## a^x =a^(x%phi(p)+phi(p)) mod (p) 当x大于phi(p) 成立

## 扩展剩余定理

求ans 使得ans=b[i](mod n[i])  
lld ext\_euclid(lld a,lld b,lld &x,lld &y)   
{   
   lld t,d;   
   if (b==0) {x=1;y=0;return a;}   
   d=ext\_euclid(b,a%b,x,y);   
   t=x;   
   x=y;   
   y=t-a/b\*y;   
   return d;   
}  
lld modE(lld a, lld b, lld n)  
{  
lld d, x, y;  
d = ext\_euclid(a, n, x, y);  
if(b % d) return -1;  
x = x\*(b/d);  
x = (x%(n/d) + n/d) % (n/d);  
return x;  
}

lld sol(lld n[], lld b[], lld len)  
{  
lld i, s = -1, d;  
for(i = 1; i < len; i++) {  
   lld xx,yy;  
   d = ext\_euclid(n[0], n[i],xx,yy);  
   if((b[0]-b[i]) % d) return -1;  
   s = modE(n[0], b[i]-b[0], n[i]);  
   if(s == -1) return -1;  
   b[0] +=s \* n[0];  
   n[0] = n[0]\*n[i]/d;  
   b[0] %= n[0];  
}  
s = modE(1, b[0], n[0]);  
return s;  
}

## 取石子

有n堆石子 每堆ai个 两人轮流取 每次每人可以从最多k堆取 每堆最多取m个 谁取到最后一个就赢

将ai全部mod m，再转2进制 然后每位单独累加mod (k+1) 最后 如果全0 就是必胜 否则必败

## baby-step-gaint-step

#include <iostream>  
#include <cmath>  
using namespace std;  
const int maxh=499997,maxm=maxh+3;  
struct baby\_step  
{  
    int tag,rem,next;  
}baby[maxm];  
int st[maxm],e;  
int gcd(int a,int b)  
{  
if (a<b){int c=a;a=b;b=c;}  
while (b!=0)  
{  
   int c=a;a=b;b=c%b;  
}  
return a;  
}  
int find(int m)  
{  
    int mm,now;  
    mm=m%maxh; now=st[mm];  
    while (now!=-1)  
    {  
        if (baby[now].rem==m) return baby[now].tag;  
        else now=baby[now].next;  
    }  
    return -1;  
}  
void insert(int m,int n)  
{  
    int mm,now;  
    mm=m%maxh; now=st[mm];  
    while (now!=-1&&baby[now].rem!=m) now=baby[now].next;  
    if (now==-1)  
    {  
        baby[e].rem=m; baby[e].tag=n; baby[e].next=st[mm];  
        st[mm]=e++;  
    }  
    else baby[now].tag=n;  
}  
\_\_int64 mood(\_\_int64 a,\_\_int64 x,\_\_int64 m)  
{  
\_\_int64 i=1;  
while (x)  
{  
   if (x&1) i=i\*a%m;  
   x>>=1;  
   a=a\*a%m;  
}  
return i;  
}  
int babystep(\_\_int64 n,\_\_int64 a,\_\_int64 b)  
{  
memset(st,-1,sizeof(st));  
a%=n;b%=n;  
\_\_int64 bb=b;  
int i,m;  
m=(int)sqrt((double) n);  
insert(b,0);  
for (i=1;i<=m;i++)  
{  
   b=b\*a%n;  
   insert(b,i);  
}  
\_\_int64 step=mood(a,m,n);  
\_\_int64 y=1;  
for (i=1;i\*m<=n;i++)  
{  
   y=y\*step%n;  
   int pai;  
   pai=find(y);  
   if (pai>=0) return i\*m-pai;  
}  
return -1;

}  
int main()  
{  
\_\_int64 a,b,n;  
while (scanf("%I64d%I64d%I64d",&a,&n,&b)!=EOF)  
{  
   int i,j,k=-1,nn,dan,dbn;  
   \_\_int64 sl=1;  
   a%=n;  
   if (b>=n) goto xxx;  
   b%=n;  
   if(b==1)  
    if(a==0) goto xxx;else {cout<<0<<endl;continue;}  
   if (b==0)  
    if (a==0) {cout<<1<<endl;continue;}  
     
   for (i=0;i<40;i++)  
   {  
    sl=sl\*a%n;  
    if (sl==b) break;  
   }  
   if (sl==b) {cout<<i+1<<endl;continue;}   
   dan=gcd(a,n);dbn=gcd(b,n);  
   nn=n/dbn;  
   if (gcd(nn,a)>1) goto xxx;  
   while ((i=gcd(n,a))>1) n/=i;  
   a%=n;b%=n;  
   k=babystep(n,a,b);  
   if (k<0) goto xxx;  
   if (k<=40)  
   {  
    i=babystep(n,a,1);  
    while (k<=40) k+=i;  
   }

   if (k>=0&&mood(a,k,n)==b)  
   {  
    printf("%d\n",k);  
    continue;  
   }  
xxx: printf("Orz,I can’t find D!\n");  
// printf("No Solution\n");  
}  
return 0;  
}

