

# Code Library short



Himemiya Nanao © Perfect Freeze

August 7, 2013

# Contents

<b>1</b>	<b>data structure</b>	<b>1</b>	5.18	Stable Marriage	22
1.1	access	1	5.19	Stoer-Wagner Algorithm	22
1.2	Binary Indexed tree	1	5.20	Strongly Connected Component	22
1.3	Leftist tree	1	<b>6</b>	<b>math</b>	<b>23</b>
1.4	Size Blanced Tree	1	6.1	cantor	23
1.5	Sparse Table - rectangle	2	6.2	Discrete logarithms - BSGS	23
1.6	Sparse Table - square	2	6.3	Divisor function	23
1.7	Sparse Table	2	6.4	Extended Euclidean Algorithm	23
1.8	Trea	2	6.5	Gaussian elimination	23
<b>2</b>	<b>dynamic programming</b>	<b>3</b>	6.6	inverse element	24
2.1	knapsack problem	3	6.7	Linear programming	24
2.2	LCIS	3	6.8	Lucas' theorem(2)	24
2.3	LCS	3	6.9	Lucas' theorem	25
<b>3</b>	<b>geometry</b>	<b>3</b>	6.10	Matrix	25
3.1	3D	3	6.11	Multiset	25
3.2	3DCH	4	6.12	Pell's equation	25
3.3	circle&poy's area	5	6.13	Pollard's rho algorithm	26
3.4	circle's area	5	6.14	Reduced Residue System	26
3.5	circle	6	6.15	System of linear congruences	26
3.6	closest point pair	6	<b>7</b>	<b>other</b>	<b>27</b>
3.7	ellipse	7	7.1	bigint	27
3.8	Graham's scan	8	7.2	java	28
3.9	half-plane intersection	8	7.3	others	28
3.10	kdtree	8	<b>8</b>	<b>search</b>	<b>28</b>
3.11	others	9	8.1	dlx	28
3.12	Pick's theorem	9	8.2	dlx - exact cover	28
3.13	PointInPoly	9	8.3	dlx - repeat cover	29
3.14	rotating caliper	9	8.4	fibonacci knapsack	29
3.15	shit	10	<b>9</b>	<b>string</b>	<b>30</b>
3.16	sort - polar angle	10	9.1	Aho-Corasick Algorithm	30
3.17	triangle	10	9.2	Gusfield's Z Algorithm	30
<b>4</b>	<b>geometry/tmp</b>	<b>11</b>	9.3	Manacher's Algorithm	30
4.1	circle	11	9.4	Morris-Pratt Algorithm	31
4.2	circles	12	9.5	smallest representation	31
4.3	halfplane	12	9.6	Suffix Array - DC3 Algorithm	31
4.4	line	12	9.7	Suffix Array - Prefix-doubling Algorithm	31
4.5	line3d	13			
4.6	plane	13			
4.7	point	14			
4.8	point3d	14			
4.9	polygon	14			
4.10	polygons	16			
<b>5</b>	<b>graph</b>	<b>16</b>			
5.1	Articulation	16			
5.2	Biconnected Component - Edge	16			
5.3	Blossom algorithm	17			
5.4	chu-liu algorithm	17			
5.5	Covering problems	17			
5.6	Difference constraints	18			
5.7	Flow network	18			
5.8	Hamiltonian circuit	18			
5.9	Hopcroft-Karp algorithm	19			
5.10	Improved Shortest Augmenting Path Algorithm	19			
5.11	k Shortest Path	19			
5.12	Kariv-Hakimi Algorithm	20			
5.13	Kuhn-Munkres algorithm	20			
5.14	Manhattan minimum spanning tree	21			
5.15	Minimum Ratio Spanning Tree	21			
5.16	Minimum-cost flow problem	22			
5.17	Spanning tree	22			

# 1 data structure

## 1.1 access

```
1 //记得随手啊.....亲.....down
2 //时记得优先检查debugup/down/select
3
4 inline void rot(int id,int tp)
5 {
6     static int k;
7     k=pre[id];
8     nxt[k][tp^1]=nxt[id][tp];
9     if(nxt[id][tp])
10         pre[nxt[id][tp]]=k;
11     if(pre[k])
12         nxt[pre[k]][k==nxt[pre[k]][1]] = id;
13     pre[id]=pre[k];
14     nxt[id][tp]=k;
15     pre[k]=id;
16     up(k);
17     up(id);
18 }
19
20 inline void down(int id) //记得随手啊.....亲.....down
21 {
22     static int i;
23     if(rev[id])
24     {
25         rev[id]=false;
26         std::swap(nxt[id][0],nxt[id][1]);
27         for(i=0;i<2;i++)
28             if(nxt[id][i])
29                 rev[nxt[id][i]]^=true;
30     }
31 }
32
33 int freshen(int id)
34 {
35     int re(id);
36     if(pre[id])
37         re=freshen(pre[id]);
38     down(id);
39     return re;
40 }
41
42 inline void splay(int id)//记得随手啊.....亲.....down
43 {
44     static int rt;
45     if(id!=rt==freshen(id))
46         for(std::swap(fa[id],fa[rt]);pre[id];rot(id,id==nxt[pre[id]][0]));
47     /* another faster method:
48     if(id!=rt)
49     {
50         std::swap(fa[id],fa[rt]);
51         do
52         {
53             rt=pre[id];
54             if(pre[rt])
55             {
56                 k=nxt[pre[rt]][0]==rt;
57                 if(nxt[rt][k]==id)
58                     rot(id,k^1);
59                 else
60                     rot(rt,k);
61                 rot(id,k);
62             }
63             else
64                 rot(id,id==nxt[rt][0]);
65         }
66         while(pre[id]);
67     }
68     */
69 }
```

## 1.2 Binary Indexed tree

```
1 inline void update(int pos,const int &val)
2 {
3     while(pos<MAX)
4     {
5         tree[pos]+=val;
6         pos=pos&~pos;
7     }
8 }
9
10 inline int read(int pos)
11 {
12     int re(0);
13     while(pos>0)
14     {
15         re+=tree[pos];
16         pos=pos&~pos;
17     }
18     return re;
19 }
```

## 1.3 Leftist tree

```
1 int merge(int a,int b)
2 {
3     if(!a)
4         return b;
5     if(!b)
6         return a;
7     if(val[a]<val[b]) // max-heap
8         std::swap(a,b);
9     if(rand()<1)
10         r[a]=merge(r[a],b);
11     else
12         l[a]=merge(l[a],b);
13     fa[l[a]]=fa[r[a]]=a; // set a as father of its sons
14     return a;
15 }
```

## 1.4 Size Blanced Tree

```
1 template<class Tp>class sbt
2 {
3     public:
```

```
4     inline void init()
5     {
6         rt=cnt=1;r[0]=sz[0]=0;
7     }
8     inline void ins(const Tp &a)
9     {
10         ins(rt,a);
11     }
12     inline void del(const Tp &a)
13     {
14         del(rt,a);
15     }
16     inline Tp pred(const Tp &a)
17     {
18         return pred(rt,a);
19     }
20     inline Tp succ(const Tp &a)
21     {
22         return succ(rt,a);
23     }
24     inline void delsmall(const Tp &a)
25     {
26         dels(rt,a);
27     }
28     inline Tp delse1(int a)
29     {
30         return delse1(rt,a);
31     }
32 private:
33     int cnt,rt,l[MAX],r[MAX],sz[MAX];
34     Tp val[MAX];
35     inline void rro(int &pos)
36     {
37         int k(l[pos]);
38         l[pos]=r[k];
39         r[k]=pos;
40         sz[k]=sz[pos];
41         sz[pos]=sz[l[pos]]+sz[r[pos]]+1;
42         pos=k;
43     }
44     inline void lro(int &pos)
45     {
46         int k(r[pos]);
47         r[pos]=l[k];
48         l[k]=pos;
49         sz[k]=sz[pos];
50         sz[pos]=sz[l[pos]]+sz[r[pos]]+1;
51         pos=k;
52     }
53     inline void mt(int &pos,bool flag)
54     {
55         if(!pos)
56             return;
57         if(flag)
58             if(sz[r[r[pos]]]>sz[l[pos]])
59                 lro(pos);
60             else
61                 if(sz[l[l[pos]]]>sz[l[pos]])
62                 {
63                     rro(r[pos]);
64                     lro(pos);
65                 }
66             else
67                 return;
68         if(sz[l[l[pos]]]>sz[r[pos]])
69             rro(pos);
70         else
71             if(sz[r[l[pos]]]>sz[r[pos]])
72             {
73                 lro(l[pos]);
74                 rro(pos);
75             }
76             else
77                 return;
78         mt(l[pos],false);
79         mt(r[pos],true);
80         mt(pos,false);
81         mt(pos,true);
82     }
83     void ins(int &pos,const Tp &a)
84     {
85         if(pos)
86         {
87             ++sz[pos];
88             if(a<val[pos])
89                 ins(l[pos],a);
90             else
91                 ins(r[pos],a);
92             mt(pos,a==val[pos]);
93             return;
94         }
95         pos=++cnt;
96         l[pos]=r[pos]=0;
97         val[pos]=a;
98         sz[pos]=1;
99     }
100     Tp del(int &pos,const Tp &a)
101     {
102         --sz[pos];
103         if(val[pos]==a || (a<val[pos] && !l[pos]) || (a>val[pos] && !r[pos]))
104         {
105             Tp ret(val[pos]);
106             if(!l[pos] || !r[pos])
107                 pos=l[pos]+r[pos];
108             else
109                 val[pos]=del(l[pos],val[pos]+1);
110             return ret;
111         }
112         else
113             if(a<val[pos])
114                 return del(l[pos],a);
115             else
116                 return del(r[pos],a);
117     }
118     void dels(int &pos,const Tp &v)
119     {
120         if(!pos)
121             return;
122         if(val[pos]<v)
123         {
124             pos=r[pos];
125             dels(pos,v);
126             return;
127         }
128         dels(l[pos],v);
129         sz[pos]=1+sz[l[pos]]+sz[r[pos]];
130     }
131     Tp delse1(int &pos,int k)
132     {
```

```

133     {
134         --sz[pos];
135         if(sz[l[pos]]+1==k)
136         {
137             Tp re(val[pos]);
138             if(!l[pos] || !r[pos])
139                 pos=l[pos]+r[pos];
140             else
141                 val[pos]=del(l[pos], val[pos]+1);
142             return re;
143         }
144         if(k>sz[l[pos]])
145             return delsel(r[pos], k-1-sz[l[pos]]);
146         return delsel(l[pos], k);
147     }
148 };

```

## 1.5 Sparse Table - rectangle

```

1 #include<iostream>
2 #include<cstdio>
3 #include<algorithm>
4
5 #define MAXX 310
6
7 int mat[MAXX][MAXX];
8 int table[9][9][MAXX][MAXX];
9 int n;
10 short lg[MAXX];
11
12 int main()
13 {
14     for(int i(2); i<MAXX;++i)
15         lg[i]=lg[i>>1]+1;
16     int T;
17     std::cin >> T;
18     while (T--)
19     {
20         std::cin >> n;
21         for (int i = 0; i < n; ++i)
22             for (int j = 0; j < n; ++j)
23             {
24                 std::cin >> mat[i][j];
25                 table[0][0][i][j] = mat[i][j];
26             }
27
28         // 从小到大计算, 保证后来用到的都已经计算过
29         for(int i=0; i<=lg[n]; ++i) // width
30         {
31             for(int j=0; j<=lg[n]; ++j) //height
32             {
33                 if(i==0&&j==0)
34                     continue;
35                 for(int ii=0; ii+(1<<j)<=n;++ii)
36                     for(int jj=0; jj+(1<<i)<=n;++jj)
37                     if(i==0)
38                         table[i][j][ii][jj]=std::min(table[i][j-1][ii][jj], table[i][j-1][ii+(1<<(j-1))][jj]);
39                     else
40                         table[i][j][ii][jj]=std::min(table[i-1][j][ii][jj], table[i-1][j][ii+(1<<(i-1))][jj]);
41             }
42         }
43         long long N;
44         std::cin >> N;
45         int r1, c1, r2, c2;
46         for (int i = 0; i < N; ++i)
47         {
48             scanf("%d%d", &r1, &c1, &r2, &c2);
49             --r1;
50             --c1;
51             --r2;
52             --c2;
53             int w=lg[c2-c1+1];
54             int h=lg[r2-r1+1];
55             printf("%d\n", std::min(table[w][h][r1][c1], std::min(table[w][h][r1][c2-(1<<w)+1], std::min(table[w][h][r2-(1<<h)+1][c1], table[w][h][r2-(1<<h)+1][c2-(1<<w)+1]))));
56         }
57     }
58     return 0;
59 }

```

## 1.6 Sparse Table - square

```

1 int num[MAXX][MAXX], max[MAXX][MAXX][10];
2 short lg[MAXX];
3
4 int main()
5 {
6     for(i=2; i<MAXX;++i)
7         lg[i]=lg[i>>1]+1;
8     scanf("%d", &n, &q);
9     for(i=0; i<n;++i)
10         for(j=0; j<n;++j)
11         {
12             scanf("%d", num[i][j]);
13             max[i][j][0]=num[i][j];
14         }
15     for(k=1; k<=lg[n]; ++k)
16     {
17         l=n+1-(1<<k);
18         for(i=0; i<l;++i)
19             for(j=0; j<l;++j)
20                 max[i][j][k]=std::max(std::max(max[i][j][k-1], max[i+(1<<(k-1))][j][k-1]), std::max(max[i][j+(1<<(k-1))][k-1], max[i+(1<<(k-1))][j+(1<<(k-1))][k-1]));
21     }
22     printf("Case %d: \n", t);
23     while(q--)
24     {
25         scanf("%d%d%d", &i, &j, &l);
26         --i;
27         --j;
28         k=lg[l];
29         printf("%d\n", std::max(std::max(max[i][j][k], max[i][j+l-(1<<k)][k]), std::max(max[i+l-(1<<k)][j][k], max[i+l-(1<<k)][j+l-(1<<k)][k])));
30     }
31 }

```

## 1.7 Sparse Table

```

1 int num[MAXX], min[MAXX][20];
2 int lg[MAXX];
3
4 int main()
5 {
6     for(i=2; i<MAXX;++i)
7         lg[i]=lg[i>>1]+1;
8     scanf("%d", &n, &q);
9     for(i=1; i<=n;++i)
10     {
11         scanf("%d", num[i]);
12         min[i][0]=num[i];
13     }
14     for(j=1; j<=lg[n]; ++j)
15     {
16         l=n+1-(1<<j);
17         j_1=j-1;
18         j_2=(1<<j_1);
19         for(i=1; i<=l; ++i)
20             min[i][j]=std::min(min[i][j_1], min[i+j_1][j_1]);
21     }
22     printf("Case %d: \n", t);
23     while(q--)
24     {
25         scanf("%d", &i, &j);
26         k=lg[j-i+1];
27         printf("%d\n", std::min(min[i][k], min[j-(1<<k)+1][k]));
28     }
29 }
30 }

```

## 1.8 Treap

```

1 #include<cstdlib>
2 #include<ctime>
3 #include<cstring>
4
5 struct node
6 {
7     node *ch[2];
8     int sz, val, key;
9     node() {memset(this, 0, sizeof(node));}
10     node(int a);
11 }*null;
12
13 node::node(int a):sz(1), val(a), key(rand()-1){ch[0]=ch[1]=null;}
14
15 class Treap
16 {
17     inline void up(node *pos)
18     {
19         pos->sz=pos->ch[0]->sz+pos->ch[1]->sz+1;
20     }
21     inline void rot(node *&pos, int tp)
22     {
23         node *k(pos->ch[tp]);
24         pos->ch[tp]=k->ch[tp^1];
25         k->ch[tp^1]=pos;
26         up(pos);
27         up(k);
28         pos=k;
29     }
30
31     void insert(node *&pos, int val)
32     {
33         if(pos!=null)
34         {
35             int t(val>pos->val);
36             insert(pos->ch[t], val);
37             if(pos->ch[t]->key<pos->key)
38                 rot(pos, t);
39             else
40                 up(pos);
41             return;
42         }
43         pos=new node(val);
44     }
45     void rec(node *pos)
46     {
47         if(pos!=null)
48         {
49             rec(pos->ch[0]);
50             rec(pos->ch[1]);
51             delete pos;
52         }
53     }
54     void del(node *&pos, int val)
55     {
56         if(pos!=null)
57         {
58             if(pos->val==val)
59             {
60                 int t(pos->ch[1]->key<pos->ch[0]->key);
61                 if(pos->ch[t]==null)
62                 {
63                     delete pos;
64                     pos=null;
65                     return;
66                 }
67                 rot(pos, t);
68                 del(pos->ch[t^1], val);
69             }
70             else
71                 del(pos->ch[val>pos->val], val);
72             up(pos);
73         }
74     }
75     public:
76     node *rt;
77
78     Treap():rt(null){}
79     inline void insert(int val)
80     {
81         insert(rt, val);
82     }
83     inline void reset()
84     {
85         rec(rt);
86         rt=null;
87     }
88     inline void del(int val)
89     {
90         del(rt, val);
91     }
92 }treap[MAXX];
93
94 init:

```

```

95 {
96     srand(time(0));
97     null=new node();
98     null->val=0xc0c0c0c0;
99     null->sz=0;
100     null->key=RAND_MAX;
101     null->ch[0]=null->ch[1]=null;
102     for(i=0;i<MAXX;++i)
103         treap[i].rt=null;
104 }

```

## 2 dynamic programming

### 2.1 knapsack problem

```

1 multiple-choice knapsack problem:
2
3 for 所有的组k
4     for v=V..0
5         for 所有的属于组ik
6             f[v]=max{f[v],f[v-c[i]]+w[i]}

```

### 2.2 LCIS

```

1 #include<cstdio>
2 #include<cstring>
3 #include<vector>
4
5 #define MAXX 1111
6
7 int T;
8 int n,m,p,i,j,k;
9 std::vector<int>>the[2];
10 int dp[MAXX],path[MAXX];
11 int ans[MAXX];
12
13 int main()
14 {
15     the[0].reserve(MAXX);
16     the[1].reserve(MAXX);
17     {
18         scanf("%d",&n);
19         the[0].resize(n);
20         for(i=0;i<n;++i)
21             scanf("%d",&the[0][i]);
22         scanf("%d",&m);
23         the[1].resize(m);
24         for(i=0;i<m;++i)
25             scanf("%d",&the[1][i]);
26         memset(dp,0,sizeof dp);
27         for(i=0;i<the[0].size();++i)
28         {
29             n=0;
30             p=-1;
31             for(j=0;j<the[1].size();++j)
32             {
33                 if(the[0][i]==the[1][j] && n+1>dp[j])
34                 {
35                     dp[j]=n+1;
36                     path[j]=p;
37                 }
38                 if(the[1][j]<the[0][i] && n<dp[j])
39                 {
40                     n=dp[j];
41                     p=j;
42                 }
43             }
44             n=0;
45             p=-1;
46             for(i=0;i<the[1].size();++i)
47                 if(dp[i]>n)
48                     n=dp[p=i];
49             printf("%d\n",n);
50             for(i=n-1;i>=0;--i)
51             {
52                 ans[i]=the[1][p];
53                 p=path[p];
54             }
55             for(i=0;i<n;++i)
56                 printf("%d_",ans[i]);
57             puts("");
58         }
59         return 0;
60     }
61 }

```

### 2.3 LCS

```

1 #include<cstdio>
2 #include<algorithm>
3 #include<vector>
4
5 #define MAXX 111
6 #define N 128
7
8 std::vector<char>>the[2];
9 std::vector<int>>dp(MAXX),p[N];
10
11 int i,j,k;
12 char buf[MAXX];
13 int t;
14
15 int main()
16 {
17     the[0].reserve(MAXX);
18     the[1].reserve(MAXX);
19     while(gets(buf),buf[0]!='#')
20     {
21         the[0].resize(0);
22         for(i=0;buf[i];++i)
23             the[0].push_back(buf[i]);
24         the[1].resize(0);
25         gets(buf);
26         for(i=0;buf[i];++i)
27             the[1].push_back(buf[i]);
28         for(i=0;i<N;++i)
29             p[i].resize(0);
30         for(i=0;i<the[1].size();++i)
31             p[the[1][i]].push_back(i);
32         dp.resize(1);

```

```

33         dp[0]=-1;
34         for(i=0;i<the[0].size();++i)
35             for(j=p[the[0][i]].size()-1;j>=0;--j)
36             {
37                 k=p[the[0][i]][j];
38                 if(k<dp.back())
39                     dp.push_back(k);
40                 else
41                     *std::lower_bound(dp.begin(),dp.end(),k)=k;
42             }
43         printf("Case #%d: you can visit at most %d cities.\n",++t,dp.size()-1);
44     }
45     return 0;
46 }

```

## 3 geometry

### 3.1 3D

```

1 struct pv
2 {
3     double x,y,z;
4     pv() {}
5     pv(double xx,double yy,double zz):x(xx),y(yy),z(zz) {}
6     pv operator -(const pv& b)const
7     {
8         return pv(x-b.x,y-b.y,z-b.z);
9     }
10    pv operator *(const pv& b)const
11    {
12        return pv(y*b.z-z*b.y,z*b.x-x*b.z,x*b.y-y*b.x);
13    }
14    double operator &(const pv& b)const
15    {
16        return x*b.x+y*b.y+z*b.z;
17    }
18 };
19
20 //模
21 double Norm(pv p)
22 {
23     return sqrt(p&p);
24 }
25
26 //绕单位向量 V 旋转 theta 角度
27 pv Trans(pv pa,pv V,double theta)
28 {
29     double s = sin(theta);
30     double c = cos(theta);
31     double x,y,z;
32     x = V.x;
33     y = V.y;
34     z = V.z;
35     pv pp =
36         pv(
37             (x*x*(1-c)+c)*pa.x+(x*y*(1-c)-z*s)*pa.y+(x*z*(1-c)+y*s)*pa.z,
38             (y*x*(1-c)+z*s)*pa.x+(y*y*(1-c)+c)*pa.y+(y*z*(1-c)-x*s)*pa.z,
39             (x*z*(1-c)-y*s)*pa.x+(y*z*(1-c)+x*s)*pa.y+(z*z*(1-c)+c)*pa.z
40         );
41     return pp;
42 }
43
44 //经纬度转换
45 x=r*sin ()*cos ();
46 y=r*sin ()*sin ();
47 z=r*cos ();
48
49 r=sqrt(x*2+y*2+z*2);///?
50 r=sqrt(x^2+y^2+z^2);///?
51
52 r=atan(y/x);
53 r=acos(z/r);
54
55 r=0[,]
56 [0,2]
57 [0,]
58
59 lat1 [-/2,/2]
60 lng1 [-,]
61
62 pv getpv(double lat,double lng,double r)
63 {
64     lat += pi/2;
65     lng += pi;
66     return
67         pv(r*sin(lat)*cos(lng),r*sin(lat)*sin(lng),r*cos(lat));
68 }
69
70 //经纬度球面距离
71
72
73 #include<cstdio>
74 #include<math>
75
76 #define MAXX 1111
77
78 char buf[MAXX];
79 const double r=6875.0/2,pi=acos(-1.0);
80 double a,b,c,x1,x2,y2,ans;
81
82 int main()
83 {
84     double y1;
85     while(gets(buf)!=NULL)
86     {
87         gets(buf);
88         gets(buf);
89
90         scanf("%d%f%f%f%f\","%s\n",&a,&b,&c,&c,buf);
91         x1=a+b/60+c/3600;
92         x1=x1*pi/180;
93         if(buf[0]=='S')
94             x1=-x1;
95
96         scanf("%s",buf);
97         scanf("%d%f%f%f%f\","%s\n",&a,&b,&c,&c,buf);
98         y1=a+b/60+c/3600;
99         y1=y1*pi/180;
100         if(buf[0]=='W')
101             y1=-y1;
102
103         gets(buf);
104
105         scanf("%d%f%f%f%f\","%s\n",&a,&b,&c,&c,buf);
106         x2=a+b/60+c/3600;

```

```

107     x2=x2*pi/180;
108     if(buf[0]=='S')
109         x2=x2;
110
111     scanf("%s",buf);
112     scanf("%lf%lf%lf\n",&a,&b,&c,buf);
113     y2=a+b/60+c/3600;
114     y2=y2*pi/180;
115     if(buf[0]=='W')
116         y2=y2;
117
118     ans=acos(cos(x1)*cos(x2)*cos(y1-y2)+sin(x1)*sin(x2))*r;
119     printf("The distance to the iceberg is %.2lf miles.\n",ans);
120     if(ans>0.005<100)
121         puts("DANGER");
122
123     gets(buf);
124 }
125 return 0;
126 }
127
128 inline bool ZERO(const double &a)
129 {
130     return fabs(a)<eps;
131 }
132
133 //三维向量是否为零
134 inline bool ZERO(pv p)
135 {
136     return (ZERO(p.x) && ZERO(p.y) && ZERO(p.z));
137 }
138
139 //直线相交
140 bool LineIntersect(Line3D L1, Line3D L2)
141 {
142     pv s = L1.s-L1.e;
143     pv e = L2.s-L2.e;
144     pv p = s*e;
145     if (ZERO(p))
146         return false; //是否平行
147     p = (L2.s-L1.e)*(L1.s-L1.e);
148     return ZERO(p&L2.e); //是否共面
149 }
150
151 //线段相交
152 bool inter(pv a,pv b,pv c,pv d)
153 {
154     pv ret = (a-b)*(c-d);
155     pv t1 = (b-a)*(c-a);
156     pv t2 = (b-a)*(d-a);
157     pv t3 = (d-c)*(a-c);
158     pv t4 = (d-c)*(b-c);
159     return sgn(t1&ret)*sgn(t2&ret) < 0 && sgn(t3&ret)*sgn(t4&ret) < 0;
160 }
161
162 //点在直线上
163 bool OnLine(pv p, Line3D L)
164 {
165     return ZERO((p-L.s)*(L.e-L.s));
166 }
167
168 //点在线段上
169 bool OnSeg(pv p, Line3D L)
170 {
171     return (ZERO((L.s-p)*(L.e-p)) && EQ(Norm(p-L.s)+Norm(p-L.e),Norm(L.e-L.s)));
172 }
173
174 //点到直线距离
175 double Distance(pv p, Line3D L)
176 {
177     return (Norm((p-L.s)*(L.e-L.s))/Norm(L.e-L.s));
178 }
179
180 //线段夹角
181 //范围值为 之间的弧度[0,]
182 double Inclination(Line3D L1, Line3D L2)
183 {
184     pv u = L1.e - L1.s;
185     pv v = L2.e - L2.s;
186     return acos( (u & v) / (Norm(u)*Norm(v)) );
187 }
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

```

171 short ans=0;
172 for(short l=0;l<fac.size();l++)
173 {
174     for(j=0;j<l;j++)
175         if(same(l,j))
176             break;
177     if(j==l)
178         ++ans;
179 }
180 return ans;
181 }
182 //表面三角形数目
183 inline short trianglecnt()
184 {
185     return fac.size();
186 }
187 //三点构成的三角形面积*2
188 inline double area(const pv &a,const pv &b,const pv &c)
189 {
190     return (b-a)*(c-a).len();
191 }
192 //表面积
193 inline double area()
194 {
195     double ret(0);
196     for(i=0;i<fac.size();i++)
197         ret+=area(pnt[fac[i].a],pnt[fac[i].b],pnt[fac[i].c]);
198     return ret/2;
199 }
200 //体积
201 inline double volume()
202 {
203     pv o(0,0,0);
204     double ret(0);
205     for(short i(0);i<fac.size();i++)
206         ret+=vol(o,pnt[fac[i].a],pnt[fac[i].b],pnt[fac[i].c]);
207     return fabs(ret/6);
208 }
209 }
210
211
212

```

### 3.3 circle&ploy's area

```

1 bool InCircle(Point a,double r)
2 {
3     return cmp(a.x*a.x+a.y*a.y,r*r) <= 0;
4     //这里判断的时候 EPS 一定不要太小!!
5 }
6
7 double CalcArea(Point a,Point b,double r)
8 {
9     Point p[4];
10    int tot = 0;
11    p[tot++] = a;
12
13    Point tv = Point(a,b);
14    Line tmp = Line(Point(0,0),Point(tv.y,-tv.x));
15    Point near = LineToLine(Line(a,b),tmp);
16    if (cmp(near.x*near.x+near.y*near.y,r*r) <= 0)
17    {
18        double A,B,C;
19        A = near.x*near.x+near.y*near.y;
20        C = r;
21        B = C*C-A;
22        double tv1 = tv.x*tv.x+tv.y*tv.y;
23        double tmp = sqrt(B/tv1); //这样做只用一次开根
24        p[tot] = Point(near.x+tmp*tv.x,near.y+tmp*tv.y);
25        if (OnSeg(Line(a,b),p[tot]) == true) tot++;
26        p[tot] = Point(near.x-tmp*tv.x,near.y-tmp*tv.y);
27        if (OnSeg(Line(a,b),p[tot]) == true) tot++;
28    }
29    if (tot == 3)
30    {
31        if (cmp(Point(p[0],p[1]).Length(),Point(p[0],p[2]).Length()) > 0)
32            swap(p[1],p[2]);
33    }
34    p[tot++] = b;
35
36    double res = 0.0,theta,a0,a1,sgn;
37    for (int i = 0;i < tot-1;i++)
38    {
39        if (InCircle(p[i],r) == true && InCircle(p[i+1],r) == true)
40        {
41            res += 0.5*xmult(p[i],p[i+1]);
42        }
43        else
44        {
45            a0 = atan2(p[i+1].y,p[i+1].x);
46            a1 = atan2(p[i].y,p[i].x);
47            if (a0 < a1) a0 += 2*pi;
48            theta = a0-a1;
49            if (cmp(theta,pi) >= 0) theta = 2*pi-theta;
50            sgn = xmult(p[i],p[i+1])/2.0;
51            if (cmp(sgn,0) < 0) theta = -theta;
52            res += 0.5*r*r*theta;
53        }
54    }
55    return res;
56 }
57 //调用
58
59 area2 = 0.0;
60 for (int i = 0;i < resn;i++) //遍历每条边, 按照逆时针
61     area2 += CalcArea(p[i],p[(i+1)%resn],r);
62

```

### 3.4 circle's area

