

# 1 Computational Geometry

## 1.1 Geometry

```

1 const double PI=atan2(0.0, -1.0);
2 template<typename T>
3 struct point{
4     T x,y;
5     point(){}
6     point(const T&x, const T&y):x(x),y(y){}
7     point operator+(const point &b) const{
8         return point(x+b.x,y+b.y); }
9     point operator-(const point &b) const{
10        return point(x-b.x,y-b.y); }
11     point operator*(const T &b) const{
12        return point(x*b,y*b); }
13     point operator/(const T &b) const{
14        return point(x/b,y/b); }
15     bool operator==(const point &b) const{
16        return x==b.x&&y==b.y; }
17     T dot(const point &b) const{
18        return x*b.x+y*b.y; }
19     T cross(const point &b) const{
20        return x*b.y-y*b.x; }
21     point normal() const{//求法向量
22        return point(-y,x); }
23     T abs2() const{//向量長度的平方
24        return dot(*this); }
25     T rad(const point &b) const{//兩向量的弧度
26     return fabs(atan2(fabs(cross(b)),dot(b))); }
27     T getA() const{//對x軸的弧度
28        T A=atan2(y,x); //超過180度會變負的
29        if(A<=-PI/2)A+=PI*2;
30        return A;
31     };
32 };
33 template<typename T>
34 struct line{
35     line(){}
36     point<T> p1,p2;
37     T a,b,c; //ax+by+c=0
38     line(const point<T>&x, const point<T>&y):p1(x),p2(y){}
39     void pton() const{//轉成一般式
40        a=p1.y-p2.y;
41        b=p2.x-p1.x;
42        c=-a*p1.x-b*p1.y;
43     }
44     T ori(const point<T> &p) const{//點和有向直
45        線的關係 · >0左邊 · =0在線上 <0右邊
46        return (p2-p1).cross(p-p1);
47     }
48     T btw(const point<T> &p) const{//點投影落在
49        線段上 <=0
50        return (p1-p).dot(p2-p);
51     }
52     bool point_on_segment(const point<T>&p)
53         const{//點是否在線段上
54         return ori(p)==0&&btw(p)<=0;
55     }
56     T dis2(const point<T> &p, bool is_segment
57         =0) const{//點跟直線/線段的距離平方
58         point<T> v=p2-p1,v1=v-p1;
59         if(is_segment){
60             point<T> v2=p-p1;
61             if(v.dot(v1)<=0) return v1.abs2();
62             if(v.dot(v2)>=0) return v2.abs2();
63         }
64         T tmp=v.cross(v1);
65         return tmp*tmp/v.abs2();
66     }
67     T seg_dis2(const line<T> &l) const{//兩線段
68        距離平方
69        return min({dis2(l.p1,1),dis2(l.p2,1),l.
70            dis2(p1,1),l.dis2(p2,1)});
71     }
72     point<T> projection(const point<T> &p)
73         const{//點對直線的投影
74        point<T> n=(p2-p1).normal();
75        return p-n*(p-p1).dot(n)/n.abs2();
76     }
77     point<T> mirror(const point<T> &p) const{
78        //點對直線的鏡射 · 要先呼叫pton轉成一般式
79        point<T> R;
80        T d=a*b+b*b;
81        R.x=(b*b*p.x-a*a*p.x-2*a*b*p.y-2*a*c)/d;
82        R.y=(a*a*p.y-b*b*p.y-2*a*b*p.x-2*b*c)/d;
83        return R;
84     }
85     bool equal(const line &l) const{//直線相等
86        return ori(l.p1)==0&&ori(l.p2)==0;
87     }
88     bool parallel(const line &l) const{
89        return (p1-p2).cross(l.p1-l.p2)==0;
90     }
91     bool cross_seg(const line &l) const{
92        return (p2-p1).cross(l.p1-p1)*(p2-p1).
93            cross(l.p2-p1)<=0; //直線是否交線段
94     }
95     int line_intersect(const line &l) const{//
96        直線相交情況 · -1無限多點 · 1交於一點 · 0
97        不相交
98        return parallel(l)?(ori(l.p1)==0?-1:0)
99            :1;
100     }
101     int seg_intersect(const line &l) const{
102        T c1=ori(l.p1), c2=ori(l.p2);
103        T c3=l.ori(p1), c4=l.ori(p2);
104        if(c1==0&&c2==0){ //共線
105            bool b1=btw(l.p1)>=0,b2=btw(l.p2)>=0;
106            T a3=1.btw(p1),a4=1.btw(p2);
107            if(b1&&b2&&a3==0&&a4==0) return 2;
108            if(b1&&b2&&a3>=0&&a4==0) return 3;
109            if(b1&&b2&&a3>=0&&a4>=0) return 0;
110            return -1; //無限交點
111        }else if(c1*c2<=0&&c3*c4<=0) return 1;
112        return 0; //不相交
113     }
114     point<T> line_intersection(const line &l)
115         const{//直線交點*/
116        point<T> a=p2-p1,b=l.p2-l.p1,s=l.p1-p1;
117        //if(a.cross(b)==0) return INF;
118        return p1+a*(s.cross(b)/a.cross(b));
119     }
120     point<T> seg_intersection(const line &l)
121         const{//線段交點
122         int res=seg_intersect(l);