## 判素数和大整数分解

标准模板

1. #include <cstdio>
2. #include <cstdlib>
3. #include <algorithm>
4. using namespace std;
5. typedef long long ll;
6. #define maxn 2000
7. bool flag[maxn + 1];
8. int smallPrimes[maxn];
9. int ccc;
10. #define MAX 10
11. ll mulMod(ll a, ll b, ll p) {
12. ll y = (ll)((double) a \* b / p + 0.05);
13. ll r = a \* b - y \* p;
14. if (r < 0) r = r + p;
15. return r;
16. }
17. ll rand(ll x, ll y) { //random in [x, y)
18. ll a = rand();
19. a = a \* rand() \* rand() \* rand();
20. return x + a % (y - x);
21. }
22. ll powMod(ll x, ll n, ll p){
23. ll z = 1;
24. do {
25. if(n & 1) z = mulMod(z, x, p);
26. x = mulMod(x, x, p);
27. } while (n >>= 1);
28. return z;
29. }
30. bool millerRabin(ll n) {
31. if(n == 2) return true;
32. if(n < 2 || !(n & 1)) return false;
33. ll k = 0, j, a;
34. ll m = n - 1;
35. while(!(m & 1)) m >>= 1, k++;
36. for(int i = 0; i < MAX; ++i) {
37. a = powMod(rand(1, n), m, n);
38. if(a == 1) continue;
39. for(j = 0; j < k; ++j){
40. if(a == n - 1) break;
41. a = mulMod(a, a, n);
42. }
43. if(j < k) continue;
44. return false;
45. }
46. return true;
47. }
48. ll gcd(ll a, ll b){
49. return b ? gcd(b, a % b) : a;
50. }
51. ll f(ll x, ll n) {
52. ll xx = mulMod(x, x, n) + 1;
53. return xx == n ? 0 : xx;
54. }
55. ll pollard(ll n) {
56. if(n <= 2) return 0;
57. if(!(n & 1)) return 2;
58. for(int i = 1; i < MAX; ++i) {
59. ll x = rand(0, n);
60. ll xx = f(x, n);
61. ll p = gcd((xx + n - x) % n, n);
62. while(p == 1) {
63. x = f(x, n);
64. xx = f(f(xx, n), n);
65. p = gcd((xx + n - x) % n, n);
66. }
67. if(p) return p;
68. }
69. return 0;
70. }
71. ll brent(ll n) {
72. for (int i = 0; i < MAX; ++i) {
73. ll y = rand(0, n);
74. ll r = 1, q = 1;
75. ll g, ys, x;
76. do {
77. x = y;
78. for (int i = 0; i < r; ++i) {
79. y = f(y, n);
80. }
81. ll k = 0;
82. do {
83. ys = y;
84. int a = MAX > r - k ? r - k : MAX;
85. for (int i = 0; i < a; ++i) {
86. y = f(y, n);
87. q = mulMod(q, (x - y + n) % n, n);
88. }
89. g = gcd(q, n);
90. k = k + MAX;
91. } while (k < r && g <= 1);
92. r += r;
93. } while (g <= 1);
94. if (g == n) {
95. do {
96. ys = f(ys, n);
97. g = gcd((x - ys + n) % n, n);
98. } while (g <= 1);
99. }
100. if (g == n) continue;
101. return g;
102. }
103. return 0;
104. }
105. ll prime(ll a){
106. if(millerRabin(a)) return 0;
107. ll t = brent(a), p;
108. if (!t) return 0;
109. if(p = prime(t)) return p;
110. else return t;
111. }
112. pair<ll, int> factor[64];
113. int getFactors(ll n) {
114. int m = 0;
115. int e;
116. for(int i = 0; i < ccc && smallPrimes[i] \* smallPrimes[i] <= n; ++i) {
117. if(n % smallPrimes[i] == 0) {
118. n /= smallPrimes[i], e = 1;
119. while (n % smallPrimes[i] == 0) n /= smallPrimes[i], e++;
120. factor[m++] = make\_pair((ll)smallPrimes[i], e);
121. }
122. }
123. while (n > 1) {
124. ll t = prime(n);
125. if (!t) break;
126. factor[m] = make\_pair(t, 0);
127. for (; n % t == 0; n /= t) {
128. factor[m].second++;
129. }
130. m++;
131. }
132. if(n > 1) factor[m++] = make\_pair(n, 1);
133. sort(factor, factor + m);
134. return m;
135. }
136. void init() {
137. for (int i = 2; i \* i <= maxn; ++i) {
138. if (flag[i] == 0) for (int j = i \* i; j <= maxn; j += i) {
139. flag[j] = 1;
140. }
141. }
142. ccc = 0;
143. for (int i = 2; i <= maxn; ++i) {
144. if (flag[i] == 0) smallPrimes[ccc++] = i;
145. }
146. }
147. int main() {
148. init();
149. int t;
150. scanf("%d", &t);
151. while (t--) {
152. ll p;
153. scanf("%I64d", &p);
154. if (millerRabin(p)) puts("Prime");
155. else {
156. getFactors(p);
157. printf("%I64d\n", factor[0].first);
158. }
159. }
160. return 0;
161. }

