Contents

Chapter 1

data structure

1.1 binary indexed tree.cpp

```
1  int tree[MAXX];
2
3  inline int lowbit(const int &a)
4  {
5      return a&-a;
6  }
7
8  inline void update(int pos,const int &val)
9  {
10     while(pos<MAXX)
11     {
12         tree[pos]+=val;
13         pos+=lowbit(pos);
14     }
15 }
16
17 inline int read(int pos)
18 {
19     int re(0);
20     while(pos>0)
21     {
22         re+=tree[pos];
23         pos-=lowbit(pos);
24     }
25     return re;
26 }
27
28 int find_Kth(int k)
```

```
29 {
30     int now=0;
31     for (char i=20;i>=0;--i)
32     {
33         now|=(1<<i);
34         if (now>MAXX || tree[now]>=k)
35             now^=(1<<i);
36         else k-=tree[now];
37     }
38     return now+1;
39 }
```

1.2 COT.cpp

```
1  #include<cstdio>
2  #include<algorithm>
3
4  #define MAXX 100111
5  #define MAX (MAXX*23)
6  #define N 18
7
8  int sz[MAX],lson[MAX],rson[MAX],cnt;
9  int head[MAXX];
10 int pre[MAXX][N];
11 int map[MAXX],m;
12
13 int edge[MAXX],nxt[MAXX<<1],to[MAXX<<1];
14 int n,i,j,k,q,l,r,mid;
15 int num[MAXX],dg[MAXX];
16
17 int make(int l,int r)
18 {
19     if(l==r)
20         return ++cnt;
21     int id(++cnt),mid((l+r)>>1);
22     lson[id]=make(l,mid);
23     rson[id]=make(mid+1,r);
24     return id;
25 }
26
27 inline int update(int id,int pos)
28 {
29     int re(++cnt);
30     l=1;
31     r=m;
32     int nid(re);
```

```
33      sz[nid]=sz[id]+1;
34      while(l<r)
35      {
36          mid=(l+r)>>1;
37          if(pos<=mid)
38          {
39              lson[nid]=++cnt;
40              rson[nid]=rson[id];
41              nid=lson[nid];
42              id=lson[id];
43              r=mid;
44          }
45          else
46          {
47              lson[nid]=lson[id];
48              rson[nid]=++cnt;
49              nid=rson[nid];
50              id=rson[id];
51              l=mid+1;
52          }
53          sz[nid]=sz[id]+1;
54      }
55      return re;
56  }
57
58  void rr(int now,int fa)
59  {
60      dg[now]=dg[fa]+1;
61      head[now]=update(head[fa],num[now]);
62      for(int i(edge[now]);i;i=nxt[i])
63          if(to[i]!=fa)
64          {
65              j=1;
66              for(pre[to[i]][0]=now;j<N;++j)
67                  pre[to[i]][j]=pre[pre[to[i]][j-1]][j-1];
68              rr(to[i],now);
69          }
70  }
71
72  inline int query(int a,int b,int n,int k)
73  {
74      static int tmp,t;
75      l=1;
76      r=m;
77      a=head[a];
78      b=head[b];
```

```
79      t=num[n];
80      n=head[n];
81      while(l<r)
82      {
83          mid=(l+r)>>1;
84          tmp=sz[lson[a]]+sz[lson[b]]-2*sz[lson[n]]+(l<=t
                && t<=mid);
85          if(tmp>=k)
86          {
87              a=lson[a];
88              b=lson[b];
89              n=lson[n];
90              r=mid;
91          }
92          else
93          {
94              k-=tmp;
95              a=rson[a];
96              b=rson[b];
97              n=rson[n];
98              l=mid+1;
99          }
100     }
101     return l;
102 }
103
104 inline int lca(int a,int b)
105 {
106     static int i,j;
107     j=0;
108     if(dg[a]<dg[b])
109         std::swap(a,b);
110     for(i=dg[a]-dg[b];i;i>>=1,++j)
111         if(i&1)
112             a=pre[a][j];
113     if(a==b)
114         return a;
115     for(i=N-1;i>=0;--i)
116         if(pre[a][i]!=pre[b][i])
117         {
118             a=pre[a][i];
119             b=pre[b][i];
120         }
121     return pre[a][0];
122 }
123
```

```
124 int main()
125 {
126     scanf("%d %d",&n,&q);
127     for(i=1;i<=n;++i)
128     {
129         scanf("%d",num+i);
130         map[i]=num[i];
131     }
132     std::sort(map+1,map+n+1);
133     m=std::unique(map+1,map+n+1)-map-1;
134     for(i=1;i<=n;++i)
135         num[i]=std::lower_bound(map+1,map+m+1,num[i])-map
                ;
136     for(i=1;i<n;++i)
137     {
138         scanf("%d %d",&j,&k);
139         nxt[++cnt]=edge[j];
140         edge[j]=cnt;
141         to[cnt]=k;
142 
143         nxt[++cnt]=edge[k];
144         edge[k]=cnt;
145         to[cnt]=j;
146     }
147     cnt=0;
148     head[0]=make(1,m);
149     rr(1,0);
150     while(q--)
151     {
152         scanf("%d %d %d",&i,&j,&k);
153         printf("%d\n",map[query(i,j,lca(i,j),k)]);
154     }
155     return 0;
156 }
```

1.3 divide tree.cpp

```
1 //              val[0][1],val[0][n]
2 
3 template<class Tp>class DT
4 {
5     public:
6         int n;
7         Tp val[20][MAXX],sorted[MAXX];
8         inline void make()
9         {
```

```
10              std::sort(sorted+1,sorted+1+n);
11              make(1,1,n,0);
12          }
13          inline int query(const int &l,const int &r,const
                int &k)
14          {
15              return query(1,1,n,l,r,k,0);
16          }
17      private:
18          int toleft[20][MAXX],mid[MAXX<<2];
19          // toleft:
20          void make(const int &id,const int &l,const int &r
                ,const int &d)
21          {
22              if(l!=r)
23              {
24                  mid[id]=(l+r)>>1;
25                  int lsame(mid[id]-l+1),i;
26                  for(i=l;i<=r;++i)
27                      if(val[d][i]<sorted[mid[id]])
28                          --lsame;
29                  int lpos(l),rpos(mid[id]+1),same(0);
30                  for(i=l;i<=r;++i)
31                  {
32                      if(i==l)
33                          toleft[d][i]=0;
34                      else
35                          toleft[d][i]=toleft[d][i-1];
36                      if(val[d][i]<sorted[mid[id]])
37                      {
38                          ++toleft[d][i];
39                          val[d+1][lpos++]=val[d][i];
40                      }
41                      else
42                          if(val[d][i]>sorted[mid[id]])
43                              val[d+1][rpos++]=val[d][i];
44                          else
45                              if(same<lsame)
46                              {
47                                  ++same;
48                                  ++toleft[d][i];
49                                  val[d+1][lpos++]=val[d][i
                                        ];
50                              }
51                              else
52                                  val[d+1][rpos++]=val[d][i
```

```
                                        ];
53                  }
54                  make(id<<1,l,mid[id],d+1);
55                  make(id<<1|1,mid[id]+1,r,d+1);
56              }
57          }
58          int query(const int &id,const int &ll,const int &
                rr,const int &l,const int &r,const int &k,
                const int &d)
59          {
60              if(l==r)
61                  return val[d][l];
62              int s,ss;
63              if(l==ll)
64              {
65                  s=toleft[d][r];
66                  ss=0;
67              }
68              else
69              {
70                  s=toleft[d][r]-toleft[d][l-1];
71                  ss=toleft[d][l-1];
72              }
73              if(s>=k)
74              {
75                  int newl(ll+ss),newr(ll+ss+s-1);
76                  return query(id<<1,ll,mid[id],newl,newr,k
                        ,d+1);
77              }
78              int bb(l-ll-ss),b(r-l+1-s),newl(mid[id]+bb+1)
                    ,newr(mid[id]+bb+b);
79              return query(id<<1|1,mid[id]+1,rr,newl,newr,k
                    -s,d+1);
80          }
81  };
```

1.4 GSS7.cxx

```
1  #include<cstdio>
2  #include<algorithm>
3  #include<queue>
4
5  #define MAXX 100111
6  #define MAX (MAXX<<1)
7
8  struct node
```

```
 9 {
10     bool set,rev;
11     node *pre,*nxt[2],*fa;
12     int lmax,max,rmax,sum,val,sz;
13     node();
14     node(int a);
15 }*tree[MAXX],*nil,*a,*b;
16
17 node::node()
18 {
19     rev=set=false;
20     fa=pre=nil;
21     nxt[0]=nxt[1]=nil;
22     sz=lmax=max=rmax=sum=val=0;
23 }
24
25 node::node(int a)
26 {
27     set=rev=false;
28     sum=val=a;
29     sz=1;
30     lmax=max=rmax=std::max(0,a);
31     fa=pre=nxt[0]=nxt[1]=nil;
32 }
33
34 inline void add(node &x,const node &l,const node &r)
35 {
36     x.max=std::max(l.rmax+r.lmax,std::max(l.max,r.max));
37     x.lmax=std::max(l.lmax,l.sum+r.lmax);
38     x.rmax=std::max(r.rmax,r.sum+l.rmax);
39     x.sum=l.sum+r.sum;
40 }
41
42 inline void up(node *id)
43 {
44     id->sz=id->nxt[0]->sz+id->nxt[1]->sz+1;
45     id->sum=id->val+id->nxt[0]->sum+id->nxt[1]->sum;
46     id->lmax=std::max(id->nxt[0]->lmax,id->nxt[0]->sum+id
          ->val+id->nxt[1]->lmax);
47     id->rmax=std::max(id->nxt[1]->rmax,id->nxt[1]->sum+id
          ->val+id->nxt[0]->rmax);
48     id->max=std::max(id->nxt[0]->rmax+id->val+id->nxt
          [1]->lmax,std::max(id->nxt[0]->max,id->nxt[1]->max
          ));
49 }
50
```

```
51 inline void set(node *id,int val)
52 {
53     if(id==nil)
54         return;
55     id->set=true;
56     id->val=val;
57     id->sum=val*id->sz;
58     id->max=id->lmax=id->rmax=std::max(0,id->sum);
59 }
60
61 inline void down(node *id)
62 {
63     if(id==nil)
64         return;
65     if(id->rev)
66     {
67         id->rev=false;
68         for(int i(0);i<2;++i)
69             if(id->nxt[i]!=nil)
70             {
71                 id->nxt[i]->rev^=true;
72                 std::swap(id->nxt[i]->nxt[0],id->nxt[i]->
                        nxt[1]);
73                 std::swap(id->nxt[i]->lmax,id->nxt[i]->
                        rmax);
74             }
75     }
76     if(id->set)
77     {
78         for(int i(0);i<2;++i)
79             if(id->nxt[i]!=nil)
80                 set(id->nxt[i],id->val);
81         id->set=false;
82     }
83 }
84
85 inline void rot(node *id,int tp)
86 {
87     node *k(id->pre);
88     k->nxt[tp^1]=id->nxt[tp];
89     if(id->nxt[tp]!=nil)
90         id->nxt[tp]->pre=k;
91     if(k->pre!=nil)
92         k->pre->nxt[k==k->pre->nxt[1]]=id;
93     id->pre=k->pre;
94     id->nxt[tp]=k;
```

```
95      k->pre=id;
96      up(k);
97      up(id);
98  }
99
100 node *fresh(node* id)
101 {
102     node *re(id);
103     if(id->pre!=nil)
104         re=fresh(id->pre);
105     down(id);
106     return re;
107 }
108
109 inline void splay(node *id)
110 {
111     node *rt(fresh(id));
112     if(id!=rt)
113         for(std::swap(rt->fa,id->fa);id->pre!=nil;rot(id,
                id==id->pre->nxt[0]));
114 }
115
116 inline void access(node *id)
117 {
118     for(node *to(nil);id!=nil;id=id->fa)
119     {
120         splay(id);
121         id->nxt[1]->pre=nil;
122         if(id->nxt[1]!=nil)
123             id->nxt[1]->fa=id;
124         id->nxt[1]=to;
125         if(to!=nil)
126             to->pre=id;
127         to->fa=nil;
128         up(to=id);
129     }
130 }
131
132 inline void lca(node *&to,node *&id)
133 {
134     access(to);
135     splay(id);
136     for(to=nil;id->fa!=nil;splay(id=id->fa))
137     {
138         id->nxt[1]->pre=nil;
139         if(id->nxt[1]!=nil)
```

```
140                 id->nxt[1]->fa=id;
141             id->nxt[1]=to;
142             if(to!=nil)
143                 to->pre=id;
144             to->fa=nil;
145             up(to=id);
146         }
147     }
148
149     int n,i,j,k;
150     int nxt[MAX],to[MAX],edge[MAXX],cnt;
151     std::queue<int>q;
152
153     inline void add(int a,int b)
154     {
155         nxt[++cnt]=edge[a];
156         edge[a]=cnt;
157         to[cnt]=b;
158     }
159
160     void rr(int now,int fa)
161     {
162         for(int i(edge[now]);i;i=nxt[i])
163             if(to[i]!=fa)
164             {
165                 tree[to[i]]->fa=tree[now];
166                 rr(to[i],now);
167             }
168     }
169
170     /*
171     void print(node *id)
172     {
173         if(id!=nil)
174         {
175             print(id->nxt[0]);
176             printf("%2d %2d %2d %2d %2d %c %2d\n",id->val
                    ,id->sum,id->sz,id->lmax,id->max,id->rmax,id->
                    rev?'r':'n',id->pre->val);
177             print(id->nxt[1]);
178         }
179     }
180     */
181
182     int main()
183     {
```

```
184         nil=new node();
185         scanf("%d",&n);
186         for(i=1;i<=n;++i)
187         {
188             scanf("%d",&j);
189             tree[i]=new node(j);
190         }
191         for(i=1;i<n;++i)
192         {
193             scanf("%d %d",&j,&k);
194             add(j,k);
195             add(k,j);
196         }
197         tree[0]=nil;
198         rr(1,0);
199         scanf("%d",&n);
200         while(n--)
201         {
202             scanf("%d %d %d",&k,&i,&j);
203             a=tree[i];
204             b=tree[j];
205             access(a);
206             splay(a);
207             a->revˆ=true;
208             std::swap(a->nxt[0],a->nxt[1]);
209             std::swap(a->lmax,a->rmax);
210             access(b);
211             splay(b);
212             /*
213             print(b);
214             puts("");
215             printf("%d %d %d %d\n",b->sum,b->nxt[0]->sum,b->
                    val,b->nxt[1]->sum);
216             */
217             if(k==1)
218                 printf("%d\n",b->max);
219             else
220             {
221                 scanf("%d",&k);
222                 set(b,k);
223             }
224         }
225         return 0;
226 }
```