123         if(res<=0) assert(0);
124         if(res==2) return p1;
125         if(res==3) return p2;
126         return line_intersection(l);
127     }
128 };
129 template<typename T>
130 struct polygon{
131     polygon(){}
132     vector<point<T> > p; //逆時針順序
133     T area() const{//面積
134        T ans=0;
135        for(int i=p.size()-1,j=0;j<(int)p.size()
136            ;i=j++){
137            ans+=p[i].cross(p[j]);
138        }
139        return ans/2;
140     }
141     point<T> center_of_mass() const{//重心
142        T cx=0,cy=0,w=0;
143        for(int i=p.size()-1,j=0;j<(int)p.size()
144            ;i=j++){
145            T a=p[i].cross(p[j]);
146            cx+=(p[i].x+p[j].x)*a;
147            cy+=(p[i].y+p[j].y)*a;
148            w+=a;
149        }
150        return point<T>(cx/3/w,cy/3/w);
151     }
152     char ahas(const point<T>& t) const{//點是否
153        在簡單多邊形內 · 是的話回傳1 · 在邊上回
154        傳-1 · 否則回傳0
155        bool c=0;
156        for(int i=0,j=p.size()-1;i<p.size();j=i
157            ++){
158            if(line<T>(p[i],p[j]).point_on_segment
159                (t)) return -1;
160            else if((p[i].y>t.y)!=p[j].y>t.y)&&
161                t.x<(p[j].x-p[i].x)*(t.y-p[i].y)/(p[j
162                    ].y-p[i].y)+p[i].x)
163                c=!c;
164        }
165        return c;
166     }
167     char point_in_convex(const point<T>&x)
168         const{
169        int l=1,r=(int)p.size()-2;
170        while(l<r){ //點是否在凸多邊形內 · 是的話
171            回傳1 · 在邊上回傳-1 · 否則回傳0
172            int mid=(l+r)/2;
173            T a1=(p[mid]-p[0]).cross(x-p[0]);
174            T a2=(p[mid+1]-p[0]).cross(x-p[0]);
175            if(a1>=0&&a2<=0){
176                T res=(p[mid+1]-p[mid]).cross(x-p[
177                    mid]);
178                return res>0?-1:(res==0?-1:0);
179            }else if(a1<0) r=mid-1;
180            else l=mid+1;
181        }
182        return 0;
183     }
184     vector<T> getA() const{//凸包邊對x軸的夾角
185     vector<T> res; //一定是遞增的
186     for(size_t i=0;i<p.size();i++){
187         res.push_back((p[(i+1)%p.size()]-p[i])
188             .getA());
189     }
190     return res;
191 }
192 bool line_intersect(const vector<T>&A,
193     const line<T> &l) const{//0(LogN)
194     int f1=upper_bound(A.begin(),A.end(),(l.
195         p1-l.p2).getA())-A.begin();
196     int f2=upper_bound(A.begin(),A.end(),(l.
197         p2-l.p1).getA())-A.begin();
198     return l.cross_seg(line<T>(p[f1],p[f2]))
199         ;
200 }
201 polygon cut(const line<T> &l) const{//凸包
202     對直線切割 · 得到直線L左側的凸包
203     polygon ans;
204     for(int n=p.size(),i=n-1,j=0;j<n;i=j++){
205         if(l.ori(p[i])>=0){
206             ans.p.push_back(p[i]);
207             if(l.ori(p[j])<0)
208                 ans.p.push_back(l.
209                     line_intersection(line<T>(p[i
210                         ],p[j])));
211         }else if(l.ori(p[j])>0)
212             ans.p.push_back(l.line_intersection(
213                 line<T>(p[i],p[j])));
214         }
215     }
216     return ans;
217 }
218 static bool monotone_chain_cmp(const point
219     <T>& a, const point<T>& b){ //凸包排序函
220     數
221     return (a.x<b.x)|| (a.x==b.x&&a.y<b.y);
222 }
223 void monotone_chain(vector<point<T> > &s){
224     //凸包
225     sort(s.begin(),s.end(),
226         monotone_chain_cmp);
227     p.resize(s.size()+1);
228     int m=0;
229     for(size_t i=0;i<s.size();i++){
230         while(m>=2&&(p[m-1]-p[m-2]).cross(s[i
231             ]-p[m-2])<=0)--m;
232         p[m++]=s[i];
233     }
234     for(int i=s.size()-2,t=m+1;i>=0;--i){
235         while(m>=t&&(p[m-1]-p[m-2]).cross(s[i
236             ]-p[m-2])<=0)--m;
237         p[m++]=s[i];
238     }
239     if(s.size()>1)--m;
240     p.resize(m);
241 }
242 T diam() const{//直徑
243     int n=p.size(),t=1;
244     T ans=0;p.push_back(p[0]);
245     for(int i=0;i<n;i++){
246         point<T> now=p[i+1]-p[i];
247         while(now.cross(p[t+1]-p[i])>now.cross
248             (p[t]-p[i])) t=(t+1)%n;
249         ans=max(ans,(p[i]-p[t]).abs2());
250     }
251     return p.pop_back(),ans;
252 }
253 T min_cover_rectangle() const{//最小覆蓋矩形