```

1 //去重
2 {
3     for (int i = 0; i < n; i++)
4     {
5         scanf("%d%d%d%d",&c[i].x,&c[i].y,&c[i].r);
6         del[i] = false;
7     }
8     for (int i = 0; i < n; i++)
9     {
10        if (del[i] == false)
11        {
12            if (c[i].r == 0.0)
13                del[i] = true;
14            for (int j = 0; j < n; j++)
15                if (i != j)
16                    if (del[j] == false)

```

```

16         if (cmp(Point(c[i].c,c[j].c).Len()+c[i].r,c[j].r) <= 0)
17             del[i] = true;
18     }
19     tn = n;
20     n = 0;
21     for (int i = 0; i < tn; i++)
22         if (del[i] == false)
23             c[n++] = c[i];
24 }
25
26 //ans[i表示被覆盖]次的面积
27 const double pi = acos(-1.0);
28 const double eps = 1e-8;
29 struct Point
30 {
31     double x,y;
32     Point(){}
33     Point(double _x,double _y)
34     {
35         x = _x;
36         y = _y;
37     }
38     double Length()
39     {
40         return sqrt(x*x+y*y);
41     }
42 };
43 struct Circle
44 {
45     Point c;
46     double r;
47 };
48 struct Event
49 {
50     double tim;
51     int typ;
52     Event(){}
53     Event(double _tim,int _typ)
54     {
55         tim = _tim;
56         typ = _typ;
57     }
58 };
59
60 int cmp(const double& a,const double& b)
61 {
62     if (fabs(a-b) < eps) return 0;
63     if (a < b) return -1;
64     return 1;
65 }
66
67 bool Eventcmp(const Event& a,const Event& b)
68 {
69     return cmp(a.tim,b.tim) < 0;
70 }
71
72 double Area(double theta,double r)
73 {
74     return 0.5*r*r*(theta-sin(theta));
75 }
76
77 double xmult(Point a,Point b)
78 {
79     return a.x*b.y-a.y*b.x;
80 }
81
82 int n,cur,tote;
83 Circle c[1000];
84 double ans[1001],pre[1001],A,B,C,B,C,theta,fai,a0,a1;
85 Event e[4000];
86 Point lab;
87
88 int main()
89 {
90     while (scanf("%d",&n) != EOF)
91     {
92         for (int i = 0;i < n;i++)
93             scanf("%d%d%d",&c[i].c.x,&c[i].c.y,&c[i].r);
94         for (int i = 1;i <= n;i++)
95             ans[i] = 0.0;
96         for (int i = 0;i < n;i++)
97         {
98             tote = 0;
99             e[tote++] = Event(-pi,1);
100            e[tote++] = Event(pi,-1);
101            for (int j = 0;j < n;j++)
102                if (j != i)
103                {
104                    lab = Point(c[j].c.x-c[i].c.x,c[j].c.y-c[i].c.y);
105                    AB = lab.Length();
106                    AC = c[i].r;
107                    BC = c[j].r;
108                    if (cmp(AB,AC+BC) <= 0)
109                    {
110                        e[tote++] = Event(-pi,1);
111                        e[tote++] = Event(pi,-1);
112                        continue;
113                    }
114                    if (cmp(AB,BC+AC) <= 0) continue;
115                    if (cmp(AB,AC+BC) > 0) continue;
116                    theta = atan2(lab.y,lab.x);
117                    fai = acos((AC*AC+AB*AB-BC*BC)/(2.0*AC*AB));
118                    a0 = theta-fai;
119                    if (cmp(a0,-pi) < 0) a0 += 2*pi;
120                    a1 = theta+fai;
121                    if (cmp(a1,pi) > 0) a1 -= 2*pi;
122                    if (cmp(a0,a1) > 0)
123                    {
124                        e[tote++] = Event(a0,1);
125                        e[tote++] = Event(pi,-1);
126                        e[tote++] = Event(-pi,1);
127                        e[tote++] = Event(a1,-1);
128                    }
129                    else
130                    {
131                        e[tote++] = Event(a0,1);
132                        e[tote++] = Event(a1,-1);
133                    }
134                }
135            sort(e,e+tote,Eventcmp);
136            cur = 0;
137            for (int j = 0;j < tote;j++)
138            {
139                if (cur != 0 && cmp(e[j].tim,pre[cur]) != 0)
140                {
141                    ans[cur] += Area(e[j].tim-pre[cur],c[i].r);
142                    ans[cur] += xmult(Point(c[i].c.x+c[i].r*cos(pre[cur]),c[i].c.y+c[i].r*sin(pre[cur])),c[i].c.x+c[i].r*cos(e[j].tim),c[i].c.y+c[i].r*sin(e[j].tim));
143                }
144            }
145        }
146    }
147 }

```

```

144         }
145         cur += e[j].typ;
146         pre[cur] = e[j].tim;
147     }
148 }
149 for (int i = 1; i < n; i++)
150     ans[i] -= ans[i+1];
151 for (int i = 1; i <= n; i++)
152     printf("%d]_=%.3f\n", i, ans[i]);
153 }
154 return 0;
155 }

```

## 3.5 circle

```

1 //单位圆覆盖
2 #include<stdio>
3 #include<math>
4 #include<vector>
5 #include<algorithm>
6
7 #define MAXX 333
8 #define eps 1e-8
9
10 struct pv
11 {
12     double x,y;
13     pv(){}
14     pv(const double &xx,const double &yy):x(xx),y(yy){}
15     inline pv operator-(const pv &i)const
16     {
17         return pv(x-i.x,y-i.y);
18     }
19     inline double cross(const pv &i)const
20     {
21         return x*i.y-y*i.x;
22     }
23     inline void print()
24     {
25         printf("%d f,%d f\n",x,y);
26     }
27     inline double len()
28     {
29         return sqrt(x*x+y*y);
30     }
31 }pnt[MAXX];
32
33 struct node
34 {
35     double k;
36     bool flag;
37     node(){}
38     node(const double &kk,const bool &ff):k(kk),flag(ff){}
39     inline bool operator<(const node &i)const
40     {
41         return k<i.k;
42     }
43 };
44
45 std::vector<node>alpha;
46
47 short n,i,j,k,l;
48 short ans,sum;
49 double R=2;
50 double theta,phi,d;
51 const double pi(acos(-1.0));
52
53 int main()
54 {
55     alpha.reserve(MAXX<1);
56     while(scanf("%d",&n),n)
57     {
58         for(i=0;i<n;i++)
59             scanf("%f,%f",&pnt[i].x,&pnt[i].y);
60         ans=0;
61         for(i=0;i<n;i++)
62         {
63             alpha.resize(0);
64             for(j=0;j<n;j++)
65                 if(i!=j)
66                 {
67                     if((d=(pnt[i]-pnt[j]).len())>R)
68                         continue;
69                     if(((theta=atan2(pnt[j].y-pnt[i].y,pnt[j].x-pnt[i].x)<0)
70                         theta+=2*pi;
71                         phi=acos(d/R);
72                         alpha.push_back(node(theta-phi,true));
73                         alpha.push_back(node(theta+phi,false));
74                     }
75                     std::sort(alpha.begin(),alpha.end());
76                     for(j=0;j<alpha.size();j++)
77                     {
78                         if(alpha[j].flag)
79                             +=sum;
80                         else
81                             -=sum;
82                     }
83                     ans=std::max(ans,sum);
84                 }
85             printf("%d\n",ans+1);
86         }
87         return 0;
88     }
89
90 //最小覆盖圆
91
92 #include<stdio>
93 #include<math>
94
95 #define MAXX 511
96 #define eps 1e-8
97
98 struct pv
99 {
100     double x,y;
101     pv(){}
102     pv(const double &xx,const double &yy):x(xx),y(yy){}
103     inline pv operator-(const pv &i)const
104     {
105         return pv(x-i.x,y-i.y);
106     }
107     inline pv operator+(const pv &i)const
108     {
109         return pv(x+i.x,y+i.y);
110     }
111     inline double cross(const pv &i)const

```

```

112     {
113         return x*i.y-y*i.x;
114     }
115     inline double len()
116     {
117         return sqrt(x*x+y*y);
118     }
119     inline pv operator/(const double &a)const
120     {
121         return pv(x/a,y/a);
122     }
123     inline pv operator*(const double &a)const
124     {
125         return pv(x*a,y*a);
126     }
127 }pnt[MAXX],o,t1,lt,aa,bb,cc,dd;
128
129 short n,i,j,k,l;
130 double r,u;
131
132 inline pv ins(const pv &a1,const pv &a2,const pv &b1,const pv &b2)
133 {
134     t1=a2-a1;
135     lt=b2-b1;
136     u=(b1-a1).cross(lt)/(t1).cross(lt);
137     return a1+t1*u;
138 }
139
140 inline pv get(const pv &a,const pv &b,const pv &x)
141 {
142     aa=(a+b)/2;
143     bb.x=aa.x-a.y+b.y;
144     bb.y=aa.y+a.x-b.x;
145     cc=(a+c)/2;
146     dd.x=cc.x-a.y+c.y;
147     dd.y=cc.y+a.x-c.x;
148     return ins(aa,bb,cc,dd);
149 }
150
151 int main()
152 {
153     while(scanf("%d",&n),n)
154     {
155         for(i=0;i<n;i++)
156             scanf("%d f,%d f",&pnt[i].x,&pnt[i].y);
157         opnt[0];
158         r=0;
159         for(i=1;i<n;i++)
160             if((pnt[i]-o).len()>r+eps)
161             {
162                 opnt[i];
163                 r=0;
164                 for(j=0;j<i;j++)
165                     if((pnt[j]-o).len()>r+eps)
166                     {
167                         o=(pnt[i]+pnt[j])/2;
168                         r=(o-pnt[j]).len();
169                         for(k=0;k<j;k++)
170                             if((o-pnt[k]).len()>r+eps)
171                             {
172                                 o=get(pnt[i],pnt[j],pnt[k]);
173                                 r=(o-pnt[i]).len();
174                             }
175                     }
176             }
177         printf("%.21f_%.21f_%.21f\n",o.x,o.y,r);
178     }
179     return 0;
180 }
181
182 //两原面积交
183 double dis(int x,int y)
184 {
185     return sqrt((double)(x*x+y*y));
186 }
187
188 double area(int x1,int y1,int x2,int y2,double r1,double r2)
189 {
190     double s=dis(x2-x1,y2-y1);
191     if(r1+r2<s) return 0;
192     else if(r2-r1>s) return PI*r1*r1;
193     else if(r1-r2>s) return PI*r2*r2;
194     double q1=acos((r1*r1+s*s-r2*r2)/(2*r1*s));
195     double q2=acos((r2*r2+s*s-r1*r1)/(2*r2*s));
196     return (r1*r1*q1+r2*r2*q2-r1*s*sin(q1));
197 }
198
199 //三角形外接圆
200 {
201     for (int i = 0; i < 3; i++)
202         scanf("%d f,%d f",&p[i].x,&p[i].y);
203     tp = pv(p[0].x+p[1].x)/2,(p[0].y+p[1].y)/2);
204     l[0] = Line(tp,pv(tp.x-(p[1].y-p[0].y),tp.y+(p[1].x-p[0].x)));
205     tp = pv(p[0].x+p[2].x)/2,(p[0].y+p[2].y)/2);
206     l[1] = Line(tp,pv(tp.x-(p[2].y-p[0].y),tp.y+(p[2].x-p[0].x)));
207     tp = LineToLine(l[0],l[1]);
208     r = pv(tp,p[0]).Length();
209     printf("(%f,%f,%f)\n",tp.x,tp.y,r);
210 }
211
212 //三角形内切圆
213 {
214     for (int i = 0; i < 3; i++)
215         scanf("%d f,%d f",&p[i].x,&p[i].y);
216     if (xmult(pv(p[0],p[1]),pv(p[0],p[2])) < 0)
217         swap(p[1],p[2]);
218     for (int i = 0; i < 3; i++)
219         len[i] = pv(p[i],p[(i+1)%3]).Length();
220     tr = (len[0]+len[1]+len[2])/2;
221     r = sqrt((tr-len[0])*(tr-len[1])*(tr-len[2])/tr);
222     for (int i = 0; i < 2; i++)
223     {
224         v = pv(p[i],p[i+1]);
225         tv = pv(-v.y,v.x);
226         tr = tv.Length();
227         tv = pv(tv.x*tr/tr,tv.y*tr/tr);
228         tp = pv(p[i].x+tv.x,p[i].y+tv.y);
229         l[i].s = tp;
230         tp = pv(p[i+1].x+tv.x,p[i+1].y+tv.y);
231         l[i].e = tp;
232     }
233     tp = LineToLine(l[0],l[1]);
234     printf("(%f,%f,%f)\n",tp.x,tp.y,r);
235 }

```

## 3.6 closest point pair



```

1 //演算法笔记1
2
3 struct Point {double x, y; } p[10], t[10];
4 bool cmpx(const Point& i, const Point& j) {return i.x < j.x;}
5 bool cmpy(const Point& i, const Point& j) {return i.y < j.y;}
6
7 double DnC(int L, int R)
8 {
9     if (L>=R) return 1e9; // 沒有點、只有一個點。
10
11     /* : 把所有點分成左右兩側，點數盡量一樣多。Divide */
12
13     int M = (L + R) / 2;
14
15     /* : 左側、右側分別遞迴求解。Conquer */
16
17     double d = min(DnC(L,M), DnC(M+1,R));
18     // if (d == 0.0) return d; // 提早結束
19
20     /* : 尋找靠近中線的點，並依座標排序。MergeYO(NlogN)。 */
21
22     int N = 0; // 靠近中線的點數目
23     for (int i=M; i>=L &&& p[M].x - p[i].x < d; --i) t[N++] = p[i];
24     for (int i=M+1; i<=R &&& p[i].x - p[M].x < d; ++i) t[N++] = p[i];
25     sort(t, t+N, cmpy); // Quicksort O(NlogN)
26
27     /* : 尋找橫跨兩側的最近點對。MergeO(N)。 */
28
29     for (int i=0; i<N-1; ++i)
30         for (int j=1; j<=2 &&& i+j<N; ++j)
31             d = min(d, distance(t[i], t[i+j]));
32
33     return d;
34 }
35
36 double closest_pair()
37 {
38     sort(p, p+10, cmpx);
39     return DnC(0, N-1);
40 }
41
42 //演算法笔记2
43
44 struct Point {double x, y; } p[10], t[10];
45 bool cmpx(const Point& i, const Point& j) {return i.x < j.x;}
46 bool cmpy(const Point& i, const Point& j) {return i.y < j.y;}
47
48 double DnC(int L, int R)
49 {
50     if (L>=R) return 1e9; // 沒有點、只有一個點。
51
52     /* : 把所有點分成左右兩側，點數盡量一樣多。Divide */
53
54     int M = (L + R) / 2;
55
56     // 先把中線的座標記起來，因為待會重新排序之後會跑掉。X
57     double x = p[M].x;
58
59     /* : 左側、右側分別遞迴求解。Conquer */
60
61     // 遞迴求解，並且依照座標重新排序。Y
62     double d = min(DnC(L,M), DnC(M+1,R));
63     // if (d == 0.0) return d; // 提早結束
64
65     /* : 尋找靠近中線的點，並依座標排序。MergeYO(N)。 */
66
67     // 尋找靠近中線的點，先找左側。各點已照座標排序了。Y
68     int N = 0; // 靠近中線的點數目
69     for (int i=0; i<=M; ++i)
70         if (x - p[i].x < d)
71             t[N++] = p[i];
72
73     // 尋找靠近中線的點，再找右側。各點已照座標排序了。Y
74     int P = N; // 為分隔位置P
75     for (int i=M+1; i<=R; ++i)
76         if (p[i].x - x < d)
77             t[N++] = p[i];
78
79     // 以座標排序。使用YMerge 方式，合併已排序的兩陣列。Sort
80     inplace_merge(t, t+P, t+N, cmpy);
81
82     /* : 尋找橫跨兩側的最近點對。MergeO(N)。 */
83
84     for (int i=0; i<N; ++i)
85         for (int j=1; j<=2 &&& i+j<N; ++j)
86             d = min(d, distance(t[i], t[i+j]));
87
88     /* : 重新以座標排序所有點。MergeYO(N)。 */
89
90     // 如此一來，更大的子問題就可以直接使用Merge 。Sort
91     inplace_merge(p+L, p+M+1, p+R+1, cmpy);
92
93     return d;
94 }
95
96 double closest_pair()
97 {
98     sort(p, p+10, cmpx);
99     return DnC(0, N-1);
100 }
101
102 //mzry
103 //分治
104 double calc_dis(Point &a ,Point &b) {
105     return sqrt((a.x-b.x)*(a.x-b.x) + (a.y-b.y)*(a.y-b.y));
106 }
107 //別忘了排序
108 bool operator<(const Point &a ,const Point &b) {
109     if(a.y != b.y) return a.x < b.x;
110     return a.x < b.x;
111 }
112
113 double Gao(int l ,int r ,Point pnts[]) {
114     double ret = inf;
115     if(l == r) return ret;
116     if(l+1 == r) {
117         ret = min(calc_dis(pnts[l], pnts[l+1]), ret);
118         return ret;
119     }
120     if(l+2 == r) {
121         ret = min(calc_dis(pnts[l], pnts[l+1]), ret);
122         ret = min(calc_dis(pnts[l], pnts[l+2]), ret);
123         ret = min(calc_dis(pnts[l+1], pnts[l+2]), ret);
124         return ret;
125     }
126
127     int mid = l+r>>1;
128     ret = min (ret ,Gao(l ,mid,pnts));
129     ret = min (ret , Gao(mid+1, r,pnts));

```

```

130
131     for(int c = 1 ; c<=r; c++)
132         for(int d = c+1; d<=c+7 &&& d<=r; d++) {
133             ret = min(ret , calc_dis(pnts[c],pnts[d]));
134         }
135     return ret;
136 }
137
138 //增量
139 #include <iostream>
140 #include <cstdio>
141 #include <cstring>
142 #include <map>
143 #include <vector>
144 #include <cmath>
145 #include <algorithm>
146 #define Point pair<double,double>
147 using namespace std;
148
149 const int step[9][2] = {{-1,-1},{-1,0},{-1,1},{0,-1},{0,0},{0,1},{1,-1},{1,0},{1,1}};
150 int n,x,y,nx,ny;
151 map<pair<int,int>,vector<Point >> > g;
152 vector<Point > tmp;
153 Point p[20000];
154 double tx,ty,ans,nowans;
155 vector<Point >::iterator it,op,ed;
156 pair<int,int> gird;
157 bool flag;
158
159 double Dis(Point p0,Point p1)
160 {
161     return sqrt((p0.first-p1.first)*(p0.first-p1.first)+
162                (p0.second-p1.second)*(p0.second-p1.second));
163 }
164
165 double CalcDis(Point p0,Point p1,Point p2)
166 {
167     return Dis(p0,p1)+Dis(p0,p2)+Dis(p1,p2);
168 }
169
170 void build(int n,double w)
171 {
172     g.clear();
173     for (int i = 0;i < n;i++)
174         g[make_pair((int)floor(p[i].first/w),(int)floor(p[i].second/w))].push_back(p[i]);
175 }
176
177 int main()
178 {
179     int t;
180     scanf("%d",&t);
181     for (int ft = 1;ft <= t;ft++)
182     {
183         scanf("%d",&n);
184         for (int i = 0;i < n;i++)
185         {
186             scanf("%d %d",&tx,&ty);
187             p[i] = make_pair(tx,ty);
188         }
189         random_shuffle(p,p+n);
190         ans = CalcDis(p[0],p[1],p[2]);
191         build(3,ans/2.0);
192         for (int i = 3;i < n;i++)
193         {
194             x = (int)floor(2.0*p[i].first/ans);
195             y = (int)floor(2.0*p[i].second/ans);
196             tmp.clear();
197             for (int k = 0;k < 9;k++)
198             {
199                 nx = x+step[k][0];
200                 ny = y+step[k][1];
201                 gird = make_pair(nx,ny);
202                 if (g.find(gird) != g.end())
203                 {
204                     op = g[gird].begin();
205                     ed = g[gird].end();
206                     for (it = op;it != ed;it++)
207                         tmp.push_back(*it);
208                 }
209             }
210             flag = false;
211             for (int j = 0;j < tmp.size();j++)
212                 for (int k = j+1;k < tmp.size();k++)
213                 {
214                     nowans = CalcDis(p[i],tmp[j],tmp[k]);
215                     if (nowans < ans)
216                     {
217                         ans = nowans;
218                         flag = true;
219                     }
220                 }
221             if (flag == true)
222                 build(i+1,ans/2.0);
223             else
224                 g[make_pair((int)floor(2.0*p[i].first/ans),(int)floor(2.0*p[i].second/ans))].push_back(p[i]);
225         }
226         printf("%.3f\n",ans);
227     }
228 }

```

## 3.7 ellipse

```

1 sq(x-h)/sq(q) + sq(y-k)/sq(b) = 1
2
3 x=h+a*cos(t);
4 y=k+b*sin(t);
5
6 area: pi*a*b;
7 distance from center to focus: f=sqrt(sq(a)-sq(b));
8 eccentricity: e=sqrt(a-sq(b/a))=f/a;
9 focal parameter: sq(b)/sqrt(sq(a)-sq(b))=sq(b)/f;
10
11 double circumference(double a,double b) // accuracy: pow(0.5,53);
12 {
13     double x=a;
14     double y=b;
15     if(x<y)
16         std::swap(x,y);
17     double digits=53,tol=sqrt(pow(0.5,digits));
18     if(digits*y<tol*x)
19         return 4*x;
20     double s=0,m=1;
21     while(x>(tol+1)*y)
22     {
23         double tx=x;
24         double ty=y;

```

```

25     x=0.5f*(tx+ty);
26     y=sqrt(tx*ty);
27     m/=2;
28     s+=m*pow(x-y,2);
29 }
30 return pi*(pow(a+b,2)-s)/(x+y);
31 }

```

## 3.8 Graham's scan

```

1  pv pnt[MXN];
2
3  inline bool com(const pv &a,const pv &b)
4  {
5      if(fabs(t=(a-pnt[0]).cross(b-pnt[0]))>eps)
6          return t>0;
7      return (a-pnt[0]).len()<(b-pnt[0]).len();
8  }
9
10 inline void graham(std::vector<pv> &ch,const int n)
11 {
12     std::nth_element(pnt,pnt+n,n);
13     std::sort(pnt+1,pnt+n,com);
14     ch.resize(0);
15     ch.push_back(pnt[0]);
16     ch.push_back(pnt[1]);
17     static int i;
18     for(i=2;i<n;i++)
19         if(fabs((pnt[i]-ch[0]).cross(ch[1]-ch[0]))>eps)
20         {
21             ch.push_back(pnt[i]);
22             break;
23         }
24         else
25             ch.back()=pnt[i];
26     for(;i<n;i++)
27     {
28         while((ch.back()-ch[ch.size()-2]).cross(pnt[i]-ch[ch.size()-2])<eps)
29             ch.pop_back();
30         ch.push_back(pnt[i]);
31     }
32 }

```

## 3.9 half-plane intersection

```

1  //解析几何方式abc
2  inline pv ins(const pv &p1,const pv &p2)
3  {
4      u=fabs(a*p1.x+b*p1.y+c);
5      v=fabs(a*p2.x+b*p2.y+c);
6      return pv((p1.x*v+p2.x*u)/(u+v),(p1.y*v+p2.y*u)/(u+v));
7  }
8
9  inline void get(const pv& p1,const pv& p2,double &a,double &b,double &c)
10 {
11     a=p2.y-p1.y;
12     b=p1.x-p2.x;
13     c=p2.x*p1.y-p2.y*p1.x;
14 }
15
16 inline pv ins(const pv &x,const pv &y)
17 {
18     get(x,y,d,e,f);
19     return pv((b*f-c*e)/(a*e-b*d),(a*f-c*d)/(b*d-a*e));
20 }
21
22 std::vector<pv> p[2];
23 inline bool go()
24 {
25     k=0;
26     p[k].resize(0);
27     p[k].push_back(pv(-inf,inf));
28     p[k].push_back(pv(-inf,-inf));
29     p[k].push_back(pv(inf,-inf));
30     p[k].push_back(pv(inf,inf));
31     for(i=0;i<n;i++)
32     {
33         get(pnt[i],pnt[(i+1)%n],a,b,c);
34         c+=he*sqrt(a*a+b*b);
35         p[k].resize(0);
36         for(i=0;i<p[k].size();i++)
37             if(a*p[k][i].x+b*p[k][i].y+c<eps)
38                 p[k].push_back(p[k][i]);
39         else
40         {
41             m=(1+p[k].size()-1)%p[k].size();
42             if(a*p[k][m].x+b*p[k][m].y+c<eps)
43                 p[k].push_back(ins(p[k][m],p[k][1]));
44             m=(1+1)%p[k].size();
45             if(a*p[k][m].x+b*p[k][m].y+c<eps)
46                 p[k].push_back(ins(p[k][m],p[k][1]));
47         }
48         k=k%2;
49         if(p[k].empty())
50             break;
51     }
52     //结果在p[k]中
53     return p[k].empty();
54 }
55
56 //计算几何方式
57 //本例求多边形核
58
59 inline pv ins(const pv &a,const pv &b)
60 {
61     u=fabs(ln.cross(a-pnt[1]));
62     v=fabs(ln.cross(b-pnt[1]));u;
63     t=l=b-a;
64     return pv(u*t1.x/v+a.x,u*t1.y/v+a.y);
65 }
66
67 int main()
68 {
69     j=0;
70     for(i=0;i<n;i++)
71     {
72         l=pnt[(i+1)%n]-pnt[i];
73         p[!j].resize(0);
74         for(k=0;k<p[j].size();k++)
75             if(ln.cross(p[j][k]-pnt[i])<=0)
76                 p[!j].push_back(p[j][k]);
77         else
78         {
79             l=(k-1+p[j].size())%p[j].size();
80             if(ln.cross(p[j][l]-pnt[i])<0)

```

```

81         p[!j].push_back(ins(p[j][k],p[j][l]));
82         l=(k+1)%p[j].size();
83         if(ln.cross(p[j][l]-pnt[i])<0)
84             p[!j].push_back(ins(p[j][k],p[j][l]));
85     }
86     j=!j;
87 }
88 //结果在p[j]中
89 }
90
91 //mrzy
92
93 bool HPlcmp(Line a, Line b)
94 {
95     if (fabs(a.k - b.k) > eps)
96         return a.k < b.k;
97     return ((a.s - b.s) * (b.e - b.s)) < 0;
98 }
99
100 Line Q[100];
101
102 void HPI(Line line[], int n, Point res[], int &resn)
103 {
104     int tot = n;
105     std::sort(line, line + n, HPlcmp);
106     tot = 1;
107     for (int i = 1; i < n; i++)
108         if (fabs(line[i].k - line[i - 1].k) > eps)
109             line[tot++] = line[i];
110     int head = 0, tail = 1;
111     Q[0] = line[0];
112     Q[1] = line[1];
113     resn = 0;
114     for (int i = 2; i < tot; i++)
115     {
116         if (fabs((Q[tail].e-Q[tail].s)*(Q[tail-1].e-Q[tail-1].s))<eps || fabs((Q[head].e-Q[head].s)*(Q[head+1].e-Q[head+1].s))<eps)
117             return;
118         while (head < tail && (((Q[tail]&Q[tail-1]) - line[i].s) * (line[i].e-line[i].s) > eps))
119             --tail;
120         while (head < tail && (((Q[head]&Q[head+1]) - line[i].s) * (line[i].e-line[i].s) > eps))
121             ++head;
122         Q[tot++] = line[i];
123     }
124     while (head < tail && (((Q[tail]&Q[tail-1]) - Q[head].s) * (Q[head].e-Q[head].s)) > eps)
125         tail--;
126     while (head < tail && (((Q[head]&Q[head+1]) - Q[tail].s) * (Q[tail].e-Q[tail].s)) > eps)
127         head++;
128     if (tail <= head + 1)
129         return;
130     for (int i = head; i < tail; i++)
131         res[resn++] = Q[i] & Q[i + 1];
132     if (head < tail + 1)
133         res[resn++] = Q[head] & Q[tail];
134 }

```

## 3.10 kdtree

```

1  #include <iostream>
2  #include <cstdio>
3  #include <cstdlib>
4  #include <algorithm>
5  #include <stack>
6  #include <algorithm>
7  using namespace std;
8  #define MXN 100010
9  typedef long long ll;
10 struct Point{
11     ll x,y;
12     void operator =(const Point &p){
13         x=p.x; y=p.y;
14     }
15     ll dis(const Point &a){
16         return (x-a.x)*(x-a.x)+(y-a.y)*(y-a.y);
17     }
18 }point[MXN],pp[MXN];
19
20 struct Node{
21     int split;//{0,1} 表示垂直于 0轴的超平面, 表示垂直于轴的超平面x1y
22     Point p;//点
23 }tree[MXN*4];
24
25 bool cmpx(const Point &a,const Point &b)
26 {
27     return a.x<b.x;
28 }
29
30 bool cmpy(const Point &a,const Point &b)
31 {
32     return a.y<b.y;
33 }
34
35 void initTree(int x,int y,int split,int pos)
36 {
37     if(y<x) return ;
38     int mid=(x+y)>>1;
39     random_shuffle(point+x,point+y);
40     if(split==0) nth_element(point+x,point+mid,point+y+1,cmpx);
41     else nth_element(point+x,point+mid,point+y+1,cmpy);
42     tree[pos].split=split;
43     tree[pos].p=point[mid];
44     initTree(x,mid-1,(split^1),2*pos);
45     initTree(mid+1,y,(split^1),2*pos+1);
46 }
47
48 ll ans;
49 void insert(int x,int y,Point &p,int pos)
50 {
51     if(y<x) return ;
52     int mid=(x+y)>>1;
53     ll temp=p.dis(tree[pos].p);
54     if(temp==0) ans=min(ans,temp);
55     if(tree[pos].split==0){
56         if(p.x<tree[pos].p.x){
57             insert(x,mid-1,p,2*pos);
58             if(ans>=p.x-tree[pos].p.x)*(p.x-tree[pos].p.x)
59                 insert(mid+1,y,p,2*pos+1);
60         }
61     }
62     else{
63         insert(mid+1,y,p,2*pos+1);
64         if(ans>=p.x-tree[pos].p.x)*(p.x-tree[pos].p.x)
65             insert(x,mid-1,p,2*pos);
66     }
67 }

```