1.5 OTOCI.cpp

```
1  #include<cstdio>
2  #include<algorithm>
3
4  #define MAXX 30111
5
6  int nxt[MAXX][2],fa[MAXX],pre[MAXX],val[MAXX],sum[MAXX];
7  bool rev[MAXX];
8
9  inline void up(int id)
10 {
11     static int i;
12     sum[id]=val[id];
13     for(i=0;i<2;++i)
14         if(nxt[id][i])
15             sum[id]+=sum[nxt[id][i]];
16 }
17
18 inline void rot(int id,int tp)
19 {
20     static int k;
21     k=pre[id];
22     nxt[k][tp^1]=nxt[id][tp];
23     if(nxt[id][tp])
24         pre[nxt[id][tp]]=k;
25     if(pre[k])
26         nxt[pre[k]][k==nxt[pre[k]][1]]=id;
27     pre[id]=pre[k];
28     nxt[id][tp]=k;
29     pre[k]=id;
30     up(k);
31     up(id);
32 }
33
34 inline void down(int id)
35 {
36     static int i;
37     if(rev[id])
38     {
39         rev[id]=false;
40         std::swap(nxt[id][0],nxt[id][1]);
41         for(i=0;i<2;++i)
42             if(nxt[id][i])
43                 rev[nxt[id][i]]^=true;
```

```
44      }
45  }
46
47  int freshen(int id)
48  {
49      int re(id);
50      if(pre[id])
51          re=freshen(pre[id]);
52      down(id);
53      return re;
54  }
55
56  inline void splay(int id)
57  {
58      static int rt;
59      if(id!=(rt=freshen(id)))
60          for(std::swap(fa[id],fa[rt]);pre[id];rot(id,id==
                nxt[pre[id]][0]));
61  }
62
63  inline void access(int id)
64  {
65      static int to;
66      for(to=0;id;id=fa[id])
67      {
68          splay(id);
69          if(nxt[id][1])
70          {
71              pre[nxt[id][1]]=0;
72              fa[nxt[id][1]]=id;
73          }
74          nxt[id][1]=to;
75          if(to)
76          {
77              pre[to]=id;
78              fa[to]=0;
79          }
80          up(to=id);
81      }
82  }
83
84  inline int getrt(int id)
85  {
86      access(id);
87      splay(id);
88      while(nxt[id][0])
```

```
89      {
90          id=nxt[id][0];
91          down(id);
92      }
93      return id;
94  }
95
96  inline void makert(int id)
97  {
98      access(id);
99      splay(id);
100     if(nxt[id][0])
101         rev[id]^=true;
102 }
103
104 int n,i,j,k,q;
105 char buf[11];
106
107 int main()
108 {
109     scanf("%d",&n);
110     for(i=1;i<=n;++i)
111         scanf("%d",val+i);
112     scanf("%d",&q);
113     while(q--)
114     {
115         scanf("%s %d %d",buf,&i,&j);
116         switch(buf[0])
117         {
118             case 'b':
119                 if(getrt(i)==getrt(j))
120                     puts("no");
121                 else
122                 {
123                     puts("yes");
124                     makert(i);
125                     fa[i]=j;
126                 }
127                 break;
128             case 'p':
129                 access(i);
130                 splay(i);
131                 val[i]=j;
132                 up(i);
133                 break;
134             case 'e':
```

```
135                     if(getrt(i)!=getrt(j))
136                         puts("impossible");
137                     else
138                     {
139                         makert(i);
140                         access(j);
141                         splay(j);
142                         printf("%d\n",sum[j]);
143                     }
144                     break;
145             }
146         }
147         return 0;
148 }
```

1.6 segment tree - discretization.cpp

```
1  std::map<double,short>map;   //      hash
2  std::map<double,short>::iterator it;
3  double rmap[inf];   //      hash
4
5  short mid[MAX],cnt[MAX];
6  double len[MAX];
7
8  void make(const short &id,const short &l,const short &r)
9  {
10     mid[id]=(l+r)>>1;
11     if(l!=r)
12     {
13         make(id<<1,l,mid[id]);
14         make(id<<1|1,mid[id]+1,r);
15     }
16 }
17
18 void update(const short &id,const short &ll,const short &
       rr,const short &l,const short &r,const char &val)
19 {
20     if(ll==l && rr==r)
21     {
22         cnt[id]+=val;
23         if(cnt[id])
24             len[id]=rmap[r]-rmap[l-1];
25         else
26             if(l!=r)
27                 len[id]=len[id<<1]+len[id<<1|1];
28             else
```

```
29                      len[id]=0;
30          return;
31      }
32      if(mid[id]>=r)
33          update(id<<1,ll,mid[id],l,r,val);
34      else
35          if(mid[id]<l)
36              update(id<<1|1,mid[id]+1,rr,l,r,val);
37          else
38          {
39              update(id<<1,ll,mid[id],l,mid[id],val);
40              update(id<<1|1,mid[id]+1,rr,mid[id]+1,r,val);
41          }
42      if(!cnt[id])
43          len[id]=len[id<<1]+len[id<<1|1];
44  }
45
46  int main()
47  {
48      n<<=1;
49      map.clear();
50      for(i=0;i<n;++i)
51      {
52          scanf("%lf%lf%lf%lf%lf",&x1,&y1,&x2,&y2,&d);
53          if(x1>x2)
54              std::swap(x1,x2);
55          if(y1>y2)
56              std::swap(y1,y2);
57          sum+=(x2-x1)*(y2-y1)*d;
58          ln[i].l=x1;
59          ln[i].r=x2;
60          ln[i].h=y1;
61          ln[i].up=1;
62          ln[++i].l=x1;
63          ln[i].r=x2;
64          ln[i].h=y2;
65          ln[i].up=-1;
66          map[x1]=1;
67          map[x2]=1;
68      }
69      k=1;
70      for(it=map.begin();it!=map.end();++it,++k) //

71      {
72          it->second=k;
73          rmap[k]=it->first;
```

```
74      }
75      std::sort(ln,ln+n);
76      update(1,1,inf,map[ln[0].l]+1,map[ln[0].r],ln[0].up);
77      for(i=1;i<n;++i)
78      {
79          //                  len[1]
80          //ln[i].h-ln[i-1].h

81          update(1,1,inf,map[ln[i].l]+1,map[ln[i].r],ln[i].
                up);
82      }
83  }
```

1.7 size-blanced binary search tree.cpp

```
1   template<class Tp>class sbt
2   {
3       public:
4           inline void init()
5           {
6               rt=cnt=l[0]=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 bool find(const Tp &a)
17          {
18              return find(rt,a);
19          }
20          inline Tp pred(const Tp &a)
21          {
22              return pred(rt,a);
23          }
24          inline Tp succ(const Tp &a)
25          {
26              return succ(rt,a);
27          }
28          inline bool empty()
29          {
30              return !sz[rt];
31          }
```

```
32          inline Tp min()
33          {
34              return min(rt);
35          }
36          inline Tp max()
37          {
38              return max(rt);
39          }
40          inline void delsmall(const Tp &a)
41          {
42              dels(rt,a);
43          }
44          inline int rank(const Tp &a)
45          {
46              return rank(rt,a);
47          }
48          inline Tp sel(const int &a)
49          {
50              return sel(rt,a);
51          }
52          inline Tp delsel(int a)
53          {
54              return delsel(rt,a);
55          }
56      private:
57          int cnt,rt,l[MAXX],r[MAXX],sz[MAXX];
58          Tp val[MAXX];
59          inline void rro(int &pos)
60          {
61              int k(l[pos]);
62              l[pos]=r[k];
63              r[k]=pos;
64              sz[k]=sz[pos];
65              sz[pos]=sz[l[pos]]+sz[r[pos]]+1;
66              pos=k;
67          }
68          inline void lro(int &pos)
69          {
70              int k(r[pos]);
71              r[pos]=l[k];
72              l[k]=pos;
73              sz[k]=sz[pos];
74              sz[pos]=sz[l[pos]]+sz[r[pos]]+1;
75              pos=k;
76          }
77          inline void mt(int &pos,bool flag)
```

```
78          {
79              if(!pos)
80                  return;
81              if(flag)
82                  if(sz[r[r[pos]]]>sz[l[pos]])
83                      lro(pos);
84                  else
85                      if(sz[l[r[pos]]]>sz[l[pos]])
86                      {
87                          rro(r[pos]);
88                          lro(pos);
89                      }
90                      else
91                          return;
92              else
93                  if(sz[l[l[pos]]]>sz[r[pos]])
94                      rro(pos);
95                  else
96                      if(sz[r[l[pos]]]>sz[r[pos]])
97                      {
98                          lro(l[pos]);
99                          rro(pos);
100                     }
101                     else
102                         return;
103             mt(l[pos],false);
104             mt(r[pos],true);
105             mt(pos,false);
106             mt(pos,true);
107         }
108         void ins(int &pos,const Tp &a)
109         {
110             if(pos)
111             {
112                 ++sz[pos];
113                 if(a<val[pos])
114                     ins(l[pos],a);
115                 else
116                     ins(r[pos],a);
117                 mt(pos,a>=val[pos]);
118                 return;
119             }
120             pos=++cnt;
121             l[pos]=r[pos]=0;
122             val[pos]=a;
123             sz[pos]=1;
```

```
124         }
125         Tp del(int &pos,const Tp &a)
126         {
127             --sz[pos];
128             if(val[pos]==a || (a<val[pos] && !l[pos]) ||
                    (a>val[pos] && !r[pos]))
129             {
130                 Tp ret(val[pos]);
131                 if(!l[pos] || !r[pos])
132                     pos=l[pos]+r[pos];
133                 else
134                     val[pos]=del(l[pos],val[pos]+1);
135                 return ret;
136             }
137             else
138                 if(a<val[pos])
139                     return del(l[pos],a);
140                 else
141                     return del(r[pos],a);
142         }
143         bool find(int &pos,const Tp &a)
144         {
145             if(!pos)
146                 return false;
147             if(a<val[pos])
148                 return find(l[pos],a);
149             else
150                 return (val[pos]==a || find(r[pos],a));
151         }
152         Tp pred(int &pos,const Tp &a)
153         {
154             if(!pos)
155                 return a;
156             if(a>val[pos])
157             {
158                 Tp ret(pred(r[pos],a));
159                 if(ret==a)
160                     return val[pos];
161                 else
162                     return ret;
163             }
164             return pred(l[pos],a);
165         }
166         Tp succ(int &pos,const Tp &a)
167         {
168             if(!pos)
```

```
169                 return a;
170             if(a<val[pos])
171             {
172                 Tp ret(succ(l[pos],a));
173                 if(ret==a)
174                     return val[pos];
175                 else
176                     return ret;
177             }
178             return succ(r[pos],a);
179         }
180         Tp min(int &pos)
181         {
182             if(l[pos])
183                 return min(l[pos]);
184             else
185                 return val[pos];
186         }
187         Tp max(int &pos)
188         {
189             if(r[pos])
190                 return max(r[pos]);
191             else
192                 return val[pos];
193         }
194         void dels(int &pos,const Tp &v)
195         {
196             if(!pos)
197                 return;
198             if(val[pos]<v)
199             {
200                 pos=r[pos];
201                 dels(pos,v);
202                 return;
203             }
204             dels(l[pos],v);
205             sz[pos]=1+sz[l[pos]]+sz[r[pos]];
206         }
207         int rank(const int &pos,const Tp &v)
208         {
209             if(val[pos]==v)
210                 return sz[l[pos]]+1;
211             if(v<val[pos])
212                 return rank(l[pos],v);
213             return rank(r[pos],v)+sz[l[pos]]+1;
214         }
```

```
215         Tp sel(const int &pos,const int &v)
216         {
217             if(sz[l[pos]]+1==v)
218                 return val[pos];
219             if(v>sz[l[pos]])
220                 return sel(r[pos],v-sz[l[pos]]-1);
221             return sel(l[pos],v);
222         }
223         Tp delsel(int &pos,int k)
224         {
225             --sz[pos];
226             if(sz[l[pos]]+1==k)
227             {
228                 Tp re(val[pos]);
229                 if(!l[pos] || !r[pos])
230                     pos=l[pos]+r[pos];
231                 else
232                     val[pos]=del(l[pos],val[pos]+1);
233                 return re;
234             }
235             if(k>sz[l[pos]])
236                 return delsel(r[pos],k-1-sz[l[pos]]);
237             return delsel(l[pos],k);
238         }
239 };
```