```

```

211 int n=p.size(),t=1,r=1,l;
212 if(n<3)return 0;//也可以做最小周長矩形
213 T ans=1e99;p.push_back(p[0]);
214 for(int i=0;i<n;i++){
215     point<T> now=p[i+1]-p[i];
216     while(now.cross(p[t+1]-p[i])>now.cross
217           (p[t]-p[i]))t=(t+1)%n;
218     while(now.dot(p[r+1]-p[i])>now.dot(p[r
219           ]-p[i]))r=(r+1)%n;
220     if(!i)l=r;
221     while(now.dot(p[l+1]-p[i])<=now.dot(p[
222           l]-p[i]))l=(l+1)%n;
223     T d=now.abs2();
224     T tmp=now.cross(p[t]-p[i])*(now.dot(p[
225           r]-p[i])-now.dot(p[l]-p[i]))/d;
226     ans=min(ans,tmp);
227 }
228 return p.pop_back(),ans;
229
230 T dis2(polygon &p1){//凸包最近距離平方
231     vector<point<T>> > &P=p,&Q=p1.p;
232     int n=P.size(),m=Q.size(),l=0,r=0;
233     for(int i=0;i<n;++i)if(P[i].y<P[l].y)l=i;
234     for(int i=0;i<m;++i)if(Q[i].y<Q[r].y)r=i;
235     P.push_back(P[0]),Q.push_back(Q[0]);
236     T ans=1e99;
237     for(int i=0;i<n;++i){
238         while((P[l+1]-P[l]).cross(Q[r+1]-Q[r])
239               <0)r=(r+1)%m;
240         ans=min(ans,line<T>(P[l],P[l+1]).
241               seg_dis2(line<T>(Q[r],Q[r+1])));
242         l=(l+1)%n;
243     }
244     return P.pop_back(),Q.pop_back(),ans;
245 }
246
247 static char sign(const point<T>&t){
248     return (t.y==0?t.x:t.y)<0;
249 }
250
251 static bool angle_cmp(const line<T>& A,
252                       const line<T>& B){
253     point<T> a=A.p2-A.p1,b=B.p2-B.p1;
254     return sign(a)<sign(b)||((sign(a)==sign(b)
255                               )&&a.cross(b)>0);
256 }
257
258 int halfplane_intersection(vector<line<T>
259                             > &s){//半平面交
260     sort(s.begin(),s.end(),angle_cmp);//線段
261     //左側為該線段半平面
262     int L,R,n=s.size();
263     vector<point<T>> > px(n);
264     vector<line<T>> > q(n);
265     q[L=R=0]=s[0];
266     for(int i=1;i<n;++i){
267         while(L<R&&s[i].ori(px[R-1])<=0)--R;
268         while(L<R&&s[i].ori(px[L])<=0)+L;
269         q[++R]=s[i];
270         if(q[R].parallel(q[R-1])){
271             --R;
272             if(q[R].ori(s[i].p1)>0)q[R]=s[i];
273         }
274         if(L<R)px[R-1]=q[R-1].
275             line_intersection(q[R]);
276     }
277     while(L<R&&q[L].ori(px[R-1])<=0)--R;
278     p.clear();
279
280     if(R-L<=1)return 0;
281     px[R]=q[R].line_intersection(q[L]);
282     for(int i=L;i<R;++i)p.push_back(px[i]);
283     return R-L+1;
284 }
285
286 template<typename T>
287 struct triangle{
288     point<T> a,b,c;
289     triangle(){
290         triangle(const point<T> &a,const point<T>
291                 &b,const point<T> &c):a(a),b(b),c(c){}
292     }
293     T area()const{
294         T t=(b-a).cross(c-a)/2;
295         return t>0?t:-t;
296     }
297     point<T> barycenter()const{//重心
298         return (a+b+c)/3;
299     }
300     point<T> circumcenter()const{//外心
301         static line<T> u,v;
302         u.p1=(a+b)/2;
303         u.p2=point<T>(u.p1.x-a.y+b.y,u.p1.y+a.x-
304                       b.x);
305         v.p1=(a+c)/2;
306         v.p2=point<T>(v.p1.x-a.y+c.y,v.p1.y+a.x-
307                       c.x);
308         return u.line_intersection(v);
309     }
310     point<T> incenter()const{//內心
311         T A=sqrt((b-c).abs2()),B=sqrt((a-c).abs2
312               ()),C=sqrt((a-b).abs2());
313         return point<T>(A*a.x+B*b.x+C*c.x,A*a.y+
314                       B*b.y+C*c.y)/(A+B+C);
315     }
316     point<T> perpercenter()const{//垂心
317         return barycenter()*3-circumcenter()*2;
318     }
319 }
320
321 template<typename T>
322 struct point3D{
323     T x,y,z;
324     point3D(){
325         point3D(const T&x,const T&y,const T&z):x(x
326               ),y(y),z(z){}
327     }
328     point3D operator+(const point3D &b)const{
329         return point3D(x+b.x,y+b.y,z+b.z);
330     }
331     point3D operator-(const point3D &b)const{
332         return point3D(x-b.x,y-b.y,z-b.z);
333     }
334     point3D operator*(const T &b)const{
335         return point3D(x*b,y*b,z*b);
336     }
337     point3D operator/(const T &b)const{
338         return point3D(x/b,y/b,z/b);
339     }
340     bool operator==(const point3D &b)const{
341         return x==b.x&&y==b.y&&z==b.z;
342     }
343     T dot(const point3D &b)const{
344         return x*b.x+y*b.y+z*b.z;
345     }
346     point3D cross(const point3D &b)const{
347         return point3D(y*b.z-z*b.y,z*b.x-x*b.z,x
348               *b.y-y*b.x);
349     }
350     T abs2()const{//向量長度的平方
351         return dot(*this);
352     }
353     T area2(const point3D &b)const{//和b、原點
354         //圍成面積的平方