```
66     }
67     else
68     {
69         if(p.y<=tree[pos].p.y){
70             insert(x,mid-1,p,2*pos);
71             if(ans>=p.y-tree[pos].p.y)*(p.y-tree[pos].p.y))
72                 insert(mid+1,y,p,2*pos+1);
73         }
74         else{
75             insert(mid+1,y,p,2*pos+1);
76             if(ans>=p.y-tree[pos].p.y)*(p.y-tree[pos].p.y))
77                 insert(x,mid-1,p,2*pos);
78         }
79     }
80 }
81
82 int main()
83 {
84     int cases,n;
85     scanf("%d",&cases);
86     while(cases--){
87         {
88             scanf("%d",&n);
89             for(int i=1;i<=n;i++){
90                 scanf("%d%d",&pp[i].x,&pp[i].y);
91                 point[i]=pp[i];
92             }
93             initTree(1,n,0,1);
94             for(int i=1;i<=n;i++){
95                 ans=LL<<62;
96                 insert(1,n,pp[i],1);
97                 printf("%d\n",ans);
98             }
99         }
100         return 0;
101     }
```

```
16     ray.s = p;
17     ray.e.y = p.y;
18     ray.e.x = -1; //-, 注意取值防止越界! INF
19
20     for (i = 0; i < n; i++)
21     {
22         side.s = poly[i];
23         side.e = poly[(i+1)%n];
24
25         if(OnSeg(p, side))
26             return 1;
27
28         // 如果平行轴则不考虑side
29         if (side.s.y == side.e.y)
30             continue;
31
32         if (OnSeg(side.s, ray))
33         {
34             if (side.s.y > side.e.y)
35                 count++;
36         }
37         else
38         {
39             if (OnSeg(side.e, ray))
40             {
41                 if (side.e.y > side.s.y)
42                     count++;
43             }
44             else
45                 if (inter(ray, side))
46                     count++;
47         }
48         return ((count % 2 == 1) ? 0 : 2);
49     }
```

## 3.11 others

```
1  eps如果
2
3  sqrt(a), asin(a), acos(a) 中的是你自己算出来并传进来的,那就得小心了。如果本来应该是,由于浮点误差,可能实际是一个绝对值很小的负数(比如aa0-1e),这样-12sqrt(a)应得的,直接因0不在定义域而出错。类似地,如果本来应该是 ±aa1则,asin(a)、acos(a)也有可能出错。因此,对于此种函数,必需事先对进行校正。a现在考虑一种情况,题目要求输出保留两位小数,有个的正确答案的精确值是
4
5  case0按理应该输出,但你的结果可能是恭喜:005,0:010:005000000001(),也有可能是悲剧0:004999999999(),如果按照printf("%.2lf", a)输出,那你的遭遇将和括号里的字相同。如果为正,则输出aa + eps, 否则输出a - .eps不要输出
6
7  -0.000注意的数据范围
8
9  double
10
11  a==b fabs(a-b)<eps
12  a!=b fabs(a-b)>eps
13  a<b a+eps<b
14  a<=b a<b+eps
15  a>b a>b+eps
16  a>=b a+eps>b三角函数
17
18  cos/sin/tan 输入弧度
19
20  acos 输入, 输出 [-1,+1][0.]
21
22  asin 输入, 输出 [-1,+1][-2,+2]
23
24  atan 输出 [-2,+2]
25
26  atan2 输入(y,x)注意顺序()返回,tan(y/x) 。 ,[-,+ ]都是零的时候会发生除零错误xy
27
28  other
29
30  log 自然对数(ln)
31  log10 你猜……
32  ceil 向上
33  floor 向下
34
35  round
36
37  cpp: 四舍六入五留双
38  java: add 0.5, then floor
39  cpp: (←) 当尾数小于或等于时, 直接将尾数舍去。
40  4 (二) 当尾数大于或等于时, 将尾数舍去并向前一位进位。
41  6 (三) 当尾数为, 而尾数后面的数字均为时, 应看尾数 " " 的前一位: 若前一位数字此时为奇数, 就应向前进一位; 若前一位数字此时为偶数, 则应将尾数舍去。数字 " " 在此时应被视为偶数。
42  5050 (四) 当尾数为 " " 的后面还有任何不是的数字时, 无论前一位在此时为奇数还是偶数, 也无论 " " 后面几个数字在哪一位上, 都应向前进一位。
43
44  55050
45
46  rotate mat:
47  [ cos(theta) -sin(theta) ]
48  [ sin(theta) cos(theta) ]
```

## 3.12 Pick’s theorem

```
1  给定顶点座标均是整点(或正方形格点)的简单多边形
2
3  A面积:
4  i内部格点数目:
5  b边上格点数目:
6  A = i + b/2 - . 1取格点的组成图形的面积为二单位。在平行四边形格点, 皮克定理依然成立。套用于任意三角形格点, 皮克定理则是
7
8
9  A = 2i + b - . 2
```

## 3.13 PointInPoly

```
1  /*射线法
2  , 多边形可以是凸的或凹的的顶点数目要大于等于
3  poly3返回值为:
4
5  0 -- 点在inpoly
6  1 -- 点在边界上poly
7  2 -- 点在外poly
8  */
9
10 int inPoly(pv p,pv poly[], int n)
11 {
12     int i, count;
13     Line ray, side;
14
15     count = 0;
```

```
16     ray.s = p;
17     ray.e.y = p.y;
18     ray.e.x = -1; //-, 注意取值防止越界! INF
19
20     for (i = 0; i < n; i++)
21     {
22         side.s = poly[i];
23         side.e = poly[(i+1)%n];
24
25         if(OnSeg(p, side))
26             return 1;
27
28         // 如果平行轴则不考虑side
29         if (side.s.y == side.e.y)
30             continue;
31
32         if (OnSeg(side.s, ray))
33         {
34             if (side.s.y > side.e.y)
35                 count++;
36         }
37         else
38         {
39             if (OnSeg(side.e, ray))
40             {
41                 if (side.e.y > side.s.y)
42                     count++;
43             }
44             else
45                 if (inter(ray, side))
46                     count++;
47         }
48         return ((count % 2 == 1) ? 0 : 2);
49     }
```

## 3.14 rotating caliper

```
1 //最远点对
2
3 inline double go()
4 {
5     l=ans=0;
6     for(i=0;i<n;i++)
7     {
8         t1=pnt[(i+1)%n]-pnt[i];
9         while(abs(t1.cross(pnt[(i+1)%n]-pnt[i]))>=abs(t1.cross(pnt[1]-pnt[i])))
10             l=(i+1)%n;
11         ans=std::max(ans,std::max(dist(pnt[1],pnt[i]),dist(pnt[1],pnt[(i+1)%n])));
12     }
13     return ans;
14 }
15
16 //两凸包最近距离
17 double go()
18 {
19     sq=sp=0;
20     for(i=1;i<ch[1].size();i++)
21         if(ch[1][sq]<ch[1][i])
22             sq=i;
23
24     tp=sp;
25     tq=sq;
26     ans=(ch[0][sp]-ch[1][sq]).len();
27     do
28     {
29         a1=ch[0][sp];
30         a2=ch[0][(sp+1)%ch[0].size()];
31         b1=ch[1][sq];
32         b2=ch[1][(sq+1)%ch[1].size()];
33         tpv=b1-(b2-a1);
34         tpv.y = b1.y - (b2.y - a1.y);
35         len=(tpv-a1).cross(a2-a1);
36         if(fabs(len)<eps)
37         {
38             ans=std::min(ans,p2l(a1,b1,b2));
39             ans=std::min(ans,p2l(a2,b1,b2));
40             ans=std::min(ans,p2l(b1,a1,a2));
41             ans=std::min(ans,p2l(b2,a1,a2));
42             sp=(sp+1)%ch[0].size();
43             sq=(sq+1)%ch[1].size();
44         }
45         else
46         {
47             if(len<-eps)
48             {
49                 ans=std::min(ans,p2l(b1,a1,a2));
50                 sp=(sp+1)%ch[0].size();
51             }
52             else
53             {
54                 ans=std::min(ans,p2l(a1,b1,b2));
55                 sq=(sq+1)%ch[1].size();
56             }
57         }
58     }while(tp!=sp || tq!=sq);
59     return ans;
60 }
61
62 //外接矩形 by mzy
63 inline void solve()
64 {
65     resa = resb = 1e100;
66     double dis1,dis2;
67     Point xp[4];
68     Line l[4];
69     int a,b,c,d;
70     int sa,sb,sc,sd;
71     a = b = c = d = 0;
72     sa = sb = sc = sd = 0;
73     Point va,vb,vc,vd;
74     for (a = 0; a < n; a++)
75     {
76         va = Point(p[a],p[(a+1)%n]);
77         vc = Point(-va.x,-va.y);
78         vb = Point(-va.y,va.x);
79         vd = Point(-vb.x,-vb.y);
80         if (sb < sa)
81         {
82             b = a;
83             sb = sa;
84         }
85     }
86     while (xmult(vb,Point(p[b],p[(b+1)%n])) < 0)
87     {
88         b = (b+1)%n;
89         sb++;
90     }
91     if (sc < sb)
92     {
93         c = b;
94         sc = sb;
```

```

92     }
93     while (xmult(vc,Point(p[c],p[(c+1)%n])) < 0)
94     {
95         c = (c+1)%n;
96         sc++;
97     }
98     if (sd < sc)
99     {
100         d = c;
101         sd = sc;
102     }
103     while (xmult(vd,Point(p[d],p[(d+1)%n])) < 0)
104     {
105         d = (d+1)%n;
106         sd++;
107     }
108
109     //卡在 p[a],p[b],p[c],p[d] 上
110     sa++;
111 }
112
113 //合并凸包给定凸多边形
114 P = { p(1), ..., p(m) } 和 Q = { q(1), ..., q(n), 一个点对 (p(i), q(j)) 形成 P 和 Q 之间的桥当且仅当:
115
116 (p(i), q(j)) 形成一个并踵点对。
117 p(i-1), p(i+1), q(j-1), q(j+1) 都位于由 (p(i), q(j)) 组成的线的同一侧。假设多边形以标准形式给出并且
118 顶点是以顺时针序排列, 算法如下: 、 分别计算
119
120 1 P 和 Q 拥有最大 y 坐标的顶点。如果存在不止一个这样的点, 取 x 坐标最大的。、 构造这些点的逆平行切线,
121 2 以多边形处于其右侧为正向 (因此他们指向 x 轴正方向)。、 同时顺时针旋转两条切线直到其中一条与边相交,
122 3 得到一个新的并踵点对 (p(i), q(j))。对于平行的情况, 得到三个并踵点对。、 对于所有有效的并踵点对
123 4 (p(i), q(j)): 判定 p(i-1), p(i+1), q(j-1), q(j+1) 是否都位于连接点 (p(i), q(j)) 形成的线的同一侧。如果是, 这个并踵点对就形成了一个桥, 并标记他。、 重复执行步骤和步骤直到切线回到他们原来的位置
124 534、 所有可能的桥此时都已经确定了。
125 6 通过连续连接桥间对应的凸包链来构造合并凸包。上述的结论确定了算法的正确性。运行时间受步骤, 约束。
126
127 156 他们都为 O(N) 运行时间 (N 是顶点总数)。因此算法拥有现行的时间复杂度。一个凸多边形间的桥实际上确定了一个有用的概念: 多边形间公切线。同时, 桥也是计算凸多边形交的算法核心。
128
129
130 //临界切线、计算
131 1 P 上 y 坐标值最小的顶点 (称为 yminP ) 和 Q 上 y 坐标值最大的顶点 (称为)。yminQ 为多边形在
132 2 yminP 和 ymaxQ 处构造两条切线 LP 和 LQ 使得他们对应的多边形位于他们的右侧。此时 LP 和 LQ 拥有不同的
133 方向, 并且 yminP 和 ymaxQ 成为了多边形间的一个对踵点对。、 令
134 3 p(i)=, yminP q(j)=, ymaxQ (p(i), q(j)) 构成了多边形间的一个对踵点对。检测是否
135 有 p(i-1),p(i+1) 在线 (p(i), q(j)) 的一侧, 并且 q(j-1),q(j+1) 在另一侧。如果成立,
136 (p(i), q(j)) 确定了一条线。CS、 旋转这两条线。
137 4 直到其中一条和其对应的多边形的边重合。、 一个新的对踵点对确定了。
138 5 如果两条线都与边重合, 总共三对对踵点对 (原先的顶点和新的顶点的组合) 需要考虑。对于所有的对踵点对, 执行上面
139 645 直到新的点对为(yminP,ymaxQ)。、 输出
140 7线。CS
141
142 //最小最大周长面积外接矩形//、 计算全部四个多边形的端点,
143 1 称之为, xminP, xmaxP, yminP, ymaxP, 通过四个点构造
144 2 P 的四条切线。他们确定了两个“卡壳”集合。、 如果一条 (或两条) 线与一条边重合,
145 3 那么计算由四条线决定的矩形的面积, 并且保存为当前最小值。否则将当前最小值定义为无穷大。、 顺时针旋转线直到其中一条和多边形的一条边重合。
146
147 4、 计算新矩形的周长面积,
148 5/ 并且和当前最小值比较。如果小于当前最小值则更新, 并保存确定最小值的矩形信息。、 重复步骤和步骤,
149 645 直到线旋转过角度大于度。90、 输出外接矩形的最小周长。
150 7

```

## 3.15 shit

```

1 struct pv
2 {
3     double x,y;
4     pv():x(0),y(0){}
5     pv(double xx,double yy):x(xx),y(yy){}
6     inline pv operator+(const pv &i)const
7     {
8         return pv(x+i.x,y+i.y);
9     }
10    inline pv operator-(const pv &i)const
11    {
12        return pv(x-i.x,y-i.y);
13    }
14    inline bool operator==(const pv &i)const
15    {
16        return fabs(x-i.x)<eps && fabs(y-i.y)<eps;
17    }
18    inline bool operator<(const pv &i)const
19    {
20        return y<=i.y?x<i.x:y<i.y;
21    }
22    inline double cross(const pv &i)const
23    {
24        return x*i.y-y*i.x;
25    }
26    inline double dot(const pv &i)const
27    {
28        return x*i.x+y*i.y;
29    }
30    inline double len()
31    {
32        return sqrt(x*x+y*y);
33    }
34    inline pv rotate(pv p,double theta)
35    {
36        static pv v;
37        v=this-p;
38        static double c,s;
39        c=cos(theta);
40        s=sin(theta);
41        return pv(p.x+v.x*c-v.y*s,p.y+v.x*s+v.y*c);
42    }
43 };
44
45 inline int dblcmp(double d)
46 {
47     if(fabs(d)<eps)
48         return 0;
49     return d>eps?1:-1;
50 }
51
52 inline int cross(pv *a,pv *b) // 不相交0 不规范1 规范2
53 {
54     int d1=dblcmp((a[1]-a[0]).cross(b[0]-a[0]));
55     int d2=dblcmp((a[1]-a[0]).cross(b[1]-a[0]));
56     int d3=dblcmp((b[1]-b[0]).cross(a[0]-b[0]));
57     int d4=dblcmp((b[1]-b[0]).cross(a[1]-b[0]));

```

```

58     if ((d1^d2)==2 && (d3^d4)==2)
59         return 2;
60     return ((d1==0 && dblcmp((b[0]-a[0]).dot(b[0]-a[1]))<=0) ||
61             (d2==0 && dblcmp((b[1]-a[0]).dot(b[1]-a[1]))<=0) ||
62             (d3==0 && dblcmp((a[0]-b[0]).dot(a[0]-b[1]))<=0) ||
63             (d4==0 && dblcmp((a[1]-b[0]).dot(a[1]-b[1]))<=0));
64 }
65
66 inline bool pntonseg(const pv &p,const pv *a)
67 {
68     return fabs((p-a[0]).cross(p-a[1])<eps && (p-a[0]).dot(p-a[1])<eps;
69 }
70
71 pv rotate(pv v,pv p,double theta,double sc=1) // rotate vector v, theta [0,2]
72 {
73     static pv re;
74     re=p;
75     v=v-p;
76     p.x=sc*cos(theta);
77     p.y=sc*sin(theta);
78     re.x+=v.x*p.x-v.y*p.y;
79     re.y+=v.x*p.y+v.y*p.x;
80     return re;
81 }
82
83 struct line
84 {
85     pv pnt[2];
86     line(double a,double b,double c) // a*x + b*y + c = 0
87     {
88         #define maxl 1e2 //precision should not be too high ( compare with eps )
89         if(fabs(b)>eps)
90         {
91             pnt[0]=pv(maxl,(c+a*maxl)/(-b));
92             pnt[1]=pv(-maxl,(c-a*maxl)/(-b));
93         }
94         else
95         {
96             pnt[0]=pv(-c/a,maxl);
97             pnt[1]=pv(-c/a,-maxl);
98         }
99     }
100     #undef maxl
101 }
102
103 pv cross(const line &v)const
104 {
105     double a=(v.pnt[1]-v.pnt[0]).cross(pnt[0]-v.pnt[0]);
106     double b=(v.pnt[1]-v.pnt[0]).cross(pnt[1]-v.pnt[0]);
107     return pv((pnt[0].x*b-pnt[1].x*a)/(b-a),(pnt[0].y*b-pnt[1].y*a)/(b-a));
108 }
109
110 inline std::pair<pv,double> getcircle(const pv &a,const pv &b,const pv &c)
111 {
112     static pv ct;
113     ct=line(2*(b.x-a.x),2*(b.y-a.y),a.len()-b.len()).cross(line(2*(c.x-b.x),2*(c.y-b.y),b.len()-c.len()));
114     return std::make_pair(ct,sqrt((ct-a).len()));
115 }

```

## 3.16 sort - polar angle

```

1 inline bool cmp(const Point& a,const Point& b)
2 {
3     if (a.y*b.y <= 0)
4     {
5         if (a.y > 0 || b.y > 0)
6             return a.y < b.y;
7         if (a.y == 0 && b.y == 0)
8             return a.x < b.x;
9     }
10    return a.cross(b) > 0;
11 }

```

## 3.17 triangle

```

1 Area:
2 p=(a+b+c)/2
3 area=sqrt(p*(p-a)*(p-b)*(p-c));
4 area=a*b*sin(C)/2;
5 area=sq(a)*sin(B)*sin(C)/2/sin(B/2);
6 area=sq(a)/2/(cot(B)+cot(C));
7
8 centroid:
9 center of mass
10 intersection of triangle's three triangle medians
11
12 Trigonometric conditions:
13 tan(A/2)*tan(B/2)+tan(B/2)*tan(C/2)+tan(A/2)*tan(C/2)=1
14 sq(sin(A/2))+sq(sin(B/2))+sq(sin(C/2))+2*sin(A/2)*sin(B/2)*sin(C/2)=1
15
16 Circumscribed circle:
17 diameter=a*b*c/(2*area);
18 diameter=sqrt(2*area/sin(A)/sin(B)/sin(c));
19 diameter=a/sin(A)=b/sin(B)=c/sin(C);
20
21 Incircle:
22 inradius=2*area/(a+b+c);
23 coordinates(x,y)=a*{xa,ya}/(a+b+c)+b*{xb,yb}/(a+b+c)+c*{xc,yc}/(a+b+c);
24
25 Excircles:
26 radius[a]=2*area/(b+c-a);
27 radius[b]=2*area/(a+c-b);
28 radius[c]=2*area/(a+b-c);
29
30 Steiner circumellipse (least area circumscribed ellipse)
31 area= area * 4*pi/3/sqrt(3);
32 center is the triangle's centroid.
33
34 Steiner inellipse ( maximum area inellipse )
35 area= area * pi/3/sqrt(3);
36 center is the triangle's centroid.
37
38 Fermat Point: 当有一个内角不小于 ° 时, 费马点为此角对应顶点。
39 120当三角形的内角都小于 ° 时
40
41 120以三角形的每一边为底边, 向外做三个正三角形
42 ABC 'BCA' 'CAR' 连接
43 CG 'BA' 'AA' 则三条线段的交点就是所求的点。
44

```

## 4 geometry/tmp

### 4.1 circle

```
1 struct circle
2 {
3     point p;
4     double r;
5     circle(){}
6     circle(point _p,double _r):
7     p(_p),r(_r){};
8     circle(double x,double y,double _r):
9     p(point(x,y)),r(_r){};
10    circle(point a,point b,point c)//三角形的外接圆
11    {
12        p=line(a.add(b).div(2),a.add(b).div(2).add(b.sub(a).rotleft()))).crosspoint(line(c.add(b).div(2),c.add(b).div(2).add(b.sub(c).rotleft())));
13        r=p.distance(a);
14    }
15    circle(point a,point b,point c,bool t)//三角形的内切圆
16    {
17        line u,v;
18        double m=atan2(b.y-a.y,b.x-a.x),n=atan2(c.y-a.y,c.x-a.x);
19        u=a.add(point(cos((n+m)/2),sin((n+m)/2)));
20        v=b.add(point(cos((n+m)/2),sin((n+m)/2)));
21        m=atan2(a.y-b.y,a.x-b.x),n=atan2(c.y-b.y,c.x-b.x);
22        v=b.v.a.add(point(cos((n+m)/2),sin((n+m)/2)));
23        p=u.crosspoint(v);
24        r=line(a,b).dispointtoseg(p);
25    }
26    void input()
27    {
28        p.input();
29        scanf("%lf",&r);
30    }
31    void output()
32    {
33        printf("%.2lf%.2lf%.2lf\n",p.x,p.y,r);
34    }
35    bool operator==(circle v)
36    {
37        return ((p==v.p)&&dblcmp(r-v.r)==0);
38    }
39    bool operator<(circle v)const
40    {
41        return ((p<v.p)||((p==v.p)&&dblcmp(r-v.r)<0);
42    }
43    double area()
44    {
45        return pi*sqr(r);
46    }
47    double circumference()
48    {
49        return 2*pi*r;
50    }
51    //0 圆外
52    //1 圆上
53    //2 圆内
54    int relation(point b)
55    {
56        double dst=b.distance(p);
57        if (dblcmp(dst-r)<0)return 2;
58        if (dblcmp(dst-r)==0)return 1;
59        return 0;
60    }
61    int relationseg(line v)
62    {
63        double dst=v.dispointtoseg(p);
64        if (dblcmp(dst-r)<0)return 2;
65        if (dblcmp(dst-r)==0)return 1;
66        return 0;
67    }
68    int relationline(line v)
69    {
70        double dst=v.dispointtoline(p);
71        if (dblcmp(dst-r)<0)return 2;
72        if (dblcmp(dst-r)==0)return 1;
73        return 0;
74    }
75    //过a 两点b 半径的两个圆r
76    int getcircle(point a,point b,double r,circle &c1,circle &c2)
77    {
78        circle x(a,r),y(b,r);
79        int t=x.pointcrosscircle(y,c1.p,c2.p);
80        if (t)return 0;
81        c1.r=c2.r-r;
82        return t;
83    }
84    //与直线相切u 过点q 半径的圆r1
85    int getcircle(line u,point q,double r1,circle &c1,circle &c2)
86    {
87        double dis=u.dispointtoline(q);
88        if (dblcmp(dis-r1*2)>0)return 0;
89        if (dblcmp(dis)==0)
90        {
91            c1.p=q.add(u.b.sub(u.a).rotleft().trunc(r1));
92            c2.p=q.add(u.b.sub(u.a).rotright().trunc(r1));
93            c1.r=c2.r=r1;
94            return 2;
95        }
96        line u1=line(u.a.add(u.b.sub(u.a).rotleft().trunc(r1)),u.b.add(u.b.sub(u.a).rotright().trunc(r1)));
97        line u2=line(u.a.add(u.b.sub(u.a).rotright().trunc(r1)),u.b.add(u.b.sub(u.a).rotleft().trunc(r1)));
98        circle c=circle(q,r1);
99        point p1,p2;
100        if (!cc.pointcrossline(u1,p1,p2))cc.pointcrossline(u2,p1,p2);
101        c1=circle(p1,r1);
102        if (p1==p2)
103        {
104            c2=c1;return 1;
105        }
106        c2=circle(p2,r1);
107        return 2;
108    }
109    //同时与直线u,相切v 半径的圆r1
110    int getcircle(line u,line v,double r1,circle &c1,circle &c2,circle &c3,circle &c4)
111    {
112        if (u.parallel(v))return 0;
113        line u1=line(u.a.add(u.b.sub(u.a).rotleft().trunc(r1)),u.b.add(u.b.sub(u.a).rotright().trunc(r1)));
114        line u2=line(u.a.add(u.b.sub(u.a).rotright().trunc(r1)),u.b.add(u.b.sub(u.a).rotleft().trunc(r1)));
115        line v1=line(v.a.add(v.b.sub(v.a).rotleft().trunc(r1)),v.b.add(v.b.sub(v.a).rotright().trunc(r1)));
116        line v2=line(v.a.add(v.b.sub(v.a).rotright().trunc(r1)),v.b.add(v.b.sub(v.a).rotleft().trunc(r1)));
117        line v3=line(v.a.add(v.b.sub(v.a).rotleft().trunc(r1)),v.b.add(v.b.sub(v.a).rotright().trunc(r1)));
118        c1.r=c2.r=c3.r=c4.r=r1;
119        c1.p=u1.crosspoint(v1);
120        c2.p=u1.crosspoint(v2);
121        c3.p=u2.crosspoint(v1);
122        c4.p=u2.crosspoint(v2);
123        return 4;
124    }
125    //同时与不相交圆cx,相切cy 半径为的圆r1
126    int getcircle(circle cx,circle cy,double r1,circle &c1,circle &c2)
127    {
128        circle x(cx.p,r1+cx.r),y(cy.p,r1+cy.r);
129        int t=x.pointcrosscircle(y,c1.p,c2.p);
130        if (t)return 0;
131        c1.r=c2.r=r1;
132        return t;
133    }
134    int pointcrossline(line v,point &p1,point &p2)//求与线段交要先判断relationseg
135    {
136        if (!(*this).relationline(v))return 0;
137        point a=v.lineprog(p);
138        double d=v.dispointtoline(p);
139        d=sqrt(r*r-d*d);
140        if (dblcmp(d)==0)
141        {
142            p1=a;
143            p2=a;
144            return 1;
145        }
146        p1=a.sub(v.b.sub(v.a).trunc(d));
147        p2=a.add(v.b.sub(v.a).trunc(d));
148        return 2;
149    }
150    //5 相离
151    //4 外切
152    //3 相交
153    //2 内切
154    //1 内含
155    int relationcircle(circle v)
156    {
157        double d=p.distance(v.p);
158        if (dblcmp(d-r-v.r)>0)return 5;
159        if (dblcmp(d-r-v.r)==0)return 4;
160        double l=fabs(r-v.r);
161        if (dblcmp(d-r-v.r)<0&&dblcmp(d-l)>0)return 3;
162        if (dblcmp(d-l)==0)return 2;
163        if (dblcmp(d-l)<0)return 1;
164    }
165    int pointcrosscircle(circle v,point &p1,point &p2)
166    {
167        int rel=relationcircle(v);
168        if (rel==1||rel==5)return 0;
169        double d=p.distance(v.p);
170        double l=(d+(sqr(r)-sqr(v.r))/d)/2;
171        double h=sqrt(sqr(r)-sqr(l));
172        p1=p.add(v.p.sub(p).trunc(l).add(v.p.sub(p).rotleft().trunc(h)));
173        p2=p.add(v.p.sub(p).trunc(l).add(v.p.sub(p).rotright().trunc(h)));
174        if (rel==2||rel==4)
175        {
176            return 1;
177        }
178        return 2;
179    }
180    //过一点做圆的切线 先判断点和圆关系()
181    int tangentline(point q,line &a,line &b)
182    {
183        int x=relation(q);
184        if (x==2)return 0;
185        if (x==1)
186        {
187            u=line(q,q.add(q.sub(p).rotleft()));
188            v=u;
189            return 1;
190        }
191        double d=p.distance(q);
192        double l=sqr(r)/d;
193        double h=sqrt(sqr(r)-sqr(l));
194        u=line(q,p.add(q.sub(p).trunc(l).add(q.sub(p).rotleft().trunc(h))));
195        v=line(q,p.add(q.sub(p).trunc(l).add(q.sub(p).rotright().trunc(h))));
196        return 2;
197    }
198    double areacircle(circle v)
199    {
200        int rel=relationcircle(v);
201        if (rel==4)return 0.0;
202        if (rel<=2)return min(area(),v.area());
203        double d=p.distance(v.p);
204        double hf=(r+v.r+d)/2.0;
205        double ss=2*sqr(hf*(hf-r)*(hf-v.r)*(hf-d));
206        double a1=acos((r*r+d*d-v.r*v.r)/(2.0*r*d));
207        a1=a1*r*r;
208        double a2=acos((v.r*v.r+d*d-r*r)/(2.0*v.r*d));
209        a2=a2*v.r*v.r;
210        return a1+a2-ss;
211    }
212    double areatriangle(point a,point b)
213    {
214        if (dblcmp(p.sub(a).det(p.sub(b))==0)return 0.0;
215        point q[5];
216        int len=0;
217        q[len++]=a;
218        line l(a,b);
219        point p1,p2;
220        if (pointcrossline(l,q[1],q[2])==2)
221        {
222            if (dblcmp(a.sub(q[1]).dot(b.sub(q[1]))<0)<q[1].len++)q[1]=a;
223            if (dblcmp(a.sub(q[2]).dot(b.sub(q[2]))<0)<q[1].len++)q[2]=a;
224        }
225        q[1].len++;
226        if (len==4&&dblcmp(q[0].sub(q[1]).dot(q[2].sub(q[1]))>0))swap(q[1],q[2]);
227        double res=0;
228        int i;
229        for (i=0;i<len-1;i++)
230        {
231            if (relation(q[i])==0||relation(q[i+1])==0)
232            {
233                double arg=p.rad(q[i],q[i+1]);
234                res+=r*r*arg/2.0;
235            }
236            else
237            {
238                res+=fabs(q[i].sub(p).det(q[i+1].sub(p)))/2.0;
239            }
240        }
241        return res;
242    }
243    };
244    };
245    };
246    };
247    };
248    };
249    };
250    };
251    };
252    };
253    };
254    };
255    };
256    };
257    };
258    };
259    };
260    };
261    };
262    };
263    };
264    };
265    };
266    };
267    };
268    };
269    };
270    };
271    };
272    };
273    };
274    };
275    };
276    };
277    };
278    };
279    };
280    };
281    };
282    };
283    };
284    };
285    };
286    };
287    };
288    };
289    };
290    };
291    };
292    };
293    };
294    };
295    };
296    };
297    };
298    };
299    };
300    };
301    };
302    };
303    };
304    };
305    };
306    };
307    };
308    };
309    };
310    };
311    };
312    };
313    };
314    };
315    };
316    };
317    };
318    };
319    };
320    };
321    };
322    };
323    };
324    };
325    };
326    };
327    };
328    };
329    };
330    };
331    };
332    };
333    };
334    };
335    };
336    };
337    };
338    };
339    };
340    };
341    };
342    };
343    };
344    };
345    };
346    };
347    };
348    };
349    };
350    };
351    };
352    };
353    };
354    };
355    };
356    };
357    };
358    };
359    };
360    };
361    };
362    };
363    };
364    };
365    };
366    };
367    };
368    };
369    };
370    };
371    };
372    };
373    };
374    };
375    };
376    };
377    };
378    };
379    };
380    };
381    };
382    };
383    };
384    };
385    };
386    };
387    };
388    };
389    };
390    };
391    };
392    };
393    };
394    };
395    };
396    };
397    };
398    };
399    };
400    };
401    };
402    };
403    };
404    };
405    };
406    };
407    };
408    };
409    };
410    };
411    };
412    };
413    };
414    };
415    };
416    };
417    };
418    };
419    };
420    };
421    };
422    };
423    };
424    };
425    };
426    };
427    };
428    };
429    };
430    };
431    };
432    };
433    };
434    };
435    };
436    };
437    };
438    };
439    };
440    };
441    };
442    };
443    };
444    };
445    };
446    };
447    };
448    };
449    };
450    };
451    };
452    };
453    };
454    };
455    };
456    };
457    };
458    };
459    };
460    };
461    };
462    };
463    };
464    };
465    };
466    };
467    };
468    };
469    };
470    };
471    };
472    };
473    };
474    };
475    };
476    };
477    };
478    };
479    };
480    };
481    };
482    };
483    };
484    };
485    };
486    };
487    };
488    };
489    };
490    };
491    };
492    };
493    };
494    };
495    };
496    };
497    };
498    };
499    };
500    };
501    };
502    };
503    };
504    };
505    };
506    };
507    };
508    };
509    };
510    };
511    };
512    };
513    };
514    };
515    };
516    };
517    };
518    };
519    };
520    };
521    };
522    };
523    };
524    };
525    };
526    };
527    };
528    };
529    };
530    };
531    };
532    };
533    };
534    };
535    };
536    };
537    };
538    };
539    };
540    };
541    };
542    };
543    };
544    };
545    };
546    };
547    };
548    };
549    };
550    };
551    };
552    };
553    };
554    };
555    };
556    };
557    };
558    };
559    };
560    };
561    };
562    };
563    };
564    };
565    };
566    };
567    };
568    };
569    };
570    };
571    };
572    };
573    };
574    };
575    };
576    };
577    };
578    };
579    };
580    };
581    };
582    };
583    };
584    };
585    };
586    };
587    };
588    };
589    };
590    };
591    };
592    };
593    };
594    };
595    };
596    };
597    };
598    };
599    };
600    };
601    };
602    };
603    };
604    };
605    };
606    };
607    };
608    };
609    };
610    };
611    };
612    };
613    };
614    };
615    };
616    };
617    };
618    };
619    };
620    };
621    };
622    };
623    };
624    };
625    };
626    };
627    };
628    };
629    };
630    };
631    };
632    };
633    };
634    };
635    };
636    };
637    };
638    };
639    };
640    };
641    };
642    };
643    };
644    };
645    };
646    };
647    };
648    };
649    };
650    };
651    };
652    };
653    };
654    };
655    };
656    };
657    };
658    };
659    };
660    };
661    };
662    };
663    };
664    };
665    };
666    };
667    };
668    };
669    };
670    };
671    };
672    };
673    };
674    };
675    };
676    };
677    };
678    };
679    };
680    };
681    };
682    };
683    };
684    };
685    };
686    };
687    };
688    };
689    };
690    };
691    };
692    };
693    };
694    };
695    };
696    };
697    };
698    };
699    };
700    };
701    };
702    };
703    };
704    };
705    };
706    };
707    };
708    };
709    };
710    };
711    };
712    };
713    };
714    };
715    };
716    };
717    };
718    };
719    };
720    };
721    };
722    };
723    };
724    };
725    };
726    };
727    };
728    };
729    };
730    };
731    };
732    };
733    };
734    };
735    };
736    };
737    };
738    };
739    };
740    };
741    };
742    };
743    };
744    };
745    };
746    };
747    };
748    };
749    };
750    };
751    };
752    };
753    };
754    };
755    };
756    };
757    };
758    };
759    };
760    };
761    };
762    };
763    };
764    };
765    };
766    };
767    };
768    };
769    };
770    };
771    };
772    };
773    };
774    };
775    };
776    };
777    };
778    };
779    };
780    };
781    };
782    };
783    };
784    };
785    };
786    };
787    };
788    };
789    };
790    };
791    };
792    };
793    };
794    };
795    };
796    };
797    };
798    };
799    };
800    };
801    };
802    };
803    };
804    };
805    };
806    };
807    };
808    };
809    };
810    };
811    };
812    };
813    };
814    };
815    };
816    };
817    };
818    };
819    };
820    };
821    };
822    };
823    };
824    };
825    };
826    };
827    };
828    };
829    };
830    };
831    };
832    };
833    };
834    };
835    };
836    };
837    };
838    };
839    };
840    };
841    };
842    };
843    };
844    };
845    };
846    };
847    };
848    };
849    };
850    };
851    };
852    };
853    };
854    };
855    };
856    };
857    };
858    };
859    };
860    };
861    };
862    };
863    };
864    };
865    };
866    };
867    };
868    };
869    };
870    };
871    };
872    };
873    };
874    };
875    };
876    };
877    };
878    };
879    };
880    };
881    };
882    };
883    };
884    };
885    };
886    };
887    };
888    };
889    };
890    };
891    };
892    };
893    };
894    };
895    };
896    };
897    };
898    };
899    };
900    };
901    };
902    };
903    };
904    };
905    };
906    };
907    };
908    };
909    };
910    };
911    };
912    };
913    };
914    };
915    };
916    };
917    };
918    };
919    };
920    };
921    };
922    };
923    };
924    };
925    };
926    };
927    };
928    };
929    };
930    };
931    };
932    };
933    };
934    };
935    };
936    };
937    };
938    };
939    };
940    };
941    };
942    };
943    };
944    };
945    };
946    };
947    };
948    };
949    };
950    };
951    };
952    };
953    };
954    };
955    };
956    };
957    };
958    };
959    };
960    };
961    };
962    };
963    };
964    };
965    };
966    };
967    };
968    };
969    };
970    };
971    };
972    };
973    };
974    };
975    };
976    };
977    };
978    };
979    };
980    };
981    };
982    };
983    };
984    };
985    };
986    };
987    };
988    };
989    };
990    };
991    };
992    };
993    };
994    };
995    };
996    };
997    };
998    };
999    };
1000    };
```

## 4.2 circles