我的程序

搞了半死 终于搞出来了

素数判断用miller法 分解用pollard法 关键有几点

1:用2分法作64位乘法必须用unsigned \_\_Int64 否则位移的时候会带符号(符号位移不掉)

2:pollard会陷入死循环 所以要加卡时 如果超过多少次还没出来就return 1,换个初始数继续

3:所有<<号,>>号必须全部加括号 像b1=(x<<32)>>32这种等号后面没加括号的是错误的 应该是b1=((x<<32)>>32);

4:发现pollard算法中用x\*x-1产生随机数，如果那个-1改成其他数 效率会不一样 根据frkstyc大牛的代码 x\*x+16381要将近快一倍

#include <iostream>  
#include <time.h>  
#include <cmath>  
using namespace std;  
unsigned \_\_int64 chen(unsigned \_\_int64 x,unsigned \_\_int64 y,unsigned \_\_int64 n)//计算x\*y%n  
{  
if ((x>>32)==0&&(y>>32)==0) return x\*y%n;  
unsigned \_\_int64 a1,b1,a2,b2;  
a1=(x>>32);  
b1=((x<<32)>>32);  
a2=(y>>32);  
b2=((y<<32)>>32);  
unsigned \_\_int64 t1,t2,t3,t4;  
t1=b1\*b2%n;t2=b1\*a2%n;t3=a1\*b2%n;t4=a1\*a2%n;  
int i;  
for (i=0;i<32;i++){ t2=(t2<<1)%n;t3=(t3<<1)%n;}  
for (i=0;i<64;i++) t4=(t4<<1)%n;  
return ((t1+t2)%n+(t3+t4)%n)%n;  
}  
unsigned \_\_int64 mood(unsigned \_\_int64 a,unsigned \_\_int64 u,unsigned \_\_int64 n) //计算a^u%n  
{  
unsigned \_\_int64 b,k;  
int i;  
i=k=0;  
while (u>0)  
{  
   i++;  
   k=(k<<1)+(u&1);  
   u>>=1;  
}  
b=1;  
while (i-->0)  
{  
   b=chen(b,b,n);  
   if (k&1) b=chen(b,a,n);  
   k>>=1;  
}  
return b;  
}  
unsigned \_\_int64 witness(unsigned \_\_int64 a, unsigned \_\_int64 n)//计算a^(n-1)%n 其中如果遇到1的非平凡平方根 就直接返回合数  
{  
unsigned \_\_int64 i,x,y,t,u;  
u=n-1;t=0;  
while ((u&1)==0)  
{  
   u>>=1;  
   t++;  
}  
x=mood(a,u,n);  
for (i=0;i<t;i++)  
{  
   y=chen(x,x,n);  
   if (y==1&&x!=1&&x!=n-1)  
    return x;  
   x=y;  
}  
return x;  
}  
unsigned \_\_int64 miller(unsigned \_\_int64 n)//miller部分 多测几次  
{  
int i;  
unsigned \_\_int64 j;  
for (i=0;i<50;i++)  
{  
   j=rand()%(n-2)+1;  
   unsigned \_\_int64 k=witness(j,n);  
   if (k!=1)  
   {  
    return k;  
   }  
}  
return 1;  
}  
unsigned \_\_int64 gcd(unsigned \_\_int64 a,unsigned \_\_int64 b) //最大公约数  
{  
unsigned \_\_int64 c;  
if (a<b){c=a;a=b;b=c;}  
while (b!=0)  
{  
   c=a;a=b;b=c%b;  
}  
return a;  
}  
unsigned \_\_int64 pollard(unsigned \_\_int64 n,unsigned \_\_int64 x)//pollard tot用于卡时防止死掉  
{  
unsigned \_\_int64 i,k,y,d;  
unsigned \_\_int64 tot=1<<20;  
i=1;y=x;k=2;  
while (1)  
{  
   i++;  
   tot--;  
   x=(chen(x,x,n)-1+n)%n;  
   d=gcd((y-x+n)%n,n);  
   if (d!=1&&d!=n) return d;  
   if (i==k)  
   {  
    y=x;  
    k=(k<<1);  
   }  
   if (tot==0) return 1;  
  
}  
}  
int main()  
{  
unsigned \_\_int64 n,m;  
int cass;  
srand(time(0));  
for (cin>>cass;cass>0;cass--)