355         return cross(b).abs2()/4;
356     }
357 };
358
359 template<typename T>
360 struct line3D{
361     point3D<T> p1,p2;
362     line3D(){
363         line3D(const point3D<T> &p1,const point3D<
364               T> &p2):p1(p1),p2(p2){}
365     }
366     T dis2(const point3D<T> &p,bool is_segment
367           =0)const{//點跟直線/線段的距離平方
368         point3D<T> v=p2-p1,v1=p-p1;
369         if(is_segment){
370             point3D<T> v2=p-p2;
371             if(v.dot(v1)<=0)return v1.abs2();
372             if(v.dot(v2)>=0)return v2.abs2();
373         }
374         point3D<T> tmp=v.cross(v1);
375         return tmp.abs2()/v.abs2();
376     }
377     pair<point3D<T>,point3D<T>> closest_pair(
378           const line3D<T> &l1)const{
379         point3D<T> v1=(p1-p2),v2=(l1.p1-l.p2);
380         point3D<T> N=v1.cross(v2),ab(p1-l.p1);
381         //if(N.abs2()==0)return NULL;平行或重合
382         T tmp=N.dot(ab),ans=tmp*tmp/N.abs2();//
383         //最近點距離
384         point3D<T> d1=p2-p1,d2=l.p1-l.p1,D=d1.
385               cross(d2),G=l.p1-p1;
386         T t1=(G.cross(d2)).dot(D)/D.abs2();
387         T t2=(G.cross(d1)).dot(D)/D.abs2();
388         return make_pair(p1+d1*t1,l.p1+d2*t2);
389     }
390     bool same_side(const point3D<T> &a,const
391           point3D<T> &b)const{
392         return (p2-p1).cross(a-p1).dot((p2-p1).
393               cross(b-p1))>0;
394     }
395 };
396
397 template<typename T>
398 struct plane{
399     point3D<T> p0,n;//平面上的點和法向量
400     plane(){
401         plane(const point3D<T> &p0,const point3D<T>
402               &n):p0(p0),n(n){}
403     }
404     T dis2(const point3D<T> &p)const{//點到平
405           面距離的平方
406         T tmp=(p-p0).dot(n);
407         return tmp*tmp/n.abs2();
408     }
409     point3D<T> projection(const point3D<T> &p)
410           const{
411         return p-n*(p-p0).dot(n)/n.abs2();
412     }
413     point3D<T> line_intersection(const line3D<
414           T> &l1)const{
415         T tmp=n.dot(l1.p2-l.p1);//等於0表示平行或
416               重合該平面
417         return l.p1+(l.p2-l.p1)*(n.dot(p0-l.p1)/
418               tmp);
419     }
420     line3D<T> plane_intersection(const plane &
421           p1)const{
422         point3D<T> e=n.cross(p1.n),v=n.cross(e);
423         T tmp=p1.n.dot(v);//等於0表示平行或重合
424               該平面
425     }
426     point3D<T> q=p0+(v*(p1.n.dot(p1.p0-p0))/
427           tmp);
428     return line3D<T>(q,q+e);
429 }
430
431 template<typename T>
432 struct triangle3D{
433     point3D<T> a,b,c;
434     triangle3D(){
435         triangle3D(const point3D<T> &a,const
436               point3D<T> &b,const point3D<T> &c):a(a
437               ),b(b),c(c){}
438     }
439     bool point_in(const point3D<T> &p)const{//
440           點在該平面上的投影在三角形中
441         return line3D<T>(b,c).same_side(p,a)&&
442               line3D<T>(a,c).same_side(p,b)&&
443               line3D<T>(a,b).same_side(p,c);
444     }
445 };
446
447 template<typename T>
448 struct tetrahedron{//四面體
449     point3D<T> a,b,c,d;
450     tetrahedron(){
451         tetrahedron(const point3D<T> &a,const
452               point3D<T> &b,const point3D<T> &c,
453               const point3D<T> &d):a(a),b(b),c(c),d(
454               d){}
455     }
456     T volume6()const{//體積的六倍
457         return (d-a).dot((b-a).cross(c-a));
458     }
459     point3D<T> centroid()const{
460         return (a+b+c+d)/4;
461     }
462     bool point_in(const point3D<T> &p)const{
463         return triangle3D<T>(a,b,c).point_in(p)
464               &&triangle3D<T>(c,d,a).point_in(p);
465     }
466 };
467
468 template<typename T>
469 struct convexhull3D{
470     static const int MAXN=1005;
471     struct face{
472         int a,b,c;
473         face(int a,int b,int c):a(a),b(b),c(c){}
474     };
475     vector<point3D<T>> pt;
476     vector<face> ans;
477     int fid[MAXN][MAXN];
478     void build(){
479         int n=pt.size();
480         ans.clear();
481         memset(fid,0,sizeof(fid));
482         ans.emplace_back(0,1,2);//注意不能共線
483         ans.emplace_back(2,1,0);
484         int ftop = 0;
485         for(int i=3, ftop=1; i<n; ++i,++ftop){
486             vector<face> next;
487             for(auto &f:ans){
488                 T d=(pt[i]-pt[f.a]).dot((pt[f.b]-pt[
489                       f.a]).cross(pt[f.c]-pt[f.a]));
490                 if(d<=0) next.push_back(f);
491                 int ff=0;
492                 if(d>0) ff=ftop;
493                 else if(d<0) ff=-ftop;
494             }
495         }
496     }
497 };

