Algorithms

Algorithms by Yu Dongfeng
First version on November 28, 2015
Latest version on February 11, 2016

Contents

CHAPTER 1

Computational Geometry

1.1   Convex Hull

Convex Hull.hpp (1063 bytes, 36 lines)

```cpp
#include<bits/stdc++.h>
using namespace std;
template<class T>struct ConvexHull{
    struct point{
        T x,y;
        point(T _x,T _y):
            x(_x),y(_y){
        }
        point operator-(point a){
            return point(x-a.x,y-a.y);
        }
        T operator*(point a){
            return x*a.y-y*a.x;
        }
        int operator<(point a){
            return x==a.x?y<a.y:x<a.x;
        }
    };
    static int check(point a,point b,point c){
        return (a-c)*(b-c)<=0;
    }
    static vector<vector<point> >run(vector<point>a){
        sort(a.begin(),a.end());
        vector<point>u,d;
        for(int i=0;i<a.size();u.push_back(a[i++]))
            while(u.size()>1&&check(a[i],u.back(),u[u.size()-2]))
                u.pop_back();
        for(int i=int(a.size()-1);i>=0;d.push_back(a[i--]))
            while(d.size()>1&&check(a[i],d.back(),d[d.size()-2]))
                d.pop_back();
        vector<vector<point> >r;
        r.push_back(u);
        r.push_back(d);
        return r;
    }
};
```

1.2 Delaunay Triangulation

Delaunay Triangulation.hpp (4889 bytes, 159 lines)

```cpp
#include<bits/stdc++.h>
using namespace std;
template<class T>struct DelaunayTriangulation{
    const static double E;
    struct poi{
        T x,y;
        poi(T _x=0,T _y=0):
            x(_x),y(_y){
        }
        poi operator-(poi b){
            return poi(x-b.x,y-b.y);
        }
        int operator<(poi b)const{
            if(fabs(x-b.x)<E)
                return y<b.y;
            return x<b.x;
        }
    };
    int n;
    vector<pair<poi,int> >pts;
    vector<vector<int> >egs;
    T det(poi a,poi b){
        return a.x*b.y-a.y*b.x;
    }
    T dot(poi a,poi b){
        return a.x*b.x+a.y*b.y;
    }
    int dir(poi a,poi b,poi c){
        T r=det(c-a,b-a);
        if(r<-E)
            return -1;
        return r>E?1:0;
    }
    int inc(poi a,poi b,poi c,poi d){
        a=a-d;
        b=b-d;
        c=c-d;
```

```
38          T az=a.x*a.x+a.y*a.y,bz=b.x*b.x+b.y*b.y,cz=c.x*c.x+c.y*c.y;
39          return a.x*b.y*cz+b.x*c.y*az+c.x*a.y*bz−a.x*bz*c.y−b.x*a.y*cz−c.x*
     b.y*az>E;
40      }
41      int crs(poi a,poi b,poi c,poi d){
42          return dir(a,b,c)*dir(a,b,d)==−1&&dir(c,d,a)*dir(c,d,b)==−1;
43      }
44      DelaunayTriangulation():
45          n(0),pts(1){
46      }
47      void add(T x,T y){
48          poi a;
49          a.x=x;
50          a.y=y;
51          pts.push_back(make_pair(a,++n));
52      }
53      poi&pot(int a){
54          return pts[a].first;
55      }
56      void con(int a,int b){
57          egs[a].push_back(b);
58          egs[b].push_back(a);
59      }
60      void dco(int a,int b){
61          egs[a].erase(find(egs[a].begin(),egs[a].end(),b));
62          egs[b].erase(find(egs[b].begin(),egs[b].end(),a));
63      }
64      void dnc(int l,int r){
65          if(r==l)
66              return;
67          if(r==l+1){
68              con(l,r);
69              return;
70          }
71          if(r==l+2){
72              if(dir(pot(l),pot(l+1),pot(r)))
73                  con(l,l+1),con(l+1,r),con(l,r);
74              else{
75                  if(dot(pot(l+1)−pot(l),pot(r)−pot(l))<0)
76                      con(l,l+1),con(l,r);
77                  else if(dot(pot(l)−pot(l+1),pot(r)−pot(l+1))<0)
```

```
78                      con(l,l+1),con(l+1,r);
79                  else
80                      con(l,r),con(l+1,r);}
81              return;
82          }
83          int m=(l+r)/2,pl=l,pr=r;
84          dnc(l,m);
85          dnc(m+1,r);
86          for(int f=0;;f=0){
87              for(int i=0;i<egs[pl].size();++i){
88                  int a=egs[pl][i],d=dir(pot(pl),pot(pr),pot(a));
89                  if(d>0||(d==0&&dot(pot(pl)−pot(a),pot(pr)−pot(a))<0)){
90                      pl=a;
91                      f=1;
92                      break;
93                  }
94              }
95              for(int i=0;i<egs[pr].size();++i){
96                  int a=egs[pr][i],d=dir(pot(pl),pot(pr),pot(a));
97                  if(d>0||(d==0&&dot(pot(pl)−pot(a),pot(pr)−pot(a))<0)){
98                      pr=a;
99                      f=1;
100                     break;
101                 }
102             }
103             if(!f)
104                 break;
105         }
106         con(pl,pr);
107         for(int pn=−1,wh=0;;pn=−1,wh=0){
108             for(int i=0;i<egs[pl].size();++i){
109                 int a=egs[pl][i],d=dir(pot(pl),pot(pr),pot(a));
110                 if(d<0&&(pn==−1||inc(pot(pl),pot(pr),pot(pn),pot(a))))
111                     pn=a;
112             }
113             for(int i=0;i<egs[pr].size();++i){
114                 int a=egs[pr][i],d=dir(pot(pl),pot(pr),pot(a));
115                 if(d<0&&(pn==−1||inc(pot(pl),pot(pr),pot(pn),pot(a))))
116                     pn=a,wh=1;
117             }
118             if(pn==−1)
```

```
119                 break;
120             vector<int>ne;
121             if(!wh){
122                 for(int i=0;i<egs[pl].size();++i){
123                     int a=egs[pl][i];
124                     if(!crs(pot(pn),pot(pr),pot(pl),pot(a)))
125                         ne.push_back(a);
126                     else
127                         egs[a].erase(find(egs[a].begin(),egs[a].end(),pl));
128                 }
129                 egs[pl]=ne;
130                 con(pr,pn);
131                 pl=pn;
132             }else{
133                 for(int i=0;i<egs[pr].size();++i){
134                     int a=egs[pr][i];
135                     if(!crs(pot(pn),pot(pl),pot(pr),pot(a)))
136                         ne.push_back(a);
137                     else
138                         egs[a].erase(find(egs[a].begin(),egs[a].end(),pr));
139                 }
140                 egs[pr]=ne;
141                 con(pl,pn);
142                 pr=pn;
143             }
144         }
145     }
146     vector<vector<int> >run(){
147         egs.resize(n+1);
148         sort(pts.begin()+1,pts.end());
149         dnc(1,n);
150         vector<vector<int> >res(n+1);
151         for(int u=1;u<=n;++u)
152             for(int i=0;i<egs[u].size();++i){
153                 int v=egs[u][i];
154                 res[pts[u].second].push_back(pts[v].second);
155             }
156         return res;
157     }
158 };
159 template<class T>const double DelaunayTriangulation<T>::E=1e-8;
```

1.3   Dynamic Convex Hull (Set)

Dynamic Convex Hull (Set).hpp (2239 bytes, 77 lines)

```cpp
#include<bits/stdc++.h>
using namespace std;
template<class T>struct DynamicConvexHull{
    struct point{
        T x,y;
        point(T _x=0,T _y=0):
            x(_x),y(_y){
        }
        point operator-(const point&a)const{
            point p(x-a.x,y-a.y);
            return p;
        }
        T operator*(const point&a)const{
            return x*a.y-y*a.x;
        }
    };
    struct node{
        node**nxt;point p;
        node(node** _n,point _p):
            nxt(_n),p(_p){
        }
        node(const node&a):
            nxt(new node*(*a.nxt)),p(a.p){
        }
        ~node(){
            delete nxt;
        }
        int operator<(const node&a)const{
            if(ctp)
                return p.x==a.p.x?p.y<a.p.y:p.x<a.p.x;
            point p1,p2;
            int f=1;
            if(nxt)
                p1=*nxt?(*nxt)->p-p:point(0,-1),p2=a.p;
            else
                f=0,p1=*a.nxt?(*a.nxt)->p-a.p:point(0,-1),p2=p;
            T x=p1*p2;
```

```
38                return f?x<0:x>0;
39            }
40        };
41        static int ctp;
42        set<node>nds;
43        typedef typename set<node>::iterator P;
44        int check(P a,P b,P c){
45            return (b->p-a->p)*(c->p-b->p)>=0;
46        }
47        void next(P a,P b){
48            *(a->nxt)=(node*)&*b;
49        }
50        void insert(T x,T y){
51            ctp=1;
52            node t(new node*(0),point(x,y));
53            P it=nds.insert(t).first,itl1=it,itl2,itr1=it,itr2=it;
54            if(it!=nds.begin())
55                for(next(--itl1,it);itl1!=nds.begin()&&check(--(itl2=itl1),
    itl1,it);)
56                    next(itl2,it),nds.erase(itl1),itl1=itl2;
57            if(++(itr1=it)!=nds.end())
58                next(it,itr1);
59            if(itl1!=it&&itr1!=nds.end()&&check(itl1,it,itr1)){
60                next(itl1,itr1);
61                nds.erase(it);
62                return;
63            }
64            if(itr1!=nds.end())
65                for(;++(itr2=itr1)!=nds.end()&&check(it,itr1,itr2);)
66                    next(it,itr2),nds.erase(itr1),itr1=itr2;
67        }
68        int size(){
69            return nds.size();
70        }
71        pair<T,T>query(T x,T y){
72            ctp=0;
73            node t=*nds.lower_bound(node(0,point(x,y)));
74            return make_pair(t.p.x,t.p.y);
75        }
76 };
77 template<class T>int DynamicConvexHull<T>::ctp=0;
```

1.4   Dynamic Convex Hull (Treap)

Dynamic Convex Hull (Treap).hpp (9485 bytes, 327 lines)

```cpp
#include<bits/stdc++.h>
using namespace std;
template<class T>struct DynamicConvexHull{
    struct point{
        T x,y;
        point(T _x,T _y):
            x(_x),y(_y){
        }
        point operator-(const point&a)const{
            point p(x-a.x,y-a.y);
            return p;
        }
        T operator*(const point&a)const{
            return x*a.y-y*a.x;
        }
        int operator<(const point&a)const{
            return x==a.x?y<a.y:x<a.x;
        }
        int operator==(const point&a)const{
            return x==a.x&&y==a.y;
        }
    };
    struct hull{
        point*pt;
        hull*ch[2],*nb[2];
        int sz,fx;
        hull(point*_pt):
            pt(_pt),sz(1),fx(rand()*1.0/RAND_MAX*1e9){
            ch[0]=ch[1]=nb[0]=nb[1]=0;
        }
        T check(point p){
            return (nb[1]?*nb[1]->pt-*pt:point(0,-1))*p;
        }
        void update(){
            sz=1;
            for(int i=0;i<2;++i)
                if(ch[i])
```

```
38                        sz+=ch[i]->sz;
39              }
40      };
41      static int sz(hull*x){
42          return x?x->sz:0;
43      }
44      static point&pt(hull*x){
45          return*x->pt;
46      }
47      static struct memory{
48          hull*ps,*pp,**ss,**sp;
49          int pm,sm;
50          vector<hull*>ns;
51          memory():
52              ps((hull*)malloc(sizeof(hull))),pp(ps),pm(1),ss((hull**)malloc(
        sizeof(hull*))),sp(ss),sm(1){
53              ns.push_back(ps);
54          }
55          ~memory(){
56              free(ss);
57              for(int i=0;i<ns.size();++i)
58                  free(ns[i]);
59          }
60          hull*create(const hull&x){
61              if(sp!=ss){
62                  --sp;
63                  **sp=x;
64                  return*sp;
65              }
66              if(pp==ps+pm){
67                  pp=ps=(hull*)malloc(sizeof(hull)*(pm<<=1));
68                  ns.push_back(ps);
69              }
70              *pp=x;
71              return pp++;
72          }
73          void destroy(hull*x){
74              if(sp==ss+sm){
75                  hull**t=(hull**)malloc(sizeof(hull*)*sm<<1);
76                  memcpy(t,ss,sm*sizeof(hull*));
77                  free(ss);
```

```
78              sp=(ss=t)+sm;
79                sm<<=1;}
80            *(sp++)=x;
81          }
82      }me;
83      struct array{
84          hull**ps,**pp;
85          int pm;
86          array():
87              ps((hull**)malloc(sizeof(hull*))),pp(ps),pm(1){
88          }
89          ~array(){
90              free(ps);
91          }
92          int size(){
93              return pp−ps;
94          }
95          hull*operator[](int i){
96              return ps[i];
97          }
98          void push(hull*x){
99              if(pp==ps+pm){
100                 hull**t=(hull**)malloc(sizeof(hull*)*pm<<1);
101                 memcpy(t,ps,pm*sizeof(hull*));
102                 free(ps);
103                 pp=(ps=t)+pm;
104                 pm<<=1;
105             }
106             *(pp++)=x;
107         }
108     };
109     static hull*link(hull*x,hull*y,hull*lb,hull*rb,int d,array&ns){
110         hull*r=me.create(*x);
111         if(x==lb||x==rb){
112             r−>nb[d]=y;
113             if(y)
114                 y−>nb[!d]=r;
115         }else
116             r−>ch[d]=link(r−>ch[d],y,lb,rb,d,ns);
117         r−>update();
118         ns.push(r);
```

```
119            return r;
120        }
121        static hull*merge(hull*x,hull*y,hull*lb,hull*rb,array&ns){
122            if(!x)
123                return y;
124            if(!y)
125                return x;
126            int d=x->fx>y->fx;
127            hull*r=me.create(d?*x:*y);
128            r->ch[d]=d?merge(r->ch[1],y,lb,rb,ns):merge(x,y->ch[0],lb,rb,ns);
129            if(d&&x==lb||!d&&y==rb)
130                r->ch[d]=link(r->ch[d],r,lb,rb,!d,ns);
131            r->update();
132            ns.push(r);
133            return r;
134        }
135        static pair<hull*,hull*>split(hull*x,int k,array&ns){
136            if(!x)
137                return make_pair((hull*)0,(hull*)0);
138            int t=sz(x->ch[0])+1;
139            hull*r=me.create(*x);
140            ns.push(r);
141            pair<hull*,hull*>s=split(x->ch[k>=t],k-t*(k>=t),ns);
142            if(k>=t){
143                r->ch[1]=s.first;r->update();
144                return make_pair(r,s.second);
145            }else{
146                r->ch[0]=s.second;r->update();
147                return make_pair(s.first,r);
148            }
149        }
150        static void turn(hull*&x,int d,int&k){
151            k+=(sz((x=x->ch[d])->ch[!d])+1)*(2*d-1);
152        }
153        static pair<T,T>range(hull*x){
154            hull*l=x,*r=x;
155            while(l->ch[0])
156                l=l->ch[0];
157            while(r->ch[1])
158                r=r->ch[1];
159            return make_pair(pt(l).x,pt(r).x);
```

```
160          }
161      static hull*merge(hull*x,hull*y,array&ns){
162          int kp=sz(x->ch[0])+1,kq=sz(y->ch[0])+1,pd[2],qd[2];
163          pair<T,T>pr=range(x),qr=range(y);
164          int pf=1;
165          hull*p=x,*q=y;
166          if(pr.second==qr.first&&pr.first==pr.second&&p->ch[pf=0])
167              turn(p,0,kp);
168          for(point pq=pt(q)-pt(p);;pq=pt(q)-pt(p)){
169              pd[0]=(p->nb[0]&&(pt(p->nb[0])-pt(p))*pq<=0)*pf;
170              qd[1]=(q->nb[1]&&(pt(q->nb[1])-pt(q))*pq<=0);
171              pd[1]=(p->nb[1]&&(pt(p->nb[1])-pt(p))*pq<0)*pf;
172              qd[0]=(q->nb[0]&&(pt(q->nb[0])-pt(q))*pq<0);
173              if(!(pd[0]+pd[1]+qd[0]+qd[1])){
174                  hull*l=split(x,kp,ns).first,*r=split(y,kq-1,ns).second,*lb=
     l,*rb=r;
175                  while(lb->ch[1])
176                      lb=lb->ch[1];
177                  while(rb->ch[0])
178                      rb=rb->ch[0];
179                  return merge(l,r,lb,rb,ns);
180              }
181              if(!(pd[0]+pd[1]))
182                  turn(q,qd[1],kq);
183              if(!(qd[0]+qd[1]))
184                  turn(p,pd[1],kp);
185              if(pd[0]&&qd[1])
186                  turn(p,0,kp),turn(q,1,kq);
187              if(pd[1]&&qd[1])
188                  turn(q,1,kq);
189              if(pd[0]&&qd[0])turn(p,0,kp);
190              if(pd[1]&&qd[0]){
191                  point vp=pt(p->nb[1])-pt(p),vq=pt(q->nb[0])-pt(q);
192                  if(vp.x==0&&vq.x==0)
193                      turn(p,1,kp),turn(q,0,kq);
194                  else if(vp.x==0)
195                      turn(p,1,kp);
196                  else if(vq.x==0)
197                      turn(q,0,kq);
198                  else{
199                      long double m=pr.second,pb=vp.y*(m-pt(p).x),qb=vq.y*(m-
```

```
          pt(q).x);
200                           pb=pb/vp.x+pt(p).y;
201                           qb=qb/vq.x+pt(q).y;
202                           if(qb>pb+1e−8)
203                               turn(q,0,kq);
204                           else if(pb>qb+1e−8)
205                               turn(p,1,kp);
206                           else if(pt(q−>nb[0]).x+pt(p−>nb[1]).x<2*m)
207                               turn(q,0,kq);
208                           else
209                               turn(p,1,kp);
210                   }
211               }
212           }
213       }
214       hull*query(hull*x,point p){
215           for(hull*y=0;;){
216               T d=x−>check(p);
217               if(d>0)
218                   y=x,x=x−>ch[0];
219               else if(d<0)
220                   x=x−>ch[1];
221               else
222                   y=x;
223               if(!d||!x)
224                   return y;
225           }
226       }
227       struct treap{
228           int fx,ct,sz;
229           point pt;
230           treap*ch[2];
231           struct hull*ip,*hu;
232           array ns;
233           treap(point _pt):
234               fx(rand()*1.0/RAND_MAX*1e9),ct(1),sz(1),pt(_pt),ip(me.create(
          hull(&pt))),hu(ip){
235               ch[0]=ch[1]=0;
236           }
237           ~treap(){
238               for(hull**i=ns.ps;i!=ns.pp;++i)
```

```
239                     me.destroy(*i);
240                 me.destroy(ip);
241             }
242         void update(){
243             for(hull**i=ns.ps;i!=ns.pp;++i)
244                 me.destroy(*i);
245             ns.pp=ns.ps;
246             sz=1;
247             hu=ip;
248             if(ch[0])
249                 hu=merge(ch[0]->hu,hu,ns),sz+=ch[0]->sz;
250             if(ch[1])
251                 hu=merge(hu,ch[1]->hu,ns),sz+=ch[1]->sz;
252         }
253     }*root;
254     void rotate(treap*&x,int d){
255         treap*y=x->ch[d];
256         x->ch[d]=y->ch[!d];
257         y->ch[!d]=x;
258         x=y;
259     }
260     int insert(treap*&x,point p){
261         if(!x)
262             x=new treap(p);
263         else if(p==x->pt){
264             ++x->ct;
265             return 0;
266         }else{
267             int d=x->pt<p;
268             if(!insert(x->ch[d],p))
269                 return 0;
270             if(x->ch[d]->fx>x->fx)
271                 rotate(x,d),x->ch[!d]->update();
272             x->update();
273         }
274         return 1;
275     }
276     int erase(treap*&x,point p){
277         if(p==x->pt){
278             if(x->ct>1){
279                 --x->ct;
```

```
280                     return 0;
281                 }
282             treap*y=x;
283             if(!x->ch[0])
284                 x=x->ch[1],delete y;
285             else if(!x->ch[1])
286                 x=x->ch[0],delete y;
287             else{
288                 int d=x->ch[0]->fx<x->ch[1]->fx;
289                 rotate(x,d);
290                 erase(x->ch[!d],p);
291                 x->update();
292             }
293             return 1;
294         }
295         if(erase(x->ch[x->pt<p],p)){
296             x->update();
297             return 1;
298         }else{
299             --x->sz;
300             return 0;
301         }
302     }
303     void clear(treap*x){
304         if(x)
305             clear(x->ch[0]),clear(x->ch[1]),delete x;
306     }
307     DynamicConvexHull():
308         root(0){
309     }
310     ~DynamicConvexHull(){
311         clear(root);
312     }
313     int size(){
314         return root?root->sz:0;
315     }
316     void insert(T x,T y){
317         insert(root,point(x,y));
318     }
319     void erase(T x,T y){
320         erase(root,point(x,y));
```

```
321 |     }
322 |     pair<T,T>query(T x,T y){
323 |         point r=pt(query(root->hu,point(x,y)));
324 |         return make_pair(r.x,r.y);
325 |     }
326 | };
327 | template<class T>typename DynamicConvexHull<T>::memory DynamicConvexHull<T
      |     >::me;
```

1.5   Geometry 2D

Geometry 2D.hpp (5120 bytes, 164 lines)

```
 1 | #include<bits/stdc++.h>
 2 | using namespace std;
 3 | namespace Geometry2D{
 4 |     double eps=1e-8;
 5 |     long double pi=acos((long double)-1);
 6 |     template<class T>T sqr(T a){
 7 |         return a*a;
 8 |     }
 9 |     template<class T>int cmp(T a,T b){
10 |         if(typeid(T)==typeid(int)||typeid(T)==typeid(long long)){
11 |             if(a==b)
12 |                 return 0;
13 |             return a<b?-1:1;
14 |         }
15 |         if(a<b-eps)
16 |             return -1;
17 |         if(a>b+eps)
18 |             return 1;
19 |         return 0;
20 |     }
21 |     template<class T>struct Point{
22 |         T x,y;
23 |         Point(T _x=0,T _y=0):
24 |             x(_x),y(_y){
25 |         }
26 |         Point<T>&operator+=(const Point<T>&a){
```

```
27              return*this=*this+a;
28          }
29          Point<T>&operator-=(const Point<T>&a){
30              return*this=*this-a;
31          }
32      };
33      #define Vector Point
34      template<class T>Point<T>operator+(const Point<T>&a,const Point<T>&b){
35          return Point<T>(a.x+b.x,a.y+b.y);
36      }
37      template<class T>Point<T>operator-(const Point<T>&a,const Point<T>&b){
38          return Point<T>(a.x-b.x,a.y-b.y);
39      }
40      template<class T>Point<T>operator*(T a,const Point<T>&b){
41          return Point<T>(b.x*a,b.y*a);
42      }
43      template<class T>Point<T>operator*(const Point<T>&a,T b){
44          return b*a;
45      }
46      template<class T>Point<T>operator/(const Point<T>&a,T b){
47          return Point<T>(a.x/b,a.y/b);
48      }
49      template<class T>bool operator==(const Point<T>&a,const Point<T>&b){
50          return !cmp(a.x,b.x)&&!cmp(a.y,b.y);
51      }
52      template<class T>bool operator!=(const Point<T>&a,const Point<T>&b){
53          return !(a==b);
54      }
55      template<class T>bool operator<(const Point<T>&a,const Point<T>&b){
56          int t=cmp(a.x,b.x);
57          if(t)
58              return t<0;
59          return cmp(a.y,b.y)<0;
60      }
61      template<class T>bool operator>(const Point<T>&a,const Point<T>&b){
62          return b<a;
63      }
64      template<class T>Point<T>NaP(){
65          T t=numeric_limits<T>::max();
66          return Point<T>(t,t);
67      }
```

```
68      template<class T>T det(const Point<T>&a,const Point<T>&b){
69          return a.x*b.y−a.y*b.x;
70      }
71      template<class T>T dot(const Point<T>&a,const Point<T>&b){
72          return a.x*b.x+a.y*b.y;
73      }
74      template<class T>T abs(const Point<T>&a){
75          return sqrt(sqr(a.x)+sqr(a.y));
76      }
77      template<class T>T dis(const Point<T>&a,const Point<T>&b){
78          return abs(a−b);
79      }
80      template<class T>istream&operator>>(istream&s,Point<T>&a){
81          return s>>a.x>>a.y;
82      }
83      template<class T>ostream&operator<<(ostream&s,const Point<T>&a){
84          return s<<a.x<<" "<<a.y;
85      }
86      template<class T>struct Segment;
87      template<class T>struct Line{
88          Point<T>u,v;
89          Line(const Point<T>&_u=Point<T>(),const Point<T>&_v=Point<T>()):
90              u(_u),v(_v){
91          }
92          Line(const Segment<T>&a):
93              u(a.u),v(a.v){
94          }
95      };
96      template<class T>Point<T>nor(const Line<T>&a){
97          Point<T>t=a.v−a.u;
98          return Point<T>(t.y,−t.x);
99      }
100     template<class T>Point<T>dir(const Line<T>&a){
101         return a.v−a.u;
102     }
103     template<class T>int dir(const Line<T>a,const Point<T>b){
104         return cmp(det(b−a.u,a.v−a.u),T(0));
105     }
106     template<class T>Point<T>operator&(const Line<T>&a,const Line<T>&b){
107         T p=det(b.u−a.v,b.v−b.u),q=det(a.u−b.v,b.v−b.u);
108         return (a.u*p+a.v*q)/(p+q);
```

```
109        }
110    template<class T>struct Segment{
111        Point<T>u,v;
112        Segment(const Point<T>&_u=Point<T>(),const Point<T>&_v=Point<T>()):
113            u(_u),v(_v){
114        }
115    };
116    template<class T>Point<T>nor(const Segment<T>&a){
117        Point<T>t=a.v−a.u;
118        return Point<T>(t.y,−t.x);
119    }
120    template<class T>Point<T>dir(const Segment<T>&a){
121        return a.v−a.u;
122    }
123    template<class T>int dir(const Segment<T>a,const Point<T>b){
124        return cmp(b−a.u,a.v−a.u);
125    }
126    template<class T>Point<T>operator&(const Line<T>&a,const Segment<T>&b){
127        if(dir(a,b.u)*dir(a,b.v)<=0)
128            return a&Line<T>(b);
129        return NaP<T>();
130    }
131    template<class T>Point<T>operator&(const Segment<T>&a,const Line<T>&b){
132        return b&a;
133    }
134    template<class T>pair<T,T>dis(const Segment<T>&a,const Point<T>&b){
135        pair<T,T>d(dis(a.u,b),dis(a.v,b));
136        if(d.first>d.second)
137            swap(d.first,d.second);
138        Point<T>t=Line<T>(b,b+nor(a))&a;
139        if(t!=NaP<T>())
140            d.first=dis(t,b);
141        return d;
142    }
143    template<class T>pair<T,T>dis(const Point<T>&a,const Segment<T>&b){
144        return dis(b,a);
145    }
146    template<class T>struct Circle{
147        Point<T>c;
148        T r;
149        Circle(const Point<T>&_c=Point<T>(),T _r=0):
```

```
150              c(_c),r(_r){
151          }
152      };
153      template<class T>T abs(const Circle<T>&a){
154          return pi*sqr(a.r);
155      }
156      template<class T>bool col(const Point<T>&a,const Point<T>&b,const Point
         <T>&c){
157          return !cmp(det(a−c,b−c),T(0));
158      }
159      template<class T>T read(){
160          T t;
161          cin>>t;
162          return t;
163      }
164  }
```