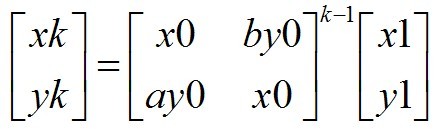
{  
   scanf("%I64d",&n);

   if (n==2)  
    goto isprim;  
   if ((n&1)==0)  
   {  
    cout<<2<<endl;  
    continue;  
   }

   if (miller(n)==1)  
   {  
isprim:    cout<<"Prime"<<endl;  
    continue;  
   }  
   unsigned \_\_int64 d=pollard(n,2);  
   n/=d;  
   unsigned \_\_int64 mind=d;  
   if (mind==1) mind=99999999999999;  
   while (miller(n)!=1)  
   {  
    m=rand()%(n-1)+1;  
    d=pollard(n,m);  
    if (d>1&&mind>d) mind=d;  
    n/=d;  
   }  
   if (mind>n) mind=n;  
   printf("%I64d\n",mind);  
}  
return 0;  
}

## 佩尔方程

特殊的方程 http://hiphotos.baidu.com/aekdycoin/pic/item/e5224da90625909bca130c0c.jpg

假设我们得到了以上方程的特解: x0 y0 (x0,y0>0,并是最小的满足条件的解)  
  
2.继续求  
http://hiphotos.baidu.com/aekdycoin/pic/item/e5224da90625909bca130c0c.jpg  
的一个最小特解.  
  
假设是x1,y1(x1,y1>0)  
  
3.  
假设你要求第k个解,那么有  
  


下面是百科的解释

若一个丢番图方程具有以下的形式：

x^2 - ny^2= \pm 1

且*n*为正整数，则称此方程为**佩尔方程**(英文：Pell's equation 德文：Pellsche Gleichung)

若*n*是完全平方数，则这个方程式只有解(\pm 1, 0)（实际上对任意的*n*，(\pm 1, 0)都是解）。对于其余情况，拉格朗日证明了佩尔方程总有解。而这些解可由\sqrt{n}的[连分数](http://zh.wikipedia.org/zh-cn/%E9%80%A3%E5%88%86%E6%95%B8)求出

另外，当*n*为偶数时，*x*2 − *ny*2 = − 1形式的方程无整数解

设\tfrac{p_i}{q_i} 是\sqrt{n}的连分数表示:[a_{0}; a_{1}, a_{2}, a_{3}, \,\ldots ]的渐近分数列，由连分数理论知存在 *i* 使得(*p*i,*q*i) 为佩尔方程的解。取其中最小的 *i*，将对应的 (*p*i,*q*i) 称为佩尔方程的**基本解**，或**最小解**，记作(*x*1,*y*1) ，则所有的解(*x*i,*y*i) 可表示成如下形式：

x_i + y_i\sqrt n = (x_1 + y_1\sqrt n)^i.

或者由以下递推公式得到：

\displaystyle x_{i+1} = x_1 x_i + n y_1 y_i,

\displaystyle y_{i+1} = x_1 y_i + y_1 x_i.

过河问题

当n>=4 考虑最慢的2个 有2种情况

1 都由最快的陪过去

2 最快的2个先过去 再回来1 个 最慢的2个过去 最快的2个里面另外一个回来 这样就化归为n-2的情况 设最快的2个用时a，b，最慢的用时 y,z

第一种情况 用时a+a+y+z 第二种情况 用时a+b+b+z 选时间少的那组

n=1,2 直接过 n=3，最快的陪

求互质对数（i，j),其中1<=i<=m,1<=j<=n m,n可到10^6

设f[i]表示数i的质因数的个数 但是 如果i 有某个质因数 在i内的指数超过1 就f[i]=-1

比如 f[1]=0;f[2.3.5]=1;f[6]=2;但是f[4]=-1没用

然后

for i=1;i<=min(m,n);i++) if (f[i]>=0)

if (f[i]%2) tot-=(n/i)\*(m/i);

else tot+=(n/i)\*(m/i);