```

```

424     fid[f.a][f.b]=fid[f.b][f.c]=fid[f.c]
        ][f.a]=ff;
425 }
426 for(auto &f:ans){
427     if(fid[f.a][f.b]>0 && fid[f.a][f.b]
        !=fid[f.b][f.a])
428         next.emplace_back(f.a,f.b,i);
429     if(fid[f.b][f.c]>0 && fid[f.b][f.c]
        !=fid[f.c][f.b])
430         next.emplace_back(f.b,f.c,i);
431     if(fid[f.c][f.a]>0 && fid[f.c][f.a]
        !=fid[f.a][f.c])
432         next.emplace_back(f.c,f.a,i);
433 }
434 ans=next;
435 }
436 }
437 point3D<T> centroid()const{
438     point3D<T> res(0,0,0);
439     T vol=0;
440     for(auto &f:ans){
441         T tmp=pt[f.a].dot(pt[f.b].cross(pt[f.c]
            ));
442         res=res+(pt[f.a]+pt[f.b]+pt[f.c])*tmp;
443         vol+=tmp;
444     }
445     return res/(vol*4);
446 }
447 };

```

## 1.2 SmallestCircle

```

1 using PT=point<T>; using CPT=const PT;
2 PT circumcenter(CPT &a,CPT &b,CPT &c){
3     PT u=b-a, v=c-a;
4     T c1=u.abs2()/2, c2=v.abs2()/2;
5     T d=u.cross(v);
6     return PT(a.x+(v.y*c1-u.y*c2)/d, a.y+(u.x*
            c2-v.x*c1)/d);
7 }
8 void solve(PT p[],int n,PT &c,T &r2){
9     random_shuffle(p,p+n);
10    c=p[0]; r2=0; // c,r2 = 圓心,半徑平方
11    for(int i=1;i<n;i++)if((p[i]-c).abs2(>r2){
12        c=p[i]; r2=0;
13    }
14    for(int j=0;j<i;j++)if((p[j]-c).abs2(>r2){
15        c.x=(p[i].x+p[j].x)/2;
16        c.y=(p[i].y+p[j].y)/2;
17        r2=(p[j]-c).abs2();
18    }
19    for(int k=0;k<j;k++)if((p[k]-c).abs2(>r2){
20        c=circumcenter(p[i],p[j],p[k]);
21        r2=(p[i]-c).abs2();
22    }
23 }