1.8 sparse table - rectangle.cpp

```
 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          //


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][jj+(1<<(
                                    i-1))]);
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%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  }
```

1.9 sparse table - square.cpp

```
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("%hd %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 %hd:\n",t);
23      while(q--)
24      {
25          scanf("%hd %hd %hd",&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  }
```

1.10 sparse table.cpp

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

1.11 treap.cpp

```
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      inline int sel(node *pos,int k)
55      {
56          while(pos->ch[0]->sz+1!=k)
57              if(pos->ch[0]->sz>=k)
58                  pos=pos->ch[0];
59              else
60              {
61                  k-=pos->ch[0]->sz+1;
62                  pos=pos->ch[1];
63              }
64          return pos->val;
65      }
66      void del(node *&pos,int val)
67      {
68          if(pos!=null)
69          {
70              if(pos->val==val)
71              {
72                  int t(pos->ch[1]->key<pos->ch[0]->key);
73                  if(pos->ch[t]==null)
74                  {
75                      delete pos;
76                      pos=null;
77                      return;
78                  }
79                  rot(pos,t);
80                  del(pos->ch[t^1],val);
81              }
82              else
83                  del(pos->ch[val>pos->val],val);
84              up(pos);
85          }
86      }
87      public:
88      node *rt;
89
90      Treap():rt(null){}
91      inline void insert(int val)
92      {
93          insert(rt,val);
94      }
95      inline void reset()
96      {
97          rec(rt);
98          rt=null;
99      }
```

```
100     inline int sel(int k)
101     {
102         if(k<1 || k>rt->sz)
103             return 0;
104         return sel(rt,rt->sz+1-k);
105     }
106     inline void del(int val)
107     {
108         del(rt,val);
109     }
110     inline int size()
111     {
112         return rt->sz;
113     }
114 }treap[MAXX];
115
116 init:
117 {
118     srand(time(0));
119     null=new node();
120     null->val=0xc0c0c0c0;
121     null->sz=0;
122     null->key=RAND_MAX;
123     null->ch[0]=null->ch[1]=null;
124     for(i=0;i<MAXX;++i)
125         treap[i].rt=null;
126 }
```

Chapter 2

geometry

2.1 3D.cpp

```
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 //
21 double Norm(pv p)
22 {
23         return sqrt(p&p);
24 }
25
26 //'              V       t h e t a       '
```

```
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 //
45
46 x=r*sin(  )*cos(  );
47 y=r*sin(  )*sin(  );
48 z=r*cos(  );
49
50 r=sqrt(x*2+y*2+z*2);//??
51 r=sqrt(x^2+y^2+z^2);//??
52
53   =atan(y/x);
54   =acos(z/r);
55
56  r   [0,    ]
57       [0,2   ]
58       [0,   ]
59
60  lat1  [-   /2,   /2]
61  lng1  [-   ,   ]
62
63 pv getpv(double lat,double lng,double r)
64 {
65         lat += pi/2;
66         lng += pi;
67         return
68     pv(r*sin(lat)*cos(lng),r*sin(lat)*sin(lng),r*cos(lat)
           );
```

```
69 }
70
71 //
72
73 #include<cstdio>
74 #include<cmath>
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("%lf^%lf'%lf\" %s\n",&a,&b,&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("%lf^%lf'%lf\" %s\n",&a,&b,&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("%lf^%lf'%lf\" %s\n",&a,&b,&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\" %s\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: %.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     //
134     inline bool ZERO(pv p)
135     {
136         return (ZERO(p.x) && ZERO(p.y) && ZERO(p.z));
137     }
138
139     //
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;     //
147         p = (L2.s-L1.e)*(L1.s-L1.e);
148         return ZERO(p&L2.e);          //
149     }
150
151     //
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 //
163 bool OnLine(pv p, Line3D L)
164 {
165     return ZERO((p-L.s)*(L.e-L.s));
166 }
167
168 //
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 //
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 //
181 //              [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 }
```

2.2 3DCH.cpp

```
 1 #include<cstdio>
 2 #include<cmath>
 3 #include<vector>
 4 #include<algorithm>
 5
 6 #define MAXX 1111
 7 #define eps 1e-8
 8 #define inf 1e20
 9
10 struct pv
11 {
12     double x,y,z;
```

```
13      pv(){}
14      pv(const double &xx,const double &yy,const double &zz
            ):x(xx),y(yy),z(zz){}
15      inline pv operator-(const pv &i)const
16      {
17          return pv(x-i.x,y-i.y,z-i.z);
18      }
19      inline pv operator*(const pv &i)const //
20      {
21          return pv(y*i.z-z*i.y,z*i.x-x*i.z,x*i.y-y*i.x);
22      }
23      inline double operator^(const pv &i)const //
24      {
25          return x*i.x+y*i.y+z*i.z;
26      }
27      inline double len()
28      {
29          return sqrt(x*x+y*y+z*z);
30      }
31  };
32
33  struct pla
34  {
35      short a,b,c;
36      bool ok;
37      pla(){}
38      pla(const short &aa,const short &bb,const short &cc):
            a(aa),b(bb),c(cc),ok(true){}
39      inline void set();
40      inline void print()
41      {
42          printf("%hd %hd %hd\n",a,b,c);
43      }
44  };
45
46  pv pnt[MAXX];
47  std::vector<pla>fac;
48  short to[MAXX][MAXX];
49
50  inline void pla::set()
51  {
52      to[a][b]=to[b][c]=to[c][a]=fac.size();
53  }
54
55  inline double ptof(const pv &p,const pla &f) //
                    ?
```

```
56 {
57     return (pnt[f.b]-pnt[f.a])*(pnt[f.c]-pnt[f.a])^(p-pnt
           [f.a]);
58 }
59
60 inline double vol(const pv &a,const pv &b,const pv &c,
       const pv &d)//              *6
61 {
62     return (b-a)*(c-a)^(d-a);
63 }
64
65 inline double ptof(const pv &p,const short &f) //
       p       f
66 {
67     return fabs(vol(pnt[fac[f].a],pnt[fac[f].b],pnt[fac[f
           ].c],p)/((pnt[fac[f].b]-pnt[fac[f].a])*(pnt[fac[f
           ].c]-pnt[fac[f].a])).len());
68 }
69
70 void dfs(const short&,const short&);
71
72 void deal(const short &p,const short &a,const short &b)
73 {
74     if(fac[to[a][b]].ok)
75         if(ptof(pnt[p],fac[to[a][b]])>eps)
76             dfs(p,to[a][b]);
77         else
78         {
79             pla add(b,a,p);
80             add.set();
81             fac.push_back(add);
82         }
83 }
84
85 void dfs(const short &p,const short &now)
86 {
87     fac[now].ok=false;
88     deal(p,fac[now].b,fac[now].a);
89     deal(p,fac[now].c,fac[now].b);
90     deal(p,fac[now].a,fac[now].c);
91 }
92
93 inline void make()
94 {
95     fac.resize(0);
96     if(n<4)
```

```
97          return;
98
99      for(i=1;i<n;++i)
100         if((pnt[0]-pnt[i]).len()>eps)
101         {
102             std::swap(pnt[i],pnt[1]);
103             break;
104         }
105     if(i==n)
106         return;
107
108     for(i=2;i<n;++i)
109         if(((pnt[0]-pnt[1])*(pnt[1]-pnt[i])).len()>eps)
110         {
111             std::swap(pnt[i],pnt[2]);
112             break;
113         }
114     if(i==n)
115         return;
116
117     for(i=3;i<n;++i)
118         if(fabs((pnt[0]-pnt[1])*(pnt[1]-pnt[2])^(pnt[2]-
              pnt[i]))>eps)
119         {
120             std::swap(pnt[3],pnt[i]);
121             break;
122         }
123     if(i==n)
124         return;
125
126     for(i=0;i<4;++i)
127     {
128         pla add((i+1)%4,(i+2)%4,(i+3)%4);
129         if(ptof(pnt[i],add)>0)
130             std::swap(add.c,add.b);
131         add.set();
132         fac.push_back(add);
133     }
134     for(;i<n;++i)
135         for(j=0;j<fac.size();++j)
136             if(fac[j].ok && ptof(pnt[i],fac[j])>eps)
137             {
138                 dfs(i,j);
139                 break;
140             }
141
```

```
142     short tmp(fac.size());
143     fac.resize(0);
144     for(i=0;i<tmp;++i)
145         if(fac[i].ok)
146             fac.push_back(fac[i]);
147 }
148
149 inline pv gc() //
150 {
151     pv re(0,0,0),o(0,0,0);
152     double all(0),v;
153     for(i=0;i<fac.size();++i)
154     {
155         v=vol(o,pnt[fac[i].a],pnt[fac[i].b],pnt[fac[i].c
            ]);
156         re+=(pnt[fac[i].a]+pnt[fac[i].b]+pnt[fac[i].c])
            *0.25*v;
157         all+=v;
158     }
159     return re*(1/all);
160 }
161
162 inline bool same(const short &s,const short &t) //

163 {
164     pv &a=pnt[fac[s].a],&b=pnt[fac[s].b],&c=pnt[fac[s].c
        ];
165     return fabs(vol(a,b,c,pnt[fac[t].a]))<eps && fabs(vol
        (a,b,c,pnt[fac[t].b]))<eps && fabs(vol(a,b,c,pnt[
        fac[t].c]))<eps;
166 }
167
168 //
169 inline short facetcnt()
170 {
171     short ans=0;
172     for(short i=0;i<fac.size();++i)
173     {
174         for(j=0;j<i;++j)
175             if(same(i,j))
176                 break;
177         if(j==i)
178             ++ans;
179     }
180     return ans;
181 }
```

```
182
183 //
184 inline short trianglecnt()
185 {
186     return fac.size();
187 }
188
189 //                                  *2
190 inline double area(const pv &a,const pv &b,const pv &c)
191 {
192         return (b-a)*(c-a).len();
193 }
194
195 //
196 inline double area()
197 {
198     double ret(0);
199     for(i=0;i<fac.size();++i)
200         ret+=area(pnt[fac[i].a],pnt[fac[i].b],pnt[fac[i].
            c]);
201     return ret/2;
202 }
203
204 //
205 inline double volume()
206 {
207     pv o(0,0,0);
208     double ret(0);
209     for(short i(0);i<fac.size();++i)
210         ret+=vol(o,pnt[fac[i].a],pnt[fac[i].b],pnt[fac[i
            ].c]);
211     return fabs(ret/6);
212 }
```