1.6   Half-Plane Intersection

Half-Plane Intersection.hpp (1950 bytes, 70 lines)

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  namespace HalfPlaneIntersection{
4      const double E=1e−8;
5      struct pot{
6          pot(double a=0,double b=0):
7              x(a),y(b){
8          }
9          double x,y;
10     };
11     double ag(pot p){
12         return atan2(double(p.x),double(p.y));
13     }
14     pot operator+(pot p,pot q){
15         return pot(p.x+q.x,p.y+q.y);
16     }
17     pot operator−(pot p,pot q){
18         return pot(p.x−q.x,p.y−q.y);
```

```
19          }
20      pot operator*(pot p,double q){
21          return pot(p.x*q,p.y*q);
22      }
23      pot operator/(pot p,double q){
24          return pot(p.x/q,p.y/q);
25      }
26      double det(pot p,pot q){
27          return p.x*q.y−q.x*p.y;
28      }
29      double dot(pot p,pot q){
30          return p.x*q.x+p.y*q.y;
31      }
32      struct lin{
33          pot p,q;
34          double a;
35          lin(pot a,pot b):
36              p(a),q(b),a(ag(b−a)){
37          }
38      };
39      pot operator*(lin a,lin b){
40          double a1=det(b.p−a.q,b.q−b.p);
41          double a2=det(a.p−b.q,b.q−b.p);
42          return (a.p*a1+a.q*a2)/(a1+a2);
43      }
44      bool cmp(lin a,lin b){
45          if(fabs(a.a−b.a)>E)
46              return a.a<b.a;
47          else
48              return det(a.q−b.p,b.q−b.p)<−E;
49      }
50      bool left(lin a,lin b,lin c){
51          pot t=a*b;
52          return det(t−c.p,c.q−c.p)<−E;
53      }
54      deque<lin>run(vector<lin>lns){
55          deque<lin>ans;
56          sort(lns.begin(),lns.end(),cmp);
57          for(int i=0;i<lns.size();++i){
58              while(ans.size()>1&&!left(ans.back(),ans[ans.size()−2],lns[i]))
59                  ans.pop_back();
```

```
60          while(ans.size()>1&&!left(ans[0],ans[1],lns[i]))
61              ans.pop_front();
62          if(ans.empty()||fabs(ans.back().a−lns[i].a)>E)
63              ans.push_back(lns[i]);}
64      while(ans.size()>1&&!left(ans[ans.size()−1],ans[ans.size()−2],ans.
    front()))
65              ans.pop_back();
66      if(ans.size()<3)
67          ans.clear();
68      return ans;
69   }
70 }
```

CHAPTER 2

Data Structures

2.1   Binary Heap

Binary Heap.hpp (1629 bytes, 73 lines)

```
1   #include<bits/stdc++.h>
2   using namespace std;
3   template<class T,class C>struct BinaryHeap{
4       struct node{
5           node(int _p,T _v):
6               p(_p),v(_v){
7           }
8           int p;
9           T v;
10      };
11      vector<node*>a;
12      BinaryHeap():
13          a(1){
14      }
15      ~BinaryHeap(){
16          clear();
17      }
18      void move(int i,int j){
19          swap(a[i]->p,a[j]->p);
20          swap(a[i],a[j]);
21      }
22      int check(int i,int j){
23          if(!j||j>=a.size()||a[i]->v==a[j]->v)
24              return 0;
25          return a[i]->v<a[j]->v?-1:1;
26      }
27      int up(int i){
28          if(check(i,i>>1)<0){
29              move(i,i>>1);
30              return i>>1;
31          }else
32              return 0;
33      }
34      int down(int i){
35          if(check(i,i<<1)<=0&&check(i,i<<1^1)<=0)
36              return a.size();
37          if(check(i<<1,i<<1^1)<=0){
```

```
38              move(i,i<<1);
39              return i<<1;
40          }else{
41              move(i,i<<1^1);
42              return i<<1^1;
43          }
44      }
45      void maintain(int i){
46          for(int j=up(i);j;i=j,j=up(i));
47          for(int j=down(i);j<a.size();i=j,j=down(i));
48      }
49      void clear(){
50          for(int i=1;i<a.size();++i)
51              delete a[i];
52          a.resize(1);
53      }
54      node*push(T v){
55          a.push_back(new node(a.size(),v));
56          node*r=a.back();
57          maintain(a.size()-1);
58          return r;
59      }
60      T top(){
61          return a[1]->v;
62      }
63      void pop(){
64          move(1,a.size()-1);
65          delete a.back();
66          a.pop_back();
67          maintain(1);
68      }
69      void modify(node*x,T v){
70          x->v=v;
71          maintain(x->p);
72      }
73  };
```

2.2   Dynamic Sequence

Dynamic Sequence.hpp (4119 bytes, 177 lines)

```cpp
#include<bits/stdc++.h>
using namespace std;
template<class T>struct DynamicSequence{
    struct node{
        node(T _i):
            i(_i),v(_i),s(1),r(0){
                c[0]=c[1]=0;
                static int g;
                w=g=(214013*g+2531011);
        }
        T i,v;
        int s,r,w;
        node*c[2];
    }*rt,*sl,*sr;
    struct pool{
        node*ps,*pp,**ss,**sp;
        int pm,sm;
        vector<node*>ns;
        pool():
            ps((node*)malloc(sizeof(node))),pp(ps),pm(1),ss((node**)malloc(
    sizeof(node*))),sp(ss),sm(1){
                ns.push_back(ps);
        }
        ~pool(){
            free(ss);
            for(int i=0;i<ns.size();++i)
                free(ns[i]);
        }
        node*crt(T a){
            if(sp!=ss){
                --sp;
                **sp=node(a);
                return*sp;
            }
            if(pp==ps+pm){
                pp=ps=(node*)malloc(sizeof(node)*(pm<<=1));
                ns.push_back(ps);
```

```
37                    }
38                    *pp=node(a);
39                    return pp++;
40              }
41              void des(node*x){
42                    if(sp==ss+sm){
43                          node**t=(node**)malloc(sizeof(node*)*sm<<1);
44                          memcpy(t,ss,sm*sizeof(node*));
45                          free(ss);
46                          sp=(ss=t)+sm;
47                          sm<<=1;
48                    }
49                    *(sp++)=x;
50              }
51        }me;
52        node*bud(T*a,int l,int r){
53              if(l>r)
54                    return 0;
55              int m=l+r>>1;
56              node*t=me.crt(a[m]);
57              t->c[0]=bud(a,l,m-1);
58              t->c[1]=bud(a,m+1,r);
59              pup(t);
60              return t;
61        }
62        void pdw(node*x){
63              for(int d=0;d<2&&(x->i>x->v,1);++d)
64                    if(x->c[d])
65                          x->i>x->c[d]->i;
66              *x->i;
67              *x->v;
68              if(x->r){
69                    -x->i;
70                    for(int d=0;d<2;++d)
71                          if(x->c[d])
72                                x->c[d]->r^=1;
73                    swap(x->c[0],x->c[1]);
74                    x->r=0;
75              }
76        }
77        void pup(node*x){
```

```
78          x->i=x->v;
79          x->s=1;
80          for(int d=0;d<2;++d)
81              if(x->c[d])
82                  pdw(x->c[d]),x->s+=x->c[d]->s,x->i=d?x->i+x->c[d]->i:x->
    c[d]->i+x->i;
83      }
84      void jon(node*x){
85          rt=jon(jon(sl,x),sr);
86      }
87      node*jon(node*x,node*y){
88          if(!x)
89              return y;
90          if(!y)
91              return x;
92          pdw(x);
93          pdw(y);
94          if(x->w<y->w){
95              x->c[1]=jon(x->c[1],y);
96              pup(x);
97              return x;
98          }else{
99              y->c[0]=jon(x,y->c[0]);
100             pup(y);
101             return y;
102         }
103     }
104     node*spt(int l,int r){
105         spt(rt,l-1);
106         node*t=sl;
107         spt(sr,r-l+1);
108         swap(sl,t);
109         return t;
110     }
111     void spt(node*x,int p){
112         if(!x){
113             sl=sr=0;
114             return;
115         }
116         pdw(x);
117         int t=x->c[0]?x->c[0]->s:0;
```

```
118        if(t<p)
119            spt(x->c[1],p-t-1),x->c[1]=sl,sl=x;
120        else
121            spt(x->c[0],p),x->c[0]=sr,sr=x;
122        pup(x);
123    }
124    void clr(node*x){
125        if(x)
126            clr(x->c[0]),clr(x->c[1]),me.des(x);
127    }
128    DynamicSequence(T*a=0,int n=0){
129        rt=bud(a,1,n);
130    }
131    ~DynamicSequence(){
132        clr(rt);
133    }
134    void clear(){
135        clr(rt);
136        rt=0;
137    }
138    void insert(T a,int p){
139        insert(&a-1,1,p);
140    }
141    void insert(T*a,int n,int p){
142        spt(p+1,p);
143        jon(bud(a,1,n));
144    }
145    void erase(int p){
146        erase(p,p);
147    }
148    void erase(int l,int r){
149        clr(spt(l,r));
150        jon(0);
151    }
152    T query(int p){
153        return query(p,p);
154    }
155    T query(int l,int r){
156        node*t=spt(l,r);
157        T i=t->i;
158        jon(t);
```

```
159         return i;
160     }
161     void modify(T a,int l){
162         modify(a,l,l);
163     }
164     void modify(T a,int l,int r){
165         node*t=spt(l,r);
166         a>t−>i;
167         jon(t);
168     }
169     void reverse(int l,int r){
170         node*t=spt(l,r);
171         t−>r=1;
172         jon(t);
173     }
174     int length(){
175         return rt?rt−>s:0;
176     }
177 };
```

2.3   Fenwick Tree

Fenwick Tree.hpp (529 bytes, 25 lines)

```cpp
1  #include<bits/stdc++.h>
2  using namespace std;
3  template<class T>struct FenwickTree{
4      FenwickTree(int _n):
5          n(_n),l(log2(n)),a(n+1){
6      }
7      void add(int v,T d){
8          for(;v<=n;v+=v&−v)
9              a[v]+=d;
10     }
11     T sum(int v){
12         T r=0;
13         for(;v;v−=v&−v)
14             r+=a[v];
15         return r;
```

```
16          }
17      int kth(T k,int r=0){
18          for(int i=1<<l;i;i>>=1)
19              if(r+i<=n&&a[r+i]<k)
20                  k-=a[r+=i];
21          return r+1;
22      }
23      int n,l;
24      vector<T>a;
25  };
```

2.4   K-D Tree

K-D Tree.hpp (2467 bytes, 80 lines)

```
1   #include<bits/stdc++.h>
2   using namespace std;
3   struct KDTree{
4       struct node{
5           node(int x0,int x1,int d):
6               color(1),cover(0),dir(d){
7                   ch[0]=ch[1]=0;
8                   x[0]=mi[0]=mx[0]=x0;
9                   x[1]=mi[1]=mx[1]=x1;
10          }
11          node*ch[2];
12          int x[2],mi[2],mx[2],color,cover,dir;
13      }*root;
14      KDTree(pair<int,int>*a,int n){
15          root=build(a,1,n,0);
16      }
17      static int direct;
18      static int cmp(pair<int,int>a,pair<int,int>b){
19          if(direct)
20              return make_pair(a.second,a.first)<make_pair(b.second,b.first);
21          return a<b;
22      }
23      node*build(pair<int,int>*a,int l,int r,int d){
24          int m=(r+l)/2;
```

```
25            direct=d;
26            nth_element(a+l,a+m,a+r+1,cmp);
27            node*p=new node((a+m)−>first,(a+m)−>second,d);
28            if(l!=m)
29                p−>ch[0]=build(a,l,m−1,!d);
30            if(r!=m)
31                p−>ch[1]=build(a,m+1,r,!d);
32            for(int i=0;i<2;++i)
33                for(int j=0;j<2;++j)
34                    if(p−>ch[j]){
35                        p−>mi[i]=min(p−>mi[i],p−>ch[j]−>mi[i]);
36                        p−>mx[i]=max(p−>mx[i],p−>ch[j]−>mx[i]);
37                    }
38            return p;
39        }
40        void down(node*a){
41            if(a−>cover){
42                for(int i=0;i<2;++i)
43                    if(a−>ch[i])
44                        a−>ch[i]−>cover=a−>cover;
45                a−>color=a−>cover;
46                a−>cover=0;
47            }
48        }
49        void modify(node*a,int mi0,int mx0,int mi1,int mx1,int c){
50            if(mi0>a−>mx[0]||mx0<a−>mi[0]||mi1>a−>mx[1]||mx1<a−>mi[1])
51                return;
52            if(mi0<=a−>mi[0]&&mx0>=a−>mx[0]&&mi1<=a−>mi[1]&&mx1>=a−>mx[1]){
53                a−>cover=c;
54                return;
55            }
56            down(a);
57            if(mi0<=a−>x[0]&&mx0>=a−>x[0]&&mi1<=a−>x[1]&&mx1>=a−>x[1])
58                a−>color=c;
59            for(int i=0;i<2;++i)
60                if(a−>ch[i])
61                    modify(a−>ch[i],mi0,mx0,mi1,mx1,c);
62        }
63        void modify(int mi0,int mx0,int mi1,int mx1,int c){
64            modify(root,mi0,mx0,mi1,mx1,c);
65        }
```

```
66      int query(node*a,int x0,int x1){
67          down(a);
68          if(x0==a->x[0]&&x1==a->x[1])
69              return a->color;
70          direct=a->dir;
71          if(cmp(make_pair(x0,x1),make_pair(a->x[0],a->x[1])))
72              return query(a->ch[0],x0,x1);
73          else
74              return query(a->ch[1],x0,x1);
75      }
76      int query(int x0,int x1){
77          return query(root,x0,x1);
78      }
79  };
80  int KDTree::direct=0;
```

2.5   Link-Cut Tree

Link-Cut Tree.hpp (5518 bytes, 215 lines)

```
1   #include<bits/stdc++.h>
2   using namespace std;
3   template<class T>struct LinkCutTree{
4       struct node{
5           node():
6               ch({0,0}),pr(0),rev(0){
7           }
8           node*ch[2],*pr;
9           T ifo;
10          int rev;
11      }*ptrs;
12      LinkCutTree(int n):
13          ptrs(new node[n]-1){
14      }
15      ~LinkCutTree(){
16          delete ptrs;
17      }
18      int direct(node*x){
19          if(!x->pr)
```

```
20              return 2;
21          if(x==x->pr->ch[0])
22              return 0;
23          if(x==x->pr->ch[1])
24              return 1;
25          return 2;
26      }
27      void down(node*x){
28          if(x->rev){
29              x->ifo.reverse();
30              swap(x->ch[0],x->ch[1]);
31              for(int i=0;i<2;++i)
32                  if(x->ch[i])
33                      x->ch[i]->rev^=1;
34              x->rev=0;
35          }
36          x->ifo.down(x->ch[0]?&x->ch[0]->ifo:0,x->ch[1]?&x->ch[1]->ifo:0);
37      }
38      void up(node*x){
39          for(int i=0;i<2;++i)
40              if(x->ch[i])
41                  down(x->ch[i]);
42          x->ifo.up(x->ch[0]?&x->ch[0]->ifo:0,x->ch[1]?&x->ch[1]->ifo:0);
43      }
44      void setchild(node*x,node*y,int d){
45          x->ch[d]=y;
46          if(y)
47              y->pr=x;
48          up(x);
49      }
50      void rotate(node*x){
51          node*y=x->pr,*z=y->pr;
52          int d1=direct(x),d2=direct(y);
53          setchild(y,x->ch[!d1],d1);
54          setchild(x,y,!d1);
55          if(d2<2)
56              setchild(z,x,d2);
57          else
58              x->pr=z;
59      }
60      void release(node*x){
```

```
61          if(direct(x)<2)
62              release(x−>pr);
63          down(x);
64      }
65      void splay(node*x){
66          for(release(x);direct(x)<2;){
67              node*y=x−>pr;
68              if(direct(y)==2)
69                  rotate(x);
70              else if(direct(x)==direct(y))
71                  rotate(y),rotate(x);
72              else
73                  rotate(x),rotate(x);
74          }
75      }
76      node*access(node*x){
77          node*y=0;
78          for(;x;y=x,x=x−>pr){
79              splay(x);
80              setchild(x,y,1);
81          }
82          return y;
83      }
84      void evert(node*x){
85          access(x);
86          splay(x);
87          x−>rev=1;
88      }
89      void set(int x,T v){
90          ptrs[x].ifo=v;
91      }
92      int linked(int a,int b){
93          access((ptrs+a));
94          node*z=access((ptrs+b));
95          return z==access((ptrs+a));
96      }
97      void link(int a,int b){
98          evert((ptrs+b));
99          (ptrs+b)−>pr=(ptrs+a);
100     }
101     void cut(int a,int b){
```

```
102            access((ptrs+b));
103            node*z=access((ptrs+a));
104            if(z==(ptrs+a))
105                splay((ptrs+b)),(ptrs+b)->pr=0;
106            else
107                access((ptrs+b)),splay((ptrs+a)),(ptrs+a)->pr=0;
108        }
109        int root(int a){
110            access((ptrs+a));
111            splay((ptrs+a));
112            node*r=(ptrs+a);
113            while(r->ch[1])
114                r=r->ch[1];
115            return r-ptrs;
116        }
117        void evert(int a){
118            evert((ptrs+a));
119        }
120        int lca(int a,int b){
121            access((ptrs+a));
122            return access((ptrs+b))-ptrs;
123        }
124        T query(int a){
125            splay((ptrs+a));
126            T p=(ptrs+a)->ifo;
127            p.up(0,0);
128            return p;
129        }
130        T query(int a,int b){
131            if((ptrs+a)==(ptrs+b))
132                return query((ptrs+a));
133            access((ptrs+a));
134            node*c=access((ptrs+b));
135            T p=c.ifo;
136            if(c==(ptrs+b)){
137                splay((ptrs+a));
138                T q=(ptrs+a)->ifo;
139                q.reverse();
140                p.up(&q,0);
141                return p;
142            }else if(c==(ptrs+a))
```

```
143              p.up(0,&(ptrs+a)−>ch[1]−>ifo);
144          else{
145              splay((ptrs+a));
146              T q=(ptrs+a)−>ifo;
147              q.reverse();
148              p.up(&q,&c−>ch[1]−>ifo);
149          }
150          return p;
151      }
152      T equery(int a){
153          return query(a);
154      }
155      T equery(int a,int b){
156          access((ptrs+a));
157          node*c=access((ptrs+b));
158          if(c==(ptrs+b)){
159              splay((ptrs+a));
160              T q=(ptrs+a)−>ifo;
161              q.reverse();
162              return q;
163          }else if(c==(ptrs+a))
164              return (ptrs+a)−>ch[1]−>ifo;
165          else{
166              splay((ptrs+a));
167              node*t=c−>ch[1];
168              while(t−>ch[0])
169                  t=t−>ch[0];
170              splay(t);
171              if(t−>ch[1])
172                  down(t−>ch[1]);
173              T p=t−>ifo,q=(ptrs+a)−>ifo;
174              q.reverse();
175              p.up(&q,t−>ch[1]?&t−>ch[1]−>ifo:0);
176              return p;
177          }
178      }
179      template<class F>void modify(int a,F f){
180          splay((ptrs+a));
181          f(&(ptrs+a)−>ifo);
182          up((ptrs+a));
183      }
```

```
184    template<class F>void modify(int a,int b,F f){
185        if((ptrs+a)==(ptrs+b)){
186            splay((ptrs+a));
187            f(0,&(ptrs+a)−>ifo,0);
188            up((ptrs+a));
189            return;
190        }
191        access((ptrs+a));
192        node*c=access((ptrs+b));
193        if(c==(ptrs+b))
194            splay((ptrs+a)),f(&(ptrs+a)−>ifo,&(ptrs+b)−>ifo,0);
195        else if(c==a)
196            f(0,&(ptrs+a)−>ifo,&(ptrs+a)−>ch[1]−>ifo);
197        else
198            splay(a),f(&(ptrs+a)−>ifo,&c−>ifo,&c−>ch[1]−>ifo);
199        up(c);
200    }
201    template<class F>void emodify(int a,F f){
202        modify(a,f);
203    }
204    template<class F>void emodify(int a,int b,F f){
205        access((ptrs+a));
206        node*c=access((ptrs+b));
207        if(c==(ptrs+b))
208            splay((ptrs+a)),f(&(ptrs+a)−>ifo,0);
209        else if(c==a)
210            f(0,&(ptrs+a)−>ch[1]−>ifo);
211        else
212            splay(a),f(&(ptrs+a)−>ifo,&c−>ch[1]−>ifo);
213        up(c);
214    }
215 };
```

2.6   Pairing Heap

Pairing Heap.hpp (2226 bytes, 102 lines)

```
1 #include<bits/stdc++.h>
2 using namespace std;
```

```
 3  template<class T,class C>struct PairingHeap{
 4      PairingHeap():
 5          root(0),siz(0){
 6      }
 7      ~PairingHeap(){
 8          clear(root);
 9      }
10      struct node{
11          node(const T&_val):
12              val(_val),ch(0),br(0),pr(0){
13          }
14          T val;
15          node*ch,*br,*pr;
16      }*root;
17      int siz;
18      void merge(node*&x,node*y){
19          if(!x)
20              x=y;
21          else if(y){
22              if(C()(y->val,x->val))
23                  swap(x,y);
24              y->br=x->ch;
25              if(x->ch)
26                  x->ch->pr=y;
27              y->pr=x;
28              x->ch=y;
29          }
30      }
31      void cut(node*&x,node*y){
32          if(x==y)
33              x=0;
34          else{
35              if(y==y->pr->ch)
36                  y->pr->ch=y->br;
37              else
38                  y->pr->br=y->br;
39              if(y->br)
40                  y->br->pr=y->pr;
41              y->pr=y->br=0;
42          }
43      }
```

```
44 | node*split(node*x){
45 |     vector<node*>t;
46 |     for(node*i=x->ch;i;i=i->br)
47 |         t.push_back(i);
48 |     x->ch=0;
49 |     node*r=0;
50 |     for(int i=0;i<t.size();++i)
51 |         t[i]->pr=t[i]->br=0;
52 |     for(int i=0;i+1<t.size();i+=2)
53 |         merge(t[i],t[i+1]);
54 |     for(int i=0;i<t.size();i+=2)
55 |         merge(r,t[i]);
56 |     return r;
57 | }
58 | void clear(node*x){
59 |     if(x){
60 |         clear(x->ch);
61 |         clear(x->br);
62 |         delete x;
63 |     }
64 | }
65 | void clear(){
66 |     clear(root);
67 |     root=0;
68 |     siz=0;
69 | }
70 | node*push(T a){
71 |     node*r=new node(a);
72 |     merge(root,r);
73 |     ++siz;
74 |     return r;
75 | }
76 | void erase(node*x){
77 |     cut(root,x);
78 |     merge(root,split(x));
79 |     --siz;
80 | }
81 | T top(){
82 |     return root->val;
83 | }
84 | void pop(){
```

```
85  |         erase(root);
86  |     }
87  |     void merge(PairingHeap<T,C>&a){
88  |         merge(root,a.root);
89  |         a.root=0;
90  |         siz+=a.siz;
91  |         a.siz=0;
92  |     }
93  |     void modify(node*x,T v){
94  |         if(C()(x−>val,v))
95  |             x−>val=v,merge(root,split(x));
96  |         else
97  |             x−>val=v,cut(root,x),merge(root,x);
98  |     }
99  |     int size(){
100 |         return siz;
101 |     }
102 | };
```

2.7   Red-Black Tree

Red-Black Tree.hpp (7432 bytes, 307 lines)

```
1  | #include<bits/stdc++.h>
2  | using namespace std;
3  | template<class T,class C>struct RedBlackTree{
4  |     struct node{
5  |         node(T _v,node*l,node*r,node*_p,int _b,int _s):
6  |             v(_v),p(_p),b(_b),s(_s){
7  |             c[0]=l;
8  |             c[1]=r;
9  |         }
10 |         T v;
11 |         node*c[2],*p;
12 |         int b,s;
13 |     }*root,*nil;
14 |     void clear(node*x){
15 |         if(x!=nil){
16 |             clear(x−>c[0]);
```

