

ACM_高次同余方程 - xiaotan1314 - CSDN博客

[cpp]view plaincopy

```
118. /*poj 3243
119. *解决高次同余方程的应用，已知  $X^Y = K \bmod Z$ ，及X,Z,K的值，求 Y 的值
120. */
121. #include
122. #include
123. #include
124. using namespace std;
125. #define lint __int64
126. #define MAXN 131071
127. struct HashNode { lint data, id, next; };
128. HashNode hash[MAXN<<1];
129. bool flag[MAXN<<1];
130. lint top;
131. void Insert ( lint a, lint b )
132. {
133.     lint k = b & MAXN;
134.     if ( flag[k] == false )
135.     {
136.         flag[k] = true;
137.         hash[k].next = -1;
138.         hash[k].id = a;
139.         hash[k].data = b;
140.     }
141.     return;
142. while( hash[k].next != -1 )
143. {
144.     if( hash[k].data == b ) return;
145.     k = hash[k].next;
146. }
147. if ( hash[k].data == b ) return;
148. hash[k].next = ++top;
149. hash[top].next = -1;
150. hash[top].id = a;
151. hash[top].data = b;
152. }
153. lint Find ( lint b )
154. {
155.     lint k = b & MAXN;
156.     if( flag[k] == false ) return -1;
157.     while ( k != -1 )
158.     {
159.         if( hash[k].data == b ) return hash[k].id;
160.         k = hash[k].next;
161.     }
162.     return -1;
163. }
164. lint gcd ( lint a, lint b )
165. {
166.     return b ? gcd ( b, a % b ) : a;
167. }
168. lint ext_gcd (lint a, lint b, lint& x, lint& y )
169. {
170.     lint t, ret;
171.     if ( b == 0 )
172.     {
173.         x = 1, y = 0;
174.     }
175.     return a;
176.     ret = ext_gcd ( b, a % b, x, y );
177.     t = x, x = y, y = t - a / b * y;
178.     return ret;
179. }
180. lint mod_exp ( lint a, lint b, lint n )
```

```

181. {
182.     lint ret = 1;
183.     a = a % n;
184. while ( b >= 1 )
185.     {
186. if( b & 1 )
187.         ret = ret * a % n;
188.         a = a * a % n;
189.         b >>= 1;
190.     }
191. return ret;
192. }
193. lint BabyStep_GiantStep ( lint A, lint B, lint C )
194. {
195.     top = MAXN; B %= C;
196.     lint tmp = 1, i;
197. for ( i = 0; i < tmp=" tmp * A % C, i++ )
198. if ( tmp == B % C ) return i;
199.     lint D = 1, cnt = 0;
200. while( (tmp = gcd(A,C)) !=1 )
201.     {
202. if( B % tmp ) return -1;
203.         C /= tmp;
204.         B /= tmp;
205.         D = D * A / tmp % C;
206.         cnt++;
207.     }
208.     lint M = (lint)ceil(sqrt(C+0.0));
209. for ( tmp = 1, i = 0; i < Mtmp=" tmp * A % C, i++ )
210.         Insert ( i, tmp );
211.     lint x, y, K = mod_exp( A, M, C );
212. for ( i = 0; i < Mispan>
213.     {
214.         ext_gcd ( D, C, x, y ); // D * X = 1 ( mod C )
215.         tmp = ((B * x) % C + C) % C;
216. if( (y = Find(tmp)) != -1 )
217. return i * M + y + cnt;
218.         D = D * K % C;
219.     }
220. return -1;
221. }
222. int main()
223. {
224.     lint A, B, C;
225. while( scanf("%I64d%I64d%I64d",&A,&C,&B) !=EOF )
226.     {
227. if ( !A && !B && !C ) break;
228.         memset(flag,0,sizeof(flag));
229.         lint tmp = BabyStep_GiantStep ( A, B, C );
230. if ( tmp == -1 )puts("No Solution");
231. else printf("%I64d\n",tmp);
232.     }
233. return 0;
234. }

```