2.3 circle & ploy's area.cpp

```
1 bool InCircle(Point a,double r)
2 {
3         return cmp(a.x*a.x+a.y*a.y,r*r) <= 0;
4         //‘
                                    E P S
              ‘
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 tvl = tv.x*tv.x+tv.y*tv.y;
23                   double tmp = sqrt(B/tvl); //
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
58  //
59
60  area2 = 0.0;
61  for (int i = 0;i < resn;i++) //

62      area2 += CalcArea(p[i],p[(i+1)%resn],r);
```

2.4 circle's area.cpp

```
1   //
2   {
3       for (int i = 0; i < n; i++)
4       {
5           scanf("%lf%lf%lf",&c[i].c.x,&c[i].c.y,&c[i].r);
6           del[i] = false;
7       }
8       for (int i = 0; i < n; i++)
9           if (del[i] == false)
10          {
11              if (c[i].r == 0.0)
12                  del[i] = true;
13              for (int j = 0; j < n; j++)
14                  if (i != j)
15                      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]                    i
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],AB,AC,BC,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("%lf%lf%lf",&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])),
143                                 Point(c[i].c.x+c[i].r*cos(e[j
                                        ].tim),c[i].c.y+c[i].r*sin
                                        (e[j].tim)))/2.0;
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 }
```

2.5 circle.cpp

```
1  //
2  #include<cstdio>
3  #include<cmath>
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("%lf %lf\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("%hd",&n),n)
57      {
58          for(i=0;i<n;++i)
59              scanf("%lf %lf",&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                  ans=std::max(ans,sum);
83              }
```

```
84          }
85          printf("%hd\n",ans+1);
86      }
87      return 0;
88  }
89
90  //
91
92  #include<cstdio>
93  #include<cmath>
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,tl,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     tl=a2-a1;
135     lt=b2-b1;
136     u=(b1-a1).cross(lt)/(tl).cross(lt);
137     return a1+tl*u;
138 }
139
140 inline pv get(const pv &a,const pv &b,const pv &c)
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("%hd",&n),n)
154     {
155         for(i=0;i<n;++i)
156             scanf("%lf␣%lf",&pnt[i].x,&pnt[i].y);
157         o=pnt[0];
158         r=0;
159         for(i=1;i<n;++i)
160             if((pnt[i]-o).len()>r+eps)
161             {
162                 o=pnt[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("%.2lf %.2lf %.2lf\n",o.x,o.y,r);
178             }
179             return 0;
180 }
181
182 //
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 //
200 {
201     for (int i = 0; i < 3; i++)
202         scanf("%lf%lf",&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("(%.6f,%.6f,%.6f)\n",tp.x,tp.y,r);
210 }
211
212 //
213 {
214     for (int i = 0; i < 3; i++)
215         scanf("%lf%lf",&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*r/tr,tv.y*r/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("(%.6f,%.6f,%.6f)\n",tp.x,tp.y,r);
235 }
```

2.6 closest point pair.cpp

```
1  //              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; //
10
11     /*
           D i v i d e
           */
12
13     int M = (L + R) / 2;
14
15     /*  C o n q u e r                                    */
16
17     double d = min(DnC(L,M), DnC(M+1,R));
18     //  if (d == 0.0) return d; //
19
```

```
20      /*
            M e r g e                                   Y           O
        (NlogN)     */
21
22      int N = 0;  //
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      /*  M e r g e                                   O (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
43  //              2
44
45  struct Point {double x, y;} p[10], t[10];
46  bool cmpx(const Point& i, const Point& j) {return i.x < j
        .x;}
47  bool cmpy(const Point& i, const Point& j) {return i.y < j
        .y;}
48
49  double DnC(int L, int R)
50  {
51      if (L >= R) return 1e9; //

52
53      /*
            D i v i d e
            */
54
55      int M = (L + R) / 2;
```

```
56
57      //
                              X

58      double x = p[M].x;
59
60      /*  C o n q u e r                                          */
61
62      //                                    Y
63      double d = min(DnC(L,M), DnC(M+1,R));
64      //  if (d == 0.0) return d; //
65
66      /*
            M e r g e                                    Y              O
           (N)     */
67
68      //
                                                                      Y

69      int N = 0;  //
70      for (int i=0; i<=M; ++i)
71          if (x - p[i].x < d)
72              t[N++] = p[i];
73
74      //
                                                                      Y

75      int P = N;  //  P
76      for (int i=M+1; i<=R; ++i)
77          if (p[i].x - x < d)
78              t[N++] = p[i];
79
80      //    Y                    M e r g e
            S o r t
81      inplace_merge(t, t+P, t+N, cmpy);
82
83      /*  M e r g e                                        O (N)
              */
84
85      for (int i=0; i<N; ++i)
86          for (int j=1; j<=2 && i+j<N; ++j)
87              d = min(d, distance(t[i], t[i+j]));
88
89      /*  M e r g e          Y                    O (N)
           */
90
```

```
91      //                                                    Merge Sort
92      inplace_merge(p+L, p+M+1, p+R+1, cmpy);
93
94      return d;
95  }
96
97  double closest_pair()
98  {
99      sort(p, p+10, cmpx);
100     return DnC(0, N-1);
101 }
102
103 //mzry
104 //
105 double calc_dis(Point &a ,Point &b) {
106         return sqrt((a.x-b.x)*(a.x-b.x) + (a.y-b.y)*(a.y-
                b.y));
107 }
108 //
109 bool operator<(const Point &a ,const Point &b) {
110         if(a.y != b.y) return a.x < b.x;
111         return a.x < b.x;
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 = l ; 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 //
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("%lf%lf",&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 }
```

2.7 half-plane intersection.cpp

```
1 // a b c
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  int main()
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+=the*sqrt(a*a+b*b);
35          p[!k].resize(0);
36          for(l=0;l<p[k].size();++l)
37              if(a*p[k][l].x+b*p[k][l].y+c<eps)
38                  p[!k].push_back(p[k][l]);
39              else
40              {
41                  m=(l+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][l]))
                            ;
44                  m=(l+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][l]))
                            ;
47              }
48          k=!k;
49          if(p[k].empty())
50              break;
51      }
```

```
52      //          p [k]
53      return p[k].empty();
54  }
55
56  //
57  //
58
59  inline pv ins(const pv &a,const pv &b)
60  {
61      u=fabs(ln.cross(a-pnt[i]));
62      v=fabs(ln.cross(b-pnt[i]))+u;
63      tl=b-a;
64      return pv(u*tl.x/v+a.x,u*tl.y/v+a.y);
65  }
66
67  int main()
68  {
69      j=0;
70      for(i=0;i<n;++i)
71      {
72          ln=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      //          p [j]
89  }
90
91  //mrzy
92
93  bool HPIcmp(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, HPIcmp);
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[++tail] = 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 }
```

2.8 kdtree.cpp

```
1  #include <iostream>
2  #include <cstdio>
3  #include <cstdlib>
4  #include <algorithm>
5  #include <stack>
6  #include <algorithm>
7  using namespace std;
8  #define MAXN 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[MAXN],pp[MAXN];
19
20 struct Node{
21     int split;//{0,1}  0
                          x                    1                    y

22     Point p;//
23 }tree[MAXN*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          else{
62              insert(mid+1,y,p,2*pos+1);
63              if(ans>=(p.x-tree[pos].p.x)*(p.x-tree[pos].p.
                    x))
64                  insert(x,mid-1,p,2*pos);
65          }
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()
```

```
{
    int cases,n;
    scanf("%d",&cases);
    while(cases--)
    {
        scanf("%d",&n);
        for(int i=1;i<=n;i++){
            scanf("%I64d%I64d",&pp[i].x,&pp[i].y);
            point[i]=pp[i];
        }
        initTree(1,n,0,1);
        for(int i=1;i<=n;i++){
            ans=1LL<<62;
            insert(1,n,pp[i],1);
            printf("%I64d\n",ans);
        }
    }
    return 0;
}
```

2.9 others

```
1  eps
2
3      sqrt(a), asin(a), acos(a)
           a                                                          a
       -1e-12            sqrt(a)
             0                a                                      a
       , asin(a) acos(a)
                                                                    a


4
5                                                                   case
       :005,                  0 :01
                                  0 :005000000001(       )
                          0 :004999999999(       )
                    printf("%.2lf", a)

6        a                  a + eps,            a - eps
7
8              -0.000
9
10       double
11
12 a==b  fabs(a-b)<eps
13 a!=b  fabs(a-b)>eps
```

```
a<b     a+eps<b
a<=b    a<b+eps
a>b     a>b+eps
a>=b    a+eps>b



cos/sin/tan
acos          [-1,+1]             [0,    ]
asin          [-1,+1]             [-    /2,+    /2]
atan          [-    /2,+    /2]
atan2          (y,x)(                     ),        tan (y/x),[-    ,+    ]
           x y

other

log                    (ln)
log10
ceil
floor

round

cpp:
java: add 0.5,then floor
cpp:
                                                 4

                                                 6

                                5                                     0

                                5             5                                     0
```

2.10 Pick's theorem

```
A:
i:
b:
A = i + b/2 - 1
```

A = 2i + b − 2

2.11 PointInPoly.cpp

```
/*
          ,
 p o l y                                3

0  --       p o l y
1  --        p o l y
2  --       p o l y
*/

int inPoly(pv p,pv poly[], int n)
{
        int i, count;
        Line ray, side;

        count = 0;
        ray.s   = p;
        ray.e.y  = p.y;
        ray.e.x  = -1;  //-
            I N F

        for (i = 0; i < n; i++)
        {
                side.s = poly[i];
                side.e = poly[(i+1)%n];

                if(OnSeg(p, side))
                        return 1;

                //        s i d e       x
                if (side.s.y == side.e.y)
                        continue;

        if (OnSeg(side.s, ray))
        {
            if (side.s.y > side.e.y)
                count++;
        }
        else
            if (OnSeg(side.e, ray))
            {
                if (side.e.y > side.s.y)
                    count++;
```

```
42              }
43              else
44                  if (inter(ray, side))
45                      count++;
46          }
47          return ((count % 2 == 1) ? 0 : 2);
48  }
```