```
17              clear(x->c[1]);
18              delete x;
19          }
20      }
21      void rotate(node*x,int d){
22          node*y=x->c[!d];
23          x->c[!d]=y->c[d];
24          if(y->c[d]!=nil)
25              y->c[d]->p=x;
26          y->p=x->p;
27          if(x->p==nil)
28              root=y;
29          else
30              x->p->c[x!=x->p->c[0]]=y;
31          y->c[d]=x;
32          x->p=y;
33          y->s=x->s;
34          x->s=x->c[0]->s+x->c[1]->s+1;
35      }
36      void insert_fixup(node*z){
37          while(!z->p->b){
38              int d=z->p==z->p->p->c[0];
39              node*y=z->p->p->c[d];
40              if(!y->b)
41                  z->p->b=1,y->b=1,(z=z->p->p)->b=0;
42              else{
43                  if(z==z->p->c[d])
44                      rotate(z=z->p,!d);
45                  z->p->b=1;
46                  z->p->p->b=0;
47                  rotate(z->p->p,d);
48              }
49          }
50          root->b=1;
51      }
52      void erase(node*z){
53          node*y;
54          for(y=z;y!=nil;y=y->p)
55              --y->s;
56          if(z->c[0]==nil||z->c[1]==nil)
57              y=z;
```

```
58          else{
59              for(y=z->c[1];y->c[0]!=nil;)
60                  y=y->c[0];
61              z->v=y->v;
62              y=z->c[1];
63              while(y->c[0]!=nil)
64                  --y->s,y=y->c[0];
65          }
66          node*x=y->c[y->c[0]==nil];
67          x->p=y->p;
68          if(y->p==nil)
69              root=x;
70          else
71              y->p->c[y!=y->p->c[0]]=x;
72          if(y->b)
73              erase_fixup(x);
74          delete y;
75      }
76      void erase_fixup(node*x){
77          while(x!=root&&x->b){
78              int d=x==x->p->c[0];
79              node*w=x->p->c[d];
80              if(!w->b){
81                  w->b=1;
82                  x->p->b=0;
83                  rotate(x->p,!d);
84                  w=x->p->c[d];
85              }
86              if(w->c[0]->b&&w->c[1]->b)
87                  w->b=0,x=x->p;
88              else{
89                  if(w->c[d]->b)
90                      w->c[!d]->b=1,w->b=0,rotate(w,d),w=x->p->c[d];
91                  w->b=x->p->b;
92                  x->p->b=1;
93                  w->c[d]->b=1;
94                  rotate(x->p,!d);
95                  x=root;
96              }
97          }
98          x->b=1;
```

```
 99          }
100          node*clone(node*x,node*y){
101              if(x.size==0)
102                  return nil;
103              node*z=new node(*x);
104              z->c[0]=clone(x->c[0],z);
105              z->c[1]=clone(x->c[1],z);
106              z->p=y;
107              return z;
108          }
109          node*precursor(node*x){
110              if(x->c[0]->count){
111                  for(x=x->c[0];x->c[1]->count;)
112                      x=x->c[1];
113                  return x;
114              }else{
115                  node*y=x->p;
116                  while(y->count&&x==y->c[0])
117                      x=y,y=y->p;
118                  return y;
119              }
120          }
121          node*successor(node*x){
122              if(x->c[1]->count){
123                  for(x=x->c[1];x->c[0]->count;)
124                      x=x->c[0];
125                  return x;
126              }else{
127                  node*y=x->p;
128                  while(y->count&&x==y->c[1])
129                      x=y,y=y->p;
130                  return y;
131              }
132          }
133          RedBlackTree(){
134              root=nil=(node*)malloc(sizeof(node));
135              nil->b=1;
136              nil->s=0;
137          }
138          RedBlackTree(const RedBlackTree&a){
139              nil=new node(*a.nil);
```

```
140          root=clone(a.root,nil);
141      }
142      ~RedBlackTree(){
143          clear(root);
144          free(nil);
145      }
146      RedBlackTree&operator=(const RedBlackTree&a){
147          clear(root);
148          root=clone(a.root,nil);
149          return*this;
150      }
151      node*begin(){
152          node*z=root;
153          while(z!=nil&&z−>c[0]!=nil)
154              z=z−>c[0];
155          return z;
156      }
157      node*reverse_begin(){
158          node*z=root;
159          while(z!=nil&&z−>c[1]!=nil)
160              z=z−>c[1];
161          return z;
162      }
163      node*end(){
164          return nil;
165      }
166      node*reverse_end(){
167          return nil;
168      }
169      void clear(){
170          clear(root);
171          root=nil;
172      }
173      void insert(T a){
174          node*y=nil,*x=root;
175          while(x!=nil)
176              y=x,++x−>s,x=x−>c[C()(x−>v,a)];
177          node*z=new node(a,nil,nil,y,0,1);
178          if(y==nil)
179              root=z;
180          else
```

```
181             y−>c[C()(y−>v,z−>v)]=z;
182         insert_fixup(z);
183     }
184     void erase(T a){
185         node*z=root;
186         for(;;)
187             if(C()(a,z−>v))
188                 z=z−>c[0];
189             else if(C()(z−>v,a))
190                 z=z−>c[1];
191             else
192                 break;
193         erase(z);
194     }
195     int count(T a){
196         return count_less_equal(a)−count_less(a);
197     }
198     int count_less(T a){
199         int r=0;
200         node*z=root;
201         while(z!=nil)
202             if(C()(z−>v,a))
203                 r+=z−>c[0]−>s+1,z=z−>c[1];
204             else
205                 z=z−>c[0];
206         return r;
207     }
208     int count_less_equal(T a){
209         int r=0;
210         node*z=root;
211         while(z!=nil){
212             if(!C()(a,z−>v))
213                 r+=z−>c[0]−>s+1,z=z−>c[1];
214             else
215                 z=z−>c[0];
216         }
217         return r;
218     }
219     int count_greater(T a){
220         int r=0;
221         node*z=root;
```

```
222          while(z!=nil)
223              if(C()(a,z->v))
224                  r+=z->c[1]->s+1,z=z->c[0];
225              else
226                  z=z->c[1];
227          return r;
228      }
229      int count_greater_equal(T a){
230          int r=0;
231          node*z=root;
232          while(z!=nil)
233              if(!C()(z->v,a))
234                  r+=z->c[1]->s+1,z=z->c[0];
235              else
236                  z=z->c[1];
237          return r;
238      }
239      node*nth_element(int a){
240          node*z=root;
241          for(;;)
242              if(z->c[0]->s>=a)
243                  z=z->c[0];
244              else if((z->c[0]->s+1)<a)
245                  a-=z->c[0]->s+1,z=z->c[1];
246              else
247                  return z;
248      }
249      node*precursor(T a){
250          node*z=root,*r=nil;
251          while(z!=nil)
252              if(C()(z->v,a))
253                  r=z,z=z->c[1];
254              else
255                  z=z->c[0];
256          return r;
257      }
258      node*successor(T a){
259          node*z=root,*r=nil;
260          while(z!=nil)
261              if(C()(a,z->v))
262                  r=z,z=z->c[0];
```

```
263              else
264                  z=z->c[1];
265          return r;
266      }
267      node*find(T a){
268          node*z=root,*r=nil;
269          while(z!=nil)
270              if(C()(a,z->v))
271                  z=z->c[0];
272              else if(C()(z->v,a))
273                  z=z->c[1];
274              else
275                  break;
276          return r;
277      }
278      node*lower_bound(T a){
279          node*z=root,*r=nil;
280          while(z!=nil)
281              if(C()(z->v,a))
282                  r=z,z=z->c[1];
283              else if(C()(a,z->v))
284                  z=z->c[0];
285              else
286                  r=z,z=z->c[0];
287          return r;
288      }
289      node*upper_bound(T a){
290          return successor(a);
291      }
292      pair<node*,node*> equal_range(T a){
293          return make_pair(lower_bound(a),upper_bound(a));
294      }
295      int size(){
296          return root->s;
297      }
298      int empty(){
299          return !root->s;
300      }
301      T front(){
302          return*begin();
303      }
```

```
304        T back(){
305            return*reverse_begin();
306        }
307    };
```

2.8   Self-Adjusting Top Tree

Self-Adjusting Top Tree.hpp (12629 bytes, 443 lines)

```
1    #include<bits/stdc++.h>
2    using namespace std;
3    struct SelfAdjustingTopTree{
4        const static int inf=~0u>>1;
5        static void gmin(int&a,int b){
6            a=min(a,b);
7        }
8        static void gmax(int&a,int b){
9            a=max(a,b);
10       }
11       struct treap{
12           SelfAdjustingTopTree*tr;
13           treap(struct SelfAdjustingTopTree*a,int n):
14               tr(a),ns(n){
15           }
16           struct node{
17               node(){
18               }
19               node(int a,int b,int c,int d,int e){
20                   ch[0]=ch[1]=0;
21                   val=a;
22                   fix=rand();
23                   add=0;
24                   mi=vmi=b;
25                   mx=vmx=c;
26                   sum=vsum=d;
27                   siz=vsiz=e;
28                   sam=inf;
29               }
30               node*ch[2];
```

```
31              int val,fix,vmi,vmx,vsum,vsiz,mi,mx,sum,siz,add,sam;
32          };
33          vector<node>ns;
34          void down(node*a){
35              if(a->sam!=inf){
36                  a->mi=a->mx=a->vmi=a->vmx=a->sam;
37                  a->vsum=a->sam*a->vsiz;
38                  a->sum=a->sam*a->siz;
39                  (&tr->ns[0]+(a-&ns[0]))->viradd=0;
40                  (&tr->ns[0]+(a-&ns[0]))->virsam=a->sam;
41                  (&tr->ns[0]+(a-&ns[0]))->add=0;
42                  (&tr->ns[0]+(a-&ns[0]))->sam=a->sam;
43                  for(int i=0;i<=1;++i)
44                      if(a->ch[i])
45                          a->ch[i]->add=0,a->ch[i]->sam=a->sam;
46                  a->sam=inf;
47              }
48              if(a->add){
49                  a->mi+=a->add;
50                  a->mx+=a->add;
51                  a->vmi+=a->add;
52                  a->vmx+=a->add;
53                  a->vsum+=a->add*a->vsiz;
54                  a->sum+=a->add*a->siz;
55                  (&tr->ns[0]+(a-&ns[0]))->viradd+=a->add;
56                  (&tr->ns[0]+(a-&ns[0]))->add+=a->add;
57                  for(int i=0;i<=1;++i)
58                      if(a->ch[i])
59                          a->ch[i]->add+=a->add;
60                  a->add=0;
61              }
62          }
63          void update(node*a){
64              for(int i=0;i<=1;++i)
65                  if(a->ch[i])
66                      down(a->ch[i]);
67              a->mi=a->vmi;
68              for(int i=0;i<=1;++i)
69                  if(a->ch[i])
70                      gmin(a->mi,a->ch[i]->mi);
71              a->mx=a->vmx;
```

```
72              for(int i=0;i<=1;++i)
73                  if(a->ch[i])
74                      gmax(a->mx,a->ch[i]->mx);
75              a->sum=a->vsum;
76              for(int i=0;i<=1;++i)
77                  if(a->ch[i])
78                      a->sum+=a->ch[i]->sum;
79              a->siz=a->vsiz;
80              for(int i=0;i<=1;++i)
81                  if(a->ch[i])
82                      a->siz+=a->ch[i]->siz;
83          }
84          void rotate(node*&a,int d){
85              node*b=a->ch[d];
86              a->ch[d]=b->ch[!d];
87              b->ch[!d]=a;
88              update(a);
89              update(b);
90              a=b;
91          }
92          void insert(node*&a,node*b){
93              if(!a)
94                  a=b;
95              else{
96                  down(a);
97                  int d=b->val>a->val;
98                  insert(a->ch[d],b);
99                  update(a);
100                 if(a->ch[d]->fix<a->fix)
101                     rotate(a,d);
102             }
103         }
104         void erase(node*&a,int b){
105             down(a);
106             if(a->val==b){
107                 if(!a->ch[0])
108                     a=a->ch[1];
109                 else if(!a->ch[1])
110                     a=a->ch[0];
111                 else{
112                     int d=a->ch[1]->fix<a->ch[0]->fix;
```

```
113                     down(a->ch[d]);
114                     rotate(a,d);
115                     erase(a->ch[!d],b);
116                     update(a);
117                 }
118             }else{
119                 int d=b>a->val;
120                 erase(a->ch[d],b);
121                 update(a);
122             }
123         }
124     };
125     int n;
126     SelfAdjustingTopTree(int _n,vector<int>*to,int*we,int rt):
127         trp(this,_n+1),ns(_n+1),n(_n){
128         build(to,we,rt);
129     }
130     struct node{
131         node(){}
132         node(int a,node*b){
133             ch[0]=ch[1]=0;
134             pr=b;
135             vir=0;
136             val=a;
137             mi=mx=a;
138             siz=1;
139             rev=virsum=add=0;
140             virmi=inf;
141             virmx=-inf;
142             sam=inf;
143             virsam=inf;
144             virsiz=0;
145             viradd=0;
146         }
147         node*ch[2],*pr;
148         int val,mi,mx,sum,virmi,virmx,virsum,virsam,viradd,virsiz,rev,sam,
        siz,add;
149         treap::node*vir;
150     };
151     vector<node>ns;
152     treap trp;
```

```
153     int direct(node*a){
154         if(!a−>pr)
155             return 3;
156         else if(a==a−>pr−>ch[0])
157             return 0;
158         else if(a==a−>pr−>ch[1])
159             return 1;
160         else
161             return 2;
162     }
163     void down(node*a){
164         if(a−>rev){
165             swap(a−>ch[0],a−>ch[1]);
166             for(int i=0;i<=1;++i)
167                 if(a−>ch[i])
168                     a−>ch[i]−>rev^=1;
169             a−>rev=0;
170         }
171         if(a−>sam!=inf){
172             a−>val=a−>mi=a−>mx=a−>sam;
173             a−>sum=a−>sam*a−>siz;
174             for(int i=0;i<=1;++i)
175                 if(a−>ch[i])a−>ch[i]−>sam=a−>sam,a−>ch[i]−>add=0;
176             a−>sam=inf;
177         }
178         if(a−>add){
179             a−>val+=a−>add;
180             a−>mi+=a−>add;
181             a−>mx+=a−>add;
182             a−>sum+=a−>add*a−>siz;
183             for(int i=0;i<=1;++i)
184                 if(a−>ch[i])a−>ch[i]−>add+=a−>add;
185             a−>add=0;
186         }
187         if(a−>virsam!=inf){
188             if(a−>virsiz){
189                 a−>virmi=a−>virmx=a−>virsam;
190                 a−>virsum=a−>virsam*a−>virsiz;
191                 if(a−>vir)
192                     a−>vir−>add=0,a−>vir−>sam=a−>virsam;
193                 for(int i=0;i<=1;++i)
```

```
194                         if(a->ch[i])
195                             a->ch[i]->viradd=0,a->ch[i]->virsam=a->virsam;
196                     }
197                     a->virsam=inf;
198                 }
199                 if(a->viradd){
200                     if(a->virsiz){
201                         a->virmi+=a->viradd;
202                         a->virmx+=a->viradd;
203                         a->virsum+=a->viradd*a->virsiz;
204                         if(a->vir)a->vir->add+=a->viradd;
205                         for(int i=0;i<=1;++i)
206                             if(a->ch[i])
207                                 a->ch[i]->viradd+=a->viradd;
208                     }
209                     a->viradd=0;
210                 }
211             }
212             void update(node*a){
213                 for(int i=0;i<=1;++i)
214                     if(a->ch[i])
215                         down(a->ch[i]);
216                 if(a->vir)
217                     trp.down(a->vir);
218                 a->mi=a->val;
219                 for(int i=0;i<=1;++i)
220                     if(a->ch[i])
221                         gmin(a->mi,a->ch[i]->mi);
222                 a->virmi=inf;
223                 for(int i=0;i<=1;++i)
224                     if(a->ch[i])
225                         gmin(a->virmi,a->ch[i]->virmi);
226                 if(a->vir)
227                     gmin(a->virmi,a->vir->mi);
228                 a->mx=a->val;
229                 for(int i=0;i<=1;++i)
230                     if(a->ch[i])
231                         gmax(a->mx,a->ch[i]->mx);
232                 a->virmx=-inf;
233                 for(int i=0;i<=1;++i)
234                     if(a->ch[i])
```

```
235                        gmax(a->virmx,a->ch[i]->virmx);
236            if(a->vir)
237                gmax(a->virmx,a->vir->mx);
238            a->sum=a->val;
239            for(int i=0;i<=1;++i)
240                if(a->ch[i])
241                    a->sum+=a->ch[i]->sum;
242            a->virsum=0;
243            for(int i=0;i<=1;++i)
244                if(a->ch[i])
245                    a->virsum+=a->ch[i]->virsum;
246            if(a->vir)
247                a->virsum+=a->vir->sum;
248            a->siz=1;
249            for(int i=0;i<=1;++i)
250                if(a->ch[i])
251                    a->siz+=a->ch[i]->siz;
252            a->virsiz=0;
253            for(int i=0;i<=1;++i)
254                if(a->ch[i])
255                    a->virsiz+=a->ch[i]->virsiz;
256            if(a->vir)
257                a->virsiz+=a->vir->siz;
258        }
259        void setchd(node*a,node*b,int d){
260            a->ch[d]=b;
261            if(b)
262                b->pr=a;
263            update(a);
264        }
265        void connect(node*a,node*b){
266            down(a);
267            *(&trp.ns[0]+(a-&ns[0]))=treap::node(a-&ns[0],min(a->virmi,a->mi),
        max(a->virmx,a->mx),a->virsum+a->sum,a->virsiz+a->siz);
268            trp.insert(b->vir,&trp.ns[0]+(a-&ns[0]));
269        }
270        void disconnect(node*a,node*b){
271            trp.erase(b->vir,a-&ns[0]);
272        }
273        void rotate(node*a){
274            node*b=a->pr,*c=a->pr->pr;
```

```
275            int d1=direct(a),d2=direct(b);
276            setchd(b,a->ch[!d1],d1);
277            setchd(a,b,!d1);
278            if(d2<2)
279                setchd(c,a,d2);
280            else if(d2==2){
281                disconnect(b,c);
282                connect(a,c);
283                a->pr=c;
284            }else
285                a->pr=0;
286        }
287        void release(node*a){
288            if(direct(a)<2)
289                release(a->pr);
290            else if(a->pr)
291                disconnect(a,a->pr),connect(a,a->pr);
292            down(a);
293        }
294        void splay(node*a){
295            release(a);
296            while(direct(a)<2){
297                node*b=a->pr;
298                if(!b->pr||direct(b)>1)
299                    rotate(a);
300                else if(direct(a)==direct(b))
301                    rotate(b),rotate(a);
302                else
303                    rotate(a),rotate(a);
304            }
305        }
306        node*access(node*a){
307            node*b=0;
308            while(a){
309                splay(a);
310                if(a->ch[1])
311                    connect(a->ch[1],a);
312                if(b)
313                    disconnect(b,a);
314                setchd(a,b,1);
315                b=a;
```

```
316          a=a−>pr;
317        }
318        return b;
319      }
320      void evert(node*a){
321        access(a);
322        splay(a);
323        a−>rev=1;
324      }
325      int qchain(node*a,node*b,int d){
326        access(a);
327        node*c=access(b);
328        splay(c);
329        splay(a);
330        int ret=c−>val;
331        if(d==1){
332          if(a!=c)
333            gmin(ret,a−>mi);
334          if(c−>ch[1])
335            down(c−>ch[1]),gmin(ret,c−>ch[1]−>mi);
336        }else if(d==2){
337          if(a!=c)
338            gmax(ret,a−>mx);
339          if(c−>ch[1])
340            down(c−>ch[1]),gmax(ret,c−>ch[1]−>mx);
341        }else if(d==3){
342          if(a!=c)
343            ret+=a−>sum;
344          if(c−>ch[1])
345            down(c−>ch[1]),ret+=c−>ch[1]−>sum;
346        }
347        return ret;
348      }
349      void mchain(node*a,node*b,int u,int d){
350        access(a);
351        node*c=access(b);
352        splay(c);
353        splay(a);
354        if(d==1){
355          c−>val+=u;
356          if(a!=c)
```

```
357              a->add=u,disconnect(a,c),connect(a,c);
358          if(c->ch[1])
359              down(c->ch[1]),c->ch[1]->add=u;
360      }else if(d==2){
361          c->val=u;
362          if(a!=c)
363              a->sam=u,disconnect(a,c),connect(a,c);
364          if(c->ch[1])
365              down(c->ch[1]),c->ch[1]->sam=u;
366      }
367      update(c);
368  }
369  int qtree(node*a,int d){
370      access(a);
371      splay(a);
372      int ret=a->val;
373      if(d==1){
374          if(a->vir)
375              trp.down(a->vir),gmin(ret,a->vir->mi);
376      }else if(d==2){
377          if(a->vir)
378              trp.down(a->vir),gmax(ret,a->vir->mx);
379      }else if(d==3){
380          if(a->vir)
381              trp.down(a->vir),ret+=a->vir->sum;
382      }
383      return ret;
384  }
385  void mtree(node*a,int u,int d){
386      access(a);
387      splay(a);
388      if(d==1){
389          a->val+=u;
390          if(a->vir)
391              trp.down(a->vir),a->vir->add=u;
392      }else if(d==2){
393          a->val=u;
394          if(a->vir)
395              trp.down(a->vir),a->vir->sam=u;
396      }
397      update(a);
```

```
398          }
399      void stparent(node*a,node*b){
400          access(b);
401          if(access(a)!=a){
402              splay(a);
403              node*c=a->ch[0];
404              down(c);
405              while(c->ch[1])
406                  c=c->ch[1],down(c);
407              splay(c);
408              c->ch[1]=0;
409              update(c);
410              access(b);
411              splay(b);
412              connect(a,b);
413              a->pr=b;
414              update(b);
415          }
416      }
417      void build(vector<int>*to,int*we,int rt){
418          vector<int>pr(n);
419          vector<int>vec;
420          queue<int>qu;
421          qu.push(rt);
422          while(!qu.empty()){
423              int u=qu.front();
424              qu.pop();
425              vec.push_back(u);
426              for(int i=0;i<to[u].size();++i){
427                  int v=to[u][i];
428                  if(v!=pr[u])
429                      qu.push(v),pr[v]=u;
430              }
431          }
432          for(int i=0;i<n;++i){
433              int u=vec[i];
434              ns[u]=node(we[u],pr[u]?&ns[0]+pr[u]:0);
435          }
436          for(int i=n-1;i>=0;--i){
437              int u=vec[i];
438              update(&ns[0]+u);
```

```
439            if(pr[u])
440                connect(&ns[0]+u,&ns[0]+pr[u]);
441        }
442    }
443 };
```

2.9   Skew Heap

Skew Heap.hpp (1220 bytes, 61 lines)

```
 1  #include<bits/stdc++.h>
 2  using namespace std;
 3  template<class T,class C>struct SkewHeap{
 4      SkewHeap():
 5          root(0),siz(0){
 6      }
 7      ~SkewHeap(){
 8          clear(root);
 9      }
10      struct node{
11          node(T _val):
12              val(_val){
13              ch[0]=ch[1]=0;
14          }
15          T val;
16          node*ch[2];
17      }*root;
18      int siz;
19      node*merge(node*x,node*y){
20          if(!x)
21              return y;
22          if(!y)
23              return x;
24          if(C()(y->val,x->val))
25              swap(x,y);
26          swap(x->ch[0],x->ch[1]=merge(x->ch[1],y));
27          return x;
28      }
29      void clear(node*x){
```

```
30          if(x){
31              clear(x->ch[0]);
32              clear(x->ch[1]);
33              delete x;
34          }
35      }
36      void clear(){
37          clear(root);
38          root=0;
39          siz=0;
40      }
41      void push(T a){
42          root=merge(root,new node(a));
43          ++siz;
44      }
45      T top(){
46          return root->val;
47      }
48      void pop(){
49          root=merge(root->ch[0],root->ch[1]);
50          --siz;
51      }
52      void merge(SkewHeap<T,C>&a){
53          root=merge(root,a.root);
54          a.root=0;
55          siz+=a.siz;
56          a.siz=0;
57      }
58      int size(){
59          return siz;
60      }
61 };
```

CHAPTER 3

Graph Algorithms

3.1   Chordality Test

Chordality Test.hpp (1343 bytes, 42 lines)

```
 1  #include<bits/stdc++.h>
 2  using namespace std;
 3  struct ChordalityTest{
 4      int n,ns;
 5      vector<vector<int> >to;
 6      ChordalityTest(int _n):
 7          n(_n),ns(n),to(n+1){
 8      }
 9      void add(int u,int v){
10          to[u].push_back(v),to[v].push_back(u);
11      }
12      bool run(){
13          vector<int>pos(n+1),idx(n+2),lab(n+1),tab(n+1);
14          vector<list<int>>qu(n);
15          for(int i=1;i<=n;++i)
16              qu[0].push_back(i);
17          for(int b=0,i=1,u=0;i<=n;++i,u=0){
18              for(;u?++b,0:1;--b)
19                  for(auto j=qu[b].begin();j!=qu[b].end()&&!u;qu[b].erase(j++)
    )
20                      if(!pos[*j]&&lab[*j]==b)
21                          u=*j;
22              pos[u]=ns,idx[ns--]=u;
23              for(int v:to[u])
24                  if(!pos[v])
25                      b=max(b,++lab[v]),qu[lab[v]].push_back(v);}
26          for(int i=1,u=idx[1],v=-1;i<=n;++i,u=idx[i],v=-1){
27              for(int w:to[u])
28                  if(pos[w]>pos[u]&&(v==-1||pos[w]<pos[v]))
29                      v=w;
30              if(v!=-1){
31                  for(int w:to[v])
32                      tab[w]=1;
33                  for(int w:to[u])
34                      if(pos[w]>pos[u]&&w!=v&&!tab[w])
35                          return false;
36                  for(int w:to[v])
```

```
37                          tab[w]=0;
38                  }
39              }
40          return true;
41          }
42  };
```

3.2   Dominator Tree

Dominator Tree.hpp (2916 bytes, 94 lines)