```

1 const int maxn=500;
2 struct circles
3 {
4     circle c[maxn];
5     double ans[maxn]; //ans[i表示被覆盖了]次的面积i
6     double pre[maxn];
7     int n;
8     circles(){}
9     void add(circle cc)
10    {
11        c[n++]=cc;
12    }
13    bool inner(circle x,circle y)
14    {
15        if (x.relationcircle(y)!=1)return 0;
16        return dblcmp(x.r-y.r)<=0?1:0;
17    }
18    void init_or() //圆的面积并去掉内含的圆
19    {
20        int i,j,k=0;
21        bool mark[maxn]={0};
22        for (i=0;i<n;i++)
23        {
24            for (j=0;j<n;j++)if (i!=j&&mark[j])
25            {
26                if ((c[i]==c[j])||inner(c[i],c[j]))break;
27            }
28            if (j<n)mark[i]=1;
29        }
30        for (i=0;i<n;i++)if (!mark[i])c[k++]=c[i];
31        n=k;
32    }
33    void init_and() //圆的面积交去掉内含的圆
34    {
35        int i,j,k=0;
36        bool mark[maxn]={0};
37        for (i=0;i<n;i++)
38        {
39            for (j=0;j<n;j++)if (i!=j&&mark[j])
40            {
41                if ((c[i]==c[j])||inner(c[j],c[i]))break;
42            }
43            if (j<n)mark[i]=1;
44        }
45        for (i=0;i<n;i++)if (!mark[i])c[k++]=c[i];
46        n=k;
47    }
48    double areaarc(double th,double r)
49    {
50        return 0.5*sqr(r)*(th-sin(th));
51    }
52    void getarea()
53    {
54        int i,j,k;
55        memset(ans,0,sizeof(ans));
56        vector<pair<double,int>>v;
57        for (i=0;i<n;i++)
58        {
59            v.clear();
60            v.push_back(make_pair(-pi,1));
61            v.push_back(make_pair(pi,-1));
62            for (j=0;j<n;j++)if (i!=j)
63            {
64                point q=c[j].p.sub(c[i].p);
65                double ab=q.len(),ac=c[i].r,bc=c[j].r;
66                if (dblcmp(ab+ac-bc)<=0)
67                {
68                    v.push_back(make_pair(-pi,1));
69                    v.push_back(make_pair(pi,-1));
70                    continue;
71                }
72                if (dblcmp(ab+bc-ac)<=0)continue;
73                if (dblcmp(ab-ac-bc)>0) continue;
74                double th=atan2(q.y,q.x),fa=acos((ac*ac+ab*ab-bc*bc)/(2.0*ac*ab));
75                double a0=th-fa;
76                if (dblcmp(a0+pi)<0)a0+=2*pi;
77                double a1=th+fa;
78                if (dblcmp(a1-pi)>0)a1-=2*pi;
79                if (dblcmp(a0-a1)>0)
80                {
81                    v.push_back(make_pair(a0,1));
82                    v.push_back(make_pair(pi,-1));
83                    v.push_back(make_pair(-pi,1));
84                    v.push_back(make_pair(a1,-1));
85                }
86                else
87                {
88                    v.push_back(make_pair(a0,1));
89                    v.push_back(make_pair(a1,-1));
90                }
91            }
92            sort(v.begin(),v.end());
93            int cur=0;
94            for (j=0;j<v.size();j++)
95            {
96                if (cur&&dblcmp(v[j].first-pre[cur]))
97                {
98                    ans[cur]+=areaarc(v[j].first-pre[cur],c[i].r);
99                    ans[cur]+=0.5*point(c[i].p.x+c[i].r*cos(pre[cur]),c[i].p.y+c[i].r*sin(pre[cur]),
100                    .det(point(c[i].p.x+c[i].r*cos(v[j].first),c[i].p.y+c[i].r*sin(v[j].first),
101                    ));
102                    cur+=v[j].second;
103                    pre[cur]=v[j].first;
104                }
105                for (i=1;i<=n;i++)
106                {
107                    ans[i]=ans[i+1];
108                }
109            }
110        };

```

## 4.3 halfplane

```

1 struct halfplane:public line
2 {
3     double angle;
4     halfplane(){}
5     //表示向量 a->逆时针b左侧()的半平面
6     halfplane(point _a,point _b)
7     {
8         a=_a;
9         b=_b;

```

```

10     }
11     halfplane(line v)
12     {
13         a=v.a;
14         b=v.b;
15     }
16     void calcangle()
17     {
18         angle=atan2(b.y-a.y,b.x-a.x);
19     }
20     bool operator<(const halfplane &b)const
21     {
22         return angle<b.angle;
23     }
24 };
25 struct halfplanes
26 {
27     int n;
28     halfplane hp[maxn];
29     point p[maxn];
30     int que[maxn];
31     int st,ed;
32     void push(halfplane tmp)
33     {
34         hp[n++]=tmp;
35     }
36     void unique()
37     {
38         int m=1,i;
39         for (i=1;i<n;i++)
40         {
41             if (dblcmp(hp[i].angle-hp[i-1].angle))hp[m++]=hp[i];
42             else if (dblcmp(hp[m-1].b.sub(hp[m-1].a).det(hp[i].a.sub(hp[m-1].a))>0))hp[m]=hp[i];
43         }
44         n=m;
45     }
46     bool halfplaneinsert()
47     {
48         int i;
49         for (i=0;i<n;i++)hp[i].calcangle();
50         sort(hp,hp+n);
51         unique();
52         que[st=0]=0;
53         que[ed=1]=1;
54         p[1]=hp[0].crosspoint(hp[1]);
55         for (i=2;i<n;i++)
56         {
57             while (st<ed&&dblcmp((hp[i].b.sub(hp[i].a).det(p[ed].sub(hp[i].a)))<0)ed--;
58             while (st<ed&&dblcmp((hp[i].b.sub(hp[i].a).det(p[st+1].sub(hp[i].a)))<0)st++;
59             que[st]=i;
60             if (hp[i].parallel(hp[que[ed-1]]))return false;
61             p[ed]=hp[i].crosspoint(hp[que[ed-1]]);
62         }
63         while (st<ed&&dblcmp(hp[que[st]].b.sub(hp[que[st]].a).det(p[ed].sub(hp[que[st]].a)))<0)ed--;
64         while (st<ed&&dblcmp(hp[que[ed]].b.sub(hp[que[ed]].a).det(p[st+1].sub(hp[que[ed]].a)))<0)st++;
65         if (st+1==ed)return false;
66         return true;
67     }
68     void getconvex(polygon &con)
69     {
70         p[st]=hp[que[st]].crosspoint(hp[que[ed]]);
71         con.n=ed-st+1;
72         int j=st,i=0;
73         for (;j<=ed;j++)
74         {
75             con.p[i]=p[j];
76         }
77     }
78 };

```

## 4.4 line

```

1 struct line
2 {
3     point a,b;
4     line(){}
5     line(point _a,point _b)
6     {
7         a=_a;
8         b=_b;
9     }
10    bool operator==(line v)
11    {
12        return (a==v.a)&&(b==v.b);
13    }
14    //倾斜角angle
15    line(point p,double angle)
16    {
17        a=p;
18        if (dblcmp(angle-pi/2)==0)
19        {
20            b=a.add(point(0,1));
21        }
22        else
23        {
24            b=a.add(point(1,tan(angle)));
25        }
26    }
27    //ax+by+c=0
28    line(double _a,double _b,double _c)
29    {
30        if (dblcmp(_a)==0)
31        {
32            a=point(0,-_c/_b);
33            b=point(1,-_c/_b);
34        }
35        else if (dblcmp(_b)==0)
36        {
37            a=point(-_c/_a,0);
38            b=point(-_c/_a,1);
39        }
40        else
41        {
42            a=point(0,-_c/_b);
43            b=point(1,(-_c-_a)/_b);
44        }
45    }
46    void input()
47    {
48        a.input();
49        b.input();
50    }
51    void adjust()
52    {

```

```

53     if (b<a)swap(a,b);
54 }
55 double length()
56 {
57     return a.distance(b);
58 }
59 double angle()//直线倾斜角 0<=angle<180
60 {
61     double k=atan2(b.y-a.y,b.x-a.x);
62     if (dbclmp(k)<0)k+=pi;
63     if (dbclmp(k-pi)==0)k=pi;
64     return k;
65 }
66 //点和线段关系
67 //1 在逆时针
68 //2 在顺时针
69 //3 平行
70 int relation(point p)
71 {
72     int c=dbclmp(p.sub(a).det(b.sub(a)));
73     if (c<0)return 1;
74     if (c>0)return 2;
75     return 3;
76 }
77 bool pointonseg(point p)
78 {
79     return dbclmp(p.sub(a).det(b.sub(a)))>=0&&dbclmp(p.sub(a).dot(p.sub(b)))<=0;
80 }
81 bool parallel(line v)
82 {
83     return dbclmp(b.sub(a).det(v.b.sub(v.a)))==0;
84 }
85 //2 规范相交
86 //1 非规范相交
87 //0 不相交
88 int segcrossseg(line v)
89 {
90     int d1=dbclmp(b.sub(a).det(v.a.sub(a)));
91     int d2=dbclmp(b.sub(a).det(v.b.sub(a)));
92     int d3=dbclmp(v.b.sub(v.a).det(a.sub(v.a)));
93     int d4=dbclmp(v.b.sub(v.a).det(b.sub(v.a)));
94     if ((d1^d2)==2&&(d3^d4)==2)return 2;
95     return (d1==0&&dbclmp(v.a.sub(a).dot(v.a.sub(b)))<=0||
96           d2==0&&dbclmp(v.b.sub(a).dot(v.b.sub(b)))<=0||
97           d3==0&&dbclmp(a.sub(v.a).dot(a.sub(v.b)))<=0||
98           d4==0&&dbclmp(b.sub(v.a).dot(b.sub(v.b)))<=0);
99 }
100 int linecrossseg(line v)/*this seg v line
101 {
102     int d1=dbclmp(b.sub(a).det(v.a.sub(a)));
103     int d2=dbclmp(b.sub(a).det(v.b.sub(a)));
104     if ((d1^d2)==2)return 2;
105     return (d1==0||d2==0);
106 }
107 //0 平行
108 //1 重合
109 //2 相交
110 int linecrossline(line v)
111 {
112     if ((*this).parallel(v))
113     {
114         return v.relation(a)==3;
115     }
116     return 2;
117 }
118 point crosspoint(line v)
119 {
120     double a1=v.b.sub(v.a).det(a.sub(v.a));
121     double a2=v.b.sub(v.a).det(b.sub(v.a));
122     return point((a.x*a2-b.x*a1)/(a2-a1),(a.y*a2-b.y*a1)/(a2-a1));
123 }
124 double dispointtoline(point p)
125 {
126     return fabs(p.sub(a).det(b.sub(a)))/length();
127 }
128 double dispointtoseg(point p)
129 {
130     if (dbclmp(p.sub(b).dot(a.sub(b)))<0||dbclmp(p.sub(a).dot(b.sub(a)))<0)
131     {
132         return min(p.distance(a),p.distance(b));
133     }
134     return dispointtoline(p);
135 }
136 point lineprog(point p)
137 {
138     return a.add(b.sub(a).mul(b.sub(a).dot(p.sub(a))/b.sub(a).len2()));
139 }
140 point symmetrypoint(point p)
141 {
142     point q=lineprog(p);
143     return point(2*q.x-p.x,2*q.y-p.y);
144 }
145 };

```

## 4.5 line3d

```

1 struct line3
2 {
3     point3 a,b;
4     line3(){}
5     line3(point3 _a,point3 _b)
6     {
7         a=_a;
8         b=_b;
9     }
10    bool operator==(line3 v)
11    {
12        return (a==v.a)&&(b==v.b);
13    }
14    void input()
15    {
16        a.input();
17        b.input();
18    }
19    double length()
20    {
21        return a.distance(b);
22    }
23    bool pointonseg(point3 p)
24    {
25        return dbclmp(p.sub(a).det(p.sub(b)).len())>=0&&dbclmp(a.sub(p).dot(b.sub(p)))<=0;
26    }
27    double dispointtoline(point3 p)
28    {
29        return b.sub(a).det(p.sub(a)).len()/a.distance(b);
30    }
31    double dispointtoseg(point3 p)

```

```

{
    if (dbclmp(p.sub(b).dot(a.sub(b)))<0||dbclmp(p.sub(a).dot(b.sub(a)))<0)
    {
        return min(p.distance(a),p.distance(b));
    }
    return dispointtoline(p);
}
point3 lineprog(point3 p)
{
    return a.add(b.sub(a).trunc(b.sub(a).dot(p.sub(a))/b.distance(a)));
}
point3 rotate(point3 p,double ang)//绕此向量逆时针角度pang
{
    if (dbclmp((p.sub(a).det(p.sub(b)).len())==0)return p;
    point3 f1=b.sub(a).det(p.sub(a));
    point3 f2=b.sub(a).det(f1);
    double len=fabs(a.sub(p).det(b.sub(p)).len()/a.distance(b));
    f1=f1.trunc(len);f2=f2.trunc(len);
    point3 h=p.add(f2);
    point3 pp=h.add(f1);
    return h.add((p.sub(h)).mul(cos(ang*1.0)).add((pp.sub(h)).mul(sin(ang*1.0)));
}
};

```

## 4.6 plane

```

1 struct plane
2 {
3     point3 a,b,c,o;
4     plane(){}
5     plane(point3 _a,point3 _b,point3 _c)
6     {
7         a=_a;
8         b=_b;
9         c=_c;
10        o=pvec();
11    }
12    plane(double _a,double _b,double _c,double _d)
13    {
14        //ax+by+cz+d=0
15        o=point3(_a,_b,_c);
16        if (dbclmp(_a)!=0)
17        {
18            a=point3((-_d-_c*_b)/_a,1,1);
19        }
20        else if (dbclmp(_b)!=0)
21        {
22            a=point3(1,(-_d-_c*_a)/_b,1);
23        }
24        else if (dbclmp(_c)!=0)
25        {
26            a=point3(1,1,(-_d-_a*_b)/_c);
27        }
28    }
29    void input()
30    {
31        a.input();
32        b.input();
33        c.input();
34        o=pvec();
35    }
36    point3 pvec()
37    {
38        return b.sub(a).det(c.sub(a));
39    }
40    bool pointonplane(point3 p)//点是否在平面上
41    {
42        return dbclmp(p.sub(a).dot(o))==0;
43    }
44    //0 不在
45    //1 在边界上
46    //2 在内部
47    int pointontriangle(point3 p)//点是否在空间三角形abc
48    {
49        if (!pointonplane(p))return 0;
50        double s=a.sub(b).det(c.sub(b)).len();
51        double s1=p.sub(a).det(p.sub(b)).len();
52        double s2=p.sub(a).det(p.sub(c)).len();
53        double s3=p.sub(b).det(p.sub(c)).len();
54        if (dbclmp(s-s1-s2-s3))return 0;
55        if (dbclmp(s1)&&dbclmp(s2)&&dbclmp(s3))return 2;
56        return 1;
57    }
58    //判断两平面关系
59    //0 相交
60    //1 平行但不重合
61    //2 重合
62    bool relationplane(plane f)
63    {
64        if (dbclmp(o.det(f.o).len())return 0;
65        if (pointonplane(f.a))return 2;
66        return 1;
67    }
68    double angleplane(plane f)//两平面夹角
69    {
70        return acos(o.dot(f.o)/(o.len()*f.o.len()));
71    }
72    double dispoint(point3 p)//点到平面距离
73    {
74        return fabs(p.sub(a).dot(o)/o.len());
75    }
76    point3 pttoplane(point3 p)//点到平面最近点
77    {
78        line3 u=line3(p,p.add(o));
79        crossline(u,p);
80        return p;
81    }
82    int crossline(line3 u,point3 &p)//平面和直线的交点
83    {
84        double x=o.dot(u.b.sub(a));
85        double y=o.dot(u.a.sub(a));
86        double d=x-y;
87        if (dbclmp(fabs(d))==0)return 0;
88        p=u.a.mul(x).sub(u.b.mul(y)).div(d);
89        return 1;
90    }
91    int crossplane(plane f,line3 &u)//平面和平面的交线
92    {
93        point3 oo=o.det(f.o);
94        point3 vo=o.det(oo);
95        double d=fabs(f.o.dot(v));
96        if (dbclmp(d)==0)return 0;
97        point3 qa.add(v.mul(f.o.dot(f.a.sub(a))/d));
98        u=line3(q,q.add(oo));
99        return 1;
100    }
101 };

```

## 4.7 point

```

1 using namespace std;
2
3 #define mp make_pair
4 #define pb push_back
5
6 const double eps=1e-8;
7 const double pi=acos(-1.0);
8 const double inf=1e20;
9 const int maxp=8;
10
11 int dblcmp(double d)
12 {
13     if (fabs(d)<eps)return 0;
14     return d>eps?1:-1;
15 }
16
17 inline double sqr(double x)
18 {
19     return x*x;
20 }
21
22 struct point
23 {
24     double x,y;
25     point(){ }
26     point(double _x,double _y):
27         x(_x),y(_y){ };
28     void input()
29     {
30         scanf("%lf%lf",&x,&y);
31     }
32     void output()
33     {
34         printf("%.2f_%.2f\n",x,y);
35     }
36     bool operator==(point a)const
37     {
38         return dblcmp(a.x-x)==0&&dblcmp(a.y-y)==0;
39     }
40     bool operator<(point a)const
41     {
42         return dblcmp(a.x-x)==0&&dblcmp(y-a.y)<0&&a.x<x;
43     }
44     double len()
45     {
46         return hypot(x,y);
47     }
48     double len2()
49     {
50         return x*x+y*y;
51     }
52     double distance(point p)
53     {
54         return hypot(x-p.x,y-p.y);
55     }
56     point add(point p)
57     {
58         return point(x+p.x,y+p.y);
59     }
60     point sub(point p)
61     {
62         return point(x-p.x,y-p.y);
63     }
64     point mul(double b)
65     {
66         return point(x*b,y*b);
67     }
68     point div(double b)
69     {
70         return point(x/b,y/b);
71     }
72     double dot(point p)
73     {
74         return x*p.x+y*p.y;
75     }
76     double det(point p)
77     {
78         return x*p.y-y*p.x;
79     }
80     double rad(point a,point b)
81     {
82         point p=*this;
83         return fabs(atan2(fabs(a.sub(p).det(b.sub(p))),a.sub(p).dot(b.sub(p))));
84     }
85     point trunc(double r)
86     {
87         double l=len();
88         if (!dblcmp(l))return *this;
89         r/=l;
90         return point(x*r,y*r);
91     }
92     point rotleft()
93     {
94         return point(-y,x);
95     }
96     point rotright()
97     {
98         return point(y,-x);
99     }
100     point rotate(point p,double angle)//绕点逆时针旋转角度pangle
101     {
102         point v=this->sub(p);
103         double c=cos(angle),s=sin(angle);
104         return point(p.x+v.x*c-v.y*s,p.y+v.x*s+v.y*c);
105     }
106 };

```

## 4.8 point3d

```

1 struct point3
2 {
3     double x,y,z;
4     point3(){ }
5     point3(double _x,double _y,double _z):
6         x(_x),y(_y),z(_z){ };
7     void input()
8     {
9         scanf("%lf%lf%lf",&x,&y,&z);
10    }
11    void output()
12    {
13        printf("%.2lf_%.2lf_%.2lf\n",x,y,z);
14    }
15    bool operator==(point3 a)

```

```

16    {
17        return dblcmp(a.x-x)==0&&dblcmp(a.y-y)==0&&dblcmp(a.z-z)==0;
18    }
19    bool operator<(point3 a)const
20    {
21        return dblcmp(a.x-x)==0&&dblcmp(y-a.y)==0&&dblcmp(z-a.z)<0&&a.y<y&&a.x<x;
22    }
23    double len()
24    {
25        return sqrt(len2());
26    }
27    double len2()
28    {
29        return x*x+y*y+z*z;
30    }
31    double distance(point3 p)
32    {
33        return sqrt((p.x-x)*(p.x-x)+(p.y-y)*(p.y-y)+(p.z-z)*(p.z-z));
34    }
35    point3 add(point3 p)
36    {
37        return point3(x+p.x,y+p.y,z+p.z);
38    }
39    point3 sub(point3 p)
40    {
41        return point3(x-p.x,y-p.y,z-p.z);
42    }
43    point3 mul(double d)
44    {
45        return point3(x*d,y*d,z*d);
46    }
47    point3 div(double d)
48    {
49        return point3(x/d,y/d,z/d);
50    }
51    double dot(point3 p)
52    {
53        return x*p.x+y*p.y+z*p.z;
54    }
55    point3 det(point3 p)
56    {
57        return point3(y*p.z-p.y*z,p.x*z-x*p.z,x*p.y-p.x*y);
58    }
59    double rad(point3 a,point3 b)
60    {
61        point3 p=*this;
62        return acos(a.sub(p).dot(b.sub(p))/(a.distance(p)*b.distance(p)));
63    }
64    point3 trunc(double r)
65    {
66        r/=len();
67        return point3(x*r,y*r,z*r);
68    }
69    point3 rotate(point3 o,double r) // building?
70    {
71    }
72 };

```

## 4.9 polygon

```

1 struct polygon
2 {
3     int n;
4     point p[maxp];
5     line l[maxp];
6     void input()
7     {
8         n=4;
9         p[0].input();
10        p[2].input();
11        double dis=p[0].distance(p[2]);
12        p[1]=p[2].rotate(p[0],pi/4);
13        p[1]=p[0].add((p[1].sub(p[0])).trunc(dis/sqrt(2.0)));
14        p[3]=p[2].rotate(p[0],2*pi-pi/4);
15        p[3]=p[0].add((p[3].sub(p[0])).trunc(dis/sqrt(2.0)));
16    }
17    void add(point q)
18    {
19        p[n++]=q;
20    }
21    void getline()
22    {
23        for (int i=0;i<n;i++)
24        {
25            l[i]=line(p[i],p[(i+1)%n]);
26        }
27    }
28    struct cmp
29    {
30        point p;
31        cmp(const point &p0){p=p0;}
32        bool operator()(const point &aa,const point &bb)
33        {
34            point a=aa,b=bb;
35            int d=dblcmp(a.sub(p).det(b.sub(p)));
36            if (d==0)
37            {
38                return dblcmp(a.distance(p)-b.distance(p))<0;
39            }
40            return d>0;
41        }
42    };
43    void norm()
44    {
45        point mi=p[0];
46        for (int i=1;i<n;i++)mi=min(mi,p[i]);
47        sort(p,p+n,cmp(mi));
48    }
49    void getconvex(polygon &convex)
50    {
51        int i,j,k;
52        sort(p,p+n);
53        convex.n=n;
54        for (i=0;i<min(n,2);i++)
55        {
56            convex.p[i]=p[i];
57        }
58        if (n<=2)return;
59        int &top=convex.n;
60        top=1;
61        for (i=2;i<n;i++)
62        {
63            while (top&&convex.p[top].sub(p[i]).det(convex.p[top-1].sub(p[i]))<=0)
64                top--;
65            convex.p[++top]=p[i];
66        }
67        int temp=top;

```