2.12 rotating caliper.cpp

```
1   //
2
3   l=ans=0;
4   for(i=0;i<n;++i)
5   {
6       tl=pnt[(i+1)%n]-pnt[i];
7       while(abs(tl.cross(pnt[(l+1)%n]-pnt[i]))>abs(tl.cross
            (pnt[l]-pnt[i])))
8           l=(l+1)%n;
9       ans=std::max(ans,std::max(dist(pnt[l],pnt[i]),dist(
            pnt[l],pnt[(i+1)%n])));
10  }
11  return ans;
12
13  //
14  int main()
15  {
16      sq=sp=0;
17      for(i=1;i<ch[1].size();++i)
18          if(ch[1][sq]<ch[1][i])
19              sq=i;
20      tp=sp;
21      tq=sq;
22      ans=(ch[0][sp]-ch[1][sq]).len();
23      do
24      {
25          a1=ch[0][sp];
26          a2=ch[0][(sp+1)%ch[0].size()];
27          b1=ch[1][sq];
28          b2=ch[1][(sq+1)%ch[1].size()];
29          tpv=b1-(b2-a1);
30          tpv.x = b1.x - (b2.x - a1.x);
31          tpv.y = b1.y - (b2.y - a1.y);
32          len=(tpv-a1).cross(a2-a1);
33          if(fabs(len)<eps)
34          {
```

```
35              ans=std::min(ans,p2l(a1,b1,b2));
36              ans=std::min(ans,p2l(a2,b1,b2));
37              ans=std::min(ans,p2l(b1,a1,a2));
38              ans=std::min(ans,p2l(b2,a1,a2));
39              sp=(sp+1)%ch[0].size();
40              sq=(sq+1)%ch[1].size();
41          }
42          else
43              if(len<-eps)
44              {
45                  ans=std::min(ans,p2l(b1,a1,a2));
46                  sp=(sp+1)%ch[0].size();
47              }
48              else
49              {
50                  ans=std::min(ans,p2l(a1,b1,b2));
51                  sq=(sq+1)%ch[1].size();
52              }
53      }while(tp!=sp || tq!=sq);
54      return ans;
55  }
56
57  //              by mzry
58  inline void solve()
59  {
60      resa = resb = 1e100;
61      double dis1,dis2;
62      Point xp[4];
63      Line l[4];
64      int a,b,c,d;
65      int sa,sb,sc,sd;
66      a = b = c = d = 0;
67      sa = sb = sc = sd = 0;
68      Point va,vb,vc,vd;
69      for (a = 0; a < n; a++)
70      {
71          va = Point(p[a],p[(a+1)%n]);
72          vc = Point(-va.x,-va.y);
73          vb = Point(-va.y,va.x);
74          vd = Point(-vb.x,-vb.y);
75          if (sb < sa)
76          {
77              b = a;
78              sb = sa;
79          }
80          while (xmult(vb,Point(p[b],p[(b+1)%n])) < 0)
```

```
81          {
82              b = (b+1)%n;
83              sb++;
84          }
85          if (sc < sb)
86          {
87              c = b;
88              sc = sb;
89          }
90          while (xmult(vc,Point(p[c],p[(c+1)%n])) < 0)
91          {
92              c = (c+1)%n;
93              sc++;
94          }
95          if (sd < sc)
96          {
97              d = c;
98              sd = sc;
99          }
100         while (xmult(vd,Point(p[d],p[(d+1)%n])) < 0)
101         {
102             d = (d+1)%n;
103             sd++;
104         }
105
106         //‘      p[a],p[b],p[c],p[d]    ‘
107         sa++;
108     }
109 }
110
111 //
112                     P = { p(1) , ... , p(m) }      Q = { q
        (1) , ... , q(n) }                  (p(i), q(j))
        P      Q
113
114 (p(i), q(j))
115 p(i-1), p(i+1), q(j-1), q(j+1)                  (p(i), q(j))

116
117


118
119 1                   P      Q                  y

                                                   x
```

120 2

x
121 3

(p(i), q(j))

122 4 (p(i), q(j))
p(i-1), p(i+1), q(j-1), q(j+1)
(p(i), q(j))

123 5
3 4

124 6

125
126
1 5 6 O
(N) N

127
128

129
130 //
131 1 P y
yminP Q y
y m a x Q
132 2 yminP ymaxQ
LP LQ

LP LQ
yminP ymaxQ

133 3 p(i)= y m i n P q(j)= y m a x Q (p(i), q(j))

p(i-1),p(i+1) (p(i), q(j))
q(j-1),q(j+1)
(p(i), q(j)) C S
134 4

135 5

136 6 4 5
(yminP,ymaxQ)
137 7 C S
138
139 // / /
140 1
xminP xmaxP yminP ymaxP
141 2 P

142 3

143 4

144 5 /

145 6 4 5
9 0
146 7

2.13 sort - polar angle.cpp

```
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 }
```

2.14 triangle's fermat point

1 1 2 0

2

3 1 2 0

4

5

6 A B C
' B C A ' C A B '
7 C C ' B B ' A A '

Chapter 3

graph

3.1 2-sat.cpp

```
1  #define maxn 2008
2  struct Twosat
3  {
4      int n;
5      std::vector<int>G[maxn*2];
6      bool mark[maxn*2];
7      int s[maxn*2],c;
8
9      bool dfs(int x)
10     {
11         if(mark[x^1])return false;
12         if(mark[x])return true;
13         mark[x]=true;
14         s[c++]=x;
15         for(int i=0;i<G[x].size();++i)
16             if(!dfs(G[x][i]))return false;
17         return true;
18     }
19
20     void init(int n)
21     {
22         this->n=n;
23         for(int i=0;i<n*2;++i)
24             G[i].clear();
25         memset(mark,0,sizeof(mark));
26     }
27     void add_clause(int x,int xval,int y,int yval)//
```

```
28      {
29          x=x*2+xval;
30          y=y*2+yval;
31          G[x^1].push_back(y);
32          G[y^1].push_back(x);
33      }
34
35      bool solve()
36      {
37          for(int i=0;i<n*2;i+=2)
38              if(!mark[i]&&!mark[i+1])
39              {
40                  c=0;
41                  if(!dfs(i))
42                  {
43                      while(c>0)
44                          mark[s[--c]]=false;
45                      if(!dfs(i+1))
46                          return false;
47                  }
48              }
49          return true;
50      }
51  };
```

3.2 Articulation.cpp

```
1   void dfs(int now,int fa)  // now 1
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           if(dfn[*it]==-1)
7           {
8               dfs(*it,now);
9               ++p;
10              low[now]=std::min(low[now],low[*it]);
11              if((now==1 && p>1) || (now!=1 && low[*it]>=
                    dfn[now])) //
12                  ans.insert(now);
13          }
```

```
14          else
15              if(*it!=fa)
16                  low[now]=std::min(low[now],dfn[*it]);
17  }
```

3.3 Augmenting Path Algorithm for Maximum Cardinality Bipartite Matching.cpp

```
1  #include<cstdio>
2  #include<cstring>
3
4  #define MAXX 111
5
6  bool Map[MAXX][MAXX],visit[MAXX];
7  int link[MAXX],n,m;
8  bool dfs(int t)
9  {
10     for (int i=0; i<m; i++)
11         if (!visit[i] && Map[t][i]){
12             visit[i] = true;
13             if (link[i]==-1 || dfs(link[i])){
14                 link[i] = t;
15                 return true;
16             }
17         }
18     return false;
19 }
20 int main()
21 {
22     int k,a,b,c;
23     while (scanf("%d",&n),n){
24         memset(Map,false,sizeof(Map));
25         scanf("%d%d",&m,&k);
26         while (k--){
27             scanf("%d%d%d",&a,&b,&c);
28             if (b && c)
29                 Map[b][c] = true;
30         }
31         memset(link,-1,sizeof(link));
32         int ans = 0;
33         for (int i=0; i<n; i++){
34             memset(visit,false,sizeof(visit));
35             if (dfs(i))
36                 ans++;
37         }
```

```
38          printf("%d\n",ans);
39      }
40  }
```

3.4 best spanning tree.cpp

```
1  #include<cstdio>
2  #include<cstring>
3  #include<cmath>
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%lf",&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)+(
                            node[i].y-node[j].y)*(node[i].y-node[j
                            ].y));
66                      map[j][i].c=map[i][j].c=fabs(node[i].z-
                            node[j].z);
67                  }
68              a=0,b=mst(a);
69              while (fabs(b-a)>1e-8)
70              {
71                  a=b;
72                  b=mst(a);
73              }
74              printf("%.3lf\n",b);
75          }
76          return 0;
77
78      }
```

3.5 Biconnected Component.cpp

```
#include<cstdio>
#include<cstring>
#include<stack>
#include<queue>
#include<algorithm>

const int MAXN=100000*2;
const int MAXM=200000;

//0-based

struct edges
{
    int to,next;
    bool cut,visit;
} edge[MAXM<<1];

int head[MAXN],low[MAXN],dpt[MAXN],L;
bool visit[MAXN],cut[MAXN];
int idx;
std::stack<int> st;
int bcc[MAXM];

void init(int n)
{
    L=0;
    memset(head,-1,4*n);
    memset(visit,0,n);
}

void add_edge(int u,int v)
{
    edge[L].cut=edge[L].visit=false;
    edge[L].to=v;
    edge[L].next=head[u];
    head[u]=L++;
}

void dfs(int u,int fu,int deg)
{
    cut[u]=false;
    visit[u]=true;
    low[u]=dpt[u]=deg;
```

```
44      int tot=0;
45      for (int i=head[u]; i!=-1; i=edge[i].next)
46      {
47          int v=edge[i].to;
48          if (edge[i].visit)
49              continue;
50          st.push(i/2);
51          edge[i].visit=edge[i^1].visit=true;
52          if (visit[v])
53          {
54              low[u]=dpt[v]>low[u]?low[u]:dpt[v];
55              continue;
56          }
57          dfs(v,u,deg+1);
58          edge[i].cut=edge[i^1].cut=(low[v]>dpt[u] || edge[
                i].cut);
59          if (u!=fu) cut[u]=low[v]>=dpt[u]?1:cut[u];
60          if (low[v]>=dpt[u] || u==fu)
61          {
62              while (st.top()!=i/2)
63              {
64                  int x=st.top()*2,y=st.top()*2+1;
65                  bcc[st.top()]=idx;
66                  st.pop();
67              }
68              bcc[i/2]=idx++;
69              st.pop();
70          }
71          low[u]=low[v]>low[u]?low[u]:low[v];
72          tot++;
73      }
74      if (u==fu && tot>1)
75          cut[u]=true;
76  }
77
78  int main()
79  {
80      int n,m;
81      while (scanf("%d%d",&n,&m)!=EOF)
82      {
83          init(n);
84          for (int i=0; i<m; i++)
85          {
86              int u,v;
87              scanf("%d%d",&u,&v);
88              add_edge(u,v);
```

```
89              add_edge(v,u);
90          }
91          idx=0;
92          for (int i=0; i<n; i++)
93              if (!visit[i])
94                  dfs(i,i,0);
95      }
96      return 0;
97  }
```

3.6 Bridge.cpp

```
1   void dfs(const short &now,const short &fa)
2   {
3       dfn[now]=low[now]=cnt++;
4       for(int i(0);i<edge[now].size();++i)
5           if(dfn[edge[now][i]]==-1)
6           {
7               dfs(edge[now][i],now);
8               low[now]=std::min(low[now],low[edge[now][i]])
                    ;
9               if(low[edge[now][i]]>dfn[now]) //

                    ,
10              {
11                  if(edge[now][i]<now)
12                  {
13                      j=edge[now][i];
14                      k=now;
15                  }
16                  else
17                  {
18                      j=now;
19                      k=edge[now][i];
20                  }
21                  ans.push_back(node(j,k));
22              }
23          }
24          else
25              if(edge[now][i]!=fa)
26                  low[now]=std::min(low[now],low[edge[now][
                        i]]);
27  }
```

3.7 chu-liu algorithm.cpp

```
1  #include<cstdio>
2  #include<cstring>
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
26 int zhuliu(int root,int n,int m,Edge e[])
27 {
28     int res = 0,u,v;
29     while (true)
30     {
31         for (int i = 0; i < n; i++)
32             in[i] = inf;
33         for (int i = 0; i < m; i++)
34             if (e[i].u != e[i].v && e[i].cost < in[e[i].v
                ])
35             {
36                 pre[e[i].v] = e[i].u;
37                 in[e[i].v] = e[i].cost;
38             }
39         for (int i = 0; i < n; i++)
40             if (i != root)
41                 if (in[i] == inf)
42                     return -1;
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;)
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      freopen("asdf","r",stdin);
86      while (scanf("%d%d",&n,&m) != EOF)
87      {
88          init(n);
89          for (int i = 0; i < m; i++)
90          {
```

```
91              scanf("%d%d%d",&u,&v,&cost);
92              if (u == v) continue;
93              dis[u][v] = std::min(dis[u][v],cost);
94          }
95          L = 0;
96          for (int i = 0; i < n; i++)
97              for (int j = 0; j < n; j++)
98                  if (dis[i][j] != inf)
99                  {
100                     e[L].u = i;
101                     e[L].v = j;
102                     e[L++].cost = dis[i][j];
103                 }
104         printf("%d\n",zhuliu(0,n,L,e));
105     }
106     return 0;
107 }
```