```
1   #include<bits/stdc++.h>
2   using namespace std;
3   struct DominatorTree{
4       int n,r;
5       vector<vector<int> >to,rto,chd,rsemi;
6       vector<int>dfn,res,prt,rdfn,semi,misemi;
7       DominatorTree(int _n,int _r):n(_n),r(_r),to(n+1),rto(n+1),dfn(n+1),res(
        n+1),prt(n+1),rdfn(1),semi(n+1),misemi(n+1),chd(n+1),rsemi(n+1){
8       }
9       int fd(int a){
10          stack<int>stk;
11          for(int b=a;prt[b]!=prt[prt[b]];b=prt[b])
12              stk.push(b);
13          for(int b;stk.empty()?0:(b=stk.top(),stk.pop(),1);){
14              if(dfn[semi[misemi[prt[b]]]]<dfn[semi[misemi[b]]])
15                  misemi[b]=misemi[prt[b]];
16              prt[b]=prt[prt[b]];
17          }
18          return prt[a];
19      }
20      void add(int a,int b){
21          to[a].push_back(b);
22          rto[b].push_back(a);
23      }
24      void dfs(){
25          stack<pair<int,int> >stk;
26          semi[r]=r;
27          for(stk.push(make_pair(r,0));!stk.empty();){
```

```
28              int a=stk.top().first,i=stk.top().second;
29              stk.pop();
30              if(!i)
31                  dfn[a]=rdfn.size(),rdfn.push_back(a);
32              if(i<to[a].size()){
33                  stk.push(make_pair(a,i+1));
34                  int b=to[a][i];
35                  if(!semi[b])
36                      semi[b]=a,chd[a].push_back(b),
37                      stk.push(make_pair(b,0));
38              }
39          }
40          semi[r]=0;
41      }
42      void calcsemi(){
43          for(int i=1;i<=n;++i)
44              prt[i]=i,misemi[i]=i;
45          for(int i=rdfn.size()-1;i>=1;--i){
46              int a=rdfn[i];
47              for(int b:rto[a]){
48                  if(!dfn[b])
49                      continue;
50                  if(dfn[b]<dfn[a]){
51                      if(dfn[b]<dfn[semi[a]])
52                          semi[a]=b;
53                  }else{
54                      int c=fd(b);
55                      if(dfn[semi[c]]<dfn[semi[a]])
56                          semi[a]=semi[c];
57                      if(dfn[semi[misemi[b]]]<dfn[semi[a]])
58                          semi[a]=semi[misemi[b]];
59                  }
60              }
61              for(int b:chd[a])
62                  prt[b]=a;
63          }
64      }
65      void calcres(){
66          for(int i=1;i<=n;++i)
67              prt[i]=i,misemi[i]=i,rsemi[semi[i]].push_back(i);
68          for(int i=rdfn.size()-1;i>=1;--i){
```

```
69          int a=rdfn[i];
70          for(int b:rsemi[a]){
71              fd(b);
72              int c=misemi[b];
73              if(dfn[semi[c]]>dfn[semi[prt[b]]])
74                  c=prt[b];
75              if(semi[c]==semi[b])
76                  res[b]=semi[b];
77              else
78                  res[b]=−c;}
79          for(int b:chd[a])
80              prt[b]=a;
81      }
82      for(int i=1;i<rdfn.size();++i){
83          int a=rdfn[i];
84          if(res[a]<0)
85              res[a]=res[−res[a]];
86      }
87  }
88  vector<int>run(){
89      dfs();
90      calcsemi();
91      calcres();
92      return res;
93  }
94 };
```

3.3   Maximal Clique Count

Maximal Clique Count.hpp (927 bytes, 34 lines)

```
1 #include<bits/stdc++.h>
2 using namespace std;
3 template<int N>struct MaximalCliqueCount{
4     int n,r;
5     vector<bitset<N> >e,rht,msk;
6     MaximalCliqueCount(int _n):
7         n(_n),e(n),rht(n),msk(n),r(0){
8     }
```

```cpp
 9        void add(int u,int v){
10            e[u-1][v-1]=e[v-1][u-1]=1;
11        }
12        void dfs(int u,bitset<N>cur,bitset<N>can){
13            if(cur==can){
14                ++r;
15                return;
16            }
17            for(int v=0;v<u;++v)
18                if(can[v]&&!cur[v]&&(e[v]&rht[u]&can)==(rht[u]&can))
19                    return;
20            for(int v=u+1;v<n;++v)
21                if(can[v])
22                    dfs(v,cur|msk[v],can&e[v]);
23        }
24        int run(){
25            for(int i=1;i<=n;++i){
26                rht[i-1]=bitset<N>(string(n-i,'1')+string(i,'0'));
27                msk[i-1]=bitset<N>(1)<<i-1;
28                e[i-1]|=msk[i-1];
29            }
30            for(int i=0;i<n;++i)
31                dfs(i,msk[i],e[i]);
32            return r;
33        }
34 };
```

3.4   Maximal Planarity Test

Maximal Planarity Test.hpp (5195 bytes, 165 lines)

```cpp
1 #include<bits/stdc++.h>
2 using namespace std;
3 struct MaximalPlanarityTesting{
4     int n,m;
5     vector<set<int> >to2;
6     vector<vector<int> >to;
7     vector<int>dec,rmd,mrk,invc,rt;
8     vector<list<int>::iterator>dpos,pos;
```

```cpp
 9        bool order(int v1,int v2,int vn){
10            rt[0]=v1;
11            rt[1]=v2;
12            rt[n−1]=vn;
13            fill(invc.begin(),invc.end(),0);
14            invc[v1]=1;
15            invc[v2]=1;
16            invc[vn]=1;
17            list<int>deg;
18            dpos[vn]=deg.insert(deg.begin(),vn);
19            fill(dec.begin(),dec.end(),0);
20            dec[v1]=2;
21            dec[v2]=2;
22            dec[vn]=2;
23            for(int i=n−1;i>=2;−−i){
24                if(deg.empty())
25                    return false;
26                int v=*deg.begin();
27                deg.erase(deg.begin());
28                invc[v]=−1;
29                rt[i]=v;
30                for(int u:to[v]){
31                    if(invc[u]==1){
32                        if(u!=v1&&u!=v2&&dec[u]==2)
33                            deg.erase(dpos[u]);
34                        −−dec[u];
35                        if(u!=v1&&u!=v2&&dec[u]==2)
36                            dpos[u]=deg.insert(deg.begin(),u);
37                    }else if(invc[u]==0)
38                        invc[u]=2;
39                }
40                for(int u:to[v])
41                    if(invc[u]==2)
42                        for(int w:to[u])
43                            if(invc[w]==1){
44                                if(w!=v1&&w!=v2&&dec[w]==2)
45                                    deg.erase(dpos[w]);
46                                ++dec[w];
47                                if(w!=v1&&w!=v2&&dec[w]==2)
48                                    dpos[w]=deg.insert(deg.begin(),w);
49                                ++dec[u];
```

```
50                         }else if(invc[w]==2)
51                             ++dec[u];
52             for(int u:to[v]){
53                 if(invc[u]==2){
54                     invc[u]=1;
55                     if(dec[u]==2)
56                         dpos[u]=deg.insert(deg.begin(),u);
57                 }
58             }
59         }
60         return true;
61     }
62     bool embed(){
63         list<int>ext;
64         int mker=0;
65         fill(mrk.begin(),mrk.end(),0);
66         pos[rt[1]]=ext.insert(ext.begin(),rt[1]);
67         pos[rt[2]]=ext.insert(ext.begin(),rt[2]);
68         pos[rt[0]]=ext.insert(ext.begin(),rt[0]);
69         fill(rmd.begin(),rmd.end(),0);
70         rmd[rt[1]]=1;
71         rmd[rt[2]]=1;
72         rmd[rt[0]]=1;
73         for(int i=3;i<n;++i){
74             int v=rt[i];
75             rmd[v]=1;
76             vector<int>can;
77             ++mker;
78             for(int u:to[v])
79                 if(rmd[u])
80                     mrk[u]=mker,can.push_back(u);
81             int start=-1,end=-1;
82             for(int u:can){
83                 list<int>::iterator it=pos[u];
84                 if(it==list<int>::iterator())
85                     return false;
86                 if(it==ext.begin()){
87                     if(start!=-1)
88                         return false;
89                     start=u;
90                 }else{
```

```
91              list<int>::iterator tmp=it;
92              if(mrk[*(−−tmp)]!=mker){
93                  if(start!=−1)
94                      return false;
95                  start=u;
96              }
97          }
98          list<int>::iterator tmp=it;++tmp;
99          if(tmp==ext.end()){
100             if(end!=−1)
101                 return false;
102             end=u;
103         }else{
104             if(mrk[*tmp]!=mker){
105                 if(end!=−1)
106                     return false;
107                 end=u;
108             }
109         }
110     }
111     if(start==−1||end==−1)
112         return false;
113     for(int u:can)
114         if(u!=start&&u!=end)
115             ext.erase(pos[u]),pos[u]=list<int>::iterator();
116     pos[v]=ext.insert(pos[end],v);
117     }
118     return true;
119 }
120 bool istri(int u,int v,int w){
121     return to2[u].count(v)&&to2[v].count(w)&&to2[w].count(u);
122 }
123 MaximalPlanarityTesting(int _n):
124     n(_n),to(n),to2(n),m(0),rt(n),invc(n),dec(n),dpos(n),pos(n),rmd(n),
    mrk(n){
125 }
126 void add(int u,int v){
127     to[u−1].push_back(v−1);
128     to[v−1].push_back(u−1);
129     to2[u−1].insert(v−1);
130     to2[v−1].insert(u−1);++m;
```

```
131         }
132     bool run(){
133         if(n==1&&m==0)
134             return true;
135         if(n==2&&m==1)
136             return true;
137         if(n==3&&m==3)
138             return true;
139         if(n<=3)
140             return false;
141         if(m!=3*n−6)
142             return false;
143         int v1;
144         for(v1=0;v1<n;++v1)
145             if(to[v1].size()<3)
146                 return false;
147         for(v1=0;v1<n;++v1)
148             if(to[v1].size()<=5)
149                 break;
150         if(v1>=n)
151             return false;
152         int v2=to[v1].back();
153         for(int i=0;i+1<to[v1].size();++i){
154             int vn=to[v1][i];
155             if(istri(v1,v2,vn)){
156                 if(!order(v1,v2,vn))
157                     continue;
158                 if(!embed())
159                     continue;
160                 return true;
161             }
162         }
163         return false;
164     }
165 };
```

3.5   Maximum Flow

Maximum Flow.hpp (2311 bytes, 79 lines)

```cpp
 1  #include<bits/stdc++.h>
 2  using namespace std;
 3  template<class T>struct MaximumFlow{
 4      struct edge{
 5          int v;
 6          T c,l;
 7          edge(int _v,T _c):
 8              v(_v),c(_c),l(_c){
 9          }
10      };
11      int n,src,snk;
12      vector<edge>egs;
13      vector<vector<int> >bge;
14      vector<int>hei,gap,cur,frm;
15      MaximumFlow(int _n,int _src,int _snk):
16          bge(_n),hei(_n,_n),gap(_n+1),n(_n),cur(_n),frm(_n),src(_src−1),snk(
        _snk−1){
17      }
18      void lab(){
19          hei[snk]=0;
20          queue<int>qu;
21          qu.push(snk);
22          for(int u;qu.empty()?0:(u=qu.front(),qu.pop(),1);)
23              for(int i=0;i<bge[u].size();++i){
24                  edge&e=egs[bge[u][i]],&ev=egs[bge[u][i]^1];
25                  if(ev.c>0&&hei[e.v]==n)
26                      hei[e.v]=hei[u]+1,qu.push(e.v);
27              }
28          for(int i=0;i<n;++i)
29              ++gap[hei[i]];
30      }
31      T aug(){
32          T f=0;
33          for(int u=snk;u!=src;u=egs[frm[u]^1].v)
34              if(f<=0||f>egs[frm[u]].c)
35                  f=egs[frm[u]].c;
36          for(int u=snk;u!=src;u=egs[frm[u]^1].v)
37              egs[frm[u]].c−=f,egs[frm[u]^1].c+=f;
38          return f;
39      }
```

```
40      void add(int u,int v,T c){
41          bge[u−1].push_back(egs.size());
42          egs.push_back(edge(v−1,c));
43          bge[v−1].push_back(egs.size());
44          egs.push_back(edge(u−1,0));
45      }
46      T run(){
47          lab();
48          T r=0;
49          for(int u=src;hei[src]!=n;){
50              if(u==snk)
51                  r+=aug(),u=src;
52              int f=0;
53              for(int i=cur[u];i<bge[u].size();++i){
54                  edge&e=egs[bge[u][i]];
55                  if(e.c>0&&hei[u]==hei[e.v]+1){
56                      f=1;
57                      frm[e.v]=bge[u][i];
58                      u=e.v;
59                      break;
60                  }
61              }
62              if(!f){
63                  int mh=n−1;
64                  for(int i=0;i<bge[u].size();++i){
65                      edge&e=egs[bge[u][i]];
66                      if(e.c>0&&mh>hei[e.v])
67                          mh=hei[e.v];
68                  }
69                  if(!−−gap[hei[u]])
70                      break;
71                  ++gap[hei[u]=mh+1];
72                  cur[u]=0;
73                  if(u!=src)
74                      u=egs[frm[u]^1].v;
75              }
76          }
77          return r;
78      }
79  };
```

3.6   Maximum Matching

Maximum Matching.hpp (3123 bytes, 112 lines)

```cpp
#include<bits/stdc++.h>
using namespace std;
struct MaximumMatching{
    int n;
    vector<int>res,nxt,mrk,vis,top,prt,rnk;
    vector<vector<int> >to;
    queue<int>qu;
    MaximumMatching(int _n):
        n(_n),res(n+1),nxt(n+1),mrk(n+1),vis(n+1),top(n+1),to(n+1),prt(n+1)
    ,rnk(n+1){
    }
    int fd(int x){
        return x==prt[x]?x:prt[x]=fd(prt[x]);
    }
    void lk(int x,int y){
        if(rnk[x=fd(x)]>rnk[y=fd(y)])
            prt[y]=x;
        else if(rnk[x]<rnk[y])
            prt[x]=y;
        else
            prt[x]=y,++rnk[y];
    }
    int lca(int x,int y){
        static int t;
        ++t;
        for(;;swap(x,y))
            if(x){
                x=top[fd(x)];
                if(vis[x]==t)
                    return x;
                vis[x]=t;
                if(res[x])
                    x=nxt[res[x]];
                else
                    x=0;
            }
    }
```

```
37      void uni(int x,int p){
38          for(;fd(x)!=fd(p);){
39              int y=res[x],z=nxt[y];
40              if(fd(z)!=fd(p))
41                  nxt[z]=y;
42              if(mrk[y]==2)
43                  mrk[y]=1,qu.push(y);
44              if(mrk[z]==2)
45                  mrk[z]=1,qu.push(z);
46              int t=top[fd(z)];
47              lk(x,y);
48              lk(y,z);
49              top[fd(z)]=t;
50              x=z;
51          }
52      }
53      void aug(int s){
54          for(int i=1;i<=n;++i)
55              nxt[i]=0,mrk[i]=0,top[i]=i,prt[i]=i,rnk[i]=0;
56          mrk[s]=1;
57          qu=queue<int>();
58          for(qu.push(s);!qu.empty();){
59              int x=qu.front();
60              qu.pop();
61              for(int i=0;i<to[x].size();++i){
62                  int y=to[x][i];
63                  if(res[x]==y||fd(x)==fd(y)||mrk[y]==2)
64                      continue;
65                  if(mrk[y]==1){
66                      int z=lca(x,y);
67                      if(fd(x)!=fd(z))
68                          nxt[x]=y;
69                      if(fd(y)!=fd(z))
70                          nxt[y]=x;
71                      uni(x,z);
72                      uni(y,z);
73                  }else if(!res[y]){
74                      for(nxt[y]=x;y;){
75                          int z=nxt[y],mz=res[z];
76                          res[z]=y;
77                          res[y]=z;
```

```
78                        y=mz;
79                    }
80                    return;
81                }else{
82                    nxt[y]=x;
83                    mrk[res[y]]=1;
84                    qu.push(res[y]);
85                    mrk[y]=2;
86                }
87            }
88        }
89    }
90    void add(int x,int y){
91        to[x].push_back(y);
92        to[y].push_back(x);
93    }
94    int run(){
95        for(int i=1;i<=n;++i)
96            if(!res[i])
97                for(int j=0;j<to[i].size();++j)
98                    if(!res[to[i][j]]){
99                        res[to[i][j]]=i;
100                        res[i]=to[i][j];
101                        break;
102                    }
103        for(int i=1;i<=n;++i)
104            if(!res[i])
105                aug(i);
106        int r=0;
107        for(int i=1;i<=n;++i)
108            if(res[i])
109                ++r;
110        return r/2;
111    }
112 };
```

3.7   Minimum Cost Maximum Flow

Minimum Cost Maximum Flow.hpp (2278 bytes, 82 lines)

```cpp
#include<bits/stdc++.h>
using namespace std;
template<class F=int,class C=int>struct MinimumCostMaximumFlow{
    struct edge{
        edge(int _v,F _c,C _w):
            v(_v),c(_c),w(_w){
        }
        int v;
        F c;
        C w;
    };
    MinimumCostMaximumFlow(int _n,int _src,int _snk,F _all):
        n(_n),src(_src−1),snk(_snk−1),bg(_n),vis(n),dis(n),all(_all),flow
    (0),cost(0){}
    void add(int u,int v,F c,C w){
        bg[u−1].push_back(eg.size());
        eg.push_back(edge(v−1,c,w));
        bg[v−1].push_back(eg.size());
        eg.push_back(edge(u−1,0,−w));
    }
    int spfa(){
        vector<int>in(n,0);
        queue<int>qu;
        fill(vis.begin(),vis.end(),0);
        dis[src]=0;
        vis[src]=in[src]=1;
        qu.push(src);
        while(!qu.empty()){
            int u=qu.front();
            qu.pop();
            in[u]=0;
            for(int i=0;i<bg[u].size();++i){
                edge&e=eg[bg[u][i]];
                if(e.c!=0&&(!vis[e.v]||dis[u]+e.w<dis[e.v])){
                    dis[e.v]=dis[u]+e.w;
                    vis[e.v]=1;
                    if(!in[e.v]){
                        in[e.v]=1;
                        qu.push(e.v);
                    }
```

```
40                      }
41                  }
42              }
43              return vis[snk]&&dis[snk]<0;
44          }
45          F dfs(int u,F f){
46              if(u==snk)
47                  return f;
48              F g=f;
49              vis[u]=1;
50              for(int i=0;i<bg[u].size();++i){
51                  edge&e=eg[bg[u][i]],&ev=eg[bg[u][i]^1];
52                  if(e.c!=0&&dis[e.v]==dis[u]+e.w&&!vis[e.v]){
53                      F t=dfs(e.v,min(g,e.c));
54                      g-=t;
55                      e.c-=t;
56                      ev.c+=t;
57                      cost+=t*e.w;
58                      if(g==0)
59                          return f;
60                  }
61              }
62              return f-g;
63          }
64          pair<F,C>run(){
65              while(all!=0&&spfa()){
66                  F t;
67                  do{
68                      fill(vis.begin(),vis.end(),0);
69                      flow+=(t=dfs(src,all));
70                      all-=t;
71                  }while(t!=0);
72              }
73              return make_pair(flow,cost);
74          }
75          int n,src,snk;
76          vector<vector<int> >bg;
77          vector<edge>eg;
78          vector<int>vis;
79          vector<C>dis;
80          F all,flow;
```

```
81       C cost;
82   };
```

3.8   Minimum Spanning Arborescence

Minimum Spanning Arborescence.hpp (1933 bytes, 64 lines)

```
1   #include<bits/stdc++.h>
2   using namespace std;
3   template<class T>struct MinimumSpanningArborescence{
4       struct eg{
5           int u,v;
6           T w;
7       };
8       int n,rt;
9       vector<eg>egs;
10      vector<int>vi,in,id;
11      vector<T>inw;
12      MinimumSpanningArborescence(int _n,int _rt):
13          n(_n),rt(_rt),vi(n+1),in(n+1),inw(n+1),id(n+1){
14      }
15      void add(int u,int v,T w){
16          eg e;
17          e.u=u;
18          e.v=v;
19          e.w=w;
20          egs.push_back(e);
21      }
22      T run(){
23          int nv=0;
24          for(T r=0;;n=nv,nv=0,rt=id[rt]){
25              for(int i=1;i<=n;++i)
26                  in[i]=-1;
27              for(int i=0;i<egs.size();++i)
28                  if(egs[i].u!=egs[i].v&&(in[egs[i].v]==-1||egs[i].w<inw[egs[
    i].v]))
29                      in[egs[i].v]=egs[i].u,inw[egs[i].v]=egs[i].w;
30              for(int i=1;i<=n;++i)
31                  if(i!=rt&&in[i]==-1)
```

```
32                      return numeric_limits<T>::max();
33              for(int i=1;i<=n;++i){
34                  if(i!=rt)
35                      r+=inw[i];
36                  id[i]=−1,vi[i]=0;
37              }
38              for(int i=1;i<=n;++i)
39                  if(i!=rt&&!vi[i]){
40                      int u=i;
41                      do{
42                          vi[u]=i;
43                          u=in[u];
44                      }while(!vi[u]&&u!=rt);
45                      if(u!=rt&&vi[u]==i){
46                          int v=u;
47                          ++nv;
48                          do{
49                              id[v]=nv;
50                              v=in[v];
51                          }while(v!=u);
52                      }
53                  }
54              if(nv==0)
55                  return r;
56              for(int i=1;i<=n;++i)
57                  if(id[i]==−1)
58                      id[i]=++nv;
59              for(int i=0;i<egs.size();++i)
60                  egs[i].w−=inw[egs[i].v],egs[i].u=id[egs[i].u],
61                  egs[i].v=id[egs[i].v];
62          }
63      }
64 };
```

3.9   Minimum Spanning Tree

Minimum Spanning Tree.hpp (1049 bytes, 44 lines)

```
1 #include<bits/stdc++.h>
```

```cpp
using namespace std;
template<class T,class C=less<T> >struct MinimumSpanningTree{
    struct edge{
        T w;
        int u,v;
        int operator<(const edge&b)const{
            return C()(w,b.w);
        }
    };
    int n;
    vector<edge>egs;
    vector<int>pr;
    MinimumSpanningTree(int _n):
        n(_n),pr(n+1){
    }
    void add(int u,int v,T w){
        edge e;
        e.u=u;
        e.v=v;
        e.w=w;
        egs.push_back(e);
    }
    int fd(int x){
        return x==pr[x]?x:pr[x]=fd(pr[x]);
    }
    void lk(int x,int y){
        pr[fd(x)]=y;
    }
    pair<T,vector<edge> >run(){
        vector<edge>ret;
        T sum=0;
        sort(egs.begin(),egs.end());
        for(int i=1;i<=n;++i)
            pr[i]=i;
        for(int i=0;i<egs.size();++i){
            int u=egs[i].u,v=egs[i].v;
            T w=egs[i].w;
            if(fd(u)!=fd(v))
                lk(u,v),ret.push_back(egs[i]),sum+=w;
        }
        return make_pair(sum,ret);
```

```
43 |     }
44 | };
```

3.10   Shortest Path

Shortest Path.hpp (1279 bytes, 44 lines)

```
 1 | #include<bits/stdc++.h>
 2 | using namespace std;
 3 | template<class T>struct ShortestPath{
 4 |     int n,m;
 5 |     vector<vector<int> >to;
 6 |     vector<vector<T> >we;
 7 |     T inf;
 8 |     vector<pair<T,int> >sg;
 9 |     vector<T>di;
10 |     ShortestPath(int _n):
11 |         n(_n),m(1<<(int)ceil(log2(n)+1e−8)),to(n+1),we(n+1),inf(
        numeric_limits<T>::max()),sg(2*m,make_pair(inf,0)),di(n+1,inf){
12 |     }
13 |     void set(int u,T d){
14 |         di[u]=d;
15 |     }
16 |     void add(int u,int v,T w){
17 |         to[u].push_back(v);
18 |         we[u].push_back(w);
19 |     }
20 |     int upd(T&a,T b,T c){
21 |         if(b!=inf&&c!=inf&&b+c<a){
22 |             a=b+c;
23 |             return 1;
24 |         }
25 |         return 0;
26 |     }
27 |     void mod(int u,T d){
28 |         for(sg[u+m−1]=make_pair(d,u),u=(u+m−1)>>1;u;u>>=1)
29 |             sg[u]=min(sg[u<<1],sg[u<<1^1]);
30 |     }
31 |     vector<T>run(){
```

```
32          for(int i=1;i<=n;++i)
33              sg[i+m−1]=make_pair(di[i],i);
34          for(int i=m−1;i>=1;−−i)
35              sg[i]=min(sg[i<<1],sg[i<<1^1]);
36          for(int u=sg[1].second;sg[1].first!=inf?(mod(u,inf),1):0;u=sg[1].
    second)
37              for(int i=0;i<to[u].size();++i){
38                  int v=to[u][i];
39                  T w=we[u][i];
40                  if(upd(di[v],di[u],w))
41                      mod(v,di[v]);}
42          return di;
43      }
44  };
```

3.11   Steiner Tree

Steiner Tree.hpp (1745 bytes, 56 lines)

```
1   #include<bits/stdc++.h>
2   using namespace std;
3   template<class T>struct SteinerTree{
4       int n,k,z;
5       T inf=numeric_limits<T>::max();
6       vector<vector<T> >wei,dp;
7       vector<int>im;
8       SteinerTree(int _n):
9           n(_n),k(0),wei(n+1,vector<T>(n+1,inf)),im(n+1){
10      }
11      void set(int u){
12          if(!im[u])
13              im[z=u]=++k;
14      }
15      void add(int u,int v,T w){
16          wei[u][v]=wei[v][u]=min(w,wei[u][v]);
17      }
18      int upd(T&a,T b,T c){
19          if(b!=inf&&c!=inf&&b+c<a){
20              a=b+c;
```