## 约瑟夫问题

设人编号为0..n-1 数到第m个人出列

那么第t轮出列的人编号为f(n,m,t)=(m%n+f(n-1,m,t-1))%n

## 半平面交

#include <iostream>  
#include <cmath>  
using namespace std;  
const int maxn=210;  
const double pi=3.1415926535897932384626433832795;  
struct point//点  
{  
double x,y;  
};  
struct line//半平面直线 表示为a\*x+b\*y+c>=0  
{  
double a,b,c;  
double f;  
double func(point s)  
{  
   return a\*s.x+b\*s.y+c;  
}  
};  
point acro(line p,line q)//无视平行情况计算两条直线交点  
{  
point s;  
// if (p.a\*q.b-p.b\*q.a==0) return;  
s.y=(p.a\*q.c-p.c\*q.a)/(q.a\*p.b-p.a\*q.b);  
s.x=(p.b\*q.c-p.c\*q.b)/(p.a\*q.b-p.b\*q.a);  
return s;  
}  
line que[maxn];  
int head,tail;  
point a[maxn];  
line b[maxn];  
int n;  
int needtopop(line s,line a,line b)//是否需要退栈  
{  
if (s.func(acro(a,b))<0) return 1;  
return 0;  
}  
int addone(line s)//增加一个点  
{  
int hhead=head,ttail=tail;  
while (head<tail&&needtopop(s,que[tail],que[tail-1])) tail--;  
while (head<tail&&needtopop(s,que[head],que[head+1])) head++;  
tail++;  
que[tail]=s;  
return 1;  
}  
line linemove(line s,double d)//半平面正向平移d  
{  
line c=s;  
c.c-=d;  
return c;  
}  
int comp(const void \*a,const void \*b)//排序  
{  
line \*p=(line\*)a,\*q=(line\*)b;  
if (p->f>q->f) return 1;  
if (p->f<q->f) return -1;  
if (p->c>q->c) return 1;  
if (p->c<q->c) return -1;  
return 0;  
}  
int banpingmianjiao(double d)//主函数  
{  
line c[maxn];  
int i,j;  
for (i=0;i<n;i++) c[i]=linemove(b[i],d);//c储存要做的半平面  
qsort(c,n,sizeof(line),comp);//极角排序  
j=0;  
for (i=1;i<n;i++)  
   if (c[i-1].f==c[i].f) j++;  
   else c[i-j]=c[i];//干掉极角相同的  
c[n-j]=c[0];  
int m=n-j;  
if (m<=1) return 1;  
head=0;tail=1;  
que[0]=c[0];que[1]=c[1];  
  
for (i=2;i<m;i++)   
   if (addone(c[i])==0) //加线进去  
    return 0;  
while (head<tail&&needtopop(que[tail],que[head],que[head+1])) head++;//删掉头多余线  
while (head<tail&&needtopop(que[head],que[tail],que[tail-1])) tail--;//删掉尾巴多余线  
if (tail-head<2) return 0;// 必须有3条边 否则挂  
return 1;  
}

line poitoline(point p,point q)//将点q-〉p专成半平面 直线和法向量成右手螺旋  
{  
line c;  
c.a=q.y-p.y;  
c.b=p.x-q.x;  
c.c=p.y\*q.x-p.x\*q.y;  
double one=sqrt(c.a\*c.a+c.b\*c.b);  
c.a/=one;c.b/=one;c.c/=one;  
c.f=atan2(c.b,c.a);  
return c;  
}  
void prework() //预处理将凸包（顺时针）转出半平面来  
{  
int i;  
a[n]=a[0];  
for (i=0;i<n;i++)  
   b[i]=poitoline(a[i],a[i+1]);

const double maxl=999999999;  
point wei[5];  
wei[0].x=-maxl;wei[0].y=-maxl;  
wei[1].x=-maxl;wei[1].y=maxl;  
wei[2].x=maxl;wei[2].y=maxl;  
wei[3].x=maxl;wei[3].y=-maxl;  
wei[4]=wei[0];  
for (i=0;i<4;i++) b[n+i]=poitoline(wei[i],wei[i+1]);  
n+=4;  
}

double countarea(point\* s,int m)  
{  
int i;  
double tot=0;  
s[m]=s[0];  
for (i=0;i<m;i++) tot+=s[i].x\*s[i+1].y-s[i].y\*s[i+1].x;  
if (tot<0) return -tot/2;  
return tot/2;  
}  
void work()//二分  
{  
double l,r;  
l=0;r=10000000;  
while (l+0.000001<r)  
{  
   double mid=(l+r)/2;  
   if (banpingmianjiao(mid)) l=mid;else r=mid;  
}  
printf("%.6lf\n",l);  
}  
int main()  
{  
// freopen("in.in","r",stdin);  
while (scanf("%d",&n),n)  
{  
   int i;  
   for (i=n-1;i>=0;i--) scanf("%lf%lf",&a[i].x,&a[i].y);  
   prework();  
   work();  
}  
return 0;  
}

## 最小覆盖球

#include <stdio.h>  
#include <math.h>  
struct P  
{  
double x,y,z;  
}p[32],q;  
void main()  
{  
int n,i,k;  
double r,l,t,d;  
for(;scanf("%d",&n),n;printf("%.5lf\n",r))  
{  
for(i=0;i<n;scanf("%lf%lf%lf",&p[i].x,&p[i].y,&p[i].z),++i);   
for(d=r=200.0,q.x=q.y=q.z=0.0;d>1e-6;d\*=0.985)  
{   
    for(l=i=0;i<n;++i)  
    if((t=(q.x-p[i].x)\*(q.x-p[i].x)+(q.y-p[i].y)\*(q.y-p[i].y)+(q.z-p[i].z)\*(q.z-p[i].z))>l)  
       l=t,k=i;  
    if(r>(l=sqrt(l)))  
       r=l;  
    l = d/l;  
    q.x += (p[k].x-q.x)\*l;  
    q.y += (p[k].y-q.y)\*l;  
    q.z += (p[k].z-q.z)\*l;  
}   
}  
}