3.8 k-th shortest path.cpp

```
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     Lr=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     edger[Lr].to=u;
86     edger[Lr].cost=x;
87     edger[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[t]==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%d",&n,&m);
122     init(n);
123     for (int i=0; i<m; i++)
124     {
125         int u,v,x;
126         scanf("%d%d%d",&u,&v,&x);
127         add_edge(u-1,v-1,x);
128     }
129     int s,t;
130     scanf("%d%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 }
```

3.9 Kuhn-Munkres algorithm.cpp

```
1  bool match(int u)//
2  {
3      vx[u]=true;
4      for(int i=1;i<=n;++i)
5          if(lx[u]+ly[i]==g[u][i]&&!vy[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 inline void update()//
17 {
18     int i,j;
19     int a=1<<30;
20     for(i=1;i<=n;++i)if(vx[i])
21         for(j=1;j<=n;++j)if(!vy[j])
22             a=min(a,lx[i]+ly[j]-g[i][j]);
23     for(i=1;i<=n;++i)
```

```
24      {
25          if(vx[i])lx[i]-=a;
26          if(vy[i])ly[i]+=a;
27      }
28  }
29  void km()
30  {
31      int i,j;
32      for(i=1;i<=n;++i)
33      {
34          lx[i]=ly[i]=d[i]=0;
35          for(j=1;j<=n;++j)
36              lx[i]=max(lx[i],g[i][j]);
37      }
38      for(i=1;i<=n;++i)
39      {
40          while(true)
41          {
42              memset(vx,0,sizeof(vx));
43              memset(vy,0,sizeof(vy));
44              if(match(i))
45                  break;
46              update();
47          }
48      }
49      int ans=0;
50      for(i=1;i<=n;++i)
51          if(d[i]!=0)
52              ans+=g[d[i]][i];
53      printf("%d\n",ans);
54  }
55  int main()
56  {
57      while(scanf("%d\n",&n)!=EOF)
58      {
59          for(int i=1;i<=n;++i)gets(s[i]);
60          memset(g,0,sizeof(g));
61          for(int i=1;i<=n;++i)
62              for(int j=1;j<=n;++j)
63                  if(i!=j) g[i][j]=cal(s[i],s[j]);
64          km();
65      }
66      return 0;
67  }
68
69
```

```
70  //bupt
71
72  // k m                                          n^3
73  int dfs(int u)//
74  {
75      int v;
76      sx[u]=1;
77      for ( v=1; v<=n; v++)
78          if (!sy[v] && lx[u]+ly[v]==map[u][v])
79          {
80              sy[v]=1;
81              if (match[v]==-1 || dfs(match[v]))
82              {
83                  match[v]=u;
84                  return 1;
85              }
86          }
87      return 0;
88  }
89
90  int bestmatch(void)// k m
91  {
92      int i,j,u;
93      for (i=1; i<=n; i++)//
94      {
95          lx[i]=-1;
96          ly[i]=0;
97          for (j=1; j<=n; j++)
98              if (lx[i]<map[i][j])
99                  lx[i]=map[i][j];
100     }
101     memset(match, -1, sizeof(match));
102     for (u=1; u<=n; u++)
103     {
104         while (true)
105         {
106             memset(sx,0,sizeof(sx));
107             memset(sy,0,sizeof(sy));
108             if (dfs(u))
109                 break;
110             int dx=Inf;//
                                                                    ~~
111             for (i=1; i<=n; i++)
112             {
113                 if (sx[i])
114                     for (j=1; j<=n; j++)
```

```
115                             if(!sy[j] && dx>lx[i]+ly[j]-map[i
                                    ][j])
116                                 dx=lx[i]+ly[j]-map[i][j];
117                 }
118                 for (i=1; i<=n; i++)
119                 {
120                     if (sx[i])
121                         lx[i]-=dx;
122                     if (sy[i])
123                         ly[i]+=dx;
124                 }
125             }
126         }
127         int sum=0;
128         for (i=1; i<=n; i++)
129             sum+=map[match[i]][i];
130         return sum;
131     }
```

3.10 LCA - DA.cpp

```
1   int edge[MAXX],nxt[MAXX<<1],to[MAXX<<1],cnt;
2   int pre[MAXX][N],dg[MAXX];
3
4   inline void add(int j,int k)
5   {
6       nxt[++cnt]=edge[j];
7       edge[j]=cnt;
8       to[cnt]=k;
9   }
10
11  void rr(int now,int fa)
12  {
13      dg[now]=dg[fa]+1;
14      for(int i(edge[now]);i;i=nxt[i])
15          if(to[i]!=fa)
16          {
17              static int j;
18              j=1;
19              for(pre[to[i]][0]=now;j<N;++j)
20                  pre[to[i]][j]=pre[pre[to[i]][j-1]][j-1];
21              rr(to[i],now);
22          }
23  }
24
25  inline int lca(int a,int b)
```

```
26 {
27     static int i,j;
28     j=0;
29     if(dg[a]<dg[b])
30         std::swap(a,b);
31     for(i=dg[a]-dg[b];i;i>>=1,++j)
32         if(i&1)
33             a=pre[a][j];
34     if(a==b)
35         return a;
36     for(i=N-1;i>=0;--i)
37         if(pre[a][i]!=pre[b][i])
38         {
39             a=pre[a][i];
40             b=pre[b][i];
41         }
42     return pre[a][0];
43 }
```

3.11 LCA - tarjan - minmax.cpp

```
1  #include<cstdio>
2  #include<list>
3  #include<algorithm>
4  #include<cstring>
5
6  #define MAXX 100111
7  #define inf 0x5fffffff
8
9  short T,t;
10 int set[MAXX],min[MAXX],max[MAXX],ans[2][MAXX];
11 bool done[MAXX];
12 std::list<std::pair<int,int> >edge[MAXX];
13 std::list<std::pair<int,int> >q[MAXX];
14 int n,i,j,k,l,m;
15
16 struct node
17 {
18     int a,b,id;
19     node() {}
20     node(const int &aa,const int &bb,const int &idd): a(
           aa),b(bb),id(idd){}
21 };
22
23 std::list<node>to[MAXX];
24
```

```
25 int find(const int &a)
26 {
27     if(set[a]==a)
28         return a;
29     int b(set[a]);
30     set[a]=find(set[a]);
31     max[a]=std::max(max[a],max[b]);
32     min[a]=std::min(min[a],min[b]);
33     return set[a];
34 }
35
36 void tarjan(const int &now)
37 {
38     done[now]=true;
39     for(std::list<std::pair<int,int> >::const_iterator it
           (q[now].begin());it!=q[now].end();++it)
40         if(done[it->first])
41             if(it->second>0)
42                 to[find(it->first)].push_back(node(now,it
                       ->first,it->second));
43             else
44                 to[find(it->first)].push_back(node(it->
                       first,now,-it->second));
45     for(std::list<std::pair<int,int> >::const_iterator it
           (edge[now].begin());it!=edge[now].end();++it)
46         if(!done[it->first])
47         {
48             tarjan(it->first);
49             set[it->first]=now;
50             min[it->first]=it->second;
51             max[it->first]=it->second;
52         }
53     for(std::list<node>::const_iterator it(to[now].begin
           ());it!=to[now].end();++it)
54     {
55         find(it->a);
56         find(it->b);
57         ans[0][it->id]=std::min(min[it->b],min[it->a]);
58         ans[1][it->id]=std::max(max[it->a],max[it->b]);
59     }
60 }
61
62 int main()
63 {
64     scanf("%hd",&T);
65     for(t=1;t<=T;++t)
```

```
66      {
67          scanf("%d",&n);
68          for(i=1;i<=n;++i)
69          {
70              edge[i].clear();
71              q[i].clear();
72              to[i].clear();
73              done[i]=false;
74              set[i]=i;
75              min[i]=inf;
76              max[i]=0;
77          }
78          for(i=1;i<n;++i)
79          {
80              scanf("%d%d%d",&j,&k,&l);
81              edge[j].push_back(std::make_pair(k,l));
82              edge[k].push_back(std::make_pair(j,l));
83          }
84          scanf("%d",&m);
85          for(i=0;i<m;++i)
86          {
87              scanf("%d %d",&j,&k);
88              q[j].push_back(std::make_pair(k,i));
89              q[k].push_back(std::make_pair(j,-i));
90          }
91          tarjan(1);
92          printf("Case %hd:\n",t);
93          for(i=0;i<m;++i)
94              printf("%d %d\n",ans[0][i],ans[1][i]);
95      }
96      return 0;
97  }
```

3.12 Minimum Cost Maximum Flow.cpp

```
1   struct mcmf
2   {
3       struct Edge
4       {
5           int from,to,cap,flow,cost;
6       };
7       int n,m,s,t;
8       std::vector<Edge>edges;
9       std::vector<int>G[maxn];
10      int inq[maxn],d[maxn],p[maxn],a[maxn];
11
```

```
12      void init(int n)
13      {
14          this->n=n;
15          for(int i=0;i<n;++i)
16              G[i].clear();
17          edges.clear();
18      }
19
20      void addedge(int from,int to,int cap,int cost)
21      {
22          Edge x={from,to,cap,0,cost};
23          edges.push_back(x);
24          Edge y={to,from,0,0,-cost};
25          edges.push_back(y);
26          m=edges.size();
27          G[from].push_back(m-2);
28          G[to].push_back(m-1);
29      }
30      int mincost(int s,int t)
31      {
32          int flow=0,cost=0;
33          while(BellmanFord(s,t,flow,cost));
34          if(flow!=(n-1)/2)return -1;
35          return cost;
36      }
37  private:
38      bool BellmanFord(int s,int t,int& flow,int& cost)
39      {
40          for(int i=0;i<=n;++i)
41              d[i]=INF;
42          memset(inq,0,sizeof(inq));
43          d[s]=0; inq[s]=1; p[s]=0; a[s]=INF;
44          std::queue<int>Q;
45          Q.push(s);
46          while(!Q.empty())
47          {
48              int u=Q.front();
49              Q.pop();
50              inq[u]=0;
51              for(int i=0;i<G[u].size();++i)
52              {
53                  Edge& e=edges[G[u][i]];
54                  if(e.cap>e.flow && d[e.to]>d[u]+e.cost)
55                  {
56                      d[e.to]=d[u]+e.cost;
57                      p[e.to]=G[u][i];
```

```
58                          a[e.to]=min(a[u],e.cap-e.flow);
59                          if(!inq[e.to])
60                          {
61                              Q.push(e.to);
62                              inq[e.to]=1;
63                          }
64                      }
65                  }
66              }
67              if(d[t]==INF)
68                  return false;
69              flow+=a[t];
70              cost+=d[t]*a[t];
71              int u=t;
72              while(u!=s)
73              {
74                  edges[p[u]].flow+=a[t];
75                  edges[p[u]^1].flow-=a[t];
76                  u=edges[p[u]].from;
77              }
78              return true;
79          }
80
81      }G;
```

3.13 minimum cut.cpp

```
1   #include <iostream>
2   using namespace std;
3   const int maxn=510;
4   int map[maxn][maxn];
5   int n;
6   void contract(int x,int y)//
7   {
8       int i,j;
9        for (i=0; i<n; i++)
10       if (i!=x) map[x][i]+=map[y][i],map[i][x]+=map[i][y];
11       for (i=y+1; i<n; i++) for (j=0; j<n; j++)
12       {
13           map[i-1][j]=map[i][j];
14           map[j][i-1]=map[j][i];
15       }
16       n--;
17  }
18  int w[maxn],c[maxn];
19  int sx,tx;
```

```
20 int mincut()
21 //


22 {
23     int i,j,k,t;
24     memset(c,0,sizeof(c));
25     c[0]=1;
26     for (i=0; i<n; i++) w[i]=map[0][i];
27     for (i=1; i+1<n; i++)
28     {
29         t=k=-1;
30         for (j=0; j<n; j++) if (c[j]==0&&w[j]>k)
31         k=w[t=j];
32         c[sx=t]=1;
33         for (j=0; j<n; j++) w[j]+=map[t][j];
34     }
35     for (i=0; i<n; i++) if (c[i]==0) return w[tx=i];
36 }
37 int main()
38 {
39     int i,j,k,m;
40     while (scanf("%d%d",&n,&m)!=EOF)
41     {
42         memset(map,0,sizeof(map));
43         while (m--)
44         {
45             scanf("%d%d%d",&i,&j,&k);
46             map[i][j]+=k;
47             map[j][i]+=k;
48         }
49         int mint=999999999;
50         while (n>1)
51         {
52             k=mincut();
53             if (k<mint) mint=k;
54             contract(sx,tx);
55         }
56     printf("%d\n",mint);
57     }
58 return 0;
59 }
```