```
21                return 1;
22            }
23            return 0;
24        }
25        int ins(int s,int u){
26            return im[u]&&((s>>im[u]-1)&1);
27        }
28        T run(){
29            for(int l=1;l<=n;++l)
30                for(int i=1;i<=n;++i)
31                    for(int j=1;j<=n;++j)
32                        upd(wei[i][j],wei[i][l],wei[l][j]);
33            dp=vector<vector<T> >(1<<k-1,vector<T>(n+1,inf));
34            fill(begin(dp[0]),end(dp[0]),0);
35            for(int s=1;s<(1<<k-1);++s){
36                queue<int>qu;
37                vector<int>in(n+1);
38                for(int u=1;u<=n;++u){
39                    if(ins(s,u))
40                        continue;
41                    qu.push((u));
42                    in[u]=1;
43                    for(int t=(s-1)&s;t;t=(t-1)&s)
44                        upd(dp[s][u],dp[t][u],dp[s^t][u]);
45                    for(int v=1;v<=n;++v)
46                        if(ins(s,v))
47                            upd(dp[s][u],dp[s^(1<<im[v]-1)][v],wei[u][v]);
48                }
49                for(int u;qu.empty()?0:(u=qu.front(),qu.pop(),in[u]=0,1);)
50                    for(int v=1;v<=n;++v)
51                        if(!ins(s,v)&&upd(dp[s][v],dp[s][u],wei[u][v])&&!in[v])
52                            in[v]=1,qu.push(v);
53            }
54            return k?dp[(1<<k-1)-1][z]:0;
55        }
56  };
```

3.12   Virtual Tree

Virtual Tree.hpp (2375 bytes, 77 lines)

```cpp
#include<bits/stdc++.h>
using namespace std;
struct VirtualTree{
    int n,r,l;
    vector<vector<int> >to,vto,up;
    vector<int>lst,dp,dfn,edf,imp;
    VirtualTree(int _n,int _r):
        n(_n),r(_r),l(ceil(log2(n)+1e-8)),to(n+1),vto(n+1),up(n+1,vector<
    int>(l+1)),dp(n+1),dfn(n+1),edf(n+1),imp(n+1){
    }
    void add(int u,int v){
        to[u].push_back(v);
        to[v].push_back(u);
    }
    void vadd(int u,int v){
        vto[u].push_back(v);
    }
    int lca(int u,int v){
        if(dp[u]<dp[v])
            swap(u,v);
        for(int i=0;i<=l;++i)
            if(((dp[u]-dp[v])>>i)&1)
                u=up[u][i];
        if(u==v)
            return u;
        for(int i=l;i>=0;--i)
            if(up[u][i]!=up[v][i])
                u=up[u][i],v=up[v][i];
        return up[u][0];
    }
    void dfs(int u){
        dfn[u]=++dfn[0];
        for(int i=1;i<=l;++i)
            up[u][i]=up[up[u][i-1]][i-1];
        for(int i=0;i<to[u].size();++i){
            int v=to[u][i];
            if(v!=up[u][0])
```

```
37 |                     up[v][0]=u,dp[v]=dp[u]+1,dfs(v);
38 |             }
39 |             edf[u]=dfn[0];
40 |         }
41 |     void build(){
42 |         dfs(r);
43 |     }
44 |     void run(int*a,int m){
45 |         for(int i=0;i<lst.size();++i)
46 |             imp[lst[i]]=0,vto[lst[i]].clear();
47 |         vector<pair<int,int> >b(m+1);
48 |         for(int i=1;i<=m;++i)
49 |             imp[a[i]]=1,b[i]=make_pair(dfn[a[i]],a[i]);
50 |         sort(b.begin()+1,b.end());
51 |         vector<int>st(1,r);
52 |         lst=st;
53 |         for(int i=1;i<=m;++i){
54 |             int u=b[i].second,v=st.back();
55 |             if(u==r)
56 |                 continue;
57 |             if(dfn[u]<=edf[v])
58 |                 st.push_back(u);
59 |             else{
60 |                 int w=lca(u,v);
61 |                 while(st.size()>=2&&dp[st[st.size()-2]]>=dp[w]){
62 |                     vadd(st[st.size()-2],*st.rbegin());
63 |                     lst.push_back(*st.rbegin()),st.pop_back();
64 |                 }
65 |                 if(st.size()>=2&&w!=st[st.size()-1]){
66 |                     vadd(w,*st.rbegin()),lst.push_back(*st.rbegin());
67 |                     st.pop_back(),st.push_back(w);
68 |                 }
69 |                 st.push_back(u);
70 |             }
71 |         }
72 |         while(st.size()>=2){
73 |             vadd(st[st.size()-2],*st.rbegin());
74 |             lst.push_back(*st.rbegin()),st.pop_back();
75 |         }
76 |     }
77 | };
```

CHAPTER 4

Number Theory

4.1   Discrete Logarithm

Discrete Logarithm.hpp (1819 bytes, 74 lines)

```cpp
#include<bits/stdc++.h>
using namespace std;
namespace DiscreteLogarithm{
    typedef long long T;
    int ti[1<<16],va[1<<16],mp[1<<16],nx[1<<16],hd[1<<16],tm,nw;
    void ins(int x,int v){
        int y=x&65535;
        if(ti[y]!=tm)
            ti[y]=tm,hd[y]=0;
        for(int i=hd[y];i;i=nx[i])
            if(va[i]==x){
                mp[i]=v;
                return;
            }
        va[++nw]=x;
        mp[nw]=v;
        nx[nw]=hd[y];
        hd[y]=nw;
    }
    int get(int x){
        int y=x&65535;
        if(ti[y]!=tm)
            ti[y]=tm,hd[y]=0;
        for(int i=hd[y];i;i=nx[i])
            if(va[i]==x){
                return mp[i];
            }
        return −1;
    }
    T pow(T a,T b,T c){
        T r=1;
        for(;b;b&1?r=r*a%c:0,b>>=1,a=a*a%c);
        return r;
    }
    T gcd(T a,T b){
        return b?gcd(b,a%b):a;
    }
```

```
38    void exg(T a,T b,T&x,T&y){
39        if(!b)
40            x=1,y=0;
41        else
42            exg(b,a%b,y,x),y−=a/b*x;
43    }
44    T inv(T a,T b){
45        T x,y;
46        exg(a,b,x,y);
47        return x+b;
48    }
49    T bgs(T a,T b,T c){
50        ++tm;
51        nw=0;
52        T m=sqrt(c);
53        for(T i=m−1,u=pow(a,i,c),v=inv(a,c);i>=0;−−i,u=u*v%c)
54            ins(u,i);
55        for(T i=0,u=1,v=inv(pow(a,m,c),c);i*m<=c;++i,u=u*v%c){
56            T t=u*b%c,j;
57            if((j=get(t))!=−1)
58                return i*m+j;
59        }
60        return −1;
61    }
62    T run(T a,T b,T c){
63        T u=1,t=0;
64        a=(a%c+c)%c;
65        b=(b%c+c)%c;
66        for(int i=0;i<32;++i)
67            if(pow(a,i,c)==b)
68                return i;
69        for(T d;(d=gcd(a,c))!=1;++t,u=a/d*u%c,b/=d,c/=d)
70            if(b%d)
71                return −1;
72        return (u=bgs(a,b*inv(u,c)%c,c))<0?−1:u+t;
73    }
74 }
```

4.2   Integer Factorization (Pollard's Rho Algorithm)

Integer Factorization (Pollard's Rho Algorithm).hpp (2848 bytes, 93 lines)

```cpp
#include<bits/stdc++.h>
using namespace std;
namespace IntegerFactorization{
    template<class T>T mul(T x,T y,T z){
        if(typeid(T)==typeid(int))
            return (long long)x*y%z;
        else if(typeid(T)==typeid(long long))
            return (x*y-(T)(((long double)x*y+0.5)/z)*z+z)%z;
        else
            return x*y%z;
    }
    template<class T>T pow(T a,T b,T c){
        T r=1;
        for(;b;b&1?r=mul(r,a,c):0,b>>=1,a=mul(a,a,c));
        return r;
    }
    template<class T>int chk(T a,int c=10){
        if(a==2)
            return 1;
        if(a%2==0||a<2)
            return 0;
        static int pi[]={2,7,61},pl
    []={2,325,9375,28178,450775,9780504,1795265022};
        if(typeid(T)==typeid(int))
            c=3;
        else if(typeid(T)==typeid(long long))
            c=7;
        T u=a-1,t=0,p=1;
        for(;u%2==0;u/=2,++t);
        for(int i=0;i<c;++i){
            if(typeid(T)==typeid(int))
                p=pi[i]%a;
            else if(typeid(T)==typeid(long long))
                p=pl[i]%a;
            else
                p=(p*29+7)%a;
            if(!p||p==1||p==a-1)
```

```
37                  continue;
38              T x=pow(p,u,a);
39              if(x==1)
40                  continue;
41              for(int j=0;x!=a-1&&j<t;++j){
42                  x=mul(x,x,a);
43                  if(x==1)
44                      return 0;
45              }
46              if(x==a-1)
47                  continue;
48              return 0;
49          }
50          return 1;
51      }
52      template<class T>T gcd(T a,T b){
53          if(a<0)
54              a=-a;
55          if(b<0)
56              b=-b;
57          return b?gcd(b,a%b):a;
58      }
59      template<class T>T rho(T a,T c){
60          T x=double(rand())/RAND_MAX*(a-1),y=x;
61          for(int i=1,k=2;;){
62              x=(mul(x,x,a)+c)%a;
63              T d=gcd(y-x,a);
64              if(d!=1&&d!=a)
65                  return d;
66              if(y==x)
67                  return a;
68              if(++i==k)
69                  y=x,k=2*k;
70          }
71      }
72      template<class T>vector<pair<T,int> >run(T a){
73          if(a==1)
74              return vector<pair<T,int> >();
75          if(chk(a))
76              return vector<pair<T,int> >(1,make_pair(a,1));
77          T b=a;
```

```
78          while((b=rho(b,T(double(rand())/RAND_MAX*(a-1))))==a);
79          vector<pair<T,int> >u=run(b),v=run(a/b),r;
80          for(int pu=0,pv=0;pu<u.size()||pv<v.size();){
81              if(pu==u.size())
82                  r.push_back(v[pv++]);
83              else if(pv==v.size())
84                  r.push_back(u[pu++]);
85              else if(u[pu].first==v[pv].first)
86                  r.push_back(make_pair(u[pu].first,(u[pu].second+v[pv].second
    ))),++pu,++pv;
87              else if(u[pu].first>v[pv].first)
88                  r.push_back(v[pv++]);
89              else
90                  r.push_back(u[pu++]);}
91          return r;
92      }
93  }
```

4.3   Integer Factorization (Shanks' Square Forms Factorization)

Integer Factorization (Shanks' Square Forms Factorization).hpp (4675 bytes, 147 lines)

```
1   #include<bits/stdc++.h>
2   using namespace std;
3   namespace IntegerFactorization{
4       typedef long long ll;
5       typedef unsigned long long ull;
6       ll lim=36893488146942583261l;
7       ull srt(const ull&a){
8           ull b=sqrt(a);
9           b-=b*b>a;
10          return b+=(b+1)*(b+1)<=a;
11      }
12      int sqr(const ull&a,ll&b){
13          b=srt(a);
14          return b*b==a;
15      }
16      ull gcd(const ull&a,const ull&b){
```

```
17          return b?gcd(b,a%b):a;
18      }
19      ll amb(ll a,const ll&B,const ll&dd,const ll&D){
20          for(ll q=(dd+B/2)/a,b=q*a*2−B,c=(D−b*b)/4/a,qc,qcb,a0=a,b0=a,b1=b,
        c0=c;;b1=b,c0=c){
21              if(c0>dd)
22                  qcb=c0−b,b=c0+qcb,c=a−qcb;
23              else{
24                  q=(dd+b/2)/c0;
25                  if(q==1)
26                      qcb=c0−b,b=c0+qcb,c=a−qcb;
27                  else
28                      qc=q*c0,qcb=qc−b,b=qc+qcb,c=a−q*qcb;
29              }
30              if(a=c0,b==b1)
31                  break;
32              if(b==b0&&a==a0)
33                  return 0;
34          }
35          return a&1?a:a>>1;
36      }
37      ull fac(const ull&n){
38          if(n&1^1)
39              return 2;
40          if(n%3==0)
41              return 3;
42          if(n%5==0)
43              return 5;
44          if(srt(n)*srt(n)==n)
45              return srt(n);
46          static ll d1,d2,a1,b1,c1,dd1,L1,a2,b2,c2,dd2,L2,a,q,c,qc,qcb,D1,D2,
        bl1[1<<19],bl2[1<<19];
47          int p1=0,p2=0,ac1=1,ac2=1,j,nm4=n&3;
48          if(nm4==1)
49              D1=n,D2=5*n,d2=srt(D2),dd2=d2/2+d2%2,b2=(d2−1)|1;
50          else
51              D1=3*n,D2=4*n,dd2=srt(D2),d2=dd2*2,b2=d2;
52          d1=srt(D1),b1=(d1−1)|1,c1=(D1−b1*b1)/4,c2=(D2−b2*b2)/4,L1=srt(d1),
        L2=srt(d2),dd1=d1/2+d1%2;
53          for(int i=a1=a2=1;ac1||ac2;++i){
54              #define m(t)\
```

```
55              if(ac##t){\
56                  c=c##t;\
57                  q=c>dd##t?1:(dd##t+b##t/2)/c;\
58                  if(q==1)\
59                      qcb=c−b##t,b##t=c+qcb,c##t=a##t−qcb;\
60                  else\
61                      qc=q*c,qcb=qc−b##t,b##t=qc+qcb,c##t=a##t−q*qcb;\
62                  if((a##t=c)<=L##t)\
63                      bl##t[p##t++]=a##t;\
64              }
65          m(1)m(2)
66          if(i&1)
67              continue;
68          #define m(t)\
69          if((ac##t=ac##t&a##t!=1)&&sqr(a##t,a)){\
70              if(a<=L##t)\
71                  for(j=0;j<p##t;j++)\
72                      if(a==bl##t[j]){\
73                          a=0;\
74                          break;\
75                      }\
76              if(a>0){\
77                  if((q=gcd(a,b##t))>1)\
78                      return q*q;\
79                  q=amb(a,b##t,dd##t,D##t);\
80                  if(nm4==5−2*t&&(q=amb(a,b##t,dd##t,D##t))%(2*t+1)==0)\
81                      q/=2*t+1;\
82                  if(q>1)\
83                      return q;\
84              }\
85          }
86          m(1)m(2)
87          #undef m
88      }
89      for(int i=3;;i+=2)
90          if(n%i==0)
91              return i;
92  }
93  ll mul(const ll&x,const ll&y,const ll&z){
94      return(x*y−(ll)(((long double)x*y+0.5)/z)*z+z)%z;
95  }
```

```
96      ll pow(ll a,ll b,const ll&c){
97          ll r=1;
98          for(;b;b&1?r=mul(r,a,c):0,b>>=1,a=mul(a,a,c));
99          return r;
100     }
101     int chk(const ll&a){
102         if(a==2)
103             return 1;
104         if(a%2==0||a<2)
105             return 0;
106         static int pf[]={2,325,9375,28178,450775,9780504,1795265022};
107         ll u=a−1,t=0,p;
108         for(;u%2==0;u/=2,++t);
109         for(int i=0;i<7;++i){
110             p=pf[i]%a;
111             if(!p||p==a−1)
112                 continue;
113             ll x=pow(p,u,a);
114             if(x==1)
115                 continue;
116             for(int j=0;x!=a−1&&j<t;++j){
117                 x=mul(x,x,a);
118                 if(x==1)
119                     return 0;
120             }
121             if(x==a−1)
122                 continue;
123             return 0;
124         }
125         return 1;
126     }
127     vector<pair<ll,int> >run(const ll&a){
128         if(a==1)
129             return vector<pair<ll,int> >();
130         if(chk(a))
131             return vector<pair<ll,int> >(1,make_pair(a,1));
132         ll b=fac(a);
133         vector<pair<ll,int> >u=run(b),v=run(a/b),r;
134         for(int pu=0,pv=0;pu<u.size()||pv<v.size();){
135             if(pu==u.size())
136                 r.push_back(v[pv++]);
```

```
137              else if(pv==v.size())
138                  r.push_back(u[pu++]);
139              else if(u[pu].first==v[pv].first)
140                  r.push_back(make_pair(u[pu].first,(u[pu].second+v[pv].second
     ))),++pu,++pv;
141              else if(u[pu].first>v[pv].first)
142                  r.push_back(v[pv++]);
143              else
144                  r.push_back(u[pu++]);}
145          return r;
146      }
147 }
```

4.4   Modular Integer

Modular Integer.hpp (2886 bytes, 98 lines)

```
 1  #include<bits/stdc++.h>
 2  using namespace std;
 3  template<class T>struct ModularInteger{
 4      ModularInteger(T t=0):
 5          v(t){
 6          if(v<0||v>=p)
 7              v=(v%p+p)%p;
 8      }
 9      ModularInteger<T>&operator=(T a){
10          v=a;
11          if(v<0||v>=p)
12              v%=p;
13          return*this;
14      }
15      ModularInteger<T>operator−(){
16          return v?p−v:0;
17      }
18      ModularInteger<T>&operator+=(ModularInteger<T>a){
19          return*this=*this+a;
20      }
21      ModularInteger<T>&operator−=(ModularInteger<T>a){
22          return*this=*this−a;
```

```
23          }
24          ModularInteger<T>&operator*=(ModularInteger<T>a){
25              return*this=*this*a;
26          }
27          ModularInteger<T>&operator/=(ModularInteger<T>a){
28              return*this=*this/a;
29          }
30          T v;
31          static T p;
32   };
33   template<class T>ModularInteger<T>pow(ModularInteger<T>a,long long b){
34          ModularInteger<T>r(1);
35          for(;b;b>>=1,a=a*a)
36              if(b&1)
37                  r=r*a;
38          return r;
39   }
40   template<class T>ModularInteger<T>inv(ModularInteger<T>a){
41          return pow(a,a.p−2);
42   }
43   template<class T>vector<ModularInteger<T> >sqrt(ModularInteger<T>a){
44          vector<ModularInteger<T> >r;
45          if(!a.v)
46              r.push_back(ModularInteger<T>(0));
47          else if(pow(a,a.p−1>>1).v==1){
48              int s=a.p−1,t=0;
49              ModularInteger<T>b=1;
50              for(;pow(b,a.p−1>>1).v!=a.p−1;b=rand()*1.0/RAND_MAX*(a.p−1));
51              for(;s%2==0;++t,s/=2);
52              ModularInteger<T>x=pow(a,(s+1)/2),e=pow(a,s);
53              for(int i=1;i<t;++i,e=x*x/a)
54                  if(pow(e,1<<t−i−1).v!=1)
55                      x=x*pow(b,(1<<i−1)*s);
56              r.push_back(x);
57              r.push_back(−x);
58          }
59          return r;
60   }
61   template<class T>ModularInteger<T>operator+(ModularInteger<T>a,
        ModularInteger<T>b){
62          ModularInteger<T>c(a.v+b.v);
```

```
63        if(c.v>=a.p)
64            c.v-=a.p;
65        return c;
66  }
67  template<class T>ModularInteger<T>operator-(ModularInteger<T>a,
        ModularInteger<T>b){
68        ModularInteger<T>c(a.v-b.v);
69        if(c.v<0)
70            c.v+=a.p;
71        return c;
72  }
73  template<class T>ModularInteger<T>operator*(ModularInteger<T>a,
        ModularInteger<T>b){
74        if(typeid(T)!=typeid(int))
75            return ModularInteger<T>((a.v*b.v-(long long)(((long double)a.v*b.v
        +0.5)/a.p)*a.p+a.p)%a.p);
76        else
77            return ModularInteger<T>((long long)a.v*b.v%a.p);
78  }
79  template<class T>ModularInteger<T>operator/(ModularInteger<T>a,
        ModularInteger<T>b){
80        return a*inv(b);
81  }
82  template<class T>bool operator==(ModularInteger<T>a,ModularInteger<T>b){
83        return a.v==b.v;
84  }
85  template<class T>bool operator!=(ModularInteger<T>a,ModularInteger<T>b){
86        return a.v!=b.v;
87  }
88  template<class T>istream&operator>>(istream&s,ModularInteger<T>&a){
89        s>>a.v;
90        return s;
91  }
92  template<class T>ostream&operator<<(ostream&s,ModularInteger<T>a){
93        s<<a.v;
94        if(a.v<0||a.v>=a.p)
95            a.v%=a.p;
96        return s;
97  }
98  template<class T>T ModularInteger<T>::p=1e9+7;
```

4.5   Möbius Function

Möbius Function.hpp (534 bytes, 21 lines)

```cpp
#include<bits/stdc++.h>
using namespace std;
namespace MobiusFunction{
    vector<int>run(int n){
        vector<int>p,ntp(n+1),u(n+1);
        ntp[1]=1;
        u[1]=1;
        for(int i=2;i<=n;++i){
            if(!ntp[i])
                p.push_back(i),u[i]=-1;
            for(int j=0;j<p.size()&&p[j]*i<=n;++j){
                ntp[p[j]*i]=1;
                if(i%p[j]==0)
                    break;
                else
                    u[p[j]*i]=-u[i];
            }
        }
        return u;
    }
}
```

4.6   Number

Number.hpp (7837 bytes, 345 lines)

```cpp
#include<bits/stdc++.h>
using namespace std;
template<class T>T add(T a,T b,T p){
    return a+b<p?a+b:a+b-p;
}
template<class T>T sub(T a,T b,T p){
    return a-b<0?a-b+p:a-b;
}
template<class T>T mul(T a,T b,T p){
```

```
10        if(typeid(T)==typeid(int))
11            return (long long)a*b%p;
12        else
13            return (a*b-(T)(((long double)a*b+0.5)/p)*p+p)%p;
14  }
15  template<class T>T pow(T a,T b,T p){
16      T r=1;
17      for(;b;b&1?r=mul(r,a,p):0,b>>=1,a=mul(a,a,p));
18      return r;
19  }
20  template<class T>T inv(T a,T p){
21      return pow(a,p-2,p);
22  }
23  template<class T>T div(T a,T b,T p){
24      return mul(a,inv(b,p),p);
25  }
26  template<class T>T gcd(T a,T b){
27      if(a<0)
28          return gcd(-a,b);
29      if(b<0)
30          return gcd(a,-b);
31      return b?gcd(b,a%b):a;
32  }
33  template<class T>pair<T,T>exgcd(T a,T b){
34      if(!b)
35          return make_pair(T(1),T(0));
36      pair<T,T>t=exgcd(b,a%b);
37      swap(t.first,t.second);
38      t.second-=a/b*t.first;
39      return t;
40  }
41  template<class T>vector<T>divisor(T a){
42      vector<T>t;
43      for(T d=1;d*d<=a;++d)
44          if(a%d==0){
45              t.push_back(d);
46              if(d*d!=a)
47                  t.push_back(a/d);
48          }
49      return t;
50  }
```

```
51 | template<class T>vector<pair<T,T> >factor(T a){
52 |     vector<pair<T,T> >t;
53 |     T b=a;
54 |     for(T d=2;d*d<=a;++d)
55 |         if(b%d==0)
56 |             for(t.push_back(make_pair(d,T(0)));b%d==0;b/=d,++t.back().second
   | );
57 |     if(b!=1)
58 |         t.push_back(make_pair(b,T(1)));
59 |     return t;
60 | }
61 | template<class T>bool is_prime_number(T a){
62 |     if(a<2)
63 |         return false;
64 |     for(T d=2;d*d<=a;++d)
65 |         if(a%d==0)
66 |             return false;
67 |     return true;
68 | }
69 | template<class T>bool is_palindromic_number(T a){
70 |     string b;
71 |     stringstream s;
72 |     s<<a;
73 |     s>>b;
74 |     for(int i=0;i<b.size();++i)
75 |         if(b[i]!=b[b.size()-i-1])
76 |             return false;
77 |     return true;
78 | }
79 | template<class T>T pow(T a,T b){
80 |     T s=1;
81 |     for(;b;b/=2,a*=a)
82 |         if(b%2)
83 |             s*=a;
84 |     return s;
85 | }
86 | template<class T>T power_sum(T n,T k){
87 |     T r=0;
88 |     for(T i=1;i<=n;++i)
89 |         r+=pow(i,k);
90 |     return r;
```

```cpp
 91 }
 92 template<class T>T sqr(T a){
 93     return a*a;
 94 }
 95 int nth_prime(int n){
 96     int r=1;
 97     for(int i=1;i<=n;++i)
 98         for(++r;!is_prime_number(r);++r);
 99     return r;
100 }
101 vector<vector<int> >pythagorean_triple(int n){
102     vector<vector<int> >t;
103     for(int i=1;i*i<=n;++i)
104         for(int j=1;j<i&&i*i+j*j<=n;++j){
105             vector<int>u;
106             u.push_back(i*i−j*j);
107             u.push_back(2*i*j);
108             u.push_back(i*i+j*j);
109             t.push_back(u);
110         }
111     return t;
112 }
113 vector<vector<int> >primitive_pythagorean_triple(int n){
114     vector<vector<int> >t;
115     for(int i=1;i*i<=n;++i)
116         for(int j=1;j<i&&i*i+j*j<=n;++j)
117             if((i−j)%2&&gcd(i,j)==1){
118                 vector<int>u;
119                 u.push_back(i*i−j*j);
120                 u.push_back(2*i*j);
121                 u.push_back(i*i+j*j);
122                 t.push_back(u);
123             }
124     return t;
125 }
126 template<class T>nth_triangular_number(T n){
127     if(n%2==0)
128         return n/2*(n+1);
129     else
130         return (n+1)/2*n;
131 }
```

```cpp
template<class T>nth_pentagonal_number(T n){
    if(n%2==0)
        return n/2*(3*n-1);
    else
        return (3*n-1)/2*n;
}
template<class T>nth_hexagonal_number(T n){
    return n*(2*n-1);
}
template<class T>vector<T>collatz_sequence(T a){
    vector<T>t;
    do{
        t.push_back(a);
        if(a==1)
            return t;
        if(a%2==0)
            a/=2;
        else
            a=3*a+1;
    }while(1);
}
template<class T>T factorial(T n){
    T r=1;
    for(T i=1;i<=n;++i)
        r*=i;
    return r;
}
template<class T>T product(T a,T b){
    T r=1;
    for(T i=a;i<=b;++i)
        r*=i;
    return r;
}
template<class T>T C(T n,T k){
    return factorial(n)/factorial(k)/factorial(n-k);
}
template<class T>T P(T n,T k){
    return factorial(n)/factorial(n-k);
}
vector<int>prime(int n){
    vector<int>p,ntp(n+1);
```