```

## 1.3 最近點對

```

1 template<typename _IT=point<T>* >
2 T cloest_pair(_IT L, _IT R){
3     if(R-L <= 1) return INF;
4     _IT mid = L+(R-L)/2;
5     T x = mid->x;
6     T d = min(cloest_pair(L,mid),cloest_pair(
            mid,R));
7     inplace_merge(L, mid, R, ycmp);
8     static vector<point> b; b.clear();
9     for(auto u=L;u<R;++u){
10        if((u->x-x)*(u->x-x)>=d) continue;
11        for(auto v=b.rbegin();v!=b.rend();++v){
12            T dx=u->x-v->x, dy=u->y-v->y;
13            if(dy*dy>=d) break;
14            d=min(d,dx*dx+dy*dy);
15        }
16        b.push_back(*u);
17    }
18    return d;
19 }
20 T closest_pair(vector<point<T>> &v){
21     sort(v.begin(),v.end(),xcmp);
22     return closest_pair(v.begin(),v.end());
23 }

```

## 2 Data Structure

### 2.1 undo disjoint set

```

1 struct DisjointSet {
2     // save() is like recursive
3     // undo() is like return
4     int n, fa[MXN], sz[MXN];
5     vector<pair<int*,int>> h;
6     vector<int> sp;
7     void init(int tn) {
8         n=tn;
9         for (int i=0; i<n; i++) sz[fa[i]=i]=1;
10        sp.clear(); h.clear();
11    }
12    void assign(int *k, int v) {
13        h.PB({k, *k});
14        *k=v;
15    }
16    void save() { sp.PB(SZ(h)); }
17    void undo() {
18        assert(!sp.empty());
19        int last=sp.back(); sp.pop_back();
20        while (SZ(h)!=last) {
21            auto x=h.back(); h.pop_back();
22            *x.F=x.S;
23        }
24    }
25    int f(int x) {
26        while (fa[x]!=x) x=fa[x];
27        return x;
28    }
29    void uni(int x, int y) {
30        x=f(x); y=f(y);
31        if (x==y) return ;
32        if (sz[x]<sz[y]) swap(x, y);

```

## 2.2 DLX

```

33     assign(&sz[x], sz[x]+sz[y]);
34     assign(&fa[y], x);
35 }
36 }djs;

2.2 DLX

1 const int MAXN=4100, MAXM=1030, MAXND=16390;
2 struct DLX{
3     int n,m,sz,ansd; //高是n · 寬是m的稀疏矩陣
4     int S[MAXN],H[MAXN];
5     int row[MAXN],col[MAXN]; //每個節點代表的
        列與行
6     int L[MAXN],R[MAXN],U[MAXN],D[MAXN];
7     vector<int> ans,anst;
8     void init(int _n,int _m){
9         n=_n,m=_m;
10        for(int i=0;i<=m;++i){
11            U[i]=D[i]=i,L[i]=i-1,R[i]=i+1;
12            S[i]=0;
13        }
14        R[m]=0,L[0]=m;
15        sz=m,ansd=INT_MAX; //ansd存最優解的個數
16        for(int i=1;i<=n;++i)H[i]=-1;
17    }
18    void add(int r,int c){
19        ++S[col[++sz]=c];
20        row[sz]=r;
21        D[sz]=D[c],U[D[c]]=sz,U[sz]=c,D[c]=sz;
22        if(H[r]<0)H[r]=L[sz]=R[sz]=sz;
23        else R[sz]=R[H[r]],L[R[H[r]]]=sz,L[sz]=H
            [r],R[H[r]]=sz;
24    }
25    #define DFOR(i,A,s) for(int i=A[s];i!=s;i=
        A[i])
26    void remove(int c){ //刪除第c行和所有當前覆
        蓋到第c行的列
27        L[R[c]]=L[c],R[L[c]]=R[c]; //這裡刪除第c
            行 · 若有些行不需要處理可以在開始時呼
            叫他
28        DFOR(i,D,c)DFOR(j,R,i){U[D[j]]=U[j],D[U[
            j]]=D[j],--S[col[j]]};
29    }
30    void restore(int c){ //恢復第c行和所有當前
        覆蓋到第c行的列 · remove的逆操作
31        DFOR(i,U,c)DFOR(j,L,i){++S[col[j]],U[D[j]
            ]=j,D[U[j]]=j};
32        L[R[c]]=c,R[L[c]]=c;
33    }
34    void remove2(int nd){ //刪除nd所在的行當前
        所有點(包括虛擬節點) · 只保留nd
35        DFOR(i,D,nd)L[R[i]]=L[i],R[L[i]]=R[i];
36    }
37    void restore2(int nd){ //刪除nd所在的行當前
        所有點 · 為remove2的逆操作
38        DFOR(i,U,nd)L[R[i]]=R[L[i]]=i;
39    }
40    bool vis[MAXN];
41    int h(){ //估價函數 for IDA*
42        int res=0;