3.14 others

```
1
```

2
3
,
.
, .
4
5
,
s s s
6
7
8
9
G D D
, v u ,
v G =
G = (V;E)
U V u , v U
< u; v >
E U G G U G
; v U < u; v >
E U G G U G
10
11
12 + = V
13 =
14 =

3.15 Shortest Augmenting Path algorithm.cpp

```
1 #include <cstring>
2 #include <cstdio>
3 #include <vector>
4 #include <queue>
5 #define maxn 1005
6 #define INF 1<<30
7 using namespace std;
8 struct Edge
```

```
 9 {
10         int from,to,cap,flow;
11 };
12 vector<Edge>edges;
13 vector<int>G[maxn];
14 int num[maxn],p[maxn],n,m;
15 int st[maxn],et[maxn],nt[maxn];
16 int d[maxn],s,t,cur[maxn];
17 void addedge(int from,int to,int cap)
18 {
19     struct Edge x={from,to,cap,0};
20         edges.push_back(x);
21         struct Edge y={to,from,0,0};
22         edges.push_back(y);
23         int m=edges.size();
24         G[from].push_back(m-2);
25         G[to].push_back(m-1);
26 }
27 void bfs()//
             B F S
28 {
29         queue<int>q;
30         memset(d,0,sizeof(d));
31         d[t]=1;
32         q.push(t);
33         while(!q.empty())
34         {
35                 int u=q.front(); q.pop();
36                 for(int i=0;i<G[u].size();++i)
37                 if(G[u][i]&1)
38                 {
39                         Edge& e=edges[G[u][i]];
40                         if(!d[e.to])
41                         {
42                                 d[e.to]=d[u]+1;
43                                 q.push(e.to);
44                         }
45                 }
46         }
47 }
48 int augment()//
49 {
50         int u=t, a=INF;
51         while(u!=s)
52         {
53                 Edge& e=edges[p[u]];
```

```
54                      a=min(a,e.cap-e.flow);//


55                      u=e.from;
56              }
57              u=t;
58              while(u!=s)
59              {
60                      edges[p[u]].flow+=a;
61                      edges[p[u]^1].flow-=a;
62                      u=edges[p[u]].from;
63              }
64              return a;
65      }
66      int sap()
67      {
68              int flow=0;
69              bfs();
70              memset(num,0,sizeof(num));
71              for(int i=0;i<=t;++i)num[d[i]]++;
72              int u=s;
73              memset(cur,0,sizeof(cur));
74              while(d[s]<t)
75              {
76                      if(u==t)
77                      {
78                              flow+=augment();
79                              u=s;
80                      }
81                      int ok=0;
82                      for(int i=cur[u];i<G[u].size();++i)
83                      {
84                              Edge& e=edges[G[u][i]];
85                              if(e.cap>e.flow && d[u]==d[e.to
                                    ]+1)
86                              {
87                                      ok=1;
88                                      p[e.to]=G[u][i];
89                                      cur[u]=i;
90                                      u=e.to;
91                                      break;
92                              }
93                      }
94                      if(!ok)//
```

```
95                      {
96                              int m=t-1;
97                              for(int i=0;i<G[u].size();++i)
98                              {
99                                      Edge& e=edges[G[u][i]];
100                                     if(e.cap>e.flow) m=min(m,
                                            d[e.to]);
101                             }
102                             if(--num[d[u]]==0)break;//
                                    gap
103                             num[d[u]=m+1]++;
104                             cur[u]=0;//


105                             if(u!=s)u=edges[p[u]].from;
106                     }
107             }
108             return flow;
109     }
110     void init()
111     {
112      edges.clear();
113      for(int i=0;i<maxn;++i)G[i].clear();
114     }
115     int main()
116     {
117         int T, i;
118       //      freopen("1.txt","r",stdin);
119         scanf("%d", &T);
120         while (T--)
121         {
122             scanf("%d%d", &n, &m);
123             s=0;   t=-1;
124             int res=0;
125             init();
126             for(i=1;i<=n;++i)
127             {
128                 scanf("%d%d%d",&st[i],&et[i],&nt[i]);
129                 res+=nt[i];
130                 if(et[i]+n+1>t)t=et[i]+n+1;
131             }
132             for (i=1; i<=n; ++i)
133             {
134                 int j;
135                 addedge(s,i,nt[i]);
136                 for(j=st[i];j<=et[i];++j) addedge(i,j+n,1);
```

```
137             }
138             for(i=n+1;i<t;++i) addedge(i,t,m);
139             int ans = sap();
140             if(ans==res)printf("Yaha, Garfield, You Finish it
                    !\n");
141             else printf("Oh~~ Garfield, You are just as before
                    !\n");
142         }
143             return 0;
144 }
```

3.16 Stable Marriage.cpp

```
1  //



2
3  while(!g.empty())   //
4  {
5      if(dfn[edge[g.front()].front()]==-1)
6          dfn[edge[g.front()].front()]=g.front(); //


7      else
8      {
9          for(it=edge[edge[g.front()].front()].begin();it!=
               edge[edge[g.front()].front()].end();++it)
10             if(*it==dfn[edge[g.front()].front()] || *it==
                   g.front()) //


11                 break;
12         if(*it==g.front()) //
13         {
14             g.push_back(dfn[edge[g.front()].front()]);
15             dfn[edge[g.front()].front()]=g.front();
16         }
17         else
18             g.push_back(g.front()); //

19     }
20     edge[g.front()].pop_front(); //

21     g.pop_front();
22 }
```

3.17 Strongly Connected Component.cpp

```
1  void dfs(const short &now)
2  {
3      dfn[now]=low[now]=cnt++;
4      st.push(now);
5      for(std::list<short>::const_iterator it(edge[now].
           begin());it!=edge[now].end();++it)
6          if(dfn[*it]==-1)
7          {
8              dfs(*it);
9              low[now]=std::min(low[now],low[*it]);
10         }
11         else
12             if(sc[*it]==-1)
13                 low[now]=std::min(low[now],dfn[*it]);
14     if(dfn[now]==low[now])
15     {
16         while(sc[now]==-1)
17         {
18             sc[st.top()]=p;
19             st.pop();
20         }
21         ++p;
22     }
23 }
```

Chapter 4

math

4.1 cantor.cpp

```
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  }
```

4.2 combinations.cpp

```
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; 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)//‘ m o d                ‘
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 (n%p>=m%p)
52              ans = ans*num[n%p]%p *getInv(num[m%p]%p)%p *
                    getInv(num[n%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%d",&n,&m,&p);
67          printf("%lld\n",calc(n+m,m,p));
68      }
69      return 0;
70  }
```

4.3 euler's totient function.cpp

1 n n n
(x)=x(1-1/p1)(1-1/p2)(1-1/p3)(1-1/p4) ..(1-1/pn),
p 1 ,
p 2 p n x x 0
(1)=1 1 1 m ,

```
    n            (mn)=  (m)   (n)
2
3   int Euler(int n)
4   {
5       int ans = n;
6       for (int i=2; i<=sqrt(n); i++)
7       {
8           if (n%i==0)
9           {
10              ans = ans-ans/i;
11              while (n%i==0)
12                  n /= i;
13          }
14      }
15      if (n>1)
16          ans = ans-ans/n;
17      return ans;
18  }
19
20  //
21  inline void Euler2()
22  {
23      memset(euler,0,sizeof(euler));
24      euler[1] = 1;
25      for (int i = 2; i <= 3000000; i++)
26      {
27          if (!euler[i])
28          {
29              for (int j = i; j <= 3000000; j += i)
30              {
31                  if (!euler[j])
32                      euler[j] = j;
33                  euler[j] = euler[j]/i*(i-1);
34              }
35          }
36      }
37  }
```

4.4 extended euclidean algorithm.cpp

```
1   //      a x +by=gcd(a,b)
2   long long ex_gcd(long long a,long long b,long long &x,
        long long &y)
3   {
4       if (b)
5       {
```

```
6           long long ret = ex_gcd(b,a%b,x,y),tmp = x;
7           x = y;
8           y = tmp-(a/b)*y;
9           return ret;
10      }
11      else
12      {
13          x = 1;
14          y = 0;
15          return a;
16      }
17  }
```

4.5 inverse element.cpp

```
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; 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)// m o d
18 {
19     return power(x,mod-2);
20 }
```

4.6 lucas.cpp

```
1  #include <cstdio>
2  /*
3  Lucas              C (n,m)%p
4  */
5  void gcd(int n,int k,int &x,int &y)
6  {
7          if(k==0){x=1;y=0;return ;}
```

```
 8         else
 9         {
10                 gcd(k,n%k,x,y);
11                 int t=x;x=y;
12                 y=t-(n/k)*y;
13                 return;
14         }
15 }
16
17 int CmodP(int n,int k,int p)
18 {
19         if(k>n) return 0;
20         int a,b,flag=0,x,y;
21         a=b=1;
22         for(int i=1;i<=k;i++)
23         {
24                 x=n-i+1,y=i;
25                 while(x%p==0) x/=p,flag++;
26                 while(y%p==0) y/=p,flag--;
27                 x%=p,y%=p,a*=x,b*=y;
28                 b%=p,a%=p;
29         }
30         if(flag) return 0;
31         gcd(b,p,x,y);
32         if(x<0) x+=p;
33         a*=x,a%=p;
34         return a;
35 }
36
37 // Lucas                C(n,m) % p ,p
38 long long Lucas(long long n, long long m, long long p)
39 {
40       long long ans=1;
41       long long a,b;
42       while(m&&n&&ans)
43       {
44            ans*=(CmodP(n%p,m%p,p));
45            ans=ans%p;
46            n=n/p;
47            m=m/p;
48       }
49       return ans;
50 }
51 int main()
52 {
53     long long n,k,p,ans;
```

```
54      int cas=0;
55  //    freopen("1.txt","r",stdin);
56  //    freopen("out2.txt","w",stdout);
57      while(scanf("%I64d%I64d%I64d",&n,&k,&p)!=EOF)
58      {
59          ++cas;
60          if(k>n-k)k=n-k;
61          ans=Lucas(n+1,k,p)+n-k;
62          printf("Case #%d: %I64d\n",cas,ans%p);
63      }
64      return 0;
65  }
```

4.7 matrix.cpp

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

4.8 miller rabin.cpp

```
1  inline unsigned long long multi_mod(const unsigned long
       long &a,unsigned long long b,const unsigned long long
       &n)
2  {
3       unsigned long long exp(a%n),tmp(0);
4       while(b)
5       {
6            if(b&1)
7            {
8                 tmp+=exp;
9                 if(tmp>n)
10                     tmp-=n;
11           }
12           exp<<=1;
13           if(exp>n)
14                exp-=n;
15           b>>=1;
16      }
17      return tmp;
18 }
19
20 inline unsigned long long exp_mod(unsigned long long a,
       unsigned long long b,const unsigned long long &c)
21 {
22      unsigned long long tmp(1);
23      while(b)
24      {
25           if(b&1)
26                tmp=multi_mod(tmp,a,c);
27           a=multi_mod(a,a,c);
28           b>>=1;
29      }
30      return tmp;
31 }
32
33 inline bool miller_rabbin(const unsigned long long &n,
       short T)
34 {
35      if(n==2)
36           return true;
37      if(n<2 || !(n&1))
38           return false;
39      unsigned long long a,u(n-1),x,y;
```

```
40      short t(0),i;
41      while(!(u&1))
42      {
43          ++t;
44          u>>=1;
45      }
46      while(T--)
47      {
48          a=rand()%(n-1)+1;
49          x=exp_mod(a,u,n);
50          for(i=0;i<t;++i)
51          {
52              y=multi_mod(x,x,n);
53              if(y==1 && x!=1 && x!=n-1)
54                  return false;
55              x=y;
56          }
57          if(y!=1)
58              return false;
59      }
60      return true;
61  }
```