```
173        ntp[1]=1;
174        for(int i=2;i<=n;++i){
175            if(!ntp[i])
176                p.push_back(i);
177            for(int j=0;j<p.size()&&p[j]*i<=n;++j){
178                ntp[p[j]*i]=1;
179                if(i%p[j]==0)
180                    break;
181            }
182        }
183        return p;
184  }
185  template<class T>T digit_sum(T a){
186        T r=0;
187        for(;a;r+=a%10,a/=10);
188        return r;
189  }
190  template<class T>T digit_power_sum(T a,T b){
191        T r=0;
192        for(;a;r+=pow(a%10,b),a/=10);
193        return r;
194  }
195  template<class T>T divisor_sum(T a){
196        vector<T>d=divisor(a);
197        T s=0;
198        for(int i=0;i<d.size();++i)
199            s+=d[i];
200        return s;
201  }
202  template<class T>bool is_perfect(T a){
203        return a*2==divisor_sum(a);
204  }
205  template<class T>bool is_deficient(T a){
206        return a*2>divisor_sum(a);
207  }
208  template<class T>bool is_abundant(T a){
209        return a*2<divisor_sum(a);
210  }
211  template<class T>set<int>digit_set(T a){
212        set<int>r;
213        for(;a;r.insert(a%10),a/=10);
```

```
214        return r;
215    }
216
217    template<class T>multiset<int>digit_multiset(T a){
218        multiset<int>r;
219        for(;a;r.insert(a%10),a/=10);
220        return r;
221    }
222    template<class T>int digit_count(T a){
223        int r=0;
224        if(!a)
225            ++r;
226        for(;a;++r,a/=10);
227        return r;
228    }
229    template<class T>T digit_factorial_sum(T a){
230        T r=0;
231        for(;a;r+=factorial(a%10),a/=10);
232        return r;
233    }
234    template<class T>bool has_distinct_digit(T a){
235        return digit_count(a)==digit_set(a).size();
236    }
237    template<class T>bool has_zero(T a){
238        if(!a)
239            return true;
240        while(a){
241            if(a%10==0)
242                return true;
243            a/=10;
244        }
245        return false;
246    }
247    template<class T>T right_circular_shift(T a){
248        stringstream ss;
249        ss<<a;
250        string t;
251        ss>>t;
252        t=t.substr(t.size()-1,1)+t.substr(0,t.size()-1);
253        ss<<t;
254        ss>>a;
```

```
255        return a;
256    }
257    template<class T>bool is_circular_prime(T a){
258        for(int i=digit_count(a);i;--i,a=right_circular_shift(a))
259            if(!is_prime_number(a))
260                return false;
261        return true;
262    }
263    template<class T>string to_binary(T a){
264        string r;
265        while(a){
266            r.push_back(a%2+'0');
267            a/=2;
268        }
269        reverse(r.begin(),r.end());
270        return r;
271    }
272    template<class T>T digit_reverse(T a){
273        stringstream ss;
274        ss<<a;
275        string t;
276        ss>>t;
277        reverse(t.begin(),t.end());
278        stringstream ss2;
279        ss2<<t;
280        ss2>>a;
281        return a;
282    }
283    template<class T>bool is_truncatable_prime(T a){
284        T b=digit_reverse(a);
285        while(a){
286            if(!is_prime_number(a))
287                return false;
288            a/=10;
289        }
290        a=b;
291        while(a){
292            if(!is_prime_number(digit_reverse(a)))
293                return false;
294            a/=10;
295        }
```

```
296        return true;
297    }
298    template<class T>bool is_triangle_number(T a){
299        if(a<1)
300            return false;
301        T l=1,r=1;
302        while(nth_triangular_number(r)<=a)
303            r*=2;
304        while(l+1<r){
305            T m=l+(r−l)/2;
306            if(nth_triangular_number(m)<=a)
307                l=m;
308            else
309                r=m;
310        }
311        return a==nth_triangular_number(l);
312    }
313    template<class T>bool is_pentagonal_number(T a){
314        if(a<1)
315            return false;
316        T l=1,r=1;
317        while(nth_pentagonal_number(r)<=a)
318            r*=2;
319        while(l+1<r){
320            T m=l+(r−l)/2;
321            if(nth_pentagonal_number(m)<=a)
322                l=m;
323            else
324                r=m;
325        }
326        return a==nth_pentagonal_number(l);
327    }
328    template<class T>bool is_hexagonal_number(T a){
329        if(a<1)
330            return false;
331        T l=1,r=1;
332        while(nth_hexagonal_number(r)<=a)
333            r*=2;
334        while(l+1<r){
335            T m=l+(r−l)/2;
336            if(nth_hexagonal_number(m)<=a)
```

```
337              l=m;
338          else
339              r=m;
340      }
341      return a==nth_hexagonal_number(l);
342 }
343 template<class T>bool is_square_number(T a){
344      return sqr(T(round(sqrt(a))))==a;
345 }
```

4.7   Primality Test

Primality Test.hpp (1509 bytes, 52 lines)

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  namespace PrimalityTest{
4      template<class T>T mul(T x,T y,T z){
5          if(typeid(T)==typeid(int))
6              return (long long)x*y%z;
7          else if(typeid(T)==typeid(long long))
8              return (x*y-(T)(((long double)x*y+0.5)/z)*z+z)%z;
9          else
10             return x*y%z;
11     }
12     template<class T>T pow(T a,T b,T c){
13         T r=1;
14         for(;b;b&1?r=mul(r,a,c):0,b>>=1,a=mul(a,a,c));
15         return r;
16     }
17     template<class T>int run(T a,int c=10){
18         if(a==2)
19             return 1;
20         if(a%2==0||a<2)
21             return 0;
22         static int pi[]={2,7,61},pl
    []={2,325,9375,28178,450775,9780504,1795265022};
23         if(typeid(T)==typeid(int))
24             c=3;
```

```
25          else if(typeid(T)==typeid(long long))
26              c=7;
27          T u=a−1,t=0,p=1;
28          for(;u%2==0;u/=2,++t);
29          for(int i=0;i<c;++i){
30              if(typeid(T)==typeid(int))
31                  p=pi[i]%a;
32              else if(typeid(T)==typeid(long long))
33                  p=pl[i]%a;
34              else
35                  p=(p*29+7)%a;
36              if(!p||p==1||p==a−1)
37                  continue;
38              T x=pow(p,u,a);
39              if(x==1)
40                  continue;
41              for(int j=0;x!=a−1&&j<t;++j){
42                  x=mul(x,x,a);
43                  if(x==1)
44                      return 0;
45              }
46              if(x==a−1)
47                  continue;
48              return 0;
49          }
50          return 1;
51      }
52 }
```

4.8   Prime Number

Prime Number.hpp (473 bytes, 18 lines)

```
1 #include<bits/stdc++.h>
2 using namespace std;
3 namespace PrimeNumber{
4     pair<vector<int>,vector<int> >run(int n){
5         vector<int>p,ntp(n+1);
6         ntp[1]=1;
```

```
7          for(int i=2;i<=n;++i){
8              if(!ntp[i])
9                  p.push_back(i);
10             for(int j=0;j<p.size()&&p[j]*i<=n;++j){
11                 ntp[p[j]*i]=1;
12                 if(i%p[j]==0)
13                     break;
14             }
15         }
16         return make_pair(p,ntp);
17     }
18 }
```

4.9   Primitive Root

Primitive Root.hpp (3256 bytes, 106 lines)

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  namespace PrimitiveRoot{
4      template<class T>T mul(T x,T y,T z){
5          if(typeid(T)==typeid(int))
6              return (long long)x*y%z;
7          else
8              return (x*y-(T)(((long double)x*y+0.5)/z)*z+z)%z;
9      }
10     template<class T>T pow(T a,T b,T c){
11         T r=1;
12         for(;b;b&1?r=mul(r,a,c):0,b>>=1,a=mul(a,a,c));
13         return r;
14     }
15     template<class T>bool chk(T a,int c=10){
16         if(a==1)
17             return false;
18         T u=a-1,t=0;
19         for(;u%2==0;u/=2,++t);
20         for(int i=0;i<c;++i){
21             T x=pow(T(rand()*1.0/RAND_MAX*(a-2)+1),u,a),y;
22             for(int j=0;j<t;++j){
```

```
23              y=x;
24              x=mul(x,x,a);
25              if(x==1&&y!=1&&y!=a−1)
26                  return false;
27          }
28          if(x!=1)
29              return false;
30      }
31      return true;
32  }
33  template<class T>T gcd(T a,T b){
34      if(a<0)
35          a=−a;
36      if(b<0)
37          b=−b;
38      return b?gcd(b,a%b):a;
39  }
40  template<class T>T rho(T a,T c){
41      T x=double(rand())/RAND_MAX*(a−1),y=x;
42      for(int i=1,k=2;;){
43          x=(mul(x,x,a)+c)%a;
44          T d=gcd(y−x,a);
45          if(d!=1&&d!=a)
46              return d;
47          if(y==x)
48              return a;
49          if(++i==k)
50              y=x,k=2*k;
51      }
52  }
53  template<class T>vector<pair<T,int> >fac(T a){
54      if(a==1)
55          return vector<pair<T,int> >();
56      if(chk(a))
57          return vector<pair<T,int> >(1,make_pair(a,1));
58      T b=a;
59      while((b=rho(b,T(double(rand())/RAND_MAX*(a−1))))==a);
60      vector<pair<T,int> >u=fac(b),v=fac(a/b),r;
61      for(int pu=0,pv=0;pu<u.size()||pv<v.size();){
62          if(pu==u.size())
63              r.push_back(v[pv++]);
```

```
64              else if(pv==v.size())
65                  r.push_back(u[pu++]);
66              else if(u[pu].first==v[pv].first)
67                  r.push_back(make_pair(u[pu].first,(u[pu].second+v[pv].second
    ))),++pu,++pv;
68              else if(u[pu].first>v[pv].first)
69                  r.push_back(v[pv++]);
70              else
71                  r.push_back(u[pu++]);}
72          return r;
73      }
74      template<class T>void dfs(vector<pair<T,int> >&f,int i,T now,vector<T>&
    r){
75          if(i==f.size()){
76              r.push_back(now);
77              return;
78          }
79          for(int j=0;j<=f[i].second;++j,now*=f[i].first)
80              dfs(f,i+1,now,r);
81      }
82      template<class T>T run(T a){
83          vector<pair<T,int> >fa=fac(a),fpa;
84          if(fa.size()==0||fa.size()>2)
85              return −1;
86          if(fa.size()==1&&fa[0].first==2&&fa[0].second>2)
87              return −1;
88          if(fa.size()==2&&fa[0]!=make_pair(T(2),1))
89              return −1;
90          T pa=a;
91          for(int i=0;i<fa.size();++i)
92              pa=pa/fa[i].first*(fa[i].first−1);
93          fpa=fac(pa);
94          vector<T>fs;
95          dfs(fpa,0,1,fs);
96          for(T g=1,f=0;;++g,f=0){
97              for(int i=0;i<fs.size();++i)
98                  if(fs[i]!=pa&&pow(g,fs[i],a)==1){
99                      f=1;
100                     break;
101                 }
102             if(!f)
```

```
103                   return g;
104              }
105          }
106 }
```

4.10   Sequence

Sequence.txt (1134 bytes, 8 lines)

```
Numbers n such that a Hadamard matrix of order n exists.
1, 2, 4, 8, 12, 16, 20, 24, 28, 32, 36, 40, 44, 48, 52, 56, 60, 64, 68, 72,
 76, 80, 84, 88, 92, 96, 100, 104, 108, 112, 116, 120, 124, 128, 132, 136,
140, 144, 148, 152, 156, 160, 164, 168, 172, 176, 180, 184, 188, 192, 196,
200, 204, 208, 212, 216, 220, 224, 228, 232, 236, 240
```

```
Catalan numbers: Cₙ = 1/(n+1) (2n choose n) = (2n)!/((n+1)!n!). Also called Segner numbers.
1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862, 16796, 58786, 208012, 742900,
2674440, 9694845, 35357670, 129644790, 477638700, 1767263190, 6564120420,
24466267020, 91482563640, 343059613650, 1289904147324, 4861946401452,
18367353072152, 69533550916004, 263747951750360, 1002242216651368,
3814986502092304
```

```
Bell or exponential numbers: number of ways to partition a set of n labeled
 elements.
1, 1, 2, 5, 15, 52, 203, 877, 4140, 21147, 115975, 678570, 4213597,
27644437, 190899322, 1382958545, 10480142147, 82864869804, 682076806159,
5832742205057, 51724158235372, 474869816156751, 4506715738447323,
44152005855084346, 445958869294805289, 4638590332229999353,
49631246523618756274
```

CHAPTER 5

Numerical Algorithms

5.1   Convolution (Fast Fourier Transform)

Convolution (Fast Fourier Transform).hpp (1300 bytes, 39 lines)

```cpp
#include<bits/stdc++.h>
using namespace std;
namespace Convolution{
    typedef complex<double>T;
    void fft(vector<T>&a,int n,double s,vector<int>&rev){
        T im(0,1);
        double pi=acos(-1);
        for(int i=0;i<n;++i)
            if(i<rev[i])
                swap(a[i],a[rev[i]]);
        for(int i=1,m=2;(1<<i)<=n;++i,m<<=1){
            T wm=exp(s*im*2.0*pi/double(m)),w;
            for(int j=(w=1,0);j<n;j+=m,w=1)
                for(int k=0;k<(m>>1);++k,w*=wm){
                    T u=a[j+k],v=w*a[j+k+(m>>1)];
                    a[j+k]=u+v;
                    a[j+k+(m>>1)]=u-v;
                }
        }
    }
    vector<double>run(const vector<double>&a,const vector<double>&b){
        int l=ceil(log2(a.size()+b.size()-1)),n=1<<l;
        vector<int>rv;
        for(int i=(rv.resize(n),0);i<n;++i)
            rv[i]=(rv[i>>1]>>1)|((i&1)<<(l-1));
        vector<T>ta(n),tb(n);
        copy(a.begin(),a.end(),ta.begin());
        copy(b.begin(),b.end(),tb.begin());
        fft(ta,n,1,rv);
        fft(tb,n,1,rv);
        for(int i=0;i<n;++i)
            ta[i]*=tb[i];
        fft(ta,n,-1,rv);
        vector<double>c(a.size()+b.size()-1);
        for(int i=0;i<c.size();++i)
            c[i]=real(ta[i])/n;
        return c;
```

```
38 |      }
39 | }
```

5.2   Convolution (Karatsuba Algorithm)

Convolution (Karatsuba Algorithm).hpp (1416 bytes, 43 lines)

```cpp
 1 | #include<bits/stdc++.h>
 2 | using namespace std;
 3 | namespace Convolution{
 4 |     template<class T>void kar(T*a,T*b,int n,int l,T**r){
 5 |         T*rl=r[l],*rll=r[l-1];
 6 |         for(int i=0;i<2*n;++i)
 7 |             *(rl+i)=0;
 8 |         if(n<=30){
 9 |             for(int i=0;i<n;++i)
10 |                 for(int j=0;j<n;++j)
11 |                     *(rl+i+j)+=*(a+i)**(b+j);
12 |             return;
13 |         }
14 |         kar(a,b,n>>1,l-1,r);
15 |         for(int i=0;i<n;++i)
16 |             *(rl+i)+=*(rll+i),*(rl+i+(n>>1))+=*(rll+i);
17 |         kar(a+(n>>1),b+(n>>1),n>>1,l-1,r);
18 |         for(int i=0;i<n;++i)
19 |             *(rl+i+n)+=*(rll+i),*(rl+i+(n>>1))+=*(rll+i);
20 |         for(int i=0;i<(n>>1);++i){
21 |             *(rl+(n<<1)+i)=*(a+(n>>1)+i)-*(a+i);
22 |             *(rl+i+(n>>1)*5)=*(b+i)-*(b+(n>>1)+i);
23 |         }
24 |         kar(rl+(n<<1),rl+(n>>1)*5,n>>1,l-1,r);
25 |         for(int i=0;i<n;++i)
26 |             *(rl+i+(n>>1))+=*(rll+i);}
27 |     template<class T>vector<T>run(vector<T>a,vector<T>b){
28 |         int l=ceil(log2(max(a.size(),b.size()))+1e-8);
29 |         vector<T>rt(a.size()+b.size()-1);
30 |         a.resize(1<<l);
31 |         b.resize(1<<l);
32 |         T**r=new T*[l+1];
```

```cpp
33          for(int i=0;i<=l;++i)
34              r[i]=new T[(1<<i)*3];
35          kar(&a[0],&b[0],1<<l,l,r);
36          for(int i=0;i<rt.size();++i)
37              rt[i]=*(r[l]+i);
38          for(int i=0;i<=l;++i)
39              delete r[i];
40          delete r;
41          return rt;
42      }
43  }
```

5.3   Convolution (Number Theoretic Transform)

Convolution (Number Theoretic Transform).hpp (1620 bytes, 51 lines)

```cpp
1   #include<bits/stdc++.h>
2   using namespace std;
3   namespace Convolution{
4       typedef long long T;
5       T pow(T a,T b,T c){
6           T r=1;
7           for(;b;b&1?r=r*a%c:0,b>>=1,a=a*a%c);
8           return r;
9       }
10      void ntt(vector<T>&a,int n,int s,vector<int>&rev,T p,T g){
11          g=s==1?g:pow(g,p−2,p);
12          vector<T>wm;
13          for(int i=0;1<<i<=n;++i)
14              wm.push_back(pow(g,(p−1)>>i,p));
15          for(int i=0;i<n;++i)
16              if(i<rev[i])
17                  swap(a[i],a[rev[i]]);
18          for(int i=1,m=2;1<<i<=n;++i,m<<=1){
19              vector<T>wmk(1,1);
20              for(int k=1;k<(m>>1);++k)
21                  wmk.push_back(wmk.back()*wm[i]%p);
22              for(int j=0;j<n;j+=m)
23                  for(int k=0;k<(m>>1);++k){
```

```
24              T u=a[j+k],v=wmk[k]*a[j+k+(m>>1)]%p;
25              a[j+k]=u+v;
26              a[j+k+(m>>1)]=u−v+p;
27              if(a[j+k]>=p)
28                  a[j+k]−=p;
29              if(a[j+k+(m>>1)]>=p)
30                  a[j+k+(m>>1)]−=p;
31          }
32      }
33  }
34  vector<T>run(vector<T>a,vector<T>b,T p=15*(1<<27)+1,T g=31){
35      int tn,l=ceil(log2(tn=a.size()+b.size()−1)),n=1<<l;
36      vector<int>rv;
37      for(int i=(rv.resize(n),0);i<n;++i)
38          rv[i]=(rv[i>>1]>>1)|((i&1)<<(l−1));
39      a.resize(n);
40      b.resize(n);
41      ntt(a,n,1,rv,p,g);
42      ntt(b,n,1,rv,p,g);
43      for(int i=0;i<n;++i)
44          a[i]=a[i]*b[i]%p;
45      ntt(a,n,−1,rv,p,g);
46      n=pow(n,p−2,p);
47      for(T&v:a)
48          v=v*n%p;
49      return a.resize(tn),a;
50  }
51 }
```

5.4   Fraction

Fraction.hpp (2217 bytes, 100 lines)

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  template<class T>struct Fraction{
4      T p,q;
5      int s;
6      T gcd(T a,T b){
```

```cpp
 7          return b?gcd(b,a%b):a;
 8      }
 9      void reduce(){
10          T d=gcd(p,q);
11          p/=d;
12          q/=d;
13          if(p==0)
14              s=0;
15      }
16      Fraction(int _s=0,T _p=0,T _q=1):
17          s(_s),p(_p),q(_q){
18          reduce();
19      }
20      Fraction(string a){
21          if(a[0]=='-'){
22              s=-1;
23              a=a.substr(1,a.size()-1);
24          }else if(a[0]=='+'){
25              s=1;
26              a=a.substr(1,a.size()-1);
27          }else
28              s=1;
29          stringstream ss;
30          char tc;
31          ss<<a;
32          ss>>p>>tc>>q;
33          reduce();
34      }
35      Fraction(const char*a){
36          *this=Fraction(string(a));
37      }
38      Fraction<T>&operator=(string a){
39          return*this=Fraction<T>(a);
40      }
41      Fraction<T>&operator=(const char*a){
42          return*this=Fraction<T>(a);
43      }
44  };
45  template<class T>ostream&operator<<(ostream&s,const Fraction<T>&a){
46      if(a.s==-1)
47          s<<'-';
```

```cpp
48        return s<<a.p<<'/'<<a.q;
49  }
50  template<class T>istream&operator>>(istream&s,Fraction<T>&a){
51        string t;
52        s>>t;
53        a=t;
54        return s;
55  }
56  template<class T>vector<string>real(const Fraction<T>&a){
57        vector<string>r;
58        stringstream ss;
59        string st;
60        if(a.s<0)
61            r.push_back("−");
62        else
63            r.push_back("+");
64        T p=a.p,q=a.q;
65        ss<<p/q;
66        ss>>st;
67        r.push_back(st);
68        p%=q;
69        st.clear();
70        map<T,int>mp;
71        while(true){
72            if(p==0){
73                r.push_back(st);
74                r.push_back("");
75                return r;
76            }
77            if(mp.count(p)){
78                r.push_back(st.substr(0,mp[p]));
79                r.push_back(st.substr(mp[p],st.size()−mp[p]));
80                return r;
81            }
82            p*=10;
83            mp[p/10]=st.size();
84            st.push_back('0'+p/q);
85            p%=q;
86        }
87        return r;
88  }
```

```
 89  template<class T>string decimal(const Fraction<T>&a){
 90      string r;
 91      vector<string>t=real(a);
 92      if(t[0]=="−")
 93          r.push_back('−');
 94      r+=t[1];
 95      if(t[2].size()||t[3].size())
 96          r+="."+t[2];
 97      if(t[3].size())
 98          r+="("+t[3]+")";
 99      return r;
100  }
```

