1，m】内的值一一带入（50分）

（2）求值的问题解决了但并不起多大作用，所以要考虑缩小取值的范围。（以下内容可以向学过数学奥赛的请教） 我们发现0的一个特征就是模任何一个数都是0，但我们多模几个质数是否能一定保证该数为0呢？当然不行。

那模谁呢？x。

显然，左边 mod x= A0 mod x ，所以 A0 mod x=0 ；

也就是说x一定是A0的约数！

然后我们的算法来了：

枚举1~m的每个数i，先判断它是不是A0的约数，这只要类似于高精除单精扫一遍就行了，如果不是，那么显然 2\*i，3\*i，……都不会是A0的约数，我们类似筛法筛掉这些。

如果是的话，带入求值判断是否为0.

当然如果A0=0的话，该方程的一个解是0，然后其余的解都满足 A1+A2\*x^1+A3\*x^2+……An\*x^n-1=0，我们试A1即可。

当然，到这里，算法的复杂度还是不好估计，不过是不可能达到n\*m\*len的上界的，应该有很大一部分被筛掉了。

解题思路：可以在【1，m】范围内筛去不是A0约数的数i，及其相关数2i、3i……，然后再用秦九韶算法处理方程左式，将剩余的数一一枚举带入即可。

#include <iostream>

#include <cstdio>

#include <cstring>

#include <cstdlib>

#include <algorithm>

using namespace std;

#define maxn 1000010

const int prime[] = {10007,10917,30071};

int n,m;

long long a[110][5];

bool f[100000][5];

int cnt[maxn];

char s[10010];

bool calc(int value, int j) {

long long tmp = 0;

for (int i = n; i>=0; --i)

tmp = (tmp \* value + a[i][j]) % prime[j];

return tmp != 0;

}

int main(){

freopen("equation.in", "r", stdin);

freopen("equation.out", "w", stdout);

cin>>n>>m;

for (int i = 0; i <= n; ++i) {

scanf("%s", s);

int len = strlen(s);

int sign = 1;

for (int l = 0; l < len; ++l) {

if(s[l]=='-')

sign = -1;

else

for (int j = 0; j < 3; ++j)

a[i][j]=( a[i][j] \* 10 + s[i]-'0' ) % prime[j];

}

if (sign == -1)

for (int j = 0; j < 3; ++j)

a[i][j] = prime[j] - a[i][j];

}

for (int j = 0; j < 3; ++j)

for(int i = 0; i < prime[j]; ++i)

f[i][j]=calc(i,j);

for (int i = 1; i <=m; ++i) {

bool flag = true;

for(int j = 0; j < 3; ++j)

if(f[i % prime[j]][j]) {

flag = false; break;

}

if(flag)

cnt[++cnt[0]] = i;

}

printf("%d\n", cnt[0]);

for (int i = 1; i <= cnt[0]; ++i)

printf("%d\n",cnt[i]);

return 0;

}



