# 6.1 扩展欧几里德与贝祖定理

**6.1.1 概述**

为了引进扩展欧几里德，首先讲讲贝祖定理：如果a、b是整数，那么一定存在整数x、y使得。换句话说，如果有解，那么m一定是gcd(a,b)的若干倍。（可以来判断一个这样的式子有没有解），扩展欧几里德则是用来求解方程

**6.1.2 扩欧模板**

1. **int** exgcd(**int** a,**int** b,**int** &x,**int** &y)
2. {
3. **if**(b==0)
4. {
5. x=1;
6. y=0;
7. **return** a;
8. }
9. **int** ans=exgcd(b,a%b,x,y);
10. **int** tmp=x;
11. x=y;
12. y=tmp-(a/b)\*y;
13. **return** ans;
14. }

**6.1.3 判断方程ax+by=m或是否有解**

**思路：**当gcd(a,b)|m时 方程有解，反之则无解

**6.1.4 求解方程ax+by=m( ax≡m(mod b) )的最小正整数解**

**思路：**

1.首先判断方程是否有解，无解则直接退出

2.扩欧求x，将( ( (x\*m/gcd(a,b) )%(b/gcd(a,b) )+b/gcd(a,b))%(b/gcd(a,b)即可

**6.1.5 经典例题**

**6.1.5.1 洛谷 P1082 同余方程**

**题目描述**

求关于x的同余方程ax≡1(mod b)的最小正整数解。

**输入格式**

一行，包含两个正整数a,b,用一个空格隔开。

**输出格式**

一个正整数x0，即最小正整数解。输入数据保证一定有解。

**输入样例**

3 10

**输出样例**

7

**思路：**模板题，由于题目规定一定有解，因此a⊥b（a与b互质） ，此时既可以用扩欧求解，也可用欧拉定理求解

在这科普一下欧拉定理：

(其中，a与n均为正整数，且两者互质。) 费马小定理为欧拉定理的特殊形式

**题解：**

**扩欧：**

1. #include<bits/stdc++.h>
2. **using** **namespace** std;
3. #define int long long
4. **int** exgcd(**int** a,**int** b,**int** &x,**int** &y)
5. {
6. **if**(b==0)
7. {
8. x=1;
9. y=0;
10. **return** a;
11. }
12. **int** ans=exgcd(b,a%b,x,y);
13. **int** tmp=x;
14. x=y;
15. y=tmp-(a/b)\*y;
16. **return** ans;
17. }
18. **signed** main()
19. {
20. **int** a,b;
21. **int** x,y;
22. cin>>a>>b;
23. exgcd(a,b,x,y);
24. x=(x%b+b)%b;
25. cout<<x<<endl;
26. **return** 0;
27. }

**欧拉定理：**

1. #include<bits/stdc++.h>
2. **using** **namespace** std;
3. #define int long long
4. **int** qpow(**int** a,**int** n,**int** b)
5. {
6. **int** ans=1;
7. **while**(n)
8. {
9. **if**(n&1) ans=ans%b\*a%b;
10. a=a%b\*a%b;
11. n>>=1;
12. }
13. **return** ans;
14. }
15. **int** euler(**int** n)
16. {
17. **int** ans=n;
18. **for**(**int** i=2;i\*i<=n;i++)
19. {
20. **if**(n%i==0) ans-=ans/i;
21. **while**(n%i==0) n/=i;
22. }
23. **if**(n>1) ans-=ans/n;
24. **return** ans;
25. }
26. **signed** main()
27. {
28. **int** a,b;
29. cin>>a>>b;
30. cout<<qpow(a,euler(b)-1,b)<<endl;
31. **return** 0;
32. }

**6.1.5.2 洛谷P5656 二元一次方程**

**题目描述**

给定不定方程ax+by=c

若该方程无整数解，输出-1。

若该方程有整数解，且有正整数解，则输出其正整数解的数量，所有正整数解中x的最小值，所有正整数解中y的最小值，所有正整数解中x的最大值，以及所有正整数解中y 的最大值。

若方程有整数解，但没有正整数解，你需要输出所有整数解中x的最小正整数值，y的最小正整数值。

正整数解即为 x, y 均为正整数的解， 0 不是正整数。

整数解即为 x,y 均为整数的解。

x的最小正整数值即所有x为正整数的整数解中x的最小值，y同理。

**输入格式**

第一行一个正整数 T，代表数据组数。

接下来 T 行，每行三个由空格隔开的正整数a,b,c。

**输出格式**

T 行。

若该行对应的询问无整数解，一个数字 -1。

若该行对应的询问有整数解但无正整数解，包含 2个由空格隔开的数字，依次代表整数解中，x的最小正整数值，y 的最小正整数值。

否则包含5个由空格隔开的数字，依次代表正整数解的数量，正整数解中，x的最小值，y的最小值，x的最大值，y的最大值。

**输入样例**

7

2 11 100

3 18 6

192 608 17

19 2 60817

11 45 14

19 19 810

98 76 5432

**输出样例**

4 6 2 39 8

2 1

-1

1600 1 18 3199 30399

34 3

-1

2 12 7 50 56

**题解：**

1. #include<bits/stdc++.h>
2. **using** **namespace** std;
3. #define int long long
4. **int** exgcd(**int** a,**int** b,**int** &x,**int** &y)
5. {
6. **if**(b==0)
7. {
8. x=1;
9. y=0;
10. **return** a;
11. }
12. **int** ans=exgcd(b,a%b,x,y);
13. **int** tmp=x;
14. x=y;
15. y=tmp-(a/b)\*y;
16. **return** ans;
17. }
18. **signed** main()
19. {
20. **int** t,a,b,c;
21. **int** x,y;
22. **int** minx,miny;
23. **int** maxx,maxy;
24. **int** cnt=1;
25. scanf("%lld",&t);
26. **bool** flag=**false**;
27. **while**(t--)
28. {
29. flag=**false**;
30. cnt=0;
31. scanf("%lld %lld %lld",&a,&b,&c);
32. **int** gcd=exgcd(a,b,x,y);
33. **if**(c%gcd)
34. {
35. cout<<"-1"<<endl;
36. **continue**;
37. }
38. **int** bg=b/gcd;
39. **int** cg=c/gcd;
40. **int** ag=a/gcd;
41. **int** tmp;
42. x\*=cg;
43. y\*=cg;
44. //cout<<"x= "<<x<<"y="<<y<<endl;
45. tmp=x;
46. x=(x%bg+bg-1)%bg+1;//最小正整数解  若是最小整数解则不用-1
47. **int** ansy=(y%ag+ag-1)%ag+1;
48. **int** t=(x-tmp)/bg;
49. y-=t\*ag;
50. **if**(x>0&&y>0)
51. {
52. minx=x;
53. maxy=y;
54. flag=**true**;
55. cnt=1;
56. **int** k=y/ag;
57. **if**(y%ag==0)
58. {
59. cnt+=k-1;
60. miny=ag;
61. maxx=x+(k-1)\*bg;
62. }
63. **else**
64. {
65. cnt+=k;
66. miny=y-k\*ag;
67. maxx=x+k\*bg;
68. }
69. }
70. **if**(flag) printf("%lld %lld %lld %lld %lld\n",cnt,minx,miny,maxx,maxy);
71. **else** printf("%lld %lld\n",x,ansy);
72. }
73. **return** 0;
74. }