5.5   Integer

Integer.hpp (6378 bytes, 269 lines)

```
 1  #include<bits/stdc++.h>
 2  using namespace std;
 3  struct Integer operator+(Integer a,Integer b);
 4  Integer operator+(Integer a,int b);
 5  Integer operator−(Integer a,Integer b);
 6  Integer operator*(Integer a,Integer b);
 7  Integer operator*(Integer a,Integer b);
 8  Integer operator/(Integer a,Integer b);
 9  Integer operator%(Integer a,Integer b);
10  Integer operator%(Integer a,int b);
11  Integer operator%(Integer a,long long b);
12  bool operator!=(Integer a,int b);
13  bool operator<=(Integer a,int b);
14  struct Integer{
15      operator bool(){
16          return *this!=0;
17      }
18      Integer(long long a=0){
19          if(a<0){
20              s=−1;
21              a=−a;
22          }else
```

```
23              s=a!=0;
24          do{
25              d.push_back(a%B);
26              a/=B;
27          }while(a);
28      }
29      Integer(string a){
30          s=(a[0]=='-')?-1:(a!="0");
31          for(int i=a.size()-1;i>=(a[0]=='-');i-=L){
32              int t=0,j=max(i-L+1,int(a[0]=='-'));
33              for(int k=j;k<=i;++k)
34                  t=t*10+a[k]-'0';
35              d.push_back(t);
36          }
37      }
38      Integer(const Integer&a){
39          d=a.d;
40          s=a.s;
41      }
42      Integer&operator=(long long a){
43          return*this=Integer(a);
44      }
45      Integer&operator+=(Integer a){
46          return*this=*this+a;
47      }
48      Integer&operator-=(Integer a){
49          return*this=*this-a;
50      }
51      Integer&operator*=(Integer a){
52          return*this=*this*a;
53      }
54      Integer&operator/=(Integer a){
55          return*this=*this/a;
56      }
57      Integer&operator%=(Integer a){
58          return*this=*this%a;
59      }
60      Integer&operator++(){
61          return*this=*this+1;
62      }
63      operator string()const{
```

```
64              string r;
65              for(int i=0;i<d.size();++i){
66                  stringstream ts;
67                  ts<<d[i];
68                  string tt;
69                  ts>>tt;
70                  reverse(tt.begin(),tt.end());
71                  while(i+1!=d.size()&&tt.size()<L)
72                      tt.push_back('0');
73                  r+=tt;
74              }
75              reverse(r.begin(),r.end());
76              return r;
77          }
78          int s;
79          vector<int>d;
80          static const int B=1e8,L=8;
81      };
82      string str(const Integer&a){
83          return string(a);
84      }
85      bool operator<(Integer a,Integer b){
86          if(a.s!=b.s)
87              return a.s<b.s;
88          if(a.d.size()!=b.d.size())
89              return (a.s!=1)^(a.d.size()<b.d.size());
90          for(int i=a.d.size()−1;i>=0;−−i)
91              if(a.d[i]!=b.d[i])
92                  return (a.s!=1)^(a.d[i]<b.d[i]);
93          return false;
94      }
95      bool operator>(Integer a,Integer b){
96          return b<a;
97      }
98      bool operator<=(Integer a,Integer b){
99          return !(a>b);
100     }
101     bool operator>=(Integer a,Integer b){
102         return !(a<b);
103     }
104     bool operator==(Integer a,Integer b){
```

```
105         return !(a<b)&&!(a>b);
106   }
107   bool operator!=(Integer a,Integer b){
108         return !(a==b);
109   }
110   istream&operator>>(istream&s,Integer&a){
111         string t;
112         s>>t;
113         a=Integer(t);
114         return s;
115   }
116   ostream&operator<<(ostream&s,Integer a){
117         if(a.s==−1)
118             s<<'−';
119         for(int i=a.d.size()−1;i>=0;−−i){
120             if(i!=a.d.size()−1)
121                 s<<setw(Integer::L)<<setfill('0');
122             s<<a.d[i];
123         }
124         s<<setw(0)<<setfill(' ');
125         return s;
126   }
127   void dzero(Integer&a){
128         while(a.d.size()>1&&a.d.back()==0)
129             a.d.pop_back();
130   }
131   Integer operator−(Integer a){
132         a.s*=−1;
133         if(a.d.size()==1&&a.d[0]==0)
134             a.s=1;
135         return a;
136   }
137   Integer operator+(Integer a,int b){
138         return a+Integer(b);
139   }
140   Integer operator*(Integer a,int b){
141         return a*Integer(b);
142   }
143   Integer operator%(Integer a,int b){
144         return a%Integer(b);
145   }
```

```
146  Integer operator%(Integer a,long long b){
147      return a%Integer(b);
148  }
149  bool operator!=(Integer a,int b){
150      return a!=Integer(b);
151  }
152  bool operator<=(Integer a,int b){
153      return a<=Integer(b);
154  }
155  Integer operator+(Integer a,Integer b){
156      if(a.s*b.s!=−1){
157          Integer c;c.s=a.s?a.s:b.s;
158          c.d.resize(max(a.d.size(),b.d.size())+1);
159          for(int i=0;i<c.d.size()−1;++i){
160              if(i<a.d.size())
161                  c.d[i]+=a.d[i];
162              if(i<b.d.size())
163                  c.d[i]+=b.d[i];
164              if(c.d[i]>=Integer::B){
165                  c.d[i]−=Integer::B;
166                  ++c.d[i+1];
167              }
168          }
169          dzero(c);
170          return c;
171      }
172      return a−(−b);
173  }
174  Integer operator−(Integer a,Integer b){
175      if(a.s*b.s==1){
176          if(a.s==−1)
177              return (−b)−(−a);
178          if(a<b)
179              return −(b−a);
180          if(a==b)
181              return 0;
182          for(int i=0;i<b.d.size();++i){
183              a.d[i]−=b.d[i];
184              if(a.d[i]<0){
185                  a.d[i]+=Integer::B;
186                  −−a.d[i+1];
```

```
187                     }
188                 }
189             dzero(a);
190             return a;
191         }
192         return a+(−b);
193 }
194 Integer operator*(Integer a,Integer b){
195         vector<long long>t(a.d.size()+b.d.size());
196         for(int i=0;i<a.d.size();++i)
197             for(int j=0;j<b.d.size();++j)
198                 t[i+j]+=(long long)a.d[i]*b.d[j];
199         for(int i=0;i<t.size()−1;++i){
200             t[i+1]+=t[i]/Integer::B;
201             t[i]%=Integer::B;
202         }
203         Integer c;
204         c.s=a.s*b.s;c.d.resize(t.size());
205         copy(t.begin(),t.end(),c.d.begin());
206         dzero(c);
207         return c;
208 }
209 Integer div2(Integer a){
210         for(int i=a.d.size()−1;i>=0;−−i){
211             if(i)
212                 a.d[i−1]+=(a.d[i]&1)*Integer::B;
213             a.d[i]>>=1;
214         }
215         dzero(a);
216         if(a.d.size()==1&&a.d[0]==0)
217             a.s=0;
218         return a;
219 }
220 Integer operator/(Integer a,Integer b){
221         if(!a.s)
222             return 0;
223         if(a.s<0)
224             return−((−a)/b);
225         if(a<b)
226             return 0;
227         Integer l=1,r=1;
```

```
228        while(r*b<=a)
229            r=r*2;
230        while(l+1<r){
231            Integer m=div2(l+r);
232            if(m*b>a)
233                r=m;
234            else
235                l=m;
236        }
237        return l;
238 }
239 Integer operator%(Integer a,Integer b){
240        return a−a/b*b;
241 }
242 Integer gcd(Integer a,Integer b){
243        Integer r=1;
244        while(a!=0&&b!=0){
245            if(!(a.d[0]&1)&&!(b.d[0]&1)){
246                a=div2(a);
247                b=div2(b);
248                r=r*2;
249            }else if(!(a.d[0]&1))
250                a=div2(a);
251            else if(!(b.d[0]&1))
252                b=div2(b);
253            else{
254                if(a<b)
255                    swap(a,b);
256                a=div2(a−b);
257            }
258        }
259        if(a!=0)
260            return r*a;
261        return r*b;
262 }
263 int length(Integer a){
264        a.s=1;
265        return string(a).size();
266 }
267 int len(Integer a){
268        return length(a);
```

```
269 | }
```

5.6   Linear Programming

Linear Programming.hpp (2522 bytes, 89 lines)

```
 1  #include<bits/stdc++.h>
 2  using namespace std;
 3  struct LinearProgramming{
 4      const double E;
 5      int n,m,p;
 6      vector<int>mp,ma,md;
 7      vector<vector<double> >a;
 8      vector<double>res;
 9      LinearProgramming(int _n,int _m):
10          n(_n),m(_m),p(0),a(n+2,vector<double>(m+2)),mp(n+1),ma(m+n+2),md(m
    +2),res(m+1),E(1e-8){
11      }
12      void piv(int l,int e){
13          swap(mp[l],md[e]);
14          ma[mp[l]]=l;
15          ma[md[e]]=-1;
16          double t=-a[l][e];
17          a[l][e]=-1;
18          vector<int>qu;
19          for(int i=0;i<=m+1;++i)
20              if(fabs(a[l][i]/=t)>E)
21                  qu.push_back(i);
22          for(int i=0;i<=n+1;++i)
23              if(i!=l&&fabs(a[i][e])>E){
24                  t=a[i][e];
25                  a[i][e]=0;
26                  for(int j=0;j<qu.size();++j)
27                      a[i][qu[j]]+=a[l][qu[j]]*t;
28              }
29          if(-p==l)
30              p=e;
31          else if(p==e)
32              p=-l;
```

```cpp
33          }
34      int opt(int d){
35          for(int l=-1,e=-1;;piv(l,e),l=-1,e=-1){
36              for(int i=1;i<=m+1;++i)
37                  if(a[d][i]>E){
38                      e=i;
39                      break;
40                  }
41              if(e==-1)
42                  return 1;
43              double t;
44              for(int i=1;i<=n;++i)
45                  if(a[i][e]<-E&&(l==-1||a[i][0]/-a[i][e]<t))
46                      t=a[i][0]/-a[i][e],l=i;
47              if(l==-1)
48                  return 0;
49          }
50      }
51      double&at(int x,int y){
52          return a[x][y];
53      }
54      vector<double>run(){
55          for(int i=1;i<=m+1;++i)
56              ma[i]=-1,md[i]=i;
57          for(int i=m+2;i<=m+n+1;++i)
58              ma[i]=i-(m+1),mp[i-(m+1)]=i;
59          double t;
60          int l=-1;
61          for(int i=1;i<=n;++i)
62              if(l==-1||a[i][0]<t)
63                  t=a[i][0],l=i;
64          if(t<-E){
65              for(int i=1;i<=n;++i)
66                  a[i][m+1]=1;
67              a[n+1][m+1]=-1;
68              p=m+1;
69              piv(l,m+1);
70              if(!opt(n+1)||fabs(a[n+1][0])>E)
71                  return vector<double>();
72              if(p<0)
73                  for(int i=1;i<=m;++i)
```

```
74                          if(fabs(a[−p][i])>E){
75                              piv(−p,i);
76                              break;
77                          }
78                  for(int i=0;i<=n;++i)
79                      a[i][p]=0;
80              }
81          if(!opt(0))
82              return vector<double>();
83          res[0]=a[0][0];
84          for(int i=1;i<=m;++i)
85              if(ma[i]!=−1)
86                  res[i]=a[ma[i]][0];
87          return res;
88      }
89  };
```

5.7   Linear System

Linear System.hpp (1477 bytes, 56 lines)

```
1   #include<bits/stdc++.h>
2   using namespace std;
3   template<class T>struct LinearSystem{
4       int n;
5       vector<vector<T> >a;
6       vector<int>main,pos;
7       vector<T>ans;
8       int cmp(T a){
9           if(typeid(T)==typeid(double)||typeid(T)==typeid(long double)||
    typeid(T)==typeid(float)){
10              if(a<−1e−8)
11                  return −1;
12              if(a>1e−8)
13                  return 1;
14              return 0;
15          }
16          if(a<0)
17              return −1;
```

```cpp
18          if(a>0)
19              return 1;
20          return 0;
21      }
22      T&at(int i,int j){
23          return a[i][j];
24      }
25      vector<T>&at(int i){
26          return a[i];
27      }
28      LinearSystem(int _n):
29          n(_n),a(n+1,vector<T>(n+1)),main(n+1),pos(n+1),ans(n){
30      }
31      vector<T>run(){
32          for(int i=1;i<=n;++i){
33              int j=1;
34              for(;j<=n&&!cmp(a[i][j]);++j);
35              if(j<=n){
36                  main[i]=j;
37                  pos[j]=i;
38                  T t=a[i][j];
39                  for(int k=0;k<=n;++k)
40                      a[i][k]/=t;
41                  for(int k=1;k<=n;++k)
42                      if(k!=i&&cmp(a[k][j])){
43                          t=a[k][j];
44                          for(int l=0;l<=n;++l)
45                              a[k][l]-=a[i][l]*t;
46                      }
47              }
48          }
49          for(int i=1;i<=n;++i){
50              if(!pos[i])
51                  return vector<T>();
52              ans[i-1]=a[pos[i]][0];
53          }
54          return ans;
55      }
56  };
```

5.8　Matrix

Matrix.hpp (1457 bytes, 51 lines)

```cpp
#include<bits/stdc++.h>
template<class T,int N>struct Matrix{
    Matrix(T t=0){
        for(int i=0;i<N;++i)
            for(int j=0;j<N;++j)
                u[i][j]=i==j?t:0;
    }
    T u[N][N];
};
template<class T,int N>Matrix<T,N>operator+(const Matrix<T,N>&a,const
    Matrix<T,N>&b){
    Matrix<T,N>c;
    for(int i=0;i<N;++i)
        for(int j=0;j<N;++j)
            c.u[i][j]=a.u[i][j]+b.u[i][j];
    return c;
}
template<class T,int N>Matrix<T,N>operator*(const Matrix<T,N>&a,const
    Matrix<T,N>&b){
    Matrix<T,N>c;
    for(int i=0;i<N;++i)
        for(int j=0;j<N;++j)
            for(int k=0;k<N;++k)
                c.u[i][j]+=a.u[i][k]*b.u[k][j];
    return c;
}
template<class T,int N>Matrix<T,N>operator*(const Matrix<T,N>&a,const T&b){
    Matrix<T,N>c=a;
    for(int i=0;i<N;++i)
        for(int j=0;j<N;++j)
            c.u[i][j]*=b;
    return c;
}
template<class T,int N>Matrix<T,N>operator/(const Matrix<T,N>&a,const T&b){
    Matrix<T,N>c=a;
    for(int i=0;i<N;++i)
        for(int j=0;j<N;++j)
```

```
36              c.u[i][j]/=b;
37      return c;
38  }
39  template<class T,int N>Matrix<T,N>pow(Matrix<T,N>a,long long b){
40      Matrix<T,N>r(1);
41      for(;b;a=a*a,b>>=1)
42          if(b&1)
43              r=r*a;
44      return r;
45  }
46  template<class T,int N>ostream&operator<<(ostream&s,const Matrix<T,N>a){
47      for(int i=0;i<N;++i)
48          for(int j=0;j<N;++j)
49              s<<a.u[i][j]<<(j+1==N?'\n':' ');
50      return s;
51  }
```

5.9   Polynomial Interpolation

Polynomial Interpolation.hpp (372 bytes, 15 lines)

```
1   #include<bits/stdc++.h>
2   using namespace std;
3   template<class T>T PolynomialInterpolation(vector<T>x,vector<T>y,T x0){
4       T r=0;
5       for(int i=0;i<x.size();++i){
6           T p=1,q=1;
7           for(int j=0;j<x.size();++j)
8               if(j!=i){
9                   p*=(x0-x[j]);
10                  q*=(x[i]-x[j]);
11              }
12          r+=p/q*y[i];
13      }
14      return r;
15  }
```

CHAPTER 6

String Algorithms

6.1   Aho-Corasick Automaton

Aho-Corasick Automaton.hpp (1369 bytes, 50 lines)

```cpp
#include<bits/stdc++.h>
using namespace std;
struct AhoCorasickAutomaton{
    struct node{
        node(int m):
            tr(m),fail(0),cnt(0){
        }
        vector<node*>tr;
        node*fail;
        int cnt;
    };
    int m;
    node*root;
    vector<node*>all;
    AhoCorasickAutomaton(int _m):
        m(_m),root(new node(m)),all(1,root){
    }
    ~AhoCorasickAutomaton(){
        for(int i=0;i<all.size();++i)
            delete all[i];
    }
    node*insert(int*s){
        node*p;
        for(p=root;*s!=-1;p=p->tr[*(s++)])
            if(!p->tr[*s])
                p->tr[*s]=new node(m);
        return p;
    }
    void build(){
        queue<node*>qu;
        for(int i=0;i<m;++i)
            if(!root->tr[i])
                root->tr[i]=root;
            else
                root->tr[i]->fail=root,qu.push(root->tr[i]);
        for(node*u;qu.size()?(u=qu.front(),qu.pop(),all.push_back(u),1):0;)
            for(int i=0;i<m;++i)
```

```
38              if(!u−>tr[i])
39                  u−>tr[i]=u−>fail−>tr[i];
40              else
41                  u−>tr[i]−>fail=u−>fail−>tr[i],qu.push(u−>tr[i]);
42      }
43      void run(int*s){
44          for(node*p=root;*s!=−1;++(p=p−>tr[*(s++)])−>cnt);
45      }
46      void count(){
47          for(int i=all.size()−1;i>=1;−−i)
48              all[i]−>fail−>cnt+=all[i]−>cnt;
49      }
50  };
```

6.2   Factor Oracle

Factor Oracle.hpp (597 bytes, 18 lines)

```
1   #include<bits/stdc++.h>
2   using namespace std;
3   template<class T,int N,int M,T D>struct FactorOracle{
4       FactorOracle(){
5           memset(tr,(lrs[0]=0,sp[0]=−1),4*M);
6       }
7       void insert(T*s,int n){
8           for(int i=0,j,c=s[i]−D,u,v;i<n;c=s[++i]−D){
9               memset(tr+i+1,(lrs[i+1]=0)−1,4*M);
10              for(j=i;j>−1&&tr[j][c]<0;tr[j][c]=i+1,j=sp[u=j]);
11              if(v=sp[i+1]=j<0?0:tr[j][c]){
12                  for(v=v−1==sp[u]?u:v−1;sp[u]!=sp[v];v=sp[v]);
13                  lrs[i+1]=min(lrs[u],lrs[v])+1;
14              }
15          }
16      }
17      int sp[N+1],lrs[N+1],tr[N+1][M];
18  };
```

6.3   Longest Common Substring

Longest Common Substring.hpp (1181 bytes, 28 lines)

```cpp
#include<bits/stdc++.h>
using namespace std;
template<class T,int N,int M,T D>struct LongestCommonSubstring{
    void ins(int c){
        memset(tr+i+1,(lrs[i+1]=0)-1,4*M);
        for(j=i;j>-1&&((v=tr[j][c])>=l1+2&&v<=l1+lb+1||v<0);tr[j][c]=i+1+lb
,j=sp[u=j]);
        if(v=sp[i+1]=j<0?0:tr[j][c]-(tr[j][c]>l1+1)*lb){
            for(v=v-1==sp[u]?u:v-1;sp[u]!=sp[v];v=sp[v]);
            lrs[i+1]=min(lrs[u],lrs[v])+1;
        }
        if(sp[i+1]<=l1)
            tm[sp[i+1]]=max(tm[sp[i+1]],lrs[i+1]);
    }
    int run(vector<pair<int,T*> >s){
        swap(s[0],*min_element(s.begin(),s.end()));
        l1=s[k=lb=0].first;
        memset(mi,63,4*N+4);
        memset(tr,(lrs[0]=0,sp[0]=-1),4*M+4);
        for(i=0;i<l1;ins(*(s[0].second+i)-D),++i);
        for(k=1,ins(M);k<s.size();lb+=s[k++].first){
            memset(tm,0,4*N+4);
            for(i=l1+1;i-l1-1<s[k].first;ins(*(s[k].second+i-l1-1)-D),++i)
;
            for(i=l1;i;mi[i]=min(mi[i],tm[i]),tm[sp[i]]=max(tm[sp[i]],lrs[i
]*!!tm[i]),--i);
        }
        return min(*max_element(mi+1,mi+l1+1),l1);
    }
    int sp[2*N+2],lrs[2*N+2],tr[2*N+2][M+1],mi[N+1],tm[N+1],l1,lb,i,j,k,u,v
;
};
```

6.4   Palindromic Tree

Palindromic Tree.hpp (1327 bytes, 50 lines)

```cpp
#include<bits/stdc++.h>
using namespace std;
template<class T>struct PalindromicTree{
    struct node{
        node(int m,node*f,int l):
            nxt(m),fail(f),len(l){
        }
        vector<node*>nxt;
        node*fail;
        T val;
        int len;
    }*root;
    int m;
    vector<int>str;
    vector<node*>all;
    PalindromicTree(int _m):
        m(_m){
        node*n0=new node(m,0,-2),*n1=new node(m,n0,-1),*n2=new node(m,n1,0)
;
        all.push_back(n0);
        all.push_back(n1);
        all.push_back(n2);
        fill(n0->nxt.begin(),n0->nxt.end(),n2);
        root=n1;
    }
    ~PalindromicTree(){
        for(int i=0;i<all.size();++i)
            delete all[i];
    }
    node*find(node*x){
        while(x->fail&&str[str.size()-x->len-2]!=str[str.size()-1])
            x=x->fail;
        return x;
    }
    node*insert(node*p,int c,T v){
        if(p==root)
            str=vector<int>(1,-1);
```

```
37              str.push_back(c);
38              p=find(p);
39              if(!p->nxt[c]){
40                  node*np=(p->nxt[c]=new node(m,find(p->fail)->nxt[c],p->len+2))
        ;
41                  all.push_back(np);
42              }
43              p->nxt[c]->val+=v;
44              return p->nxt[c];
45          }
46          void count(){
47              for(int i=all.size()-1;i>=1;--i)
48                  all[i]->fail->val+=all[i]->val;
49          }
50      };
```

6.5   String Searching

String Searching.hpp (682 bytes, 25 lines)

```cpp
1   #include<bits/stdc++.h>
2   using namespace std;
3   template<class T>struct StringSearching{
4       StringSearching(T*a):
5           b(2,a[l]),f(2),l(2){
6           for(int i=0;a[l]?1:(--l,0);b.push_back(a[l++])){
7               for(;i&&a[i+1]!=a[l];i=f[i]);
8               f.push_back(i=i+(a[i+1]==a[l]));
9           }
10          for(int i=2;i<l;++i)
11              if(a[f[i]+1]==a[i+1])
12                  f[i]=f[f[i]];
13      }
14      int run(T*a,int p){
15          for(int i=p?p+l:1,j=p?f[l]:0;a[i];++i){
16              for(;j&&b[j+1]!=a[i];j=f[j]);
17              if((j+=b[j+1]==a[i])==l)
18                  return i-l+1;
19          }
```

```
20          return 0;
21      }
22      int l;
23      vector<T>b;
24      vector<int>f;
25  };
```

6.6   String

String.hpp (987 bytes, 45 lines)

```
1   #include<bits/stdc++.h>
2   using namespace std;
3   string read_all(){
4       string t;
5       getline(cin,t,char(EOF));
6       return t;
7   }
8   string delete_all(string a,char b){
9       string r;
10      for(int i=0;i<a.size();++i)
11          if(a[i]!=b)
12              r.push_back(a[i]);
13      return r;
14  }
15  string substr(string a,int l,int r){
16      return a.substr(l,r-l+1);
17  }
18  vector<string>split(string a,char b){
19      vector<string>r;
20      string t;
21      for(int i=0;i<a.size();++i)
22          if(a[i]!=b)
23              t.push_back(a[i]);
24          else{
25              r.push_back(t);
26              t="";
27          }
28      r.push_back(t);
```

```
29        return r;
30 }
31 int letter_order(char a){
32        return a>='a'&&a<='z'?a-'a'+1:a-'A'+1;
33 }
34 int letter_sum(string a){
35        int r=0;
36        for(int i=0;i<a.size();++i)
37            r+=letter_order(a[i]);
38        return r;
39 }
40 bool is_palindromic_string(string a){
41        for(int i=0;i<a.size();++i)
42            if(a[i]!=a[a.size()-i-1])
43                return false;
44        return true;
45 }
```

6.7   Suffix Array (DC3 Algorithm)

Suffix Array (DC3 Algorithm).hpp (2952 bytes, 107 lines)

```
1 #include<bits/stdc++.h>
2 using namespace std;
3 struct SuffixArray{
4        int*sa,*ht,*rk,*ts,*ct,ln;
5        SuffixArray(int*s){
6            int m=0;
7            for(ln=0;s[ln+1];)
8                m=max(m,s[++ln]);
9            crt(sa,ln);
10           crt(ht,ln);
11           crt(rk,ln);
12           crt(ts,ln);
13           crt(ct,max(ln,m));
14           dc3(s,ln,m,sa,rk);
15           for(int i=1;i<=ln;++i){
16               if(rk[i]==1){
17                   ht[1]=0;
```

```
18              continue;
19          }
20          int&d=ht[rk[i]]=max(i==1?0:ht[rk[i-1]]-1,0);
21          for(;i+d<=ln&&sa[rk[i]-1]+d<=ln&&s[i+d]==s[sa[rk[i]-1]+d];++d);
22      }
23  }
24  ~SuffixArray(){
25      del(sa);
26      del(ht);
27      del(rk);
28      del(ts);
29      del(ct);
30  }
31  void crt(int*&a,int n){
32      a=new int[n+1];
33  }
34  void del(int*a){
35      delete a;
36  }
37  #define fc(i)(p0[i]+d>n||!p0[i]?0:s[p0[i]+d])
38  int cmp(int*p0,int i,int*s,int n){
39      for(int d=0;d<3;++d)
40          if(fc(i)!=fc(i-1))
41              return 1;
42      return 0;
43  }
44  void sot(int*p0,int n0,int*s,int n,int m,int d){
45      memset(ct,0,(m+1)*4);
46      for(int i=1;i<=n0;++i)
47          ++ct[fc(i)];
48      for(int i=1;i<=m;++i)
49          ct[i]+=ct[i-1];
50      for(int i=n0;i>=1;--i)
51          ts[ct[fc(i)]--]=p0[i];
52      memcpy(p0+1,ts+1,n0*4);
53  }
54  #define fc(d)if(s[i+d]!=s[j+d])return s[i+d]<s[j+d];if(i==n-d||j==n-d)return i==n-d;
55  bool cmp(int*s,int n,int*r,int i,int j){
56      fc(0)
57      if(j%3==1)
```

```
58              return r[i+1]<r[j+1];
59          fc(1)
60          return r[i+2]<r[j+2];
61      }
62      #undef fc
63      void dc3(int*s,int n,int m,int*a,int*r){
64          int n0=n-(n/3)+1,*a0,*s0,i,j=0,k=n/3+bool(n%3)+1,l;
65          crt(s0,n0);
66          s0[k]=1;
67          crt(a0,n0+1);
68          a0[k]=0;
69          for(i=1;i<=n;i+=3)
70              a0[++j]=i,a0[j+k]=i+1;
71          for(i=2;i>=0;--i)
72              sot(a0,n0,s,n,m,i);
73          r[a0[1]]=1;
74          for(i=2;i<=n0;++i)
75              r[a0[i]]=r[a0[i-1]]+cmp(a0,i,s,n);
76          for(i=1,j=0;i<=n;i+=3)
77              s0[++j]=r[i],s0[j+k]=r[i+1];
78          if(r[a0[n0]]==n0){
79              memcpy(r+1,s0+1,n0*4);
80              for(i=1;i<=n0;++i)
81                  a0[a[i]=r[i]]=i;
82          }else
83              dc3(s0,n0,r[a0[n0]],a0,a);
84          for(i=1,j=0;i<=n;i+=3)
85              r[i]=a[++j],r[i+1]=a[j+k];
86          j=0;
87          if(n%3==0)
88              s0[++j]=n;
89          for(i=1;i<=n0;++i)
90              if(a0[i]<k){
91                  a0[i]=3*a0[i]-2;
92                  if(a0[i]!=1)
93                      s0[++j]=a0[i]-1;
94              }else
95                  a0[i]=(a0[i]-k)*3-1;
96          sot(s0,j,s,n,m,0);
97          for(i=1,k=2,l=0;i<=j||k<=n0;)
98              if(k>n0||i<=j&&cmp(s,n,r,s0[i],a0[k]))
```

```
 99                  a[++l]=s0[i++];
100              else
101                  a[++l]=a0[k++];
102          for(i=1;i<=n;++i)
103              r[a[i]]=i;
104          del(a0);
105          del(s0);
106      }
107  };
```

6.8 Suffix Array (Prefix-Doubling Algorithm)

Suffix Array (Prefix-Doubling Algorithm).hpp (1357 bytes, 55 lines)

```
 1  #include<bits/stdc++.h>
 2  using namespace std;
 3  struct SuffixArray{
 4      int*a,*h,*r,*t,*c,n,m;
 5      #define lp(u,v)for(int i=u;i<=v;++i)
 6      #define rp(u,v)for(int i=u;i>=v;--i)
 7      void sort(){
 8          memset(c+1,0,m*4);
 9          lp(1,n)
10              ++c[r[t[i]]];
11          lp(2,m)
12              c[i]+=c[i-1];
13          rp(n,1)
14              a[c[r[t[i]]]--]=t[i];
15      }
16      SuffixArray(int*s){
17          for(n=m=0;s[n+1];m=max(m,s[++n]));
18          a=new int[4*n+max(n,m)+3];
19          h=a+n;
20          r=h+n+1;
21          t=r+n+1;
22          c=t+n;
23          lp(1,n)
24              t[i]=i,r[i]=s[i];
25          sort();
```

```
26          for(int l=1;l<=n;l<<=1,r[a[n]]==n?l=n+1:m=r[a[n]]){
27              t[0]=0;
28              lp(n-l+1,n)
29                  t[++t[0]]=i;
30              lp(1,n)
31                  if(a[i]>l)
32                      t[++t[0]]=a[i]-l;
33              sort();
34              swap(r,t);
35              r[a[1]]=1;
36              lp(2,n)
37                  r[a[i]]=r[a[i-1]]+(t[a[i]]!=t[a[i-1]]||a[i]+l>n||a[i-1]+l>n
        ||t[a[i]+l]!=t[a[i-1]+l]);
38          }
39          int l=0;
40          a[0]=n+1;
41          lp(1,n){
42              if(r[i]==1)
43                  l=0;
44              l-=(l>0);
45              int j=a[r[i]-1];
46              for(;s[i+l]==s[j+l];++l);
47              h[r[i]]=l;
48          }
49      }
50      #undef lp
51      #undef rp
52      ~SuffixArray(){
53          delete a;
54      }
55 };
```