```

68     convex.p[1+top]=p[n-2];
69     for (i=n-3;i>=0;i--)
70     {
71         while (top!=n&&convex.p[top].sub(p[i]).det(convex.p[top-1].sub(p[i]))<=0)
72             top--;
73         convex.p[1+top]=p[i];
74     }
75 }
76 bool isconvex()
77 {
78     bool s[3];
79     memset(s,0,sizeof(s));
80     int i,j,k;
81     for (i=0;i<n;i++)
82     {
83         j=(i+1)%n;
84         k=(j+1)%n;
85         s[dblcmp(p[j].sub(p[i]).det(p[k].sub(p[i])))+1]=1;
86         if (s[0]&&s[2])return 0;
87     }
88     return 1;
89 }
90 //3 点上
91 //2 边上
92 //1 内部
93 //0 外部
94 int relationpoint(point q)
95 {
96     int i,j;
97     for (i=0;i<n;i++)
98     {
99         if (p[i]==q)return 3;
100     }
101     getline();
102     for (i=0;i<n;i++)
103     {
104         if (l[i].pointonseg(q))return 2;
105     }
106     int cnt=0;
107     for (i=0;i<n;i++)
108     {
109         j=(i+1)%n;
110         int k=dblcmp(q.sub(p[j]).det(p[i].sub(p[j])));
111         int u=dblcmp(p[i].y-q.y);
112         int v=dblcmp(p[j].y-q.y);
113         if (l>0&&u<0&&v>=0)cnt++;
114         if (k<0&&u<0&&v>=0)cnt--;
115     }
116     return cnt!=0;
117 }
118 //1 在多边形内长度为正
119 //2 相交或与边平行
120 //0 无任何交点
121 int relationline(line u)
122 {
123     int i,j,k=0;
124     getline();
125     for (i=0;i<n;i++)
126     {
127         if (l[i].segcrossseg(u)==2)return 1;
128         if (l[i].segcrossseg(u)==1)k=1;
129     }
130     if (!k)return 0;
131     vector<point>vp;
132     for (i=0;i<n;i++)
133     {
134         if (l[i].segcrossseg(u))
135         {
136             if (l[i].parallel(u))
137             {
138                 vp.pb(u.a);
139                 vp.pb(u.b);
140                 vp.pb(l[i].a);
141                 vp.pb(l[i].b);
142                 continue;
143             }
144             vp.pb(l[i].crosspoint(u));
145         }
146     }
147     sort(vp.begin(),vp.end());
148     int sz=vp.size();
149     for (i=0;i<sz-1;i++)
150     {
151         point mid=vp[i].add(vp[i+1]).div(2);
152         if (relationpoint(mid)==1)return 1;
153     }
154     return 2;
155 }
156 //直线切割凸多边形左侧u
157 //注意直线方向
158 void convexcut(line u,polygon &po)
159 {
160     int i,j,k;
161     int &top=po.n;
162     top=0;
163     for (i=0;i<n;i++)
164     {
165         int d1=dblcmp(p[i].sub(u.a).det(u.b.sub(u.a)));
166         int d2=dblcmp(p[(i+1)%n].sub(u.a).det(u.b.sub(u.a)));
167         if (d1>=0)po.p[top++]=p[i];
168         if (d1*d2<0)po.p[top++]=u.crosspoint(line(p[i],p[(i+1)%n]));
169     }
170 }
171 double getcircumference()
172 {
173     double sum=0;
174     int i;
175     for (i=0;i<n;i++)
176     {
177         sum+=p[i].distance(p[(i+1)%n]);
178     }
179     return sum;
180 }
181 double getarea()
182 {
183     double sum=0;
184     int i;
185     for (i=0;i<n;i++)
186     {
187         sum+=p[i].det(p[(i+1)%n]);
188     }
189     return fabs(sum)/2;
190 }
191 bool getdir()//代表逆时针1 代表顺时针0
192 {
193     double sum=0;
194     int i;
195     for (i=0;i<n;i++)
196     {
197         sum+=p[i].det(p[(i+1)%n]);
198     }
199     if (sum>0)return 1;
200     return 0;
201 }
202 point getbarycentre() // centroid
203 {
204     point ret(0,0);
205     double area=0;
206     int i;
207     for (i=1;i<n-1;i++)
208     {
209         double tmp=p[p[i].sub(p[0]).det(p[i+1].sub(p[0]))];
210         if (dblcmp(tmp)==0)continue;
211         area+=tmp;
212         ret.x+=p[0].x+p[i].x+p[i+1].x)/3*tmp;
213         ret.y+=p[0].y+p[i].y+p[i+1].y)/3*tmp;
214     }
215     if (dblcmp(area))ret=ret.div(area);
216     return ret;
217 }
218 double areaintersection(polygon po) // refer: HPI
219 {
220 }
221 double areaunion(polygon po)
222 {
223     return getarea()+po.getarea()-areaintersection(po);
224 }
225 double areacircle(circle c)
226 {
227     int i,j,k,l,m;
228     double ans=0;
229     for (i=0;i<n;i++)
230     {
231         int j=(i+1)%n;
232         if (dblcmp(p[j].sub(c.p).det(p[i].sub(c.p)))>=0)
233         {
234             ans+=c.areastriangle(p[i],p[j]);
235         }
236         else
237         {
238             ans-=c.areastriangle(p[i],p[j]);
239         }
240     }
241     return fabs(ans);
242 }
243 //多边形和圆关系
244 //0 一部分在圆外
245 //1 与圆某条边相切
246 //2 完全在圆内
247 int relationcircle(circle c)
248 {
249     getline();
250     int i,x=2;
251     if (relationpoint(c.p)!=1)return 0;
252     for (i=0;i<n;i++)
253     {
254         if (c.relationseg(l[i])==2)return 0;
255         if (c.relationseg(l[i])==1)x=1;
256     }
257     return x;
258 }
259 void find(int st,point tri[],circle &c)
260 {
261     if (!st)
262     {
263         c=circle(point(0,0),-2);
264     }
265     if (st==1)
266     {
267         c=circle(tri[0],0);
268     }
269     if (st==2)
270     {
271         c=circle(tri[0].add(tri[1]).div(2),tri[0].distance(tri[1])/2.0);
272     }
273     if (st==3)
274     {
275         c=circle(tri[0],tri[1],tri[2]);
276     }
277 }
278 void solve(int cur,int st,point tri[],circle &c)
279 {
280     find(st,tri,c);
281     if (st==3)return;
282     int i;
283     for (i=0;i<cur;i++)
284     {
285         if (dblcmp(p[i].distance(c.p)-c.r)>0)
286         {
287             tri[st]=p[i];
288             solve(i,st+1,tri,c);
289         }
290     }
291 }
292 circle mincircle()//点集最小圆覆盖
293 {
294     random_shuffle(p,p+n);
295     point tri[4];
296     circle c;
297     solve(n,0,tri,c);
298     return c;
299 }
300 int circlecover(double r)//单位圆覆盖
301 {
302     int ans=0,i,j;
303     vector<pair<double,int>>v;
304     for (i=0;i<n;i++)
305     {
306         v.clear();
307         for (j=0;j<n;j++)if (i!=j)
308         {
309             point q=p[i].sub(p[j]);
310             double d=q.len();
311             if (dblcmp(d-2*r)<=0)
312             {
313                 double arg=atan2(q.y,q.x);
314                 if (dblcmp(arg)<0)arg+=2*pi;
315                 double t=acos(d/(2*r));
316                 v.push_back(make_pair(arg-t+2*pi,-1));
317                 v.push_back(make_pair(arg+t+2*pi,1));
318             }
319         }
320     }
321     sort(v.begin(),v.end());
322     int cur=0;
323     for (j=0;j<v.size();j++)
324     {
325         if (v[j].second==1)cur++;
326         else --cur;
327     }
328 }

```

```

326         ans=max(ans,cur);
327     }
328     return ans+1;
329 }
330 int pointinpolygon(point q)//点在凸多边形内部的判定
331 {
332     if (getdir())reverse(p,p+n);
333     if (dblcmp(q.sub(p[0]).det(p[n-1].sub(p[0]))==0)
334     {
335         if (line(p[n-1],p[0]).pointonseg(q))return n-1;
336         return -1;
337     }
338     int low=1,high=n-2,mid;
339     while (low<=high)
340     {
341         mid=(low+high)>>1;
342         if (dblcmp(q.sub(p[0]).det(p[mid].sub(p[0]))>=0&&dblcmp(q.sub(p[0]).det(p[mid+1].sub(p[0]))<0)
343         {
344             polygon c;
345             c.p[0]=p[mid];
346             c.p[1]=p[mid+1];
347             c.p[2]=p[0];
348             c.n=3;
349             if (c.relationpoint(q))return mid;
350             return -1;
351         }
352         if (dblcmp(q.sub(p[0]).det(p[mid].sub(p[0]))>0)
353         {
354             low=mid+1;
355         }
356         else
357         {
358             high=mid-1;
359         }
360     }
361     return -1;
362 }
363 }
364 };

```

## 4.10 polygons

```

1 struct polygons
2 {
3     vector<polygon>p;
4     polygons()
5     {
6         p.clear();
7     }
8     void clear()
9     {
10        p.clear();
11    }
12    void push(polygon q)
13    {
14        if (dblcmp(q.getarea()))p.pb(q);
15    }
16    vector<pair<double,int>>e;
17    void ins(point s,point t,point X,int i)
18    {
19        double r=fabs(t.x-s.x)>eps?(X.x-s.x)/(t.x-s.x):(X.y-s.y)/(t.y-s.y);
20        r=min(r,1.0);r=max(r,0.0);
21        e.pb(mp(r,i));
22    }
23    double polyareaunion()
24    {
25        double ans=0.0;
26        int c0,c1,c2,i,j,k,w;
27        for (i=0;i<p.size();i++)
28        {
29            if (p[i].getdir()==0)reverse(p[i].p,p[i].p+p[i].n);
30        }
31        for (i=0;i<p.size();i++)
32        {
33            for (k=0;k<p[i].n;k++)
34            {
35                point &s=p[i].p[k],&t=p[i].p[(k+1)%p[i].n];
36                if (!dblcmp(s.det(t)))continue;
37                e.clear();
38                e.pb(mp(0.0,1));
39                e.pb(mp(1.0,-1));
40                for (j=0;j<p.size();j++)if (i!=j)
41                {
42                    for (w=0;w<p[j].n;w++)
43                    {
44                        point a=p[j].p[w],b=p[j].p[(w+1)%p[j].n],c=p[i].p[(w+1+p[j].n)%p[j].n];
45                        c0=dblcmp(t.sub(s).det(c.sub(s)));
46                        c1=dblcmp(t.sub(s).det(a.sub(s)));
47                        c2=dblcmp(t.sub(s).det(b.sub(s)));
48                        if (c1*c2<0)ins(s,t,line(s,t).crosspoint(line(a,b)),-c2);
49                        else if (!c1&&c0*c2<0)ins(s,t,a,-c2);
50                        else if (!c1&&c2)
51                        {
52                            int c3=dblcmp(t.sub(s).det(p[j].p[(w+2)%p[j].n].sub(s)));
53                            int dp=dblcmp(t.sub(s).det(b.sub(a)));
54                            if (dp&&c0)ins(s,t,a,dp>0?c0*((j>i)^(c0<0)):(-c0<0));
55                            if (dp&&c3)ins(s,t,b,dp>0?c3*((j>i)^(c3<0)):c3<0);
56                        }
57                    }
58                }
59                sort(e.begin(),e.end());
60                int ct=0;
61                double tot=0.0,last;
62                for (j=0;j<e.size();j++)
63                {
64                    if (ct==p.size())tot+=e[j].first-last;
65                    ct+=e[j].second;
66                    last=e[j].first;
67                }
68                ans+=s.det(t)*tot;
69            }
70        }
71        return fabs(ans)*0.5;
72    }
73 };

```

# 5 graph

## 5.1 Articulation

```

1 void dfs(int now,int fa) // 从开始now

```

```

2 {
3     int p(0);
4     dfn[now]=low[now]=cnt++;
5     for(std::list<int>::const_iterator it(edge[now].begin());it!=edge[now].end();++it)
6     {
7         if (dfn[*it]==-1)
8         {
9             dfs(*it,now);
10            ++p;
11            low[now]=std::min(low[now],low[*it]);
12            if ((now==1 && p>1) || (now!=1 && low[*it]>=dfn[now])) // 如果从出发点出发的子节点不
13                能由兄弟节点到达, 那么出发点为割点。如果现节点不是出发点, 但是其子孙节点不能达到祖先节
14                点, 那么该节点为割点
15                ans.insert(now);
16            }
17        else
18        {
19            if (*it!=fa)
20                low[now]=std::min(low[now],dfn[*it]);
21        }
22    }
23 }

```

## 5.2 Biconnected Component - Edge

```

1 // hdu 4612
2 #include<cstdio>
3 #include<algorithm>
4 #include<set>
5 #include<string>
6 #include<stack>
7 #include<queue>
8
9 #define MAXX 200111
10 #define MAXE (1000111*2)
11 #pragma comment(linker, "/STACK:16777216")
12
13 int edge[MAXX],to[MAXE],nxt[MAXE],cnt;
14 #define v to[i]
15 inline void add(int a,int b)
16 {
17     nxt[++cnt]=edge[a];
18     edge[a]=cnt;
19     to[cnt]=b;
20 }
21
22 int dfn[MAXX],low[MAXX],col[MAXX],belong[MAXX];
23 int idx,bcnt;
24 std::stack<int>st;
25
26 void tarjan(int now,int last)
27 {
28     col[now]=1;
29     st.push(now);
30     dfn[now]=low[now]=++idx;
31     bool flag(false);
32     for(int i(edge[now]);i;i=nxt[i])
33     {
34         if(v==last && !flag)
35         {
36             flag=true;
37             continue;
38         }
39         if(!col[v])
40         {
41             tarjan(v,now);
42             low[now]=std::min(low[now],low[v]);
43             /*
44             if (low[v]>dfn[now])
45             then this is a bridge
46             */
47         }
48         else
49         {
50             if (col[v]==1)
51                 low[now]=std::min(low[now],dfn[v]);
52         }
53     }
54     col[now]=2;
55     if (dfn[now]==low[now])
56     {
57         ++bcnt;
58         static int x;
59         do
60         {
61             x=st.top();
62             st.pop();
63             belong[x]=bcnt;
64             }while(x!=now);
65     }
66 }
67
68 std::set<int>set[MAXX];
69
70 int dist[MAXX];
71 std::queue<int>q;
72 int n,m,i,j,k;
73
74 inline int go(int s)
75 {
76     static std::set<int>::const_iterator it;
77     memset(dist,0,sizeof dist);
78     dist[s]=0;
79     q.push(s);
80     while(!q.empty())
81     {
82         s=q.front();
83         q.pop();
84         for(it=set[s].begin();it!=set[s].end();++it)
85             if (dist[*it]>dist[s]+1)
86             {
87                 dist[*it]=dist[s]+1;
88                 q.push(*it);
89             }
90     }
91     return std::max_element(dist+1,dist+1+bcnt)-dist;
92 }
93
94 int main()
95 {
96     while(scanf("%d%d",&n,&m),(n||m))
97     {
98         cnt=0;
99         memset(edge,0,sizeof edge);
100         while(m--)
101         {
102             scanf("%d%d",&i,&j);
103             add(i,j);
104             add(j,i);
105         }
106         memset(dfn,0,sizeof dfn);
107         memset(belong,0,sizeof belong);
108     }
109 }

```

```

107     memset(low,0,sizeof low);
108     memset(col,0,sizeof col);
109     bcnt=dx=0;
110     while(!st.empty())
111         st.pop();
112
113     tarjan(1,-1);
114     for(i=1;i<=bcnt;+=i)
115         set[i].clear();
116     for(i=1;i<=x;+=i)
117         for(j=edge[i];j;j=next[j])
118             set[belong[i]].insert(belong[to[j]]);
119     for(i=1;i<=bcnt;+=i)
120         set[i].erase(i);
121     /*
122     printf("%d\n",dist[go(go(1))]);
123     for(i=1;i<=bcnt;+=i)
124         printf("%d\n",dist[i]);
125     puts("");
126     */
127     printf("%d\n",bcnt-1-dist[go(go(1))]);
128 }
129 return 0;
130 }

```

## 5.3 Blossom algorithm

```

1  #include<cstdio>
2  #include<vector>
3  #include<cstring>
4  #include<algorithm>
5
6  #define MAXX 233
7
8  bool map[MAXX][MAXX];
9  std::vector<int> p[MAXX];
10 int m[MAXX];
11 int vis[MAXX];
12 int q[MAXX],*qf,*qb;
13
14 int n;
15
16 inline void label(int x,int y,int b)
17 {
18     static int i,z;
19     for(i=b+1;i<=p[x].size();+=i)
20         if(vis[z=p[x][i]]==1)
21         {
22             p[z]=p[y];
23             p[z].insert(p[z].end(),p[x].rbegin(),p[x].rend()-i);
24             vis[z]=0;
25             *qb+=z;
26         }
27 }
28
29 inline bool bfs(int now)
30 {
31     static int i,x,y,z,b;
32     for(i=0;i<=x;+=i)
33         p[i].resize(0);
34     p[now].push_back(now);
35     memset(vis,-1,sizeof vis);
36     vis[now]=0;
37     qf=q;
38     *qb+=now;
39
40     while(qf<qb)
41         for(x=*qf++;y=0;y<=x;+=y)
42             if(map[x][y] && m[y]!=y && vis[y]!=1)
43             {
44                 if(vis[y]==-1)
45                     if(m[y]==-1)
46                     {
47                         for(i=0;i<=p[x].size();+=2)
48                         {
49                             m[p[x][i]]=p[x][i+1];
50                             m[p[x][i+1]]=p[x][i];
51                         }
52                         m[x]=y;
53                         m[y]=x;
54                         return true;
55                     }
56                     else
57                     {
58                         p[z=m[y]]=p[x];
59                         p[z].push_back(y);
60                         p[z].push_back(z);
61                         vis[y]=1;
62                         vis[z]=0;
63                         *qb+=z;
64                     }
65                 else
66                 {
67                     for(b=0;b<=p[x].size() && b<=p[y].size() && p[x][b]==p[y][b];+=b);
68                     --b;
69                     label(x,y,b);
70                     label(y,x,b);
71                 }
72             }
73     return false;
74 }
75
76 int i,j,k;
77 int ans;
78
79 int main()
80 {
81     scanf("%d",&n);
82     for(i=0;i<=x;+=i)
83         p[i].reserve(n);
84     while(scanf("%d",&i,&j)!=EOF)
85     {
86         --i;
87         --j;
88         map[i][j]=map[j][i]=true;
89     }
90     memset(m,-1,sizeof m);
91     for(i=0;i<=x;+=i)
92         if(m[i]==-1)
93         {
94             if(bfs(i))
95                 ++ans;
96             else
97                 m[i]=i;
98         }
99     printf("%d\n",ans<1);
100     for(i=0;i<=x;+=i)

```

```

101         if(i<=m[i])
102             printf("%d\n",i+1+m[i]+1);
103         return 0;
104     }

```

## 5.4 chu-liu algorithm

```

1  #include<stdio>
2  #include<string>
3  #include<algorithm>
4
5  const int inf = 0x5fffffff;
6
7  int n,m,u,v,cost,dis[1001][1001],L;
8  int pre[1001],id[1001],visit[1001],in[1001];
9
10 void init(int n)
11 {
12     L = 0;
13     for (int i = 0; i < n; i++)
14         for (int j = 0; j < n; j++)
15             dis[i][j] = inf;
16 }
17
18 struct Edge
19 {
20     int u,v,cost;
21 };
22
23 Edge e[1001*1001];
24
25 int zhuliu(int root,int n,int m,Edge e[])
26 {
27     int res = 0,u,v;
28     while (true)
29     {
30         for (int i = 0; i < n; i++)
31             in[i] = inf;
32         for (int i = 0; i < m; i++)
33             if (e[i].u != e[i].v && e[i].cost < in[e[i].v])
34             {
35                 pre[e[i].v] = e[i].u;
36                 in[e[i].v] = e[i].cost;
37             }
38         for (int i = 0; i < n; i++)
39             if (i != root)
40                 if (in[i] == inf)
41                     return -1;
42
43         int tn = 0;
44         memset(id,-1,sizeof(id));
45         memset(visit,-1,sizeof(visit));
46         in[root] = 0;
47         for (int i = 0; i < n; i++)
48         {
49             res += in[i];
50             v = i;
51             while (visit[v] != i && id[v] == -1 && v != root)
52             {
53                 visit[v] = i;
54                 v = pre[v];
55             }
56             if(v != root && id[v] == -1)
57             {
58                 for(int u = pre[v] ; u != v ; u = pre[u])
59                     id[u] = tn;
60                 id[v] = tn++;
61             }
62         }
63         if(tn == 0) break;
64         for (int i = 0; i < n; i++)
65             if (id[i] == -1)
66                 id[i] = tn++;
67         for (int i = 0; i < m; i++)
68         {
69             int v = e[i].v;
70             e[i].u = id[e[i].u];
71             e[i].v = id[e[i].v];
72             if (e[i].u != e[i].v)
73                 e[i].cost -= in[v];
74             else
75                 std::swap(e[i],e[-m]);
76         }
77         n = tn;
78         root = id[root];
79     }
80     return res;
81 }
82
83 int main()
84 {
85     while (scanf("%d",&n,&m) != EOF)
86     {
87         init(n);
88         for (int i = 0; i < m; i++)
89         {
90             scanf("%d%d",&u,&v,&cost);
91             if (u == v) continue;
92             dis[u][v] = std::min(dis[u][v],cost);
93         }
94         L = 0;
95         for (int i = 0; i < n; i++)
96             for (int j = 0; j < n; j++)
97                 if (dis[i][j] != inf)
98                 {
99                     e[L].u = i;
100                     e[L].v = j;
101                     e[L].cost = dis[i][j];
102                 }
103         printf("%d\n",zhuliu(0,n,L,e));
104     }
105     return 0;
106 }

```

## 5.5 Covering problems

```

1 | 最大团以及相关知识独立集：独立集是指图的顶点集的一个子集该子集的导出子图的点互不相邻如果一个独立集不是任何
2 | 一个独立集的子集
3 |
4 | ... 那么称这个独立集是一个极大独立集一个图中包含顶点数目最多的独立集称为最大独立集，最大独立集一定是极大独
5 | 立集，但是极大独立集不一定是最大的独立集。... 支配集：与独立集相对应的就是支配集，支配集也是图顶点集的一个子集，设是图的一个支配集，则对于图中的任意一个顶点，要么属于集合

```

```

1  S G u s , 要 么 与 中 的 顶 点 相 邻 。 在 中 除 去 任 何 元 素 后 不 再 是 支 配 集 ， 则 支 配 集 是 极 小 支 配 集 。 称 所 有 支 配 集 中 顶 点 个 数 最
2  少 的 支 配 集 为 最 小 支 配 集 ， 最 小 支 配 集 中 的 顶 点 个 数 成 为 支 配 数 。 s s s s G 最 大 团 ： 图 的 顶 点 的 子 集 ， 设 是 最 大 团 2
3  则 中 任 意 两 点 相 邻 。 若 ， 是 最 大 团 ， 则
4
5  G D D m u 有 边 相 连 ， 其 补 图 v u , 没 有 边 相 连 ， 所 以 图 的 最 大 团 v G 其 补 图 的 最 大 独 立 集 。 给 定 无 向 图 G = ( V ; E ) , 如 果 属
6  于 ， 并 且 对 于 任 意 U V u 包 含 于 V U v 包 含 于 > , 则 称 是 的 完 全 子 图 ， 的 完 全 子 图 是 的 团 ， 当 且 仅 当 不 包 含
7  在 的 更 大 的 完 全 子 图 中 ， 的 最 大 团 是 指 中 所 含 顶 点 数 目 最 多 的 团 。 如 果 属 于 ， 并 且 对 于 任 意 U G U G U G U G U v ( 称
8  含 于 有 v U < u ; v 不 包 含 于 > , 则 称 是 的 空 子 图 ， 的 空 子 图 是 的 独 立 集 ， 当 且 仅 当 不 包 含 在 的 更 大 的 独 立 集 ， 的 最
9  大 团 是 指 中 所 含 顶 点 数 目 最 多 的 独 立 集 。 E L G U G U G U 性 质 ： 最 大 独 立 集 最 小 覆 盖 集
10
11  += V 最 大 团 补 图 的 最 大 独 立 集
12  = 最 小 覆 盖 集 最 大 匹 配
13  =
14  minimum cover:
15  vertex cover vertex bipartite graph = maximum cardinality bipartite matching 找 完 最 大 二 分 匹 配
16  后 ， 有 三 种 情 况 要 分 别 处 理 ： 甲 、
17
18  X 侧 未 匹 配 点 的 交 错 树 们 。 乙 、
19  Y 侧 未 匹 配 点 的 交 错 树 们 。 丙 、 层 层 叠 叠 的 交 错 树 们 （ 包 含 单 独 的 匹 配 边 ）。 这 三 个 情 况 互 不 干 涉 。 用
20
21  Graph Traversal 建 立 甲 、 乙 的 交 错 树 们 ， 剩 下 部 分 就 是 丙 。 要 找 点 覆 盖 ， 甲 、 乙 是 取 遍 奇 数 距 离 的 点 ， 丙 是 取 遍 偶 数
22  距 离 的 点 、 或 者 是 取 遍 奇 数 距 离 的 点 ， 每 塊 连 通 分 量 可 以 各 自 为 政 。 另 外 ， 小 心 处 理 的 话 ， 是 可 以 印 出 字 典 顺 序 最
23  小 的 点 覆 盖 的 。 已 经 有 最 大 匹 配 时 ， 求 点 覆 盖 的 时 间 复 杂 度 等 同 于 一 次
24
25  Graph Traversal 的 时 间 。
26
27  vertex cover edge
28
29  edge cover vertex 首 先 在 图 上 求 得 一 个
30  Maximum Matching 之 后 ， 对 于 那 些 单 身 的 点 ， 都 由 匹 配 点 连 去 。 如 此 便 形 成 了 Minimum Edge Cover 。
31
32  edge cover edge
33
34  path cover vertex
35  general graph: NP-H
36  tree: DP
37  DAG: 将 每 个 节 点 拆 分 为 入 点 和 出 点 , ans 节 点 数 匹 配 数 -=
38
39  path cover edge
40  minimize the count of euler path ( greedy is ok? )
41
42  cycle cover vertex
43  general: NP-H
44  weighted: do like path cover vertex, with KM algorithm
45
46  cycle cover edge
47  NP-H
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
71
```

```

58     {
59         while( i==find(a))
60         {
61             next[i]=a;
62             done[a]=true;
63             ++cnt;
64         }
65         while( i==find(b))
66         {
67             next[b]=i;
68             done[b]=true;
69             ++cnt;
70         }
71         if( !mat[a][b] )
72             for( i=next[a]; next[i]!=b; i=next[i] )
73                 if( mat[a][next[i]] && mat[i][b] )
74                 {
75                     for( j=next[i]; j!=b; j=next[j] )
76                         pre[next[j]]=j;
77                     for( j=b; j!=next[i]; j=pre[j] )
78                         next[j]=pre[j];
79                     std::swap(next[i],b);
80                     break;
81                 }
82             next[b]=a;
83             for( i=a; i!=b; i=next[i] )
84                 if( find(i) )
85                 {
86                     a=next[b=i];
87                     break;
88                 }
89         }
90         while( a!=b )
91         {
92             printf("%d_",a);
93             a=next[a];
94         }
95         printf("%d\n",b);
96     }
97     return 0;
98 }

```

## 5.9 Hopcroft-Karp algorithm

```

1  #include<cstdio>
2  #include<cstring>
3
4  #define MAXX 50111
5  #define MAX 150111
6
7  int nx,p;
8  int i,j,k;
9  int x,y;
10 int ans;
11 bool flag;
12
13 int edge[MAXX],nxt[MAX],to[MAX],cnt;
14
15 int cx[MAXX],cy[MAXX];
16 int px[MAXX],py[MAXX];
17
18 int q[MAXX],*qf,*qb;
19
20 bool ag(int i)
21 {
22     int j,k;
23     for( k=edge[i]; k; k=nxt[k] )
24         if( py[j=to[k]]==px[i]+1 )
25         {
26             py[j]=0;
27             if( cy[j]==-1 || ag(cy[j]) )
28             {
29                 cx[i]=j;
30                 cy[j]=i;
31                 return true;
32             }
33         }
34     return false;
35 }
36
37 int main()
38 {
39     scanf("%d%d%d",&nx,&p);
40     while( p-- )
41     {
42         scanf("%d%d",&i,&j);
43         nxt[++cnt]=edge[i];
44         edge[i]=cnt;
45         to[cnt]=j;
46     }
47     memset(cx,-1,sizeof cx);
48     memset(cy,-1,sizeof cy);
49     while( true )
50     {
51         memset(px,0,sizeof(px));
52         memset(py,0,sizeof(py));
53         qf=qb=q;
54         flag=false;
55
56         for( i=1; i<=nx; ++i )
57             if( cx[i]==-1 )
58                 *qb++=i;
59         while( qf==qb )
60             for( k=edge[i=*qf++]; k; k=nxt[k] )
61                 if( !py[j=to[k]] )
62                 {
63                     py[j]=px[i]+1;
64                     if( cy[j]==-1 )
65                         flag=true;
66                     else
67                     {
68                         px[cy[j]]=py[j]+1;
69                         *qb++=cy[j];
70                     }
71                 }
72         if( !flag )
73             break;
74         for( i=1; i<=nx; ++i )
75             if( cx[i]==-1 && ag(i) )
76                 ++ans;
77     }
78     printf("%d\n",ans);
79     return 0;
80 }

```

## 5.10 Improved Shortest Augmenting Path Algorithm