4.9 mod

```
1  (a+b)%m = (a%m+b%m)%m
2  (a-b)%m = ((a%m-b%m)%m+m)%m (a>b)
3  (a*b)%m = (a%m*b%m)%m
4
5  (a/b)%m = (a%(b*m))/b //
```

4.10 pollard rho.cpp

```
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 unsigned long long
       &n)
10 {
```

```
11      unsigned long long exp(a%n),tmp(0);
12      while(b)
13      {
14          if(b&1)
15          {
16              tmp+=exp;
17              if(tmp>n)
18                  tmp-=n;
19          }
20          exp<<=1;
21          if(exp>n)
22              exp-=n;
23          b>>=1;
24      }
25      return tmp;
26  }
27
28  inline unsigned long long exp_mod(unsigned long long a,
        unsigned long long b,const unsigned long long &c)
29  {
30      unsigned long long tmp(1);
31      while(b)
32      {
33          if(b&1)
34              tmp=multi_mod(tmp,a,c);
35          a=multi_mod(a,a,c);
36          b>>=1;
37      }
38      return tmp;
39  }
40
41  inline bool miller_rabbin(const unsigned long long &n,
        short T)
42  {
43      if(n==2)
44          return true;
45      if(n<2 || !(n&1))
46          return false;
47      unsigned long long a,u(n-1),x,y;
48      short t(0),i;
49      while(!(u&1))
50      {
51          ++t;
52          u>>=1;
53      }
54      while(T--)
```

```
55      {
56          a=rand()%(n-1)+1;
57          x=exp_mod(a,u,n);
58          for(i=0;i<t;++i)
59          {
60              y=multi_mod(x,x,n);
61              if(y==1 && x!=1 && x!=n-1)
62                  return false;
63              x=y;
64          }
65          if(y!=1)
66              return false;
67      }
68      return true;
69  }
70
71  unsigned long long gcd(const unsigned long long &a,const
        unsigned long long &b)
72  {
73      return b?gcd(b,a%b):a;
74  }
75
76  inline unsigned long long pollar_rho(const unsigned long
        long n,const unsigned long long &c)
77  {
78      unsigned long long x(rand()%(n-1)+1),y,d,i(1),k(2);
79      y=x;
80      while(true)
81      {
82          ++i;
83          x=(multi_mod(x,x,n)+c)%n;
84          d=gcd((x-y+n)%n,n);
85          if(d>1 && d<n)
86              return d;
87          if(x==y)
88              return n;
89          if(i==k)
90          {
91              k<<=1;
92              y=x;
93          }
94      }
95  }
96
97  void find(const unsigned long long &n,short c)
98  {
```

```
 99      if(n==1)
100          return;
101      if(miller_rabbin(n,6))
102      {
103          fac.push_back(n);
104          return;
105      }
106      unsigned long long p(n);
107      short k(c);
108      while(p>=n)
109          p=pollar_rho(p,c--);
110      find(p,k);
111      find(n/p,k);
112  }
113
114  int main()
115  {
116      scanf("%hd",&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  }
```

4.11 prime.cpp

```
 1  #include<vector>
 2
 3  std::vector<int>prm;
 4  bool flag[MAXX];
 5
 6  int main()
 7  {
 8      prm.reserve(MAXX); // pi(x)=x/ln(x);
 9      for(i=2;i<MAXX;++i)
10      {
```

```
11              if(!flag[i])
12                  prm.push_back(i);
13              for(j=0;j<prm.size() && i*prm[j]<MAXX;++j)
14              {
15                  flag[i*prm[j]]=true;
16                  if(i%pmr[j]==0)
17                      break;
18              }
19          }
20          return 0;
21      }
```

Chapter 5

others

5.1 .vimrc

```
set number
set history=1000000
set autoindent
set smartindent
set tabstop=4
set shiftwidth=4
set expandtab
set showmatch

set nocp
filetype plugin indent on

filetype on
syntax on
```

5.2 bigint.cpp

```
// header files
#include <cstdio>
#include <string>
#include <algorithm>
#include <iostream>

struct Bigint
{
    // representations and structures
    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.a.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 
146 
147 int main()
148 {
149     Bigint a, b, c; // declared some Bigint variables
150     ////////////////////////
151     // taking Bigint input //
152     ////////////////////////
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     ////////////////////////////////
162     // Using mathematical operators //
163     ////////////////////////////////
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     ////////////////////////////////
186     // Using conditional operators //
187     ////////////////////////////////
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 }
```

5.3 java.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.radix(); //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 preffix);
31 boolean         str.startsWith(String preffix,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(); setBig(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();
```

5.4 others

```
1  chmod +x [filename]
2
3  while true; do
4  ./gen > input
5  ./sol < input > output.sol
6  ./bf < input > output.bf
7
8  diff output.sol output.bf
9  if[ $? -ne 0];then break fi
10 done
11
12
13 1


14 2 calm_down();calm_down();calm_down();
15 3
16 4
17 5          30 min                TM                ,
```

Chapter 6

search

6.1 dlx

```
1                                   0 1

2

3
4                                   0 1

5
```

6.2 dlx - precise cover.cpp

```
1  #include<cstdio>
2
3  #define INF 0x7FFFFFFF
4  #define MAXN 1000010
5
6  int n, m, size;
7  int L[MAXN], R[MAXN], U[MAXN], D[MAXN], H[MAXN];
8  int S[MAXN], C[MAXN], X[MAXN], Q[MAXN];
9
10 void Init()
11 {
12     int i;
13     for (i = 0; i <= m; i++)
14     {
15         S[i] = 0;
16         L[i + 1] = i;
```

```
17          R[i] = i + 1;
18          U[i] = D[i] = i;
19      }
20      R[m] = 0;
21      size = m + 1;
22  }
23  void Remove(int c)
24  {
25      int i, j;
26      R[L[c]] = R[c];
27      L[R[c]] = L[c];
28      for (i = D[c]; i != c; i = D[i])
29      {
30          for (j = R[i]; j != i; j = R[j])
31          {
32              D[U[j]] = D[j];
33              U[D[j]] = U[j];
34              S[C[j]]--;
35          }
36      }
37  }
38  void Resume(int c)
39  {
40      int i, j;
41      R[L[c]] = c;
42      L[R[c]] = c;
43      for (i = D[c]; i != c; i = D[i])
44      {
45          for (j = R[i]; j != i; j = R[j])
46          {
47              U[D[j]] = j;
48              D[U[j]] = j;
49              S[C[j]]++;
50          }
51      }
52  }
53  void Link(int r, int c)
54  {
55      D[size] = D[c];
56      U[size] = c;
57      U[D[c]] = size;
58      D[c] = size;
59      if (H[r] < 0)
60          H[r] = L[size] = R[size] = size;
61      else
62      {
```

```
63          L[size] = H[r];
64          R[size] = R[H[r]];
65          L[R[H[r]]] = size;
66          R[H[r]] = size;
67      }
68      S[c]++;
69      C[size] = c;
70      X[size++] = r;
71  }
72  bool Dance(int now)
73  {
74      int i, j, c, temp;
75      if (R[0] == 0)
76          return true;
77      for (temp = INF, i = R[0]; i; i = R[i])
78      {
79          if (S[i] < temp)
80          {
81              c = i;
82              temp = S[i];
83          }
84      }
85      Remove(c);
86      for (i = D[c]; i != c; i = D[i])
87      {
88          for (j = R[i]; j != i; j = R[j])
89              Remove(C[j]);
90          if (Dance(now + 1))
91              return true;
92          for (j = L[i]; j != i; j = L[j])
93              Resume(C[j]);
94      }
95      Resume(c);
96      return false;
97  }
```

6.3 dlx - repeat cover.cpp

```
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[MAXM], R[MAXM], U[MAXM], D[MAXM];
13  int size, ans, S[MAXM], H[MAXM], C[MAXM];
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; 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; 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     }
```

6.4 fibonacci knapsack.cpp

```
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("%hd",&T);
48      for(t=1;t<=T;++t)
49      {
50          scanf("%hd%lld",&n,&carry);
51          sumw=0;
52          sumc=0;
53          ans=0;
54          for(i=0;i<n;++i)
55          {
56              scanf("%lld%lld",&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 %hd: %lld\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("%lld %lld\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 %hd: %lld\n",t,ans);
75      }
76      return 0;
77  }
```

Chapter 7

string

7.1 aho corasick.cpp

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

```
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];
49     }
50 }
```

7.2 manacher.cpp

```
1  #include<cstdio>
2  #include<vector>
3
4  #define MAXX 1111
5
6  std::vector<char>str;
7  char buf[MAXX];
8  int z[MAXX<<1];
9  int i,j,l,r;
10 int ii,n,c;
11
12 inline int match(const int &a,const int &b)
13 {
14     int i(0);
15     while(a-i>=0 && b+i<str.size() && str[a-i]==str[b+i])
16         ++i;
17     return i;
18 }
19
20 int main()
21 {
```

```
22      gets(buf);
23      str.reserve(MAXX<<1);
24      for(i=0;buf[i];++i)
25      {
26          str.push_back('$');
27          str.push_back(buf[i]);
28      }
29      str.push_back('$');
30
31      z[0]=1;
32      c=l=r=0;
33      for(i=1;i<str.size();++i)
34      {
35          ii=(l<<1)-i;
36          n=r+1-i;
37
38          if(i>r)
39          {
40              z[i]=match(i,i);
41              l=i;
42              r=i+z[i]-1;
43          }
44          else
45              if(z[ii]==n)
46              {
47                  z[i]=n+match(i-n,i+n);
48                  l=i;
49                  r=i+z[i]-1;
50              }
51              else
52                  z[i]=std::min(z[ii],n);
53          if(z[i]>z[c])
54              c=i;
55      }
56
57      for(i=c-z[c]+2,n=c+z[c];i<n;i+=2)
58          putchar(str[i]);
59      puts("");
60      return 0;
61  }
```

7.3 morris-pratt.cpp

```
1  int i,j;
2
3  inline void make(char *buf,int *fal)
```

```
4  {
5      fal[0]=-1;
6      for(i=1,j=-1;buf[i];++i)
7      {
8          while(j>=0 && buf[j+1]!=buf[i])
9              j=fal[j];
10         if(buf[j+1]==buf[i])
11             ++j;
12         fal[i]=j;
13     }
14
15 }
16
17 inline int void match(char *p,char *t,int* fal)
18 {
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             //
28             j=fal[j];
29         }
30 }
```

7.4 smallest representation.cpp

```
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  }
```

7.5 suffix array - da.cpp

```
1   int wx[maxn],wy[maxn],*x,*y,wss[maxn],wv[maxn];
2
3   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 rank[],int height[],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<m; i++)
13          wss[i]=0;
14      for(i=0; i<n; i++)
15          wss[x[i]=s[i]]++;
16      for(i=1; i<m; 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<m; i++)
30              wss[i]=0;
31          for(i=0; i<n; i++)
32              wss[wv[i]]++;
33          for(i=1; i<m; 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      for(int i=0; i<n; i++)
41          rank[sa[i]]=i;
42      for(int i=0,j=0,k=0; i<n; height[rank[i++]]=k)
43          if(rank[i]>0)
44              for(k?k--:0,j=sa[rank[i]-1]; i+k < n && j+k <
                    n && str[i+k]==str[j+k]; ++k);
45  }
```

7.6 suffix array.cpp

```
1  #include<cstdio>
2  #include<cstring>
3  #include<algorithm>
4
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+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<m;++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[0])]=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(n%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])]=i;
62      for(i=j=k=0;i<ta && j<tbc;)
63          sa[k++]=c12(str,wb[j]%3,wa[i],wb[j])?wa[i++]:wb[j
                ++];
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]; //      int
72
73  int main()
74  {
```

```
75      scanf("%d %d",&n,&j);
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: str
85
86      for(i=1;i<=n;++i) // rank
87          rk[sa[i]]=i;
88      for(i=k=0;i<n;++i) // 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=n+1-(1<<i);
107         for(j=1;j<=k;++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) //              [l,r]
            r m q
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 }
```

7.7 z algorithm.cpp

```
 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
24 for(i=1;i<len && i+z[i]<len;++i); //i=
```