最小树形图

设点为0..n-1 0是根 去掉所有自环（注意不能有自环 不然会TLE）

先从0点 dfs一遍 必须能遍历到所有点 否则无解 然后 每次 找每个点的最小入边 标记最小入边的前驱节点 如果这些入边没有环 那么结束 入边和就是最小和 否则 缩环 新建点s 先把这个环上边的权值加到tot总和里面

然后 对所有（u,v）如果u 是环上点（出边） 那么连（s,v) 边权相同

如果v是环上点（入边） 设v在环上的前驱是r 连(u,s),边权为w(u,v)-w(r,v)

这样 无论最后s是由哪个外点指向 这条边加上这个环的权值 就等于 在这个环里面去掉某个点的前驱边 将这个点的前驱点改成外点

由于每次缩环 点数至少减1 所以最多缩n-1次 缩环的代价是o(m) 所以复杂度是n\*m（邻接表）或者n^3（邻接矩阵）

double mintreemap()  
{  
int c[maxn]={0};  
int p[maxn];  
int i,j,k;  
double tot=0;  
while (1)  
{  
   for (i=0;i<n;i++) p[i]=-1;  
   for (i=1;i<n;i++)if (c[i]==0)   
   {  
    map[i][i]=-1;  
    for (j=0;j<n;j++) if (c[j]==0&&map[j][i]>=0)  
     if (p[i]==-1||map[j][i]<map[p[i]][i])  
      p[i]=j;

    for (j=p[i];j!=i&&j!=-1;j=p[j]) ;

    if (j!=i) continue;

    for (j=p[i];j!=i;j=p[j]) c[j]=2;  
    c[i]=2;

    for (j=1;j<n;j++) if (c[j]==2) tot+=map[p[j]][j];

    for (j=1;j<n;j++) if (c[j]==2)  
     for (k=0;k<n;k++) if (c[k]==0&&map[k][j]>=0)  
      map[k][j]-=map[p[j]][j];

    for (j=1;j<n;j++) if (c[j]==2)  
     for (k=0;k<n;k++) if (c[k]==0)  
     {  
      if (map[k][j]>=0)  
       if (map[k][i]<0||map[k][i]>map[k][j])  
        map[k][i]=map[k][j];  
      if (map[j][k]>=0)  
       if (map[i][k]<0||map[i][k]>map[j][k])  
        map[i][k]=map[j][k];  
     }  
    for (j=1;j<n;j++) if (c[j]==2) c[j]=1;  
    c[i]=0;  
    break;  
   }  
   if (i>=n) break;  
}  
for (i=1;i<n;i++) if (c[i]==0) tot+=map[p[i]][i];  
return tot;  
}

## 最小度限制生成树

别人模板:

1. #include <stdio.h>
2. #include <string.h>
3. const int N = 20 + 3;
4. const int INF = 0x3fffffff;
5. char names[N][12];
6. int m;
7. int get\_index(char name[])
8. {
9. int i;
10. for (i=0; i<m; i++) if (strcmp(names[i], name)==0) break;
11. if (i == m) strcpy(names[m++], name);
12. return i;
13. }
14. int g[N][N], n;
15. int tree[N][N], father[N], maxs[N][2];
16. int color[N];
17. int Prim2(int v)
18. {
19. int i, j;
20. int min, total=0;
21. int adjvex[N], lowcost[N];
22. memcpy(lowcost, g[v], sizeof(lowcost));
23. for (i=1; i<n; i++) adjvex[i] = v;
24. adjvex[v] = -1;
25. color[v] = v;
26. while (1) {
27. min = INF;
28. for (i=1; i<n; i++) {
29. if (adjvex[i]>=0 && lowcost[i]<min) {
30. min = lowcost[i];
31. j = i;
32. }
33. }
34. if (min == INF) break;
35. i = adjvex[j];
36. tree[i][j] = tree[j][i] = g[i][j];
37. total += min;
38. adjvex[j] = -1;
39. color[j] = v;
40. for (i=1; i<n; i++) {
41. if (adjvex[i]>=0 && g[j][i]<lowcost[i]) {
42. lowcost[i] = g[j][i];
43. adjvex[i] = j;
44. }
45. }
46. }
47. return total;
48. }
49. void dfs(int v)
50. {
51. int i;
52. for (i=1; i<n; i++) if (i!=father[v] && tree[v][i]!=INF) {
53. father[i] = v;
54. if (v == 0) maxs[i][0] = -1;
55. else {
56. maxs[i][0] = tree[v][i];
57. maxs[i][1] = i;
58. if (maxs[v][0] > maxs[i][0]) {
59. maxs[i][0] = maxs[v][0];
60. maxs[i][1] = maxs[v][1];
61. }
62. }
63. dfs(i);
64. }
65. }
66. int solve(int s)
67. {
68. int i, j, k, min, total = 0;
69. int mins[N], vs[N];
70. for (i=1; i<n; i++) color[i] = -1, mins[i] = INF;
71. for (i=1; i<n; i++) if (color[i]==-1) total += Prim2(i);
72. for (i=1; i<n; i++) {
73. if (g[0][i] < mins[color[i]]) {
74. mins[color[i]] = g[0][i];
75. vs[color[i]] = i;
76. }
77. }
78. for (i=1; i<n; i++) {
79. if (mins[color[i]] != INF) {
80. total += mins[color[i]];
81. tree[0][vs[color[i]]] = tree[vs[color[i]]][0] = mins[color[i]];
82. mins[color[i]] = INF;
83. --s;
84. }
85. }
86. //if (s < 0) return -1;
87. memset(father, 0, sizeof(father));
88. dfs(0);
89. while (s-- > 0)
90. {
91. min = INF;
92. for (i=1; i<n; i++) {
93. if (g[0][i]!=INF && tree[0][i]==INF && g[0][i]-maxs[i][0]<min) {
94. min = g[0][i] - maxs[i][0];
95. j = i;
96. }
97. }
98. if (min >= 0) break;
99. total += min;
100. tree[0][j] = tree[j][0] = g[0][j];
101. k = maxs[j][1];
102. tree[k][father[k]] = tree[father[k]][k] = INF;
103. father[j] = 0; maxs[j][0] = -1;
104. dfs(j);
105. }
106. return total;
107. }
108. int main()
109. {
110. int i, dis;
111. char name1[12], name2[12];
112. int a, b, s;
113. for (i=0; i<N; i++) g[0][i] = INF;
114. for (i=1; i<N; i++) memcpy(g[i], g[0], sizeof(g[0]));
115. memcpy(tree, g, sizeof(g));
116. strcpy(names[0], "Park");
117. m = 1;
118. scanf("%d", &n);
119. for (i=0; i<n; i++)
120. {
121. scanf("%s%s%d", name1, name2, &dis);
122. a = get\_index(name1);
123. b = get\_index(name2);
124. if (g[a][b] > dis) g[a][b] = g[b][a] = dis;
125. }
126. n = m;
127. scanf("%d", &s);
128. printf("Total miles driven: %d\n", solve(s));
129. return 0;
130. }

我的模板:

#include <iostream>

using namespace std;

const int maxn=30;

const int maxl=999999999;

1. int n,m,v;
2. int findname(char s[20])
3. {
4. static char name[maxn][20]={"Park"};
5. int i;
6. for (i=0;i<n;i++)
7. if (strcmp(s,name[i])==0) break;
8. if (i<n) return i;
9. strcpy(name[n],s);
10. n++;
11. return i;
12. }
13. int p[maxn];
14. int c[maxn]={0};
15. void changefa(int k,int fa)
16. {
17. if (p[k]!=-1)
18. changefa(p[k],k);
19. p[k]=fa;
20. }
21. int a[maxn][maxn];
22. struct st
23. {
24. int x,y,w;
25. } l[maxn\*maxn];
26. int comp(const void \*a,const void \*b)
27. {
28. return ((st\*)a)->w-((st\*)b)->w;
29. }
30. int mv[maxn];
31. int max(int a,int b){return a>b?a:b;}
32. int findmv(int k)
33. {
34. if (mv[k]>0) return mv[k];
35. if (p[k]==0) return 0;
36. int i=findmv(p[k]);
37. if (i==0||a[i][p[i]]<a[k][p[k]])
38. mv[k]=k;
39. else mv[k]=i;
40. return mv[k];
41. }
42. int findfa(int k,int \*pa)
43. {
44. if (pa[k]==k) return k;
45. pa[k]=findfa(pa[k],pa);
46. return pa[k];
47. }
48. int findmintree()
49. {
50. int pa[maxn];
51. int i,j,k;
52. k=0;
53. for (i=0;i<n;i++) pa[i]=i;
54. for (i=0;i<m;i++)
55. if (l[i].x!=0&&l[i].y!=0)
56. if (findfa(l[i].x,pa)!=findfa(l[i].y,pa))
57. {
58. pa[findfa(l[i].x,pa)]=findfa(l[i].y,pa);
59. changefa(l[i].x,l[i].y);
60. k++;
61. }
62. for (i=1;i<n;i++) findfa(i,pa);
63. int s[maxn];
64. for (i=1;i<n;i++) s[i]=i;
65. for (i=1;i<n;i++) if (a[0][i]<a[0][s[pa[i]]])
66. s[pa[i]]=i;
67. for (i=1;i<n;i++) if (pa[i]==i)
68. {
69. changefa(s[i],0);
70. c[s[i]]=1;
71. }
72. return n-1-k;
73. }
74. int insev(int c[maxn])
75. {
76. int i,j,k;
77. int t=-1;
78. k=maxl;
79. memset(mv,0,sizeof(mv));
80. for (i=1;i<n;i++)
81. if (a[0][i]<maxl&&!c[i])
82. {
83. j=findmv(i);
84. if (a[0][i]-a[j][p[j]]<k)
85. {
86. t=i;
87. k=a[0][i]-a[j][p[j]];
88. }
89. }
90. if (t==-1) return 0;
91. c[t]=1;
92. p[mv[t]]=-1;
93. changefa(t,0);
94. return 1;
95. }
96. int count()
97. {
98. int tot=0;
99. int i;
100. for (i=1;i<n;i++) tot+=a[i][p[i]];
101. return tot;
102. }
103. int main()
104. {
105. //freopen("in.in","r",stdin);
106. int i,j,k;
107. for (i=0;i<maxn;i++) for (j=0;j<maxn;j++) a[i][j]=maxl;
108. cin>>m;
109. n=1;
110. for (i=0;i<m;i++)
111. {
112. char s1[30],s2[30];
113. cin>>s1>>s2>>k;
114. int x=findname(s1),y=findname(s2);
115. l[i].x=x;l[i].y=y;l[i].w=k;
116. if (a[x][y]>k)
117. a[x][y]=a[y][x]=k;
118. }
119. cin>>v;
120. qsort(l,m,sizeof(st),comp);
121. for (i=0;i<n;i++) p[i]=-1;
122. v-=findmintree();
123. int mins=count();
124. while (v-->0)
125. {
126. if (insev(c)==0) break;
127. int minss=count();
128. if (minss<mins) mins=minss;
129. }
130. cout<<"Total miles driven: "<<mins<<endl;
131. return 0;
132. }