```

1  int source,sink;
2
3  inline long long go(const int N=sink)
4  {
5      static int now,N,i;
6      static long long min,mf;
7      memset(gap,0,sizeof gap);
8      memset(h,0,sizeof h);
9      memcpy(w,edge,sizeof w);
10     gap[0]=N;
11     mf=0;
12
13     pre[now=source]=-1;
14     while( h[source]<N )
15     {
16         rep:
17             if( now==sink )
18             {
19                 min=inf;
20                 for( i=pre[sink]; i!=-1; i=pre[to[i^1]] )
21                     if( min>cap[i] )
22                     {
23                         min=cap[i];
24                         now=to[i^1];
25                     }
26                 for( i=pre[sink]; i!=-1; i=pre[to[i^1]] )
27                 {
28                     cap[i]-=min;
29                     cap[i^1]+=min;
30                 }
31                 mf+=min;
32             }
33             for( int &ci(w[now]); i!=-1; i=nxt[i] )
34                 if( cap[i] && h[v^1]==h[now] )
35                 {
36                     pre[now=v]=i;
37                     goto rep;
38                 }
39             if( !--gap[h[now]] )
40                 return mf;
41             min=N;
42             for( i=w[now]=edge[now]; i!=-1; i=nxt[i] )
43                 if( cap[i] )
44                     min=std::min(min,(long long)h[v]);
45             ++gap[h[now]=min+1];
46             if( now==source )
47                 now=to[pre[now]^1];
48         }
49     }
50     return mf;
51 }

```

## 5.11 k Shortest Path

```

1  #include<cstdio>
2  #include<cstring>
3  #include<queue>
4  #include<vector>
5
6  int K;
7
8  class states
9  {
10 public:
11     int cost,id;
12 };
13
14 int dist[1000];
15
16 class cmp
17 {
18 public:
19     bool operator()(const states &i,const states &j)
20     {
21         return i.cost>j.cost;
22     }
23 };
24
25 class cmp2
26 {
27 public:
28     bool operator()(const states &i,const states &j)
29     {
30         return i.cost+dist[i.id]>j.cost+dist[j.id];
31     }
32 };
33
34 struct edges
35 {
36     int to,next,cost;
37 } edger[100000],edge[100000];
38
39 int headr[1000],head[1000],Lr,L;
40
41 void dijkstra(int s)
42 {
43     states u;
44     u.id=s;
45     u.cost=0;
46     dist[s]=0;
47     std::priority_queue<states,std::vector<states>,cmp> q;
48     q.push(u);
49     while( !q.empty() )
50     {
51         u=q.top();
52         q.pop();
53         if( u.cost!=dist[u.id] )
54             continue;
55         for( int i=headr[u.id]; i!=-1; i=edger[i].next )
56         {
57             states v=u;
58             v.id=edger[i].to;
59             if( dist[v.id]>dist[u.id]+edger[i].cost )
60             {
61                 v.cost=dist[v.id]=dist[u.id]+edger[i].cost;
62                 q.push(v);
63             }
64         }
65     }
66 }
67
68 int num[1000];
69

```

```

70 inline void init(int n)
71 {
72     L=0;
73     memset(head,-1,4*n);
74     memset(headr,-1,4*n);
75     memset(dist,63,4*n);
76     memset(num,0,4*n);
77 }
78
79 void add_edge(int u,int v,int x)
80 {
81     edge[L].to=v;
82     edge[L].cost=x;
83     edge[L].next=head[u];
84     head[u]=L++;
85     edge[Lr].to=u;
86     edge[Lr].cost=x;
87     edge[Lr].next=headr[v];
88     headr[v]=Lr++;
89 }
90
91 inline int a_star(int s,int t)
92 {
93     if (dist[s]==0x3f3f3f3f)
94         return -1;
95     std::priority_queue<states>,std::vector<states>,>cmp2 q;
96     states tmp;
97     tmp.id=s;
98     tmp.cost=0;
99     q.push(tmp);
100     while (!q.empty())
101     {
102         states u=q.top();
103         q.pop();
104         num[u.id]++;
105         if (num[u]==K)
106             return u.cost;
107         for (int i=head[u.id]; i!=-1; i=edge[i].next)
108         {
109             int v=edge[i].to;
110             tmp.id=v;
111             tmp.cost=u.cost+edge[i].cost;
112             q.push(tmp);
113         }
114     }
115     return -1;
116 }
117
118 int main()
119 {
120     int n,m;
121     scanf("%d",&n,&m);
122     init(n);
123     for (int i=0; i<m; i++)
124     {
125         int u,v,x;
126         scanf("%d%d",&u,&v,&x);
127         add_edge(u-1,v-1,x);
128     }
129     int s,t;
130     scanf("%d%d",&s,&t,&K);
131     if (s==t)
132         +K;
133     dijkstra(t-1);
134     printf("%d\n",a_star(s-1,t-1));
135 }

```

## 5.12 Kariv-Hakimi Algorithm

```

1 //Absolute Center of a graph, not only a tree
2 #include<stdio>
3 #include<algorithm>
4 #include<vector>
5 #include<cstring>
6 #include<set>
7
8 #define MAXX 211
9 #define inf 0x3f3f3f3f
10
11 int e[MAXX][MAXX],dist[MAXX][MAXX];
12 double dp[MAXX][ta;
13 int ans,d;
14 int n,m,a,b;
15 int i,j,k;
16 typedef std::pair<int,int> pii;
17 std::vector<pii>vt[2];
18 bool done[MAXX];
19 typedef std::pair<double,int> pdi;
20 std::multiset<pdi>q;
21 int pre[MAXX];
22
23 int main()
24 {
25     vt[0].reserve(MAXX);
26     vt[1].reserve(MAXX);
27     scanf("%d",&n,&m);
28     memset(e,0x3f,sizeof(e));
29     while(m--)
30     {
31         scanf("%d%d",&i,&j,&k);
32         e[i][j]=e[j][i]=std::min(e[i][j],k);
33     }
34     for(i=1;i<=n;i++)
35         e[i][i]=0;
36     memcpy(dist,e,sizeof(dist));
37     for(k=1;k<=n;k++)
38         for(i=1;i<=n;i++)
39             for(j=1;j<=n;j++)
40                 dist[i][j]=std::min(dist[i][j],dist[i][k]+dist[k][j]);
41     ans=inf;
42     for(i=1;i<=n;i++)
43         for(j=1;j<=n;j++)
44             if(e[i][j]!=inf)
45             {
46                 vt[0].resize(0);
47                 vt[1].resize(0);
48                 static int i;
49                 for(i=1;i<=n;i++)
50                     vt[0].push_back(pii(dist[i][i],dist[j][i]));
51                 std::sort(vt[0].begin(),vt[0].end());
52                 for(i=0;i<vt[0].size();i++)
53                 {
54                     while(!vt[1].empty() && vt[1].back().second<=vt[0][i].second)
55                         vt[1].pop_back();
56                     vt[1].push_back(vt[0][i]);
57                 }
58                 d=inf;

```

```

59         if(vt[1].size()==1)
60             if(vt[1][0].first<vt[1][0].second)
61             {
62                 ta=0;
63                 d=vt[1][0].first<<1;
64             }
65             else
66             {
67                 ta=e[i][j];
68                 d=vt[1][0].second<<1;
69             }
70         else
71             for(i=1;i<vt[1].size();i++)
72                 if(d>e[i][j]+vt[1][i-1].first+vt[1][i].second)
73                 {
74                     ta=e[i][j]+vt[1][i].second-vt[1][i-1].first)/(double)2.0f;
75                     d=e[i][j]+vt[1][i-1].first+vt[1][i].second;
76                 }
77             if(d<ans)
78             {
79                 ans=d;
80                 a=i;
81                 b=j;
82                 dp[i]=ta;
83                 dp[j]=e[i][j]-ta;
84             }
85         }
86     printf("%d\n",ans);
87     for(i=1;i<=n;i++)
88         if(i!=a && i!=b)
89             dp[i]=1e20;
90     q.insert(pdi(dp[a],a));
91     if(a!=b)
92         q.insert(pdi(dp[b],b));
93     if(a!=b)
94         pre[b]=a;
95     while(!q.empty())
96     {
97         k=q.begin()->second;
98         q.erase(q.begin());
99         if(done[k])
100             continue;
101         done[k]=true;
102         for(i=1;i<=n;i++)
103             if(e[k][i]!=inf && dp[k]+e[k][i]<dp[i])
104             {
105                 dp[i]=dp[k]+e[k][i];
106                 q.insert(pdi(dp[i],i));
107                 pre[i]=k;
108             }
109     }
110     vt[0].resize(0);
111     for(i=1;i<=n;i++)
112         if(pre[i])
113             if(i<pre[i])
114                 printf("%d%d\n",i,pre[i]);
115             else
116                 printf("%d%d\n",pre[i],i);
117     return 0;
118 }

```

## 5.13 Kuhn-Munkres algorithm

```

1 bool match(int u)
2 {
3     vx[u]=true;
4     for(int i=1;i<=n;i++)
5         if(1x[u]+ly[i]==g[u][i]&&ly[i])
6         {
7             vy[i]=true;
8             if(!d[i]||match(d[i]))
9             {
10                 d[i]=u;
11                 return true;
12             }
13         }
14     return false;
15 }
16
17 inline void update()
18 {
19     int i,j;
20     int a=1<<30;
21     for(i=1;i<=n;i++)if(vx[i])
22         for(j=1;j<=n;j++)if(!vy[j])
23             a=min(a,1x[i]+ly[j]-g[i][j]);
24     for(i=1;i<=n;i++)
25     {
26         if(vx[i])1x[i]-=a;
27         if(vy[i])ly[i]+=a;
28     }
29 }
30
31 void km()
32 {
33     int i,j;
34     for(i=1;i<=n;i++)
35     {
36         1x[i]=ly[i]=d[i]=0;
37         for(j=1;j<=n;j++)
38             1x[i]=max(1x[i],g[i][j]);
39     }
40     for(i=1;i<=n;i++)
41     {
42         while(true)
43         {
44             memset(vx,0,sizeof(vx));
45             memset(vy,0,sizeof(vy));
46             if(match(i))
47                 break;
48             update();
49         }
50     }
51     int ans=0;
52     for(i=1;i<=n;i++)
53         if(d[i]!=0)
54             ans+=g[d[i]][i];
55     printf("%d\n",ans);
56 }
57
58 int main()
59 {
60     while(scanf("%d",&n)!=EOF)
61     {
62         for(int i=1;i<=n;i++)gets(s[i]);
63         memset(g,0,sizeof(g));
64         for(int i=1;i<=n;i++)

```

```

65         for(int j=1;j<=n++j)
66             if(l!=j)
67                 g[i][j]=cal(s[i],s[j]);
68         km();
69     }
70     return 0;
71 }

```

## 5.14 Manhattan minimum spanning tree

```

1  #include <cstdio>
2  #include <algorithm>
3  #include <cstring>
4  #include <iostream>
5
6  using namespace std;
7
8  const int maxn = 60000;
9
10 struct node {int x, y, k[2];} b[maxn];
11 struct bian {int a, b, c;} g[maxn * 8];
12 struct point {int k[2];} d[maxn * 8];
13 long long s[maxn], ans;
14 int i, n, m, a[maxn], lim, h, mid, bh[maxn * 2], f[maxn], num, e[maxn * 2], next[maxn * 2], first[maxn], tot;
15 int comx(int p, int q) {return b[p].x < b[q].x;}
16 int comy(int p, int q) {return b[p].y < b[q].y;}
17 int comc(const bian &p, const bian &q) {return p.c < q.c;}
18 int dist(int p, int q) {return abs(b[p].x - b[q].x) + abs(b[p].y - b[q].y);}
19 int maxbh(int p, int q, int k) {return b[p].k[k] > b[q].k[k] ? p : q;}
20 int minbh(int p, int q, int k) {return b[p].k[k] < b[q].k[k] ? p : q;}
21 int getfa(int x) {if (f[x] != x) f[x] = getfa(f[x]); return f[x];}
22 long long gcd(long long p, long long q) {return (!p || !q) ? p + q : gcd(q, p % q);}
23 void link(int u, int v)
24 {
25     e[++num] = v, next[num] = first[u], first[u] = num;
26     e[++num] = u, next[num] = first[v], first[v] = num;
27 }
28 void add(int x, int k)
29 {
30     int y = h + b[x].k[1]; d[y].k[0] = minbh(d[y].k[0], x, 0);
31     for (y >= 1; y; y >= 1) d[y].k[0] = minbh(d[y << 1].k[0], d[y << 1 ^ 1].k[0], 0);
32     y = h + b[x].k[0];
33     d[y].k[1] = k ? maxbh(x, d[y].k[1], 1) : minbh(d[y].k[1], x, 1);
34     for (y >= 1; y; y >= 1)
35         d[y].k[1] = k ? maxbh(x, d[y << 1].k[1], 1) : minbh(d[y << 1 ^ 1].k[1], x, 1);
36 }
37 int ask(int l, int r, int k, int boss)
38 {
39     for (mid = 0, l += h - 1, r += h + 1; (l ^ r) != 1; l >= 1, r >= 1)
40     {
41         if (!(l & 1)) mid = boss ? maxbh(mid, d[l + 1].k[k], k) : minbh(mid, d[r - 1].k[k], k);
42         if (r & 1) mid = boss ? maxbh(mid, d[r - 1].k[k], k) : minbh(mid, d[r - 1].k[k], k);
43     } return mid;
44 }
45 void manhattan()
46 {
47     sort(bh + 1, bh + m + 1, comx);
48     b[0].k[0] = maxn * 3, b[0].k[1] = -1;
49     for (add(bh[m], 1), i = m - 1; i; add(bh[i], 1), --i)
50     {
51         g[++tot].a = bh[i], g[tot].b = ask(b[bh[i]].k[1], lim, 0, 0);
52         g[tot].c = dist(g[tot].a, g[tot].b);
53         if (g[tot].b == 0) --tot;
54         g[++tot].a = bh[i], g[tot].b = ask(1, b[bh[i]].k[0], 1, 1);
55         g[tot].c = dist(g[tot].a, g[tot].b);
56         if (g[tot].b == 0) --tot;
57     }
58     b[0].k[1] = b[0].k[0];
59     memset(d, 0, sizeof(d));
60     sort(bh + 1, bh + m + 1, comy);
61     for (add(bh[m], 0), i = m - 1; i; add(bh[i], 0), --i)
62     {
63         g[++tot].a = bh[i], g[tot].b = ask(1, b[bh[i]].k[1], 0, 0);
64         g[tot].c = dist(g[tot].a, g[tot].b);
65         if (g[tot].b == 0) --tot;
66         g[++tot].a = bh[i], g[tot].b = ask(1, b[bh[i]].k[0], 1, 0);
67         g[tot].c = dist(g[tot].a, g[tot].b);
68         if (g[tot].b == 0) --tot;
69     }
70 }
71 void kruskal()
72 {
73     sort(g + 1, g + tot + 1, comc);
74     for (i = 1; i <= tot; ++i)
75     {
76         int f1 = getfa(g[i].a), f2 = getfa(g[i].b);
77         if (f1 != f2) link(g[i].a, g[i].b), f[f1] = f2;
78     } tot = 0; memset(f, 0, sizeof(f));
79 }
80 void dfs(int x, int fa)
81 {
82     bh[++tot] = x;
83     for (int p = first[x]; p; p = next[p])
84         if (e[p] != fa) dfs(e[p], x), bh[++tot] = x;
85 }
86 void del(int l, int r)
87 {
88     if (l > r) return ;
89     for (int j = l; j <= r; ++j)
90         ans -= 1LL * f[a[j]] * f[a[j]], ans += 1LL * (--f[a[j]]) * f[a[j]];
91 }
92 void ins(int l, int r)
93 {
94     if (l > r) return ;
95     for (int j = l; j <= r; ++j)
96         ans -= 1LL * f[a[j]] * f[a[j]], ans += 1LL * (++f[a[j]]) * f[a[j]];
97 }
98 int main()
99 {
100     freopen("hose.in", "r", stdin);
101     freopen("hose.out", "w", stdout);
102     scanf("%d%d", &n, &m);
103     for (i = 1; i <= n; ++i)
104         scanf("%d", &i);
105     for (i = 1; i <= m; ++i)
106     {
107         scanf("%d", &b[bh[i]] = f[i] = i).x, &b[i].y);
108         b[i].k[0] = b[i].x + b[i].y;
109         b[i].k[1] = b[i].y - b[i].x + maxn;
110         lim = max(lim, max(b[i].k[0], b[i].k[1]));
111     }
112     for (h = 1; h <= lim; h <= 1);
113     manhattan();
114     kruskal();

```

```

115     dfs(1, 0);
116     ins(b[bh[1]].x, b[bh[1]].y);
117     for (s[1] = ans, i = 2; i <= tot; s[bh[i]] = ans, ++i)
118     {
119         ins(b[bh[i]].x, b[bh[i - 1]].x - 1);
120         ins(b[bh[i - 1]].y + 1, b[bh[i]].y);
121         del(b[bh[i - 1]].x, b[bh[i]].x - 1);
122         del(b[bh[i]].y + 1, b[bh[i - 1]].y);
123     }
124     for (i = 1; i <= m; ++i)
125     {
126         long long fz = s[i] - b[i].k[1] - 1 + maxn, fm = 1LL * (b[i].k[1] + 1 - maxn) * (b[i].k[1] - maxn);
127         long long gys = gcd(fz, fm);
128         printf("%lld/%lld\n", fz/gys, fm/gys);
129     }
130     return 0;
131 }
132
133
134
135
136
137
138
139 #include <iostream>
140 #include <cstdio>
141 #include <algorithm>
142 #include <math>
143 #include <cstring>
144 #define maxn 55000
145 #define inf 2147483647
146 using namespace std;
147 struct query
148 {
149     int l, r, s, w;
150 } a[maxn];
151 int c[maxn];
152 long long col[maxn], size[maxn], ans[maxn];
153 int n, m, cnt, len;
154
155 long long gcd(long long x, long long y)
156 {
157     return (!x)?y:gcd(y%x, x);
158 }
159
160 bool cmp(query a, query b)
161 {
162     return (a.w == b.w)?a.r < b.r : a.w < b.w;
163 }
164
165 int main()
166 {
167     //freopen("hose.in", "r", stdin);
168     scanf("%d%d", &n, &m);
169     for (int i = 1; i <= n; ++i) scanf("%d", &c[i]);
170     len = (int)sqrt(m);
171     cnt = (len * len == m)?len : len + 1;
172     for (int i = 1; i <= m; ++i)
173     {
174         scanf("%d", &a[i].l, &a[i].r);
175         if (a[i].l > a[i].r) swap(a[i].l, a[i].r);
176         size[i] = a[i].r - a[i].l + 1;
177         a[i].w = a[i].l / len + 1;
178         a[i].s = i;
179     }
180     sort(a + 1, a + m + 1, cmp);
181     int i = 1;
182     while (i <= m)
183     {
184         int now = a[i].w;
185         memset(col, 0, sizeof(col));
186         for (int j = a[i].l; j <= a[i].r; ++j) ans[a[i].s] += 2 * (col[c[j]] + 1);
187         ++i;
188         for (; a[i].w == now; ++i)
189         {
190             ans[a[i].s] = ans[a[i - 1].s];
191             for (int j = a[i - 1].r + 1; j <= a[i].r; ++j)
192                 ans[a[i].s] += 2 * (col[c[j]] + 1);
193             if (a[i - 1].l < a[i].l)
194                 for (int j = a[i - 1].l; j < a[i].l; ++j)
195                     ans[a[i].s] -= 2 * (--col[c[j]]);
196             else
197                 for (int j = a[i].l; j < a[i - 1].l; ++j)
198                     ans[a[i].s] += 2 * (col[c[j]] + 1);
199         }
200     }
201     long long all, num;
202     for (int i = 1; i <= m; ++i)
203     {
204         if (size[i] == 1) all = 1; else all = size[i] * (size[i] - 1);
205         num = gcd(ans[i], all);
206         printf("%lld/%lld\n", ans[i] / num, all / num);
207     }
208     return 0;
209 }

```

## 5.15 Minimum Ratio Spanning Tree

```

1  #include <cstdio>
2  #include <cstring>
3  #include <math>
4
5  #define MAXX 1111
6
7  struct
8  {
9      int x, y;
10     double z;
11 } node[MAXX];
12
13 struct
14 {
15     double l, c;
16 } map[MAXX][MAXX];
17
18 int n, l, f[MAXX], pre[MAXX];
19 double dis[MAXX];
20
21 double mst(double x)
22 {
23     int i, j, tmp;
24     double min, s = 0, t = 0;
25     memset(f, 0, sizeof(f));
26     f[1] = 1;
27     for (i = 2; i <= n; ++i)
28

```

```

1  #include <cstdio>
2  #include <cstring>
3  #include <math>
4
5  #define MAXX 1111
6
7  struct
8  {
9      int x, y;
10     double z;
11 } node[MAXX];
12
13 struct
14 {
15     double l, c;
16 } map[MAXX][MAXX];
17
18 int n, l, f[MAXX], pre[MAXX];
19 double dis[MAXX];
20
21 double mst(double x)
22 {
23     int i, j, tmp;
24     double min, s = 0, t = 0;
25     memset(f, 0, sizeof(f));
26     f[1] = 1;
27     for (i = 2; i <= n; ++i)
28

```

```

29     dis[i]=map[1][i].c-map[1][i].l*x;
30     pre[i]=1;
31 }
32 for (i=1; i<n; i++)
33 {
34     min=1e10;
35     for (j=1; j<=n; j++)
36         if (!f[j] && min>dis[j])
37         {
38             min=dis[j];
39             tmp=j;
40         }
41     f[tmp]=1;
42     t=map[pre[tmp]][tmp].l;
43     s=map[pre[tmp]][tmp].c;
44     for (j=1; j<=n; j++)
45         if (!f[j] && map[tmp][j].c-map[tmp][j].l*x<dis[j])
46         {
47             dis[j]=map[tmp][j].c-map[tmp][j].l*x;
48             pre[j]=tmp;
49         }
50     }
51     return s/t;
52 }
53
54 int main()
55 {
56     int i,j;
57     double a,b;
58     while (scanf("%d",&n),&n);
59     {
60         for (i=1; i<=n; i++)
61             scanf("%d%d%d",&node[i].x,&node[i].y,&node[i].z);
62         for (i=1; i<=n; i++)
63             for (j=i+1; j<=n; j++)
64             {
65                 map[j][i].l=map[i][j].l=sqrt(1.0*(node[i].x-node[j].x)*(node[i].x-node[j].x)
66                     +node[i].y-node[j].y*(node[i].y-node[j].y));
67                 map[j][i].c=map[i][j].c=fabs(node[i].z-node[j].z);
68             }
69     a=0,b=mst(a);
70     while (fabs(b-a)>1e-8)
71     {
72         a=b;
73         b=mst(a);
74     }
75     printf("%.3lf\n",b);
76     return 0;
77 }
78 }

```

## 5.16 Minimum-cost flow problem

```

1 // like Edmonds-Karp Algorithm
2 inline void add(int a,int b,int c,int d)
3 { adde(a,b,c,d);adde(b,a,0,-d);}
4
5 int dist[MX],pre[MX];
6 int source,sink;
7 std::queue<int>q;
8 bool in[MX];
9
10 inline bool go()
11 {
12     static int now,i;
13     memset(dist,0x3f,sizeof dist);
14     dist[source]=0;
15     pre[source]=-1;
16     q.push(source);
17     in[source]=true;
18     while(!q.empty())
19     {
20         in[now=q.front()]=false;
21         q.pop();
22         for(i=edge[now]; i!=-1; i=nxt[i])
23             if(cap[i] && dist[v]>dist[now]+cst[i])
24             {
25                 dist[v]=dist[now]+cst[i];
26                 pre[v]=i;
27                 if(!in[v])
28                 {
29                     q.push(v);
30                     in[v]=true;
31                 }
32             }
33     }
34     return dist[sink]!=inf;
35 }
36
37 inline int mcmf(int &flow)
38 {
39     static int ans,i;
40     flow=ans=0;
41     while(go())
42     {
43         static int min;
44         min=inf;
45         for(i=pre[sink]; i!=-1; i=pre[to[i^1]])
46             min=std::min(min,cap[i]);
47         flow+=min;
48         ans+=min*dist[sink];
49         for(i=pre[sink]; i!=-1; i=pre[to[i^1]])
50         {
51             cap[i]-=min;
52             cap[i^1]+=min;
53         }
54     }
55     return ans;
56 }

```

## 5.17 Spanning tree

```

1 Minimum Bottleneck Spanning Tree && All-pairs vertexes' Minimum Bottleneck Path
2 Minimum Diameter Spanning Tree : Kariv-Hakimi Algorithm
3 Directed Minimum Spanning Tree : -ChuLiu/Edmonds' algorithm
4 Second-best MST
5 Degree-constrained MST
6 Minimum Ratio Spanning Tree
7 Manhattan MST
8 Enumerate All MST
9 Count Spanning Trees
10 Minimum Steiner Tree
11 k-MST

```

## 5.18 Stable Marriage

```

1 //对于每个预备队列中的对象，及被匹配对象，先按照喜好程度排列匹配对象
2
3 while(!g.empty()) // 预备匹配队列
4 {
5     if(dfn[edge[g.front()].front()]==1)
6         占据
7     else
8     {
9         for(it=edge[edge[g.front()].front()].begin(); it!=edge[edge[g.front()].front()].end()
10            ());++it)
11             if(*it==dfn[edge[g.front()].front()] || *it==g.front()) //如果被匹配对象更喜欢正在
12                 被匹配的人或现在准备匹配的对象
13                 break;
14             if(*it==g.front()) //如果喜欢新的
15             {
16                 g.push_back(dfn[edge[g.front()].front()]);
17                 dfn[edge[g.front()].front()]=g.front();
18             }
19             else
20                 g.push_back(g.front()); //否则放到队尾，重新等待匹配
21     }
22     edge[g.front()].pop_front(); //每组匹配最多只考虑一次
23     g.pop_front();
24 }

```

## 5.19 Stoer-Wagner Algorithm

```

1 #include<stdio>
2 #include<cstring>
3
4 const int maxn=510;
5
6 int map[maxn][maxn];
7 int n;
8
9 void contract(int x,int y)//合并两个点
10 {
11     int i,j;
12     for (i=0; i<n; i++)
13         if (i!=x)
14         {
15             map[x][i]+=map[y][i];
16             map[i][x]+=map[i][y];
17         }
18     for (i=y+1; i<n; i++)
19         for (j=0; j<n; j++)
20         {
21             map[i-1][j]=map[i][j];
22             map[j][i-1]=map[j][i];
23         }
24     n--;
25 }
26
27 int w[maxn],c[maxn];
28 int sx,tx;
29
30 int mincut() //求最大生成树，计算最后一个点的割，并保存最后一条边的两个顶点
31 {
32     static int i,j,k,t;
33     memset(c,0,sizeof(c));
34     c[0]=1;
35     for (i=0; i<n; i++)
36         w[i]=map[0][i];
37     for (i=1; i<n; i++)
38     {
39         t=k=-1;
40         for (j=0; j<n; j++)
41             if (c[j]==0&&w[j]>k)
42                 k=w[t=j];
43         c[sx=t]=1;
44         for (j=0; j<n; j++)
45             w[j]+=map[t][j];
46     }
47     for (i=0; i<n; i++)
48         if (c[i]==0)
49             return w[tx=i];
50 }
51
52 int main()
53 {
54     int i,j,k,m;
55     while (scanf("%d",&n,&m)!=EOF)
56     {
57         memset(map,0,sizeof(map));
58         while (m--)
59         {
60             scanf("%d%d",&i,&j,&k);
61             map[i][j]+=k;
62             map[j][i]+=k;
63         }
64         int mint=999999999;
65         while (n>1)
66         {
67             k=mincut();
68             if (k<mint) mint=k;
69             contract(sx,tx);
70         }
71         printf("%d\n",mint);
72     }
73     return 0;
74 }

```

## 5.20 Strongly Connected Component

```

1 //缩点后注意自环
2 void dfs(const short &now)
3 {
4     dfn[now]=low[now]=cnt++;
5     st.push(now);
6     for(std::list<short>::const_iterator it(edge[now].begin()); it!=edge[now].end(); ++it)
7         if(dfn[*it]==-1)
8         {
9             dfs(*it);
10            low[now]=std::min(low[now],low[*it]);
11        }
12        else
13            if(sc[*it]==1)
14                low[now]=std::min(low[now],dfn[*it]);
15        if(dfn[now]==low[now])
16        {
17            while(sc[now]==1)

```



```

18     {
19         sc[st.top()] = p;
20         st.pop();
21     }
22     }
23     }
24 }

```

## 6 math

### 6.1 cantor

```

1 const int PermSize = 12;
2 int fac[PermSize] = {1, 1, 2, 6, 24, 120, 720, 5040, 40320, 362880, 3628800, 39916800};
3
4 inline int Cantor(int a[])
5 {
6     int i, j, cnt;
7     int res = 0;
8     for (i = 0; i < PermSize; ++i)
9     {
10         cnt = 0;
11         for (j = i + 1; j < PermSize; ++j)
12             if (a[i] > a[j])
13                 ++cnt;
14         res = res + cnt * fac[PermSize - i - 1];
15     }
16     return res;
17 }
18
19 bool h[13];
20
21 inline void UnCantor(int x, int res[])
22 {
23     int i, j, l, t;
24     for (i = 1; i <= 12; ++i)
25         h[i] = false;
26     for (i = 1; i <= 12; ++i)
27     {
28         t = x / fac[12 - i];
29         x = t * fac[12 - i];
30         for (j = 1, l = 0; l <= t; ++j)
31             if (h[j])
32                 ++l;
33         j--;
34         h[j] = true;
35         res[i - 1] = j;
36     }
37 }

```

### 6.2 Discrete logarithms - BSGS