```

```

43        memset(vis,0,sizeof(vis));
44        DFOR(i,R,0)if(!vis[i]){
45            vis[i]=1;
46            ++res;
47            DFOR(j,D,i)DFOR(k,R,j)vis[col[k]]=1;
48        }
49        return res;
50    }
51    bool dfs(int d){ //for精確覆蓋問題
52        if(d+h(>=)ansd)return 0; //找最佳解用 · 找
            任意解可以刪掉
53        if(!R[0]){ansd=d;return 1;}
54        int c=R[0];
55        DFOR(i,R,0)if(S[i]<S[c])c=i;
56        remove(c);
57        DFOR(i,D,c){
58            ans.push_back(row[i]);
59            DFOR(j,R,i)remove(col[j]);
60            if(dfs(d+1))return 1;
61            ans.pop_back();
62            DFOR(j,L,i)restore(col[j]);
63        }
64        restore(c);
65        return 0;
66    }
67    void dfs2(int d){ //for最小重複覆蓋問題
68        if(d+h(>=)ansd)return;
69        if(!R[0]){ansd=d;ans=anst;return;}
70        int c=R[0];
71        DFOR(i,R,0)if(S[i]<S[c])c=i;
72        DFOR(i,D,c){
73            anst.push_back(row[i]);
74            remove2(i);
75            DFOR(j,R,i)remove2(j),--S[col[j]];
76            dfs2(d+1);
77            anst.pop_back();
78            DFOR(j,L,i)restore2(j),++S[col[j]];
79            restore2(i);
80        }
81    }
82    bool exact_cover(){ //解精確覆蓋問題
83        return ans.clear(), dfs(0);
84    }
85    void min_cover(){ //解最小重複覆蓋問題
86        anst.clear(); //暫存用 · 答案還是存在ans裡
87        dfs2(0);
88    }
89    #undef DFOR
90 };

3 Flow

3.1 Gomory Hu

1 //最小割樹+求任兩點間最小割
2 //0-base, root=0
3 LL e[MAXN][MAXN]; //任兩點間最小割
4 int p[MAXN]; //parent
5 ISAP D; //估價函數 for IDA*
6 void gomory_hu(){

```

```

7 fill(p, p+n, 0);
8 fill(e[0], e[n], INF);
9 for( int s = 1; s < n; ++s ) {
10     int t = p[s];
11     ISAP F = D;
12     LL tmp = F.min_cut(s, t);
13     for( int i = 1; i < s; ++i )
14         e[s][i] = e[i][s] = min(tmp, e[t][i]);
15     for( int i = s+1; i <= n; ++i )
16         if( p[i] == t && F.vis[i] ) p[i] = s;
17 }
18 }

```

## 3.2 ISAP with cut

```

1 template<typename T>
2 struct ISAP{
3     static const int MAXN=105;
4     static const T INF=INT_MAX;
5     int n;//點數
6     int d[MAXN],gap[MAXN],cur[MAXN];
7     struct edge{
8         int v,pre;
9         T cap,r;
10         edge(int v,int pre,T cap):v(v),pre(pre),
11             cap(cap),r(cap){}
12     };
13     int g[MAXN];
14     vector<edge> e;
15     void init(int _n){
16         memset(g,-1,sizeof(int)*((n=_n)+1));
17         e.clear();
18     }
19     void add_edge(int u,int v,T cap,bool
20         directed=false){
21         e.push_back(edge(v,g[u],cap));
22         g[u]=e.size()-1;
23         e.push_back(edge(u,g[v],directed?0:cap))
24         ;
25         g[v]=e.size()-1;
26     }
27     T dfs(int u,int s,int t,T CF=INF){
28         if(u==t)return CF;
29         T tf=CF,df;
30         for(int &i=cur[u];~i;i=e[i].pre){
31             if(e[i].r&&d[u]==d[e[i].v]+1){
32                 df=dfs(e[i].v,s,t,min(tf,e[i].r));
33                 e[i].r-=df;
34                 e[i^1].r+=df;
35                 if(!(tf-=df)||d[s]==n)return CF-tf;
36             }
37         }
38         int mh=n;
39         for(int i=cur[u]=g[u];~i;i=e[i].pre){
40             if(e[i].r&&d[e[i].v]<mh)mh=d[e[i].v];
41         }
42         if(!--gap[d[u]])d[s]=n;
43         else ++gap[d[u]]+=mh;
44         return CF-tf;
45     }
46     T isap(int s,int t,bool clean=true){
47         memset(d,0,sizeof(int)*(n+1));
48         memset(gap,0,sizeof(int)*(n+1));