## 求被多边形包含 或者被边切到也算 的格子个数

1. #include <iostream>
2. #include <stdlib.h>
3. #include <cmath>
4. using namespace std;
5. const int maxn=410;
6. struct poi
7. {
8. double x,y;
9. };
10. poi a[maxn];
11. poi s1[maxn],s2[maxn];
12. poi s[maxn];
13. int ss1,ss2,ss;
14. int n;
15. int tot;
16. int get(poi a,poi b,double x,poi& c)
17. {
18. if ((x-a.x)\*(x-b.x)>=0) return 0;
19. c.x=x;
20. c.y=(x-a.x)/(b.x-a.x)\*(b.y-a.y)+a.y;
21. return 1;
22. }
23. int comp(const void \*a,const void \*b)
24. {
25. poi\* p=(poi\*)a,\*q=(poi\*) b;
26. if (p->y>q->y) return 1;
27. if (p->y<q->y) return -1;
28. if (p->x>q->x) return 1;
29. if (p->x<q->x) return -1;
30. return 0;
31. }
32. int comp2(const void \*a,const void \*b)
33. {
34. poi\* p=(poi\*)a,\*q=(poi\*) b;
35. if (p->x>q->x) return 1;
36. if (p->x<q->x) return -1;
37. if (p->y>q->y) return 1;
38. if (p->y<q->y) return -1;
39. return 0;
40. }
41. int min(int a,int b){return a>b?b:a;}
42. int max(int a,int b){return a>b?a:b;}
43. int main()
44. {
45. int i,j;
46. //freopen("in.in","r",stdin);
47. while (cin>>n,n)
48. {
49. for (i=0;i<n;i++) cin>>a[i].x>>a[i].y;
50. a[n]=a[0];
51. tot=0;
52. for (i=-2000;i<=2000;i++)
53. {
54. ss1=ss2=ss=0;
55. for (j=0;j<n;j++)
56. {
57. ss1+=get(a[j],a[j+1],(double)i+1.0/5000.0,s1[ss1]);
58. ss2+=get(a[j],a[j+1],(double)i+1.0-1.0/5000.0,s2[ss2]);
59. }
60. qsort(s1,ss1,sizeof(poi),comp);
61. qsort(s2,ss2,sizeof(poi),comp);
62. for (j=0;j<ss1;j+=2)
63. {
64. s[ss].x=min(floor(s1[j].y),floor(s2[j].y));
65. s[ss].y=max(ceil(s1[j+1].y),ceil(s2[j+1].y));
66. ss++;
67. }
68. if (ss==0) continue;
69. qsort(s,ss,sizeof(poi),comp2);
70. int t1,t2;
71. t1=s[0].x;t2=s[0].y;
72. int tot2=tot;
73. for (j=1;j<ss;j++)
74. if (s[j].x<=t2)
75. {
76. if (s[j].y>t2) t2=s[j].y;
77. }
78. else
79. {
80. tot+=t2-t1;
81. t1=s[j].x;
82. t2=s[j].y;
83. }
84. tot+=t2-t1;
85. // cout<<" "<<tot-tot2<<endl;
86. }
87. cout<<tot<<endl;
88. }
89. return 0;
90. }