```

1 //The running time of BSGS and the space complexity is O(\sqrt{n})
2 //Pollard's rho algorithm for logarithms' running time is approximately O(\sqrt{p}) where 4
3 //p is n's largest prime factor.
4 #include<cstdio>
5 #include<cmath>
6 #include<cstring>
7
8 struct Hash // std::map is bad. clear()时会付出巨大的代价
9 {
10     static const int mod=100003; // prime is good
11     static const int MAXX=4711; // bigger than sqrt(c)
12     int hd[mod],nxt[MAXX],cnt;
13     long long v[MAXX],k[MAXX]; // a^k v (mod c)
14     inline void init()
15     {
16         memset(hd,0,sizeof hd);
17         cnt=0;
18     }
19     inline long long find(long long v)
20     {
21         static int now;
22         for(now=hd[v%mod];now;nxt[now])
23             if(this->v[now]==v)
24                 return k[now];
25         return -1;
26     }
27     inline void insert(long long k,long long v)
28     {
29         if(find(v)!=-1)
30             return;
31         nxt[++cnt]=hd[v%mod];
32         hd[v%mod]=cnt;
33         this->v[cnt]=v;
34         this->k[cnt]=k;
35     }
36 }hash;
37
38 long long gcd(long long a,long long b)
39 {
40     return b?gcd(b,a%b):a;
41 }
42
43 long long exgcd(long long a,long long b,long long &x,long long &y)
44 {
45     if(b)
46     {
47         long long re(exgcd(b,a%b,x,y)),tmp(x);
48         x=y;
49         y=tmp-(a/b)*y;
50         return re;
51     }
52     x=1;
53     y=0;
54     return a;
55 }
56
57 inline long long bsgs(long long a,long long b,long long c) // a^x = b (mod c)
58 {
59     static long long x,y,d,g,m,am,k;
60     static int i,cnt;
61     d=gcd(c);
62     b%=c;
63     x=1; // if c==1....
64     for(i=0;i<100;++i)
65     {
66         if(x==b)
67             return i;
68         x=x*a%c;
69     }
70     d=gcd(c);
71 }

```

```

70 cnt=0;
71 while((g=gcd(a,c))!=1)
72 {
73     if(!g)
74         return -1;
75     ++cnt;
76     c/=g;
77     b/=g;
78     d=a/g*d%c;
79 }
80 hash.init();
81 m=sqrt((double)c); // maybe need a ceil
82 am=1;
83 hash.insert(0,am);
84 for(i=1;i<=m;++i)
85 {
86     am=am*d%c;
87     hash.insert(i,am);
88 }
89 for(i=0;i<=m;++i)
90 {
91     g=exgcd(d,c,x,y);
92     x=(x*b/g+c)%c;
93     k=hash.find(x);
94     if(k!=-1)
95         return i*m+k+cnt;
96     d=d*a%c;
97 }
98 return -1;
99 }
100
101 long long k,p,n;
102
103 int main()
104 {
105     while(scanf("%lld_%lld_%lld",&k,&p,&n)!=EOF)
106     {
107         if(n>p || (k==bsgs(k,n,p))!=-1)
108             puts("Orz,I cant find D!");
109         else
110             printf("%lld\n",k);
111     }
112     return 0;
113 }

```

### 6.3 Divisor function

```

1 sum of positive divisors function
2 (n)=(pow(p[0],a[0]+1)-1)/(p[0]-1)* (pow(p[1],a[1]+1)-1)/(p[1]-1)* ... (pow(p[n-1],a[n-1]+1)-1)/(p[n-1]-1);

```

### 6.4 Gaussian elimination

```

1 #define N
2
3 inline int ge(int a[N][N],int n) // 返回系数矩阵的秩
4 {
5     static int i,j,k,l;
6     for(j=0;j<N;j++) //第j行,第j列
7     {
8         for(k=i;k<N;k++)
9             if(a[k][j])
10                 break;
11         if(k==n)
12             continue;
13         for(l=0;l<N;l++)
14             std::swap(a[i][l],a[k][l]);
15         for(l=0;l<N;l++)
16             if(l!=i && a[l][j])
17                 for(k=0;k<N;k++)
18                     a[l][k]^=a[i][k];
19         ++i;
20     }
21     for(j=i;j<N;j++)
22         if(a[j][n])
23             return -1; //无解
24     return i;
25 }
26 /*
27 */
28 void dfs(int v)
29 {
30     if(v==n)
31     {
32         static int x[MAXX],ta[MAXX][MAXX];
33         static int tmp;
34         memcpy(x,ans,sizeof(x));
35         memcpy(ta,a,sizeof(ta));
36         for(i=1;i>=0;i--)
37         {
38             for(j=i+1;j<N;j++)
39                 ta[i][n]^=(x[j]&&ta[j][i]); //迭代消元求解
40             x[i]=ta[i][n];
41         }
42         for(tmp=i=0;i<N;i++)
43             if(x[i])
44                 ++tmp;
45         cnt=std::min(cnt,tmp);
46         return;
47     }
48     ans[v]=0;
49     dfs(v+1);
50     ans[v]=1;
51     dfs(v+1);
52 }
53
54 inline int ge(int a[N][N],int n)
55 {
56     static int i,j,k,l;
57     for(i=0;j<N;j++)
58     {
59         for(k=i;k<N;k++)
60             if(a[k][i])
61                 break;
62         if(k==n)
63             continue;
64         for(l=0;l<N;l++)
65             std::swap(a[i][l],a[k][l]);
66         for(k=0;k<N;k++)
67             if(k!=i && a[k][i])
68                 for(l=0;l<N;l++)
69                     a[k][l]^=a[i][l];
70         ++i;
71     }
72 }

```

```

73     else //将不定元交换到后面去
74     {
75         l=n-1-j+i;
76         for(k=0;k<=r+k)
77             std::swap(a[k][l],a[k][i]);
78     }
79 }
80 if(l==n)
81 {
82     for(i=cnt=0;i<=r+i)
83         if(a[i][n])
84             ++cnt;
85     printf("%d\n",cnt);
86     continue;
87 }
88 for(j=i;j<=r+j)
89     if(a[j][n])
90         break;
91 if(j<n)
92     puts("impossible");
93 else
94 {
95     memset(ans,0,sizeof(ans));
96     cnt=1ll;
97     dfs(l=i);
98     printf("%d\n",cnt);
99 }
100 }
101
102 /*
103 */
104
105 inline void ge(int a[N][N],int m,int n) // m*n
106 {
107     static int i,j,k,l,b,c;
108     for(i=j=0;i<=n&& j<=r+j)
109     {
110         for(k=i;k<=r+k)
111             if(a[k][j])
112                 break;
113         if(k==n)
114             continue;
115         for(l=0;l<=r+l)
116             std::swap(a[i][l],a[k][l]);
117         for(k=i;k<=r+k)
118             if(k==i&& a[k][j])
119             {
120                 b=a[k][j];
121                 c=a[i][j];
122                 for(l=0;l<=r+l)
123                     a[k][l]=(a[k][l]*(a[k][l]*c-a[i][l]*b)%7+7)%7;
124             }
125         ++i;
126     }
127     for(j=i;j<=r+j)
128         if(a[j][n])
129             break;
130     if(j<n)
131     {
132         puts("Inconsistent_data.");
133         return;
134     }
135     if(i<n)
136         puts("Multiple_solutions.");
137     else
138     {
139         memset(ans,0,sizeof(ans));
140         for(i=n-1;i>=0;-i)
141         {
142             k=a[i][n];
143             for(j=i+1;j<=r+j)
144                 k=(k-a[j][j]*ans[j])%7+7)%7;
145             while(k%a[i][i])
146                 k+=7;
147             ans[i]=(k/a[i][i])%7;
148         }
149         for(i=0;i<=r+i)
150             printf("%d%c",ans[i],i+l==n?'\\n':' ');
151     }
152 }

```

## 6.5 inverse element

```

1 inline void getInv2(int x,int mod)
2 {
3     inv[1]=1;
4     for (int i=2; i<=x; i++)
5         inv[i]=(mod-(mod/i)*inv[mod/i]%mod)%mod;
6 }
7
8 long long power(long long x,long long y,int mod)
9 {
10     long long ret=1;
11     for (long long a=x%mod; y;>>=1;a=a*a%mod)
12         if (y&1)
13             ret=ret*a%mod;
14     return ret;
15 }
16
17 inline int getInv(int x,int mod)//为素数mod
18 {
19     return power(x,mod-2);
20 }

```

## 6.6 Linear programming

```

1 #include<cstdio>
2 #include<cstring>
3 #include<cmath>
4 #include<algorithm>
5
6 #define MAXN 33
7 #define MAXM 33
8 #define eps 1e-8
9
10 double a[MAXN][MAXM],b[MAXN],c[MAXN];
11 double x[MAXN],d[MAXN][MAXM];
12 int ix[MAXN][MAXM];
13 double ans;
14 int n,m;
15 int i,j,k,r,s;
16 double D;
17
18 inline bool simplex()
19 {

```

```

20     r=n;
21     s=m+1;
22     for(i=0;i<=r+m++i)
23         ix[i]=i;
24     memset(d,0,sizeof d);
25     for(i=0;i<=r+i)
26     {
27         for(j=0;j<=k+m+j)
28             d[i][j]=-a[i][j];
29         d[i][m]=1;
30         d[i][m]=b[i];
31         if(d[r][m]>d[i][m])
32             r=i;
33     }
34     for(j=0;j<=k+m+j)
35         d[n][j]=c[j];
36     d[n+1][m]=1;
37     while(true)
38     {
39         if(r<n)
40         {
41             std::swap(ix[s],ix[r+m]);
42             d[r][s]=1./d[r][s];
43             for(j=0;j<=r+m+j)
44                 if(j!=s)
45                     d[r][j]*=-d[r][s];
46             for(i=0;i<=r+l+i)
47                 if(i!=r)
48                 {
49                     for(j=0;j<=r+m+j)
50                         if(j!=s)
51                             d[i][j]+=d[r][j]*d[i][s];
52                     d[i][s]*=d[r][s];
53                 }
54             }
55         r=-1;
56         s=-1;
57         for(j=0;j<=r+m+j)
58             if((s<0 || ix[s]>ix[j]) &&& (d[n+1][j]>eps || (d[n+1][j]>-eps &&& d[n][j]>eps)))
59                 s=j;
60         if(s<0)
61             break;
62         for(i=0;i<=r+i)
63             if(d[i][s]<eps &&& (r<0 || (D=(d[r][m]/d[r][s]-d[i][m]/d[i][s]))<-eps || (D<eps &&& ix[r+m]>ix[i+m])))
64                 r=i;
65         if(r<0)
66             return false;
67     }
68     if(d[n+1][m]<-eps)
69         return false;
70     for(i=m;i<=r+m+i)
71         if(ix[i]+1<=n)
72             x[ix[i]]=d[i-m][m]; // answer
73     ans=d[n][m]; // maximum value
74     return true;
75 }
76
77 int main()
78 {
79     while(scanf("%d",&n)&&n)!=EOF)
80     {
81         for(i=0;i<=r+m++i)
82             scanf("%d",&c[i]); // max{ sum{c[i]*x[i]} }
83         for(i=0;i<=r+i)
84         {
85             for(j=0;j<=k+m+j)
86                 scanf("%d",&a[i][j]); // sum{ a[i]*x[j] } <= b
87             scanf("%d",&b[i]);
88             b[i]*=n;
89         }
90         simplex();
91         printf("Nasa_can_spend %.01f_taka.\n",ceil(ans));
92     }
93     return 0;
94 }

```

## 6.7 Lucas' theorem

```

1 #include<cstdio>
2 #include<cstring>
3 #include<iostream>
4
5 int mod;
6 long long num[100000];
7 int ni[100],mi[100];
8 int len;
9
10 void init(int p)
11 {
12     mod=p;
13     num[0]=1;
14     for (int i=1; i<p; i++)
15         num[i]=i*num[i-1]%p;
16 }
17
18 void get(int n,int ni[],int p)
19 {
20     for (int i = 0; i < 100; i++)
21         ni[i] = 0;
22     int tlen = 0;
23     while (n != 0)
24     {
25         ni[tlen++] = n%p;
26         n /= p;
27     }
28     len = tlen;
29 }
30
31 long long power(long long x,long long y)
32 {
33     long long ret=1;
34     for (long long a=x%mod; y;>>=1;a=a*a%mod)
35         if (y&1)
36             ret=ret*a%mod;
37     return ret;
38 }
39
40 long long getInv(long long x)//mod 为素数
41 {
42     return power(x,mod-2);
43 }
44
45 long long calc(int n,int m,int p)//C(n,m)%p
46 {
47     init(p);
48     long long ans=1;

```

```

49     for ( ; n && m && ans; n/=p,m/=p)
50     {
51         if (r[p]>=m%p)
52             ans = ans*num[r[p]%p *getInv(num[m%p]%p) *getInv(num[r[p-m%p])%p];
53         else
54             ans==0;
55     }
56     return ans;
57 }
58
59 int main()
60 {
61     int t;
62     scanf("%d",&t);
63     while (t--)
64     {
65         int n,m,p;
66         scanf("%d%d",&n,&m,&p);
67         printf("%ld\n",calc(n,m,p));
68     }
69     return 0;
70 }

```

## 6.8 Matrix

```

1 struct Matrix
2 {
3     const int N(52);
4     int a[N][N];
5     inline Matrix operator*(const Matrix &b)const
6     {
7         static Matrix res;
8         static int i,j,k;
9         for(i=0;i<N;++i)
10            for(j=0;j<N;++j)
11            {
12                res.a[i][j]=0;
13                for(k=0;k<N;++k)
14                    res.a[i][j]+=a[i][k]*b.a[k][j];
15            }
16        return res;
17    }
18    inline Matrix operator^(int y)const
19    {
20        static Matrix res,x;
21        static int i,j;
22        for(i=0;i<N;++i)
23        {
24            for(j=0;j<N;++j)
25            {
26                res.a[i][j]=0;
27                x.a[i][j]=a[i][j];
28            }
29            res.a[i][i]=1;
30        }
31        for(;y;>=1;x=x*x)
32            if(y&1)
33                res=res*x;
34        return res;
35    }
36 };
37
38 Fibonacci Matrix
39 [1 1]
40 [1 0]

```

## 6.9 Multiset

```

1 Permutation:
2 MultiSet S={1 m,4 s,4 i,2 p}
3 P(S)=(1+4+4+2)!/1!/4!/4!/2!
4
5 Combination:
6 MultiSet S={ a1∞, a2,... ∞ ak}
7 C(S,r)=(r+k-1)!/r!/(k-1)!=C(r,r+k-1)
8
9 if(r>min{count(element[i])})
10     you have to resolve this problem with inclusion-exclusion principle.
11
12 MS T={3 a,4 b,5 c}
13 MS Too*={ a∞, b∞, c}
14 A1=C(T*,10)/count(a)>3 // C(6,8)
15 A2=C(T*,10)/count(b)>4 // C(5,7)
16 A3=C(T*,10)/count(c)>5 // C(4,6)
17
18 C(T,10)=C(T*,10)-(|A1|+|A2|+|A3|)+(|A1 A2|+|A1 A3|+|A2 A3|)-|A1 A2 A3|
19 C(10,12) C(1,3) C(0,2) 0 0
20
21 ans=6

```

## 6.10 Pell's equation

```

1 /*
2 find the (x,y)pair that x^2-n*y^2=1
3 these is not solution if and only if n is a square number.
4
5 solution:
6 simply brute-force search the integer y, get (x1,y1). ( toooo slow in some situation )
7 or we can enumerate the continued fraction of sqrt(n), as (x/y), it will be much more
8 faster
9
10 other solution pairs' matrix:
11 [ x1 n*y1 ]
12 [ y1 x1 ]
13 k-th solution is pow({matrix},k)
14 */
15
16 import java.util.*;
17 import java.math.*;
18
19 public class Main
20 {
21     static BigInteger p,q,p1,p2,p3,q1,q2,q3,a1,a2,a0,h1,h2,g1,g2,n0;
22     static int n,t;
23     static void solve()
24     {
25         p2=BigInteger.ONE;
26         p1=BigInteger.ZERO;
27         q2=BigInteger.ZERO;
28         q1=BigInteger.ONE;
29         a0=a1=BigInteger.valueOf((long)Math.sqrt(n));
30         g1=BigInteger.ZERO;
31         h1=BigInteger.ONE;

```

```

31 n0=BigInteger.valueOf(n);
32 while(true)
33 {
34     g2=a1.multiply(h1).subtract(g1);
35     h2=(n0.subtract(g2.multiply(g2))).divide(h1);
36     a2=(g2.add(a0)).divide(h2);
37     p=p2.multiply(a1).add(p1);
38     q=q2.multiply(a1).add(q1);
39     if(p.multiply(p).subtract(n0.multiply(q.multiply(q))).equals(BigInteger.ONE)
40         return ;
41     a1=a2;
42     g1=g2;
43     h1=h2;
44     p1=p2;
45     p2=p;
46     q1=q2;
47     q2=q;
48 }
49
50 public static void main(String[] args)
51 {
52     Scanner in=new Scanner(System.in);
53     t=in.nextInt();
54     for(int i=0;i<t;++i)
55     {
56         n=in.nextInt();
57         solve();
58         System.out.println(p1+" "+q);
59     }
60 }
61 }

```

## 6.11 Pollard's rho algorithm

```

1 #include<cstdio>
2 #include<cstdlib>
3 #include<list>
4
5 short T;
6 unsigned long long a;
7 std::list<unsigned long long>fac;
8
9 inline unsigned long long multi_mod(const unsigned long long &a,unsigned long long b,const
10     unsigned long long &n)
11 {
12     unsigned long long exp(a%n,tmp(0));
13     while(b)
14     {
15         if(b&1)
16         {
17             tmp=exp;
18             if(tmp>n)
19                 tmp=n;
20         }
21         exp<<=1;
22         if(exp>n)
23             exp=n;
24         b>>=1;
25     }
26     return tmp;
27 }
28
29 inline unsigned long long exp_mod(unsigned long long a,unsigned long long b,const unsigned
30     long long &n)
31 {
32     unsigned long long tmp(1);
33     while(b)
34     {
35         if(b&1)
36             tmp=multi_mod(tmp,a,c);
37         a=multi_mod(a,a,c);
38         b>>=1;
39     }
40     return tmp;
41 }
42
43 inline bool miller_rabbin(const unsigned long long &n,short T)
44 {
45     if(n==2)
46         return true;
47     if(n<2 || !(n&1))
48         return false;
49     unsigned long long a,u(n-1),x,y;
50     short t(0),i;
51     while(!(u&1))
52     {
53         ++t;
54         u>>=1;
55     }
56     while(T--)
57     {
58         a=rand()%(n-1)+1;
59         x=exp_mod(a,u,n);
60         for(i=0;i<t;++i)
61         {
62             y=multi_mod(x,x,n);
63             if(y==1 && x!=1 && x!=n-1)
64                 return false;
65             x=y;
66         }
67         if(y!=1)
68             return false;
69     }
70     return true;
71 }
72
73 unsigned long long gcd(const unsigned long long &a,const unsigned long long &b)
74 {
75     return b?gcd(b,a%b):a;
76 }
77
78 inline unsigned long long pollar_rho(const unsigned long long n,const unsigned long long &
79     c)
80 {
81     unsigned long long x(rand()%(n-1)+1),y,d,i(1),k(2);
82     y=x;
83     while(true)
84     {
85         ++i;
86         x=(multi_mod(x,x,n)+c)%n;
87         d=gcd((x-y)%n,n);
88         if(d>1 && d<n)
89             return d;
90         if(x==y)
91             return n;
92         if(i==k)
93             {

```

```

91         k<=&=1;
92         y=x;
93     }
94 }
95 }
96 void find(const unsigned long long &n,short c)
97 {
98     if(n==1)
99         return;
100     if(miller_rabbin(n,6))
101     {
102         fac.push_back(n);
103         return;
104     }
105     unsigned long long p(n);
106     short k(c);
107     while(p>=n)
108         p=pollar_rho(p,c--);
109     find(p,k);
110     find(n/p,k);
111 }
112 }
113
114 int main()
115 {
116     scanf("%d",&T);
117     while(T--)
118     {
119         scanf("%llu",&a);
120         fac.clear();
121         find(a,120);
122         if(fac.size()==1)
123             puts("Prime");
124         else
125         {
126             fac.sort();
127             printf("%llu\n",fac.front());
128         }
129     }
130     return 0;
131 }

```

## 6.12 Reduced Residue System

```

1 Euler's totient function: 对正整数，欧拉函数 是少于或等于的数中与互质的数的数目，也就是对的简化剩余系的大小。
2
3 mnmn
4 (1) (唯一和互质的数就是本身)。=111若
5 m互质，n(nm)=n(n)。对于来说，所有这样的数的和为
6 nn*(n)/2
7
8 inline long long phi(int n)
9 {
10     static int i;
11     static int re;
12     re=n;
13     for(i=0;p[i]*p[i]<=n;i++)
14         if(n%p[i]==0)
15             {
16                 re=re/p[i];
17                 do
18                     n/=p[i];
19                 while(n%p[i]!=0);
20             }
21     if(n!=1)
22         re=re/n;
23     return re;
24 }
25
26 inline void Euler()
27 {
28     static int i,j;
29     phi[1]=1;
30     for(i=2;i<=MAX;i++)
31         if(!phi[i])
32             for(j=i;j<=MAX;j+=i)
33                 {
34                     if(!phi[j])
35                         phi[j]=j;
36                     phi[j]=phi[j]/i*(i-1);
37                 }
38 }
39
40 Multiplicative_order:
41
42 the multiplicative_order_of_a_modulo_n_is_the_smallest_positive_integer_k_with
43 a^k=1(mod n). 对的简化剩余系中的所有
44
45 mx,ord(x)都一定是 (m)的一个约数(aka Euler's totient theorem)求
46
47 :
48 method、根据定义，对 1(m)分解素因子之后暴力枚举所有 (m)的约数，找到最小的一个，满足d pow(x,d,m)=1
49 method、2
50 inline long long ord(long long x,long long m)
51 {
52     static long long ans;
53     static int i,j;
54     ans=phi(m);
55     for(i=0;i<fac.size();i++)
56         for(j=0;j<fac[i].second && pow(x,ans/fac[i].first,m)!=1;j++)
57             ans/=fac[i].first;
58     return ans;
59 }
60
61 Primitive root:若
62
63 ord(x)=m，则为的一个原根因此只需检查所有
64 pow(x,d) 为 d(m)的约数 找到使 pow(x,d)%m=1 的所有，当且仅当这样的只有一个，并且为 dd(m)的时候，x
65 的一个原根当且仅当
66
67 n= 1,2,4,pow(p,n),2*pow(p,n) {为奇质数p,为正整数n} 时，存在原根n // 应该是指存在对于完全剩余系的原根4
68     ....? 当存在原根时，原根数目为
69
70 m((m))求：枚举每一个简化剩余系中的数，若对于的每一个质因子
71
72 iip[j].pow(i,(m)/p[j])%m都不为，那么为的一个原根。也就是说，mimord(i)=n。最小原根通常极小。
73
74 Carmichael function:
75
76 (n) is defined as the smallest positive integer m such that
77 pow(a,m)%n=1 { for a=1 && gcd(a,n)=1 }也就是简化剩余系完全剩余系中存在乘法群中无法得到的数
78 (1)中所有的x lcm(ord(x))
79
80

```

```

81 if n==pow(p[0],a[0])*pow(p[1],a[1])*...*pow(p[m-1],a[m-1])
82 then (n)=lcm ((pow(p[0],a[0])),(pow(p[1],a[1])),...,(pow(p[m-1],a[m-1])));
83
84 if n==pow(2,a)*pow(p[0],a[0])*pow(p[1],a[1])*...*pow(p[m-1],a[m-1])
85 then (n)=lcm(pow(2,c),(pow(p[0],a[0])),(pow(p[1],a[1])),...,(pow(p[m-1],a[m-1])));
86 { c=0 if a<2; c=1 if a==2; c=a-2 if a>3; }
87
88 Carmichael's theorem:
89 if gcd(a,n)=1
90 then pow(a,(n))%n=1

```

## 6.13 System of linear congruences

```

1 // minimal val that for all (m,a) , va%n=a
2 #include<stdio>
3
4 #define MAX 11
5
6 int T,t;
7 int m[MAX],a[MAX];
8 int n,i,j,k;
9 int x,y,c,d;
10 int lcm;
11
12 int exgcd(int a,int b,int &x,int &y)
13 {
14     if(b)
15     {
16         int re(exgcd(b,a/b,x,y)),tmp(x);
17         x=y;
18         y=tmp-(a/b)*y;
19         return re;
20     }
21     x=1;
22     y=0;
23     return a;
24 }
25
26 int main()
27 {
28     scanf("%d",&T);
29     for(t=1;t<=T;t++)
30     {
31         scanf("%d",&n);
32         lcm=1;
33         for(i=0;i<T;i++)
34         {
35             scanf("%d",&m[i]);
36             lcm*=m[i]/exgcd(lcm,m[i],x,y);
37         }
38         for(i=0;i<T;i++)
39             scanf("%d",&a[i]);
40         for(i=1;i<T;i++)
41         {
42             c=a[i]-a[0];
43             d=exgcd(m[i],m[0],x,y);
44             if(d%d)
45                 break;
46             y=m[i]/d;
47             c/=d;
48             x=(x*d+y)%y;
49             a[0]=(a[0]+m[0]*x;
50             m[0]*=y;
51         }
52         printf("Case %d: %d\n",t,i<T?-1:(a[0]?a[0]:lcm));
53     }
54     return 0;
55 }

```

## 7 other

### 7.1 bigint

```

1 // header files
2 #include <stdio>
3 #include <string>
4 #include <algorithm>
5 #include <iostream>
6
7 struct Bigint
8 {
9     // representations and structures
10     std::string a; // to store the digits
11     int sign; // sign = -1 for negative numbers, sign = 1 otherwise
12     // constructors
13     Bigint() {} // default constructor
14     Bigint( std::string b ) { (*this) = b; } // constructor for std::string
15     // some helpful methods
16     int size() // returns number of digits
17     {
18         return a.size();
19     }
20     Bigint inverseSign() // changes the sign
21     {
22         sign *= -1;
23         return (*this);
24     }
25     Bigint normalize( int newSign ) // removes leading 0, fixes sign
26     {
27         for( int i = a.size() - 1; i > 0 && a[i] == '0'; i-- )
28             a.erase(a.begin() + i);
29         sign = ( a.size() == 1 && a[0] == '0' ) ? 1 : newSign;
30         return (*this);
31     }
32     // assignment operator
33     void operator = ( std::string b ) // assigns a std::string to Bigint
34     {
35         a = b[0] == '-' ? b.substr(1) : b;
36         reverse( a.begin(), a.end() );
37         this->normalize( b[0] == '-' ? -1 : 1 );
38     }
39     // conditional operators
40     bool operator < ( const Bigint &b ) const // less than operator
41     {
42         if( sign != b.sign )
43             return sign < b.sign;
44         if( a.size() != b.a.size() )
45             return sign == 1 ? a.size() < b.a.size() : a.size() > b.a.size();
46         for( int i = a.size() - 1; i >= 0; i-- )
47             if( a[i] != b.a[i] )
48                 return sign == 1 ? a[i] < b.a[i] : a[i] > b.a[i];

```

```

49     return false;
50 }
51 bool operator == ( const Bigint &b ) const // operator for equality
52 {
53     return a == b.a && sign == b.sign;
54 }
55
56 // mathematical operators
57 Bigint operator + ( Bigint b ) // addition operator overloading
58 {
59     if ( sign != b.sign )
60         return (*this) - b.inverseSign();
61     Bigint c;
62     for (int i = 0, carry = 0; i < a.size() || i < b.size() || carry; i++)
63     {
64         carry += (i < a.size() ? a[i] - 48 : 0) + (i < b.size() ? b.a[i] - 48 : 0);
65         c.a += (carry % 10 + 48);
66         carry /= 10;
67     }
68     return c.normalize(sign);
69 }
70
71 Bigint operator - ( Bigint b ) // subtraction operator overloading
72 {
73     if ( sign != b.sign )
74         return (*this) + b.inverseSign();
75     int s = sign; sign = b.sign = 1;
76     if ( (*this) < b )
77         return ((b - (*this)).inverseSign()).normalize(-s);
78     Bigint c;
79     for (int i = 0, borrow = 0; i < a.size(); i++)
80     {
81         borrow = a[i] - borrow - (i < b.size() ? b.a[i] : 48);
82         c.a += borrow >= 0 ? borrow + 48 : borrow + 58;
83         borrow = borrow >= 0 ? 0 : 1;
84     }
85     return c.normalize(s);
86 }
87 Bigint operator * ( Bigint b ) // multiplication operator overloading
88 {
89     Bigint c("0");
90     for (int i = 0, k = a[i] - 48; i < a.size(); i++, k = a[i] - 48)
91     {
92         while (k--)
93             c = c + b; // ith digit is k, so, we add k times
94         b.a.insert(b.a.begin(), '0'); // multiplied by 10
95     }
96     return c.normalize(sign * b.sign);
97 }
98 Bigint operator / ( Bigint b ) // division operator overloading
99 {
100     if ( b.size() == 1 && b.a[0] == '0' )
101         b.a[0] /= ( b.a[0] - 48 );
102     Bigint c("0"), d;
103     for (int j = 0; j < a.size(); j++)
104         d.a += "0";
105     int dSign = sign * b.sign;
106     b.sign = 1;
107     for (int i = a.size() - 1; i >= 0; i--)
108     {
109         c.a.insert( c.a.begin(), '0' );
110         c = c + a.substr( i, 1 );
111         while ( !( c < b ) )
112         {
113             c = c - b;
114             d.a[i]++;
115         }
116     }
117     return d.normalize(dSign);
118 }
119 Bigint operator % ( Bigint b ) // modulo operator overloading
120 {
121     if ( b.size() == 1 && b.a[0] == '0' )
122         b.a[0] /= ( b.a[0] - 48 );
123     Bigint c("0");
124     b.sign = 1;
125     for (int i = a.size() - 1; i >= 0; i--)
126     {
127         c.a.insert( c.a.begin(), '0' );
128         c = c + a.substr( i, 1 );
129         while ( !( c < b ) )
130             c = c - b;
131     }
132     return c.normalize(sign);
133 }
134
135 // output method
136 void print()
137 {
138     if ( sign == -1 )
139         putchar('-');
140     for (int i = a.size() - 1; i >= 0; i--)
141         putchar(a[i]);
142 }
143 };
144
145 int main()
146 {
147     Bigint a, b, c; // declared some Bigint variables
148     // taking Bigint input
149     // taking Bigint input
150     // taking Bigint input
151     // taking Bigint input
152     // taking Bigint input
153
154     std::string input; // std::string to take input
155     std::cin >> input; // take the Big integer as std::string
156     a = input; // assign the std::string to Bigint a
157
158     std::cin >> input; // take the Big integer as std::string
159     b = input; // assign the std::string to Bigint b
160
161     // Using mathematical operators //
162     // Using mathematical operators //
163     // Using mathematical operators //
164
165     c = a + b; // adding a and b
166     c.print(); // printing the Bigint
167     puts(""); // newline
168
169     c = a - b; // subtracting b from a
170     c.print(); // printing the Bigint
171     puts(""); // newline
172
173     c = a * b; // multiplying a and b
174     c.print(); // printing the Bigint
175     puts(""); // newline
176
177     c = a / b; // dividing a by b

```