6.9   Suffix Array (Treap)

Suffix Array (Treap).hpp (3803 bytes, 147 lines)

```
1 #include<bits/stdc++.h>
2 using namespace std;
3 template<class T>struct SuffixArray{
```

```
4       struct node{
5           node*c[2],*p;
6           T v;
7           int f,s,l,h,m;
8           double t;
9           node(node*_p,T _v,int _l):
10              f(rand()*1.0/RAND_MAX*1e9),p(_p),v(_v),s(1),l(_l),h(0),m(0),t(5
        e8){
11              c[0]=c[1]=0;
12          }
13      }*root;
14      vector<T>a;
15      SuffixArray():
16          root(new node(0,0,0)),a(1){
17      }
18      ~SuffixArray(){
19          clear(root);
20      }
21      void relabel(node*x,double l,double r){
22          x->t=(l+r)/2;
23          if(x->c[0])
24              relabel(x->c[0],l,x->t);
25          if(x->c[1])
26              relabel(x->c[1],x->t,r);
27      }
28      void update(node*x){
29          x->s=1;
30          x->m=x->h;
31          for(int i=0;i<2;++i)
32              if(x->c[i])
33                  x->s+=x->c[i]->s,x->m=min(x->m,x->c[i]->m);
34      }
35      void rotate(node*&x,int d){
36          node*y=x->c[d];
37          x->c[d]=y->c[!d];
38          y->c[!d]=x;
39          y->s=x->s;
40          y->m=x->m;
41          update(x);
42          x=y;
43      }
```

```
44     void clear(node*x){
45         if(!x)
46             return;
47         clear(x->c[0]);
48         clear(x->c[1]);
49         delete x;
50     }
51     node*insert(node*&x,node*p,T v,node*l,node*r){
52         int d=x->v!=v?x->v<v:x->p->t<p->t;
53         double tl=l?l->t:0,tr=r?r->t:1e9;
54         node*y;
55         if(d)
56             l=x;
57         else
58             r=x;
59         if(!x->c[d]){
60             y=new node(p,v,p->l+1);
61             y->t=((l?l->t:0)+(r?r->t:1e9))/2;
62             y->m=y->h=l->v==y->v?lcp(l->p,y->p)+1:0;
63             if(r)
64                 r->h=r->v==y->v?lcp(r->p,y->p)+1:0;
65             x->c[d]=y;
66         }else
67             y=insert(x->c[d],p,v,l,r);
68         update(x);
69         if(x->c[d]->f>x->f)
70             rotate(x,d),relabel(x,tl,tr);
71         return y;
72     }
73     node*insert(node*p,T v){
74         a.push_back(v);
75         return insert(root,p,v,0,0);
76     }
77     void erase(node*&x,node*y){
78         if(x==y){
79             if(!x->c[0]){
80                 x=x->c[1];
81                 delete y;
82             }else if(!x->c[1]){
83                 x=x->c[0];
84                 delete y;
```

```
85                }else{
86                    int d=x->c[0]->f<x->c[1]->f;
87                    rotate(x,d);
88                    erase(x->c[!d],y);
89                    --x->s;
90                }
91            }else
92                erase(x->c[x->t<y->t],y),update(x);
93        }
94        void erase(node*y){
95            erase(root,y);
96            a.pop_back();
97        }
98        bool check(node*x,T*y,node*&p,int&l){
99            if(p){
100                int t=x->c[p->t>x->t]?x->c[p->t>x->t]->m:~0u>>1;
101                if(p->t>x->t)
102                    t=min(t,p->h);
103                else
104                    t=min(t,x->h);
105                if(t<l)
106                    return x->t<p->t;
107            }
108            for(p=x;l+1<=x->l&&y[l+1];++l)
109                if(a[x->l-l]!=y[l+1])
110                    return a[x->l-l]<y[l+1];
111            return y[l+1]!=0;
112        }
113        int count(node*x,T*y){
114            int r=0,l=0;
115            for(node*p=0;x;)
116                if(check(x,y,p,l))
117                    r+=(x->c[0]?x->c[0]->s:0)+1,x=x->c[1];
118                else
119                    x=x->c[0];
120            return r;
121        }
122        int count(T*y){
123            T*t=y;
124            while(*(t+1))
125                ++t;
```

```
126          int r=-count(root,y);
127          ++*t;
128          r+=count(root,y);
129          --*t;
130          return r;
131      }
132      int lcp(node*x,double u,double v,double l,double r){
133          if(v<l||u>r||!x)
134              return ~0u>>1;
135          if(u<l&&v>=r)
136              return x->m;
137          int t=u<x->t&&v>=x->t?x->h:~0u>>1;
138          t=min(t,lcp(x->c[0],u,v,l,x->t));
139          t=min(t,lcp(x->c[1],u,v,x->t,r));
140          return t;
141      }
142      int lcp(node*x,node*y){
143          if(x->t>y->t)
144              swap(x,y);
145          return lcp(root,x->t,y->t,0,1e9);
146      }
147 };
```

6.10   Suffix Automaton

Suffix Automaton.hpp (1694 bytes, 59 lines)

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  template<class T>struct SuffixAutomaton{
4      struct node{
5          node(vector<node*>&all,int m,node*_pr=0,int _ln=0,T _va=T()):
6              pr(_pr),tr(m),ln(_ln),va(_va){
7              all.push_back(this);
8          }
9          T va;
10         int ln;
11         node*pr;
12         vector<node*>tr;
```

```
13          };
14          SuffixAutomaton(int _m):
15              root(new node(all,m)),m(_m){
16          }
17          ~SuffixAutomaton(){
18              for(int i=0;i<all.size();++i)
19                  delete all[i];
20          }
21          node*insert(node*lst,int c,T v){
22              node*p=lst,*np=p->tr[c]?0:new node(all,m,0,lst->ln+1,v);
23              for(;p&&!p->tr[c];p=p->pr)
24                  p->tr[c]=np;
25              if(!p)np->pr=root;
26              else{
27                  node*q=p->tr[c];
28                  if(p==lst)
29                      np=q;
30                  if(q->ln==p->ln+1)
31                      p==lst?(q->va+=v):(np->pr=q,0);
32                  else{
33                      node*nq=new node(all,m,q->pr,p->ln+1,p==lst?v:T());
34                      nq->tr=q->tr;
35                      q->pr=np->pr=nq;
36                      if(p==lst)
37                          np=nq;
38                      for(;p&&p->tr[c]==q;p=p->pr)
39                          p->tr[c]=nq;
40                  }
41              }
42              return np;
43          }
44          void count(){
45              vector<int>cnt(all.size());
46              vector<node*>tmp=all;
47              for(int i=0;i<tmp.size();++i)
48                  ++cnt[tmp[i]->ln];
49              for(int i=1;i<cnt.size();++i)
50                  cnt[i]+=cnt[i-1];
51              for(int i=0;i<tmp.size();++i)
52                  all[--cnt[tmp[i]->ln]]=tmp[i];
53              for(int i=int(all.size())-1;i>0;--i)
```

```
54              all[i]->pr->va+=all[i]->va;
55          }
56      int m;
57      node*root;
58      vector<node*>all;
59  };
```

6.11   Suffix Tree

Suffix Tree.hpp (2899 bytes, 116 lines)

```
 1  #include<bits/stdc++.h>
 2  using namespace std;
 3  template<class T=char,int N=int(1e5),int M=27,T D='a'-1>struct SuffixTree{
 4      struct node;
 5      struct edge{
 6          edge():
 7              l(0),r(0),t(0){
 8          }
 9          int length(){
10              return r-l;
11          }
12          T*l,*r;
13          node*t;
14      }pe[2*N],*ep=pe;
15      edge*newedge(T*l,T*r,node*t){
16          ep->l=l;
17          ep->r=r;
18          ep->t=t;
19          return ep++;
20      }
21      struct node{
22          node():
23              s(0),c({0}){
24          }
25          node*s;
26          edge*c[M];
27      }pn[2*N+1],*np=pn;
28      SuffixTree():
```

```
29              root(np++),ct(0){
30          }
31      void extend(T*s){
32          for(;ae&&al>=ae->length();){
33              s+=ae->length();
34              al-=ae->length();
35              an=ae->t;
36              ae=al?an->c[*s-D]:0;
37          }
38      }
39      bool extend(int c){
40          if(ae){
41              if(*(ae->l+al)-D-c)
42                  return true;
43              ++al;
44          }else{
45              if(!an->c[c])
46                  return true;
47              ae=an->c[c];
48              al=1;
49              if(pr)
50                  pr->s=an;
51          }
52          extend(ae->l);
53          return false;
54      }
55      void insert(T*s,int n){
56          ct+=n;
57          an=root;
58          ae=0;
59          al=0;
60          for(T*p=s;p!=s+n;++p)
61              for(pr=0;extend(*p-D);){
62                  edge*x=newedge(p,s+n,np++);
63                  if(!ae)
64                      an->c[*p-D]=x;
65                  else{
66                      edge*&y=an->c[*ae->l-D];
67                      y=newedge(ae->l,ae->l+al,np++);
68                      y->t->c[*(ae->l+=al)-D]=ae;
69                      y->t->c[*p-D]=x;
```

```
70                      ae=y;
71                  }
72                  if(pr)
73                      pr->s=ae?ae->t:an;
74                  pr=ae?ae->t:an;
75                  int r=1;
76                  if(an==root&&!al)
77                      break;
78                  if(an==root)
79                      --al;
80                  else{
81                      an=an->s?an->s:root;
82                      r=0;
83                  }
84                  if(al){
85                      T*t=ae->l+(an==root)*r;
86                      ae=an->c[*t-D];
87                      extend(t);
88                  }else
89                      ae=0;
90              }
91          }
92      void build(node*u=0,int d=0){
93          if(!u)
94              u=root;
95          int t=0,s=0;
96          for(int i=0;i<M;++i)
97              if(u->c[i]){
98                  if(!t)
99                      t=1;
100                 else if(!s){
101                     s=1;
102                     *sp++=d;
103                 }
104                 build(u->c[i]->t,d+u->c[i]->length());
105             }
106         if(s)
107             --sp;
108         else if(!t&&sp!=sk){
109             *hp++=*(sp-1);
110             *fp++=ct-d+1;
```

```
111  |          }
112  |      }
113  |      edge*ae;
114  |      node*root,*an,*pr;
115  |      int al,ct,sk[N],*sp=sk,height[N],*hp=height,suffix[N],*fp=suffix;
116  |  };
```

CHAPTER 7

Utility Tools

7.1   Checker

Checker.bat (166 bytes, 7 lines)

```
1  :again
2  generator > input.txt
3  program1 < input.txt > output1.txt
4  program2 < input.txt > output2.txt
5  fc output1.txt output2.txt
6  if errorlevel 1 pause
7  goto again
```

7.2   Date

Date.hpp (3596 bytes, 145 lines)

```
1   #include<bits/stdc++.h>
2   using namespace std;
3   struct Date{
4       int y,m,d,w;
5       Date&operator++(){
6           return*this=*this+1;
7       }
8       bool leap(int a)const{
9           return a%400==0||(a%4==0&&a%100!=0);
10      }
11      int month_sum(int a,int b)const{
12          if(b==0)
13              return 0;
14          if(b==1)
15              return 31;
16          return 59+leap(a)+30*(b−2)+(b+1)/2−1+(b>=8&&b%2==0);
17      }
18      string month_name(int a)const{
19          if(a==1)
20              return"January";
21          if(a==2)
22              return"February";
23          if(a==3)
```

```
24              return"March";
25          if(a==4)
26              return"April";
27          if(a==5)
28              return"May";
29          if(a==6)
30              return"June";
31          if(a==7)
32              return"July";
33          if(a==8)
34              return"August";
35          if(a==9)
36              return"September";
37          if(a==10)
38              return"October";
39          if(a==11)
40              return"November";
41          if(a==12)
42              return"December";
43      }
44      string day_name(int a)const{
45          if(a==0)
46              return"Sunday";
47          if(a==1)
48              return"Monday";
49          if(a==2)
50              return"Tuesday";
51          if(a==3)
52              return"Wednesday";
53          if(a==4)
54              return"Thursday";
55          if(a==5)
56              return"Friday";
57          if(a==6)
58              return"Saturday";
59      }
60      operator int()const{
61          int t=(y-1)*365+(y-1)/4-(y-1)/100+(y-1)/400+month_sum(y,m-1)+d;
62          if(y==1752&&m>=9&&d>2||y>1752)
63              t-=11;
64          t-=min(y-1,1700)/400-min(y-1,1700)/100;
```

```
65          if(y<=1700&&y%400!=0&&y%100==0&&m>2)
66              ++t;
67          return t;
68      }
69      Date(int _y,int _m,int _d):
70          y(_y),m(_m),d(_d),w((int(*this)+5)%7){
71      }
72      Date(int a){
73          int yl=0,yr=1e7;
74          while(yl+1<yr){
75              int ym=(yl+yr)/2;
76              if(int(Date(ym,12,31))<a)
77                  yl=ym;
78              else
79                  yr=ym;
80          }
81          y=yr;
82          int ml=0,mr=12;
83          while(ml+1<mr){
84              int mm=(ml+mr)/2,mt;
85              if(mm==2){
86                  if(y<=1700)
87                      mt=28+(y%4==0);
88                  else
89                      mt=28+(y%4==0&&y%100!=0||y%400==0);
90              }else if(mm<=7)
91                  mt=30+mm%2;
92              else
93                  mt=31-mm%2;
94              if(int(Date(y,mm,mt))<a)
95                  ml=mm;
96              else
97                  mr=mm;
98          }
99          m=mr;
100         for(int i=1;;++i){
101             if(y==1752&&m==9&&i>2&&i<14)
102                 continue;
103             if(int(Date(y,m,i))==a){
104                 d=i;
105                 break;
```

```
106                }
107            }
108            w=(5+a)%7;
109        }
110        operator string()const{
111            stringstream s;
112            string t;
113            s<<day_name(w)+", "+month_name(m)+" "<<d<<", "<<y;
114            getline(s,t);
115            return t;
116        }
117 };
118 ostream&operator<<(ostream&s,const Date&a){
119     return s<<string(a);
120 }
121 int operator-(const Date&a,const Date&b){
122     return int(a)-int(b);
123 }
124 Date operator+(const Date&a,int b){
125     return Date(int(a)+b);
126 }
127 Date operator-(const Date&a,int b){
128     return Date(int(a)-b);
129 }
130 bool operator<(const Date&a,const Date&b){
131     if(a.y==b.y&&a.m==b.m)
132         return a.d<b.d;
133     if(a.y==b.y)
134         return a.m<b.m;
135     return a.y<b.y;
136 }
137 bool operator>(const Date&a,const Date&b){
138     return b<a;
139 }
140 bool operator!=(const Date&a,const Date&b){
141     return a.y!=b.y||a.m!=b.m||a.d!=b.d;
142 }
143 bool operator==(const Date&a,const Date&b){
144     return !(a!=b);
145 }
```

7.3   Fast Reader

Fast Reader.hpp (1251 bytes, 61 lines)

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  struct FastReader{
4      FILE*f;
5      char*p,*e;
6      vector<char>v;
7      void ipt(){
8          for(int i=1,t;;i<<=1){
9              v.resize(v.size()+i);
10             if(i!=(t=fread(&v[0]+v.size()−i,1,i,f))){
11                 p=&v[0],e=p+v.size()−i+t;
12                 break;
13             }
14         }
15     }
16     void ign(){
17         while(p!=e&&isspace(*p))
18             ++p;
19     }
20     int isc(){
21         return p!=e&&!isspace(*p);
22     }
23     int isd(){
24         return p!=e&&isdigit(*p);
25     }
26     FastReader(FILE*_f):
27         f(_f){
28         ipt();
29     }
30     FastReader(string _f):
31         f(fopen(_f.c_str(),"r")){
32         ipt();
33     }
34     ~FastReader(){
35         fclose(f);
36     }
37     template<class T>FastReader&operator>>(T&a){
```

```
38          int n=1;
39          ign();
40          if(*p=='-')
41              n=-1,++p;
42          for(a=0;isd();)
43              a=a*10+*p++-'0';
44          a*=n;
45          return*this;
46      }
47      FastReader&operator>>(char&a){
48          ign();
49          a=*p++;
50          return*this;
51      }
52      FastReader&operator>>(char*a){
53          for(ign();isc();)
54              *a++=*p++;
55          *a=0;
56          return*this;
57      }
58      char get(){
59          return*p++;
60      }
61  };
```

7.4   Fast Writer

Fast Writer.hpp (866 bytes, 39 lines)

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  struct FastWriter{
4      FILE*f;
5      vector<char>p;
6      FastWriter(FILE*_f):
7          f(_f){
8      }
9      FastWriter(string _f):
10         f(fopen(_f.c_str(),"w")){
```

```
11          }
12      ~FastWriter(){
13          if(p.size())
14              fwrite(&p[0],1,p.size(),f);
15          fclose(f);
16      }
17      FastWriter&operator<<(char a){
18          p.push_back(a);
19          return*this;
20      }
21      FastWriter&operator<<(const char*a){
22          while(*a)
23              p.push_back(*a++);
24          return*this;
25      }
26      template<class T>FastWriter&operator<<(T a){
27          if(a<0)
28              p.push_back('−'),a=−a;
29          static char t[19];
30          char*q=t;
31          do{
32              T b=a/10;
33              *q++=a−b*10+'0',a=b;
34          }while(a);
35          while(q>t)
36              p.push_back(*−−q);
37          return*this;
38      }
39  };
```

7.5   Number Speller

Number Speller.hpp (2143 bytes, 72 lines)

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  namespace NumberSpeller{
4      template<class T>string run(T a){
5          map<T,string>m;
```

```
6          m[0]="zero";
7          m[1]="one";
8          m[2]="two";
9          m[3]="three";
10         m[4]="four";
11         m[5]="five";
12         m[6]="six";
13         m[7]="seven";
14         m[8]="eight";
15         m[9]="nine";
16         m[10]="ten";
17         m[11]="eleven";
18         m[12]="twelve";
19         m[13]="thirteen";
20         m[14]="fourteen";
21         m[15]="fifteen";
22         m[16]="sixteen";
23         m[17]="seventeen";
24         m[18]="eighteen";
25         m[19]="nineteen";
26         m[20]="twenty";
27         m[30]="thirty";
28         m[40]="forty";
29         m[50]="fifty";
30         m[60]="sixty";
31         m[70]="seventy";
32         m[80]="eighty";
33         m[90]="ninety";
34         if(a<0)
35             return"minus "+run(-a);
36         if(m.count(a))
37             return m[a];
38         if(a<100)
39             return run(a/10*10)+"-"+run(a%10);
40         if(a<1000&&a%100==0)
41             return run(a/100)+" hundred";
42         if(a<1000)
43             return run(a/100*100)+" and "+run(a%100);
44         vector<string>t;
45         t.push_back("thousand");
46         t.push_back("million");
```

```
47          t.push_back("billion");
48          t.push_back("trillion");
49          t.push_back("quadrillion");
50          t.push_back("quintillion");
51          t.push_back("sextillion");
52          t.push_back("septillion");
53          t.push_back("octillion");
54          t.push_back("nonillion");
55          t.push_back("decillion");
56          t.push_back("undecillion");
57          t.push_back("duodecillion");
58          t.push_back("tredecillion");
59          t.push_back("quattuordecillion");
60          t.push_back("quindecillion");
61          string r=a%1000?run(a%1000):"";
62          a/=1000;
63          for(int i=0;a;++i,a/=1000)
64              if(a%1000){
65                  if(!i&&r.find("and")==string::npos&&r.find("hundred")==
        string::npos&&r.size())
66                      r=run(a%1000)+" "+t[i]+" and "+r;
67                  else
68                      r=run(a%1000)+" "+t[i]+(r.size()?", ":"")+r;
69              }
70          return r;
71      }
72  }
```

7.6   Utility

Utility.hpp (4146 bytes, 167 lines)

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  typedef long long ll;
4  typedef long double ld;
5  #define lp(i,a,b)for(auto i=a;i<=b;++i)
6  void set_read(string a){
7      freopen(a.c_str(),"r",stdin);
```

```
 8 │ }
 9 │ void set_write(string a){
10 │     freopen(a.c_str(),"w",stdout);
11 │ }
12 │ template<class T>void write(T a){
13 │     cout<<a;
14 │ }
15 │ template<class T>void writeln(T a){
16 │     cout<<a<<endl;
17 │ }
18 │ template<class T>T read(){
19 │     T a;
20 │     cin>>a;
21 │     return a;
22 │ }
23 │ void sync(int t){
24 │     ios::sync_with_stdio(t);
25 │ }
26 │ template<class T1,class T2>T2 convert(T1 a){
27 │     stringstream s;
28 │     s<<a;
29 │     T2 t;
30 │     s>>t;
31 │     return t;
32 │ }
33 │
34 │ template<class T1,class T2>ostream&operator<<(ostream&s,const pair<T1,T2>&a
   │     ){
35 │     return s<<a.first<<" "<<a.second;
36 │ }
37 │ template<class T1,class T2,class T3>struct triple{
38 │     triple(T1 a,T2 b,T3 c):
39 │         first(a),second(b),third(c){
40 │     }
41 │     T1 first;
42 │     T2 second;
43 │     T3 third;
44 │ };
45 │ template<class T1,class T2,class T3>ostream&operator<<(ostream&s,const
   │     triple<T1,T2,T3>&a){
46 │     return s<<a.first<<" "<<a.second<<" "<<a.third;
```

```
47  }
48  template<class T1,class T2,class T3>triple<T1,T2,T3>make_triple(T1 a,T2 b,
        T3 c){
49      return triple<T1,T2,T3>(a,b,c);
50  }
51  template<class T>T sum(triple<T,T,T>a){
52      return a.first+a.second+a.third;
53  }
54  template<class T>T product(triple<T,T,T>a){
55      return a.first*a.second*a.third;
56  }
57  template<class T>vector<T>sort(vector<T>a){
58      sort(a.begin(),a.end());
59      return a;
60  }
61  template<class T,class F>vector<T>foreach(vector<T>a,F f){
62      for(int i=0;i<a.size();++i)
63          f(a[i]);
64      return a;
65  }
66  template<class T>T sum(const vector<T>&a){
67      T r=0;
68      for(int i=0;i<a.size();++i)
69          r+=a[i];
70      return r;
71  }
72  template<class T>T sum(const set<T>&a){
73      T r=0;
74      for(typename set<T>::iterator i=a.begin();i!=a.end();++i)
75          r+=*i;
76      return r;
77  }
78  template<class T>ostream&operator<<(ostream&s,const vector<T>&a){
79      for(int i=0;i<a.size();++i){
80          if(i)
81              cout<<' ';
82          cout<<a[i];
83      }
84      return s;
85  }
86  template<class T>vector<T>unique(vector<T>a){
```

```
87       sort(a.begin(),a.end());
88       a.erase(unique(a.begin(),a.end()),a.end());
89       return a;
90  }
91  template<class T>vector<T>combination(T a){
92       vector<T>r;
93       for(int i=0;i<(1<<a.size());++i){
94           T t;
95           for(int j=0;j<a.size();++j)
96               if((i>>j)&1)
97                   t.push_back(a[j]);
98           r.push_back(t);
99       }
100      return r;
101 }
102 template<class T>vector<T>permutation(T a){
103      sort(a.begin(),a.end());
104      vector<T>r;
105      for(int i=0;i<(1<<a.size());++i){
106          T t;
107          for(int j=0;j<a.size();++j)
108              if((i>>j)&1)
109                  t.push_back(a[j]);
110          do{
111              r.push_back(t);
112          }while(next_permutation(t.begin(),t.end()));
113      }
114      return r;
115 }
116 template<class T>vector<T>permutation(T a,int b){
117      sort(a.begin(),a.end());
118      vector<T>r;
119      for(int i=0;i<(1<<a.size());++i){
120          T t;
121          for(int j=0;j<a.size();++j)
122              if((i>>j)&1)
123                  t.push_back(a[j]);
124          if(t.size()!=b)
125              continue;
126          do{
127              r.push_back(t);
```

```
128         }while(next_permutation(t.begin(),t.end())));
129     }
130     return r;
131 }
132 template<class T>set<T>operator+(set<T>a,set<T>b){
133     for(typename set<T>::iterator i=a.begin();i!=a.end();++i)
134         b.insert(*i);
135     return b;
136 }
137 template<class T>vector<T>operator+(vector<T>a,vector<T>b){
138     for(int i=0;i<b.size();++i)
139         a.push_back(b[i]);
140     return a;
141 }
142 template<class T>set<T>operator&(set<T>a,set<T>b){
143     set<T>c;
144     for(typename set<T>::iterator i=a.begin();i!=a.end();++i)
145         if(b.count(*i))
146             c.insert(*i);
147     return c;
148 }
149 string str(int a){
150     return convert<int,string>(a);
151 }
152 string str(ll a){
153     return convert<ll,string>(a);
154 }
155 int to_int(string a){
156     return convert<string,int>(a);
157 }
158 ll to_ll(string a){
159     return convert<string,ll>(a);
160 }
161 template<class T,class F>vector<T>delete_if(vector<T>a,F b){
162     vector<T>c;
163     for(int i=0;i<a.size();++i)
164         if(!b(a[i]))
165             c.push_back(a[i]);
166     return c;
167 }
```