```

178     c.print(); // printing the Bigint
179     puts(""); // newline
180
181     c = a % b; // a modulo b
182     c.print(); // printing the Bigint
183     puts(""); // newline
184
185     // Using conditional operators //
186     // Using conditional operators //
187     // Using conditional operators //
188
189     if ( a == b )
190         puts("equal"); // checking equality
191     else
192         puts("not equal");
193
194     if ( a < b )
195         puts("a is smaller than b"); // checking less than operator
196
197     return 0;
198 }

```

## 7.2 java

```

1 //Scanner
2
3 Scanner in = new Scanner(new FileReader("asdf"));
4 PrintWriter pw = new PrintWriter(new FileWriter("out"));
5 boolean in.hasNext();
6 String in.next();
7 BigDecimal in.nextBigDecimal();
8 BigInteger in.nextBigInteger();
9 BigInteger in.nextBigInteger(int radix);
10 double in.nextDouble();
11 int in.nextInt();
12 int in.nextInt(int radix);
13 String in.nextLine();
14 long in.nextLong();
15 long in.nextLong(int radix);
16 short in.nextShort();
17 short in.nextShort(int radix);
18 int in.nextInt(); //Returns this scanner's default radix.
19 Scanner in.useRadix(int radix); // Sets this scanner's default radix to the specified
    radix.
20 void in.close(); //Closes this scanner.
21
22 //String
23
24 char str.charAt(int index);
25 int str.compareTo(String anotherString); // <0 if less. ==0 if equal. >0 if
    greater.
26 int str.compareToIgnoreCase(String str);
27 String str.concat(String str);
28 boolean str.contains(CharSequence s);
29 boolean str.endsWith(String suffix);
30 boolean str.startsWith(String prefix);
31 boolean str.startsWith(String prefix, int toffset);
32 int str.hashCode();
33 int str.indexOf(int ch);
34 int str.indexOf(int ch, int fromIndex);
35 int str.indexOf(String str);
36 int str.indexOf(String str, int fromIndex);
37 int str.lastIndexOf(int ch);
38 int str.lastIndexOf(int ch, int fromIndex);
39 // (ry
40 int str.length();
41 String str.substring(int beginIndex);
42 String str.substring(int beginIndex, int endIndex);
43 String str.toLowerCase();
44 String str.toUpperCase();
45 String str.trim(); // Returns a copy of the string, with leading and trailing
    whitespace omitted.
46
47 //StringBuilder
48 StringBuilder str.insert(int offset, ...);
49 StringBuilder str.reverse();
50 void str.setCharAt(int index, int ch);
51
52 //BigInteger
53 compareTo(); equals(); doubleValue(); longValue(); hashCode(); toString(); toString(int
    radix); max(); min(); mod(); modPow(BigInteger exp, BigInteger m); nextProbablePrime
    (); pow();
54 andNot(); and(); xor(); not(); or(); getLowestSetBit(); bitCount(); bitLength(); setBit(
    int n); shiftLeft(int n); shiftRight(int n);
55 add(); divide(); divideAndRemainder(); remainder(); multiply(); subtract(); gcd(); abs();
    signum(); negate();
56
57 //BigDecimal
58 movePointLeft(); movePointRight(); precision(); stripTrailingZeros(); toBigInteger();
    toPlainString();
59
60 //sort
61 class pii implements Comparable
62 {
63     public int a, b;
64     public int compareTo(Object i)
65     {
66         pii c = (pii) i;
67         return a == c.a ? c.b - b : c.a - a;
68     }
69 }
70
71 class Main
72 {
73     public static void main(String[] args)
74     {
75         pii[] theNew pii[2];
76         the[0] = new pii();
77         the[1] = new pii();
78         the[0].a = 1;
79         the[0].b = 1;
80         the[1].a = 1;
81         the[1].b = 2;
82         Arrays.sort(the);
83         for (int i = 0; i < 2; i++)
84             System.out.printf("%d %d\n", the[i].a, the[i].b);
85     }
86 }
87

```

## 7.3 others

```

1 god damn it windows:
2 #pragma comment(linker, "/STACK:16777216")
3 #pragma comment(linker, "/STACK:102400000,102400000")

```

```

4
5
6 chmod +x [filename]
7
8 while true; do
9 ./gen > input
10 ./sol < input > output.sol
11 ./bf < input > output.bf
12
13 diff output.sol output.bf
14 if[ $? -ne 0];then break fi
15 done, 状态状态状态状态状态状态状态状态状态
16
17
18 1、
19 2.calm_down();calm_down();calm_down();、读完题目读完题目读完题目
20 3、不盲目跟版
21 4、考虑换题换想法
22 5/、对数商线
23 6/./hash观察问题本身点 区间互转//、对数调整精度
24 6.1 or 将乘法转换成加法、点化区间、区间化点
25 6.2、数组大小……
26 7

```

## 8 search

### 8.1 dlx

```

1 精确覆盖：给定一个矩阵，现在要选择一些行，使得每一列有且仅有一个。
2 011每次选定一个元素个数最少的列，从该列中选择一行加入答案，删除该行所有的列以及与该行冲突的行。重复覆盖；
3 定一个矩阵，现在要选择一些行，使得每一列至少有一个。
4
5 011每次选定一个元素个数最少的列，从该列中选择一行加入答案，删除该行所有的列。与该行冲突的行可能满足重复覆盖；

```

### 8.2 dlx - exact cover

```

1 #include<stdio>
2 #include<cstring>
3 #include<algorithm>
4 #include<vector>
5
6 #define N 256
7 #define MAXN N*22
8 #define MAXMN N*5
9 #define inf 0x3f3f3f3f
10 const int MAXX[MAXN*MAXN];
11
12 bool mat[MAXN][MAXN];
13
14 int u[MAXN],d[MAXN],l[MAXN],r[MAXN],ch[MAXN],rh[MAXN];
15 int sz[MAXN];
16 std::vector<int>ans(MAXX);
17 int hd,cnt;
18
19 inline int node(int up,int down,int left,int right)
20 {
21     u[cnt]=up;
22     d[cnt]=down;
23     l[cnt]=left;
24     r[cnt]=right;
25     u[down]=d[up]=l[right]=r[left]=cnt;
26     return cnt++;
27 }
28
29 inline void init(int n,int m)
30 {
31     cnt=0;
32     hd=node(0,0,0,0);
33     static int i,j,k,r;
34     for(j=1;j<=m;j++)
35     {
36         ch[j]=node(cnt,cnt,l[hd],hd);
37         sz[j]=0;
38     }
39     for(i=1;i<=n;i++)
40     {
41         r=-1;
42         for(j=1;j<=m;j++)
43             if(mat[i][j])
44             {
45                 if(r==1)
46                 {
47                     r=node(u[ch[j]],ch[j],cnt,cnt);
48                     rh[r]=i;
49                     ch[r]=ch[j];
50                 }
51                 else
52                 {
53                     k=node(u[ch[j]],ch[j],l[r],r);
54                     rh[k]=i;
55                     ch[k]=ch[j];
56                 }
57                 ++sz[j];
58             }
59     }
60 }
61
62 inline void mn(int c)
63 {
64     l[r[c]]=l[c];
65     r[l[c]]=r[c];
66     static int i,j;
67     for(i=d[c];i!=c;i=d[i])
68         for(j=r[i];j!=i;j=r[j])
69         {
70             u[d[j]]=u[j];
71             d[u[j]]=d[j];
72             --sz[ch[j]];
73         }
74 }
75
76 inline void add(int c)
77 {
78     static int i,j;
79     for(i=u[c];i!=c;i=u[i])
80         for(j=l[i];j!=i;j=l[j])
81         {
82             ++sz[ch[j]];
83             u[d[j]]=d[u[j]]=j;
84         }
85     l[r[c]]=r[l[c]]=c;

```

```

86 }
87
88 bool dlx(int k)
89 {
90     if(hd==hd)
91     {
92         ans.resize(k);
93         return true;
94     }
95     int s=inf,c;
96     int i,j;
97     for(i=r[hd];i!=hd;i=r[i])
98         if(sz[i]<s)
99         {
100             s=sz[i];
101             c=i;
102         }
103     mn(c);
104     for(i=d[c];i!=c;i=d[i])
105     {
106         ans[k]=rh[i];
107         for(j=r[i];j!=i;j=r[j])
108             mn(ch[j]);
109         if(dlz(k+1))
110             return true;
111         for(j=l[i];j!=i;j=l[j])
112             add(ch[j]);
113     }
114     add(c);
115     return false;
116 }
117
118 #include <stdio>
119 #include <cstring>
120
121 #define N 1024
122 #define M 1024*110
123 using namespace std;
124
125 int l[M], r[M], d[M], u[M], col[M], row[M], h[M], res[N], cntcol[N];
126 int dcnt = 0;
127 inline void addnode(int &x)
128 {
129     ++x;
130     r[x] = l[x] = u[x] = d[x] = x;
131 }
132 inline void insert_row(int rowx, int x)
133 {
134     r[l[rowx]] = x;
135     l[x] = l[rowx];
136     r[x] = rowx;
137     l[rowx] = x;
138 }
139 inline void insert_col(int colx, int x)
140 {
141     d[u[colx]] = x;
142     u[x] = u[colx];
143     d[x] = colx;
144     u[colx] = x;
145 }
146 inline void dlx_init(int cols)
147 {
148     memset(h, -1, sizeof(h));
149     memset(cntcol, 0, sizeof(cntcol));
150     dcnt = -1;
151     addnode(dcnt);
152     for (int i = 1; i <= cols; ++i)
153     {
154         addnode(dcnt);
155         insert_row(0, dcnt);
156     }
157 }
158 inline void remove(int c)
159 {
160     l[r[c]] = l[c];
161     r[l[c]] = r[c];
162     for (int i = d[c]; i != c; i = d[i])
163         for (int j = r[i]; j != i; j = r[j])
164         {
165             u[d[j]] = u[j];
166             d[u[j]] = d[j];
167             cntcol[col[j]]--;
168         }
169 }
170 inline void resume(int c)
171 {
172     for (int i = u[c]; i != c; i = u[i])
173         for (int j = l[i]; j != i; j = l[j])
174         {
175             u[d[j]] = j;
176             d[u[j]] = j;
177             cntcol[col[j]]++;
178         }
179     l[r[c]] = c;
180     r[l[c]] = c;
181 }
182 bool DLX(int deep)
183 {
184     if (r[0] == 0)
185     {
186         //Do anything you want to do here
187         printf("%d", deep);
188         for (int i = 0; i < deep; ++i) printf("_%d", res[i]);
189         puts("");
190         return true;
191     }
192     int min = INT_MAX tempc;
193     for (int i = r[0]; i != 0; i = r[i])
194         if (cntcol[i] < min)
195         {
196             min = cntcol[i];
197             tempc = i;
198         }
199     remove(tempc);
200     for (int i = d[tempc]; i != tempc; i = d[i])
201     {
202         res[deep] = row[i];
203         for (int j = r[i]; j != i; j = r[j]) remove(col[j]);
204         if (DLX(deep + 1)) return true;
205         for (int j = l[i]; j != i; j = l[j]) resume(col[j]);
206     }
207     resume(tempc);
208     return false;
209 }
210 //插入矩阵中的节点"1"
211 inline void insert_node(int x, int y)
212 {
213     cntcol[y]++;
214     addnode(dcnt);

```

```

215 row[dcnt] = x;
216 col[dcnt] = y;
217 insert_col(y, dcnt);
218 if (h[x] == -1) h[x] = dcnt;
219 else insert_row(h[x], dcnt);
220 }
221 int main()
222 {
223     int n, m;
224     while (~scanf("%d%d", &n, &m))
225     {
226         dlx_init(m);
227         for (int i = 1; i <= n; ++i)
228         {
229             int k, x;
230             scanf("%d", &k);
231             while (k--)
232             {
233                 scanf("%d", &x);
234                 insert_node(i, x);
235             }
236         }
237         if (DLX(0))
238             puts("NO");
239     }
240     return 0;
241 }

```

## 8.3 dlx - repeat cover

```

1  #include<cstdio>
2  #include<cstring>
3  #include<algorithm>
4
5  #define MAXN 110
6  #define MAXM 1000000
7  #define INF 0x7FFFFFFF
8
9  using namespace std;
10
11 int G[MAXN][MAXN];
12 int L[MAXN], R[MAXN], U[MAXN], D[MAXN];
13 int size, ans, S[MAXN], H[MAXN], C[MAXN];
14 bool vis[MAXN * 100];
15 void Link(int r, int c)
16 {
17     U[size] = c;
18     D[size] = D[c];
19     U[D[c]] = size;
20     D[c] = size;
21     if (H[r] < 0)
22         H[r] = L[size] = R[size] = size;
23     else
24     {
25         L[size] = H[r];
26         R[size] = R[H[r]];
27         L[R[H[r]]] = size;
28         R[H[r]] = size;
29     }
30     S[c]++;
31     C[size++] = c;
32 }
33 void Remove(int c)
34 {
35     int i;
36     for (i = D[c]; i != c; i = D[i])
37     {
38         L[R[i]] = L[i];
39         R[L[i]] = R[i];
40     }
41 }
42 void Resume(int c)
43 {
44     int i;
45     for (i = D[c]; i != c; i = D[i])
46         L[R[i]] = R[L[i]] = i;
47 }
48 int A()
49 {
50     int i, j, k, res;
51     memset(vis, false, sizeof(vis));
52     for (res = 0, i = R[0]; i = R[i])
53     {
54         if (!vis[i])
55         {
56             res++;
57             for (j = D[i]; j != i; j = D[j])
58             {
59                 for (k = R[j]; k != j; k = R[k])
60                     vis[C[k]] = true;
61             }
62         }
63     }
64     return res;
65 }
66 void Dance(int now)
67 {
68     if (R[0] == 0)
69         ans = min(ans, now);
70     else if (now + A() < ans)
71     {
72         int i, j, temp, c;
73         for (temp = INF, i = R[0]; i = R[i])
74         {
75             if (temp > S[i])
76             {
77                 temp = S[i];
78                 c = i;
79             }
80         }
81         for (i = D[c]; i != c; i = D[i])
82         {
83             Remove(i);
84             for (j = R[i]; j != i; j = R[j])
85                 Remove(j);
86             Dance(now + 1);
87             for (j = L[i]; j != i; j = L[j])
88                 Resume(j);
89             Resume(i);
90         }
91     }
92 }
93 void Init(int m)
94 {
95     int i;
96     for (i = 0; i <= m; i++)
97     {

```

```

98         R[i] = i + 1;
99         L[i + 1] = i;
100         U[i] = D[i] = i;
101         S[i] = 0;
102     }
103     R[m] = 0;
104     size = m + 1;
105 }

```

## 8.4 fibonacci knapsack

```

1  #include<stdio.h>
2  #include<stdlib.h>
3  #include<algorithm>
4
5  #define MAXX 71
6
7  struct mono
8  {
9      long long weig, cost;
10 } goods[MAXX];
11
12 short n, T, t, i;
13 long long carry, sumw, sumc;
14 long long ans, las[MAXX];
15
16 int com(const void *n, const void *m)
17 {
18     struct mono *a = (struct mono *)n, *b = (struct mono *)m;
19     if (a->weig < b->weig)
20         return a->weig - b->weig;
21     else
22         return b->cost - a->cost;
23 }
24
25 bool comp(const struct mono a, const struct mono b)
26 {
27     if (a.weig < b.weig)
28         return a.weig < b.weig;
29     else
30         return b.cost < a.cost;
31 }
32
33 void dfs(short i, long long cost_n, long long carry_n, short last)
34 {
35     if (ans < cost_n)
36         ans = cost_n;
37     if (i == n || goods[i].weig > carry_n || cost_n + las[i] <= ans)
38         return;
39     if (last || (goods[i].weig == goods[i - 1].weig && goods[i].cost > goods[i - 1].cost))
40         dfs(i + 1, cost_n + goods[i].cost, carry_n - goods[i].weig, 1);
41     dfs(i + 1, cost_n, carry_n, 0);
42 }
43
44 int main()
45 {
46     // freopen("asdf", "r", stdin);
47     scanf("%d", &T);
48     for (t = 1; t <= T; ++t)
49     {
50         scanf("%d%d", &n, &carry);
51         sumw = 0;
52         sumc = 0;
53         ans = 0;
54         for (i = 0; i < n; ++i)
55         {
56             scanf("%d%d", &goods[i].weig, &goods[i].cost);
57             sumw += goods[i].weig;
58             sumc += goods[i].cost;
59         }
60         if (sumw <= carry)
61         {
62             printf("Case %d: %d\n", t, sumc);
63             continue;
64         }
65         // qsort(goods, n, sizeof(struct mono), com);
66         std::sort(goods, goods + n, comp);
67         for (i = 0; i < n; ++i)
68         {
69             // printf("%d %d\n", goods[i].weig, goods[i].cost);
70             las[i] = sumc;
71             sumc = goods[i].cost;
72         }
73         dfs(0, 0, carry, 1);
74         printf("Case %d: %d\n", t, ans);
75     }
76     return 0;
77 }

```

## 9 string

### 9.1 Aho-Corasick Algorithm

```

1  //trie graph
2  #include<cstring>
3  #include<queue>
4
5  #define MAX 1000111
6  #define N 26
7
8  int nxt[MAX][N], fail[MAX], cnt;
9  bool ed[MAX];
10 char buf[MAX];
11
12 inline void init(int a)
13 {
14     memset(nxt[a], 0, sizeof(nxt[0]));
15     fail[a] = 0;
16     ed[a] = false;
17 }
18
19 inline void insert()
20 {
21     static int i, p;
22     for (i = p; buf[i]; ++i)
23     {
24         if (!nxt[p][map[buf[i]]])
25             init(nxt[p][map[buf[i]]] = ++cnt);
26         p = nxt[p][map[buf[i]]];
27     }
28     ed[p] = true;
29 }

```

```

30 inline void make()
31 {
32     static std::queue<int> q;
33     int i, now, p;
34     q.push(0);
35     while(!q.empty())
36     {
37         now=q.front();
38         q.pop();
39         for(i=0; i<N++i)
40             if(nxt[now][i])
41             {
42                 q.push(p=nxt[now][i]);
43                 if(now)
44                     fal[p]=nxt[fal[now]][i];
45                 ed[p]=ed[fal[p]];
46             }
47             else
48                 nxt[now][i]=nxt[fal[now]][i]; // 使用本身的存串的时候注意已被重载nxt
49     }
50 }
51 // normal version
52 #define N 128
53 char buf[MAXN];
54 int cnt[1111];
55 struct node
56 {
57     node *fal,*nxt[N];
58     int idx;
59     node() { memset(this,0,sizeof node); }
60 }*rt;
61 std::queue<node*> Q;
62 void free(node *p)
63 {
64     for(int i(0); i<N++i)
65         if(p>nxt[i])
66             free(p>nxt[i]);
67     delete p;
68 }
69 inline void add(char *s,int idx)
70 {
71     static node *p;
72     for(p=rt;*s++s)
73     {
74         if(!p>nxt[*s])
75             p>nxt[*s]=new node();
76         p=p>nxt[*s];
77     }
78     p>idx=idx;
79 }
80 inline void make()
81 {
82     Q.push(rt);
83     static node *p,*q;
84     static int i;
85     while(!Q.empty())
86     {
87         p=Q.front();
88         Q.pop();
89         for(i=0; i<N++i)
90             if(p>nxt[i])
91             {
92                 q=p>fal;
93                 while(q)
94                 {
95                     if(q>nxt[i])
96                     {
97                         p>nxt[i]->fal=q>nxt[i];
98                         break;
99                     }
100                     q=q>fal;
101                 }
102                 if(!q)
103                     p>nxt[i]->fal=rt;
104                 Q.push(p>nxt[i]);
105             }
106     }
107 }
108 inline void match(const char *s)
109 {
110     static node *p,*q;
111     for(p=rt;*s++s)
112     {
113         while(p!=rt && !p>nxt[*s])
114             p=p>fal;
115         p=p>nxt[*s];
116         if(!p)
117             p=rt;
118         for(q=p;q!=rt && q>idx;q=q>fal) // why q>idx ? looks like not necessary at all,
119             I delete it in an other solution
120             ++cnt[q>idx];
121     }
122 }
123 //可以考虑一下，拉直指针来跳过无效的匹配dfs fal
124 //在线调整关键字存在性的时候，可以考虑欧拉序压扁之后使用或者线段树进行区间修改BIT

```

## 9.2 Gusfield's Z Algorithm

```

1 inline void make(int *z,char *buf)
2 {
3     int i,j,l,r;
4     l=0;
5     r=1;
6     z[0]=strlen(buf);
7     for(i=1;i<z[0];++i)
8         if(r<=i || z[i-l]>=r-i)
9         {
10             j=std::max(i,r);
11             while(j<z[0] && buf[j]==buf[j-i])
12                 ++j;
13             z[i]=j-i;
14             if(i<j)
15             {
16                 l=i;
17                 r=j;
18             }
19         }
20     else

```

```

21         z[i]=z[i-l];
22     }
23     for(i=1;i<len && i+z[i]<len;++i); //i可能最小循环节长度=
24

```

## 9.3 Manacher's Algorithm

```

1 inline int match(const int a,const int b,const std::vector<int> &str)
2 {
3     static int i;
4     i=0;
5     while(a-i>=0 && b+i<str.size() && str[a-i]==str[b+i])
6         ++i;
7     return i;
8 }
9
10 inline void go(int *z,const std::vector<int> &str)
11 {
12     static int c,l,r,i,ii,n;
13     z[0]=1;
14     c=l=r=0;
15     for(i=1;i<str.size();++i)
16     {
17         ii=(l<=l)-i;
18         r=++l-i;
19
20         if(i>r)
21         {
22             z[i]=match(i,i,str);
23             l=i;
24             r=++z[i]-1;
25         }
26         else
27             if(z[i]==n)
28             {
29                 z[i]=n+match(i-n,i+n,str);
30                 l=i;
31                 r=++z[i]-1;
32             }
33             else
34                 z[i]=std::min(z[ii],n);
35         if(z[i]>z[c])
36             c=i;
37     }
38 }
39
40 inline bool check(int *z,int a,int b) //检查子串[a,b]是否回文
41 {
42     a=a*2-1;
43     b=b*2-1;
44     int m=(a+b)/2;
45     return z[m]>=b-m+1;
46 }

```

## 9.4 Morris-Pratt Algorithm

```

1 inline void make(char *buf,int *fal)
2 {
3     static int i,j;
4     fal[0]=-1;
5     for(i=1,j=-1;buf[i];++i)
6     {
7         while(j>=0 && buf[j+1]!=buf[i])
8             j=fal[j];
9         if(buf[j+1]==buf[i])
10             ++j;
11         fal[i]=j;
12     }
13 }
14
15 inline int match(char *p,char *t,int* fal)
16 {
17     static int i,j,re;
18     re=0;
19     for(i=0,j=-1;t[i];++i)
20     {
21         while(j>=0 && p[j+1]!=t[i])
22             j=fal[j];
23         if(p[j+1]==t[i])
24             ++j;
25         if(!p[j+1])
26         {
27             ++re;
28             j=fal[j];
29         }
30     }
31     return re;
32 }
33

```

## 9.5 smallest representation

```

1 int min(char a[],int len)
2 {
3     int i=0,j=1,k=0;
4     while (i < len && j < len && k < len)
5     {
6         int cmp = a[(j+k)%len]-a[(i+k)%len];
7         if (cmp == 0)
8             k++;
9         else
10         {
11             if (cmp > 0)
12                 j += k+1;
13             else
14                 i += k+1;
15             if (i == j) j++;
16             k = 0;
17         }
18     }
19     return std::min(i,j);
20 }

```

## 9.6 Suffix Array - DC3 Algorithm

```

1 #include<cstdio>
2 #include<cstring>
3 #include<algorithm>
4

```



## 9.7 Suffix Array - Prefix-doubling Algorithm

```

5 #define MAXX 1111
6 #define F(x) ((x)/3+((x)%3==1?0:tb))
7 #define G(x) ((x)<tb?(x)*3+1:((x)-tb)*3+2)
8
9 int wa[MAXX],wb[MAXX],wv[MAXX],ws[MAXX];
10
11 inline bool c0(const int *str,const int &a,const int &b)
12 {
13     return str[a]==str[b] || str[a]==str[b] && str[a+1]==str[b+1] && str[a+2]==str[b+2];
14 }
15
16 inline bool c12(const int *str,const int &k,const int &a,const int &b)
17 {
18     if(k==2)
19         return str[a]<str[b] || str[a]==str[b] && c12(str,1,a+1,b+1);
20     else
21         return str[a]<str[b] || str[a]==str[b] && wv[a+1]<wv[b+1];
22 }
23
24 inline void sort(int *str,int *a,int *b,const int &n,const int &m)
25 {
26     memset(ws,0,sizeof(ws));
27     int i;
28     for(i=0;i<n;i++)
29         ++ws[wv[i]=str[a[i]]];
30     for(i=1;i<n;i++)
31         ws[i]+=ws[i-1];
32     for(i=n-1;i>=0;i--)
33         b[--ws[wv[i]]]=a[i];
34 }
35
36 inline void dc3(int *str,int *sa,const int &n,const int &m)
37 {
38     int *strn(str+n);
39     int *san(sa+n),tb((n+1)/3),ta(0),tbc(0),i,j,k;
40     str[n]=str[n+1]=0;
41     for(i=0;i<n;i++)
42         if(i%3)
43             wa[tbc++]=i;
44     sort(str+2,wa,wb,tbc,m);
45     sort(str+1,wb,wa,tbc,m);
46     sort(str,wa,wb,tbc,m);
47     for(i=j=1,strn[F(wb[i])]=0;i<tbc;i++)
48         strn[F(wb[i])]=c0(str,wb[i-1],wb[i])?j-1:j++;
49     if(j<tbc)
50         dc3(strn,san,tbc,j);
51     else
52         for(i=0;i<tbc;i++)
53             san[strn[i]]=i;
54     for(i=0;i<tbc;i++)
55         if(san[i]<tb)
56             wb[ta++]=san[i]*3;
57     if(tb%3==1)
58         wb[ta++]=n-1;
59     sort(str,wb,wa,ta,m);
60     for(i=0;i<tbc;i++)
61         wv[wb[i]]=G(san[i]);
62     for(i=j=k=0;i<ta && j<tbc;)
63         sa[k++]=c12(str,wb[j]%3,wa[i]);
64     while(i<ta)
65         sa[k++]=wa[i++];
66     while(j<tbc)
67         sa[k++]=wb[j++];
68 }
69
70 int rk[MAXX],lcpa[MAXX],sa[MAXX*3];
71 int str[MAXX*3]; //必须int
72
73 int main()
74 {
75     scanf("%d",&n,&m);
76     for(i=0;i<n;i++)
77     {
78         scanf("%d",&k);
79         num[i]=k-j+100;
80         j=k;
81     }
82     num[n]=0;
83
84     dc3(num,sa,n+1,191); //191: 中取值范围, 桶排序str
85
86     for(i=1;i<=n;i++) // 数组rank
87         rk[sa[i]]=i;
88     for(i=k=0;i<=n;i++) // 数组lcp
89         if(!rk[i])
90             lcpa[0]=0;
91     else
92     {
93         j=sa[rk[i]-1];
94         if(k>0)
95             --k;
96         while(num[i+k]==num[j+k])
97             ++k;
98         lcpa[rk[i]]=k;
99     }
100
101
102     for(i=1;i<=n;i++)
103         sptb[0][i]=i;
104     for(i=1;i<=lg[n];i++) //sparse table RMQ
105     {
106         k=i+1;(1<<i);
107         for(j=1;j<=n;j++)
108         {
109             a=sptb[i-1][j];
110             b=sptb[i-1][j+(1<<(i-1))];
111             sptb[i][j]=lcpa[a]<lcpa[b]?a:b;
112         }
113     }
114 }
115
116 inline int ask(int l,int r)
117 {
118     a=lg[r-l+1];
119     r=(1<<a)-1;
120     l=sptb[a][l];
121     r=sptb[a][r];
122     return lcpa[l]<lcpa[r]?l:r;
123 }
124
125 inline int lcp(int l,int r) // 字符串上[l,r区间的]lrmq
126 {
127     l=rk[l];
128     r=rk[r];
129     if(l>r)
130         std::swap(l,r);
131     return lcpa[ask(l+1,r)];
132 }

```

```

1 int wx[maxn],wy[maxn],*x,*y,wss[maxn],wv[maxn];
2
3 inline bool cmp(int *r,int n,int a,int b,int l)
4 {
5     return a+l<n && b+l<n && r[a]==r[b] && r[a+l]==r[b+l];
6 }
7 void da(int str[],int sa[],int n,int m)
8 {
9     int *s = str;
10    int *x=wx,*y=wy,*t,p;
11    int i,j;
12    for(i=0; i<n; i++)
13        wss[i]=0;
14    for(i=0; i<n; i++)
15        wss[x[i]=s[i]]++;
16    for(i=1; i<n; i++)
17        wss[i]+=wss[i-1];
18    for(i=n-1; i>=0; i--)
19        sa[--wss[x[i]]]=i;
20    for(j=1,p=1; p<n && j<n; j*=2,m=p)
21    {
22        for(i=n-j,p=0; i<n; i++)
23            y[p++]=i;
24        for(i=0; i<n; i++)
25            if(sa[i]-j>=0)
26                y[p++]=sa[i]-j;
27        for(i=0; i<n; i++)
28            wv[i]=x[y[i]];
29        for(i=0; i<n; i++)
30            wss[i]=0;
31        for(i=0; i<n; i++)
32            wss[wv[i]]++;
33        for(i=1; i<n; i++)
34            wss[i]+=wss[i-1];
35        for(i=n-1; i>=0; i--)
36            sa[--wss[wv[i]]]=y[i];
37        for(t=x,x=y,y=t,p=1,i=1,x[sa[0]]=0; i<n; i++)
38            x[sa[i]]=cmp(y,n,sa[i-1],sa[i],j)?p-1:p++;
39    }
40 }

```