```

```

46 memcpy(cur,g,sizeof(int)*(n+1));
47 if(clean) for(size_t i=0;i<e.size();++i)
48     e[i].r=e[i].cap;
49 T MF=0;
50 for(gap[0]=n;d[s]<n;)MF+=dfs(s,s,t);
51 return MF;
52 }
53 vector<int> cut_e;//最小割邊集
54 bool vis[MAXN];
55 void dfs_cut(int u){
56     vis[u]=1;//表示u屬於source的最小割集
57     for(int i=g[u];~i;i=e[i].pre)
58         if(e[i].r>0&&!vis[e[i].v])dfs_cut(e[i].v);
59 }
60 T min_cut(int s,int t){
61     T ans=isap(s,t);
62     memset(vis,0,sizeof(bool)*(n+1));
63     dfs_cut(s), cut_e.clear();
64     for(int u=0;u<n;++u)if(vis[u])
65         for(int i=g[u];~i;i=e[i].pre)
66             if(!vis[e[i].v])cut_e.push_back(i);
67     return ans;
68 }
69 };

```

## 3.3 maxFlow 拷貝

```

1 #include<bits/stdc++.h>
2 using namespace std;
3 typedef int Graph[55][55];
4 Graph C, R; //Capacity, Flow, Remain
5 int parent[55], flow[55];
6 int n, m;
7 int bfs(int s, int t){
8     memset(parent, 0, sizeof(parent));
9     memset(flow, 0, sizeof(flow));
10     queue<int> q;
11     q.push(s);
12     parent[s] = s;
13     flow[s] = INT_MAX;
14     int current;
15     while(!q.empty()){
16         current = q.front();
17         q.pop();
18         for(int i=1;i<n;i++){
19             if(R[current][i] > 0 && parent[i] == 0){
20                 parent[i] = current;
21                 flow[i] = min(R[current][i], flow[current]);
22             }
23             if(i == t){
24                 return flow[t];
25             }
26             q.push(i);
27         }
28     }
29     return 0;
30 }
31 void fordFulkerson(int s, int t){
32     memcpy(R, C, sizeof(C));

```

```

33 while(1){
34     int f = bfs(1,2);
35     if(f == 0)break;
36     int s = parent[2], t = 2;
37     while(s != t){
38         R[s][t] -= f;
39         t = s;
40         s = parent[t];
41     }
42 }
43 for(int i=1;i<n;i++){
44     if(flow[i] > 0){
45         for(int j=1;j<n;j++){
46             if(flow[j] == 0 && C[i][j] != 0){
47                 cout<<i<<" "<<j<<endl;
48             }
49         }
50     }
51 }
52 cout<<endl;
53 }
54 int main(){
55     while(cin>>n>>m){
56         if(n == 0 && m == 0)break;
57         memset(R,0, sizeof(R));
58         memset(C,0, sizeof(C));
59         for(int i=0;i<m;i++){
60             int a, b, c;
61             cin>>a>>b>>c;
62             C[a][b] = c;
63             C[b][a] = c;
64         }
65         fordFulkerson(1,2);
66     }
67     return 0;
68 }

```

## 3.4 MinCostMaxFlow

```

1 template<typename TP>
2 struct MCMF{
3     static const int MAXN=440;
4     static const TP INF=999999999;
5     struct edge{
6         int v,pre;
7         TP r,cost;
8         edge(int v,int pre,TP r,TP cost):v(v),
9             pre(pre),r(r),cost(cost){}
10     };
11     int n,S,T;
12     TP dis[MAXN],PIS,ans;
13     bool vis[MAXN];
14     vector<edge> e;
15     int g[MAXN];
16     void init(int _n){
17         memset(g,-1,sizeof(int)*((n=_n)+1));
18         e.clear();
19     }
20     void add_edge(int u,int v,TP r,TP cost,
21         bool directed=false){
22         e.push_back(edge(v,g[u],r,cost));
23         g[u]=e.size()-1;

```

```

22     e.push_back(
23         edge(u,g[v],directed?0:r,-cost));
24         g[v]=e.size()-1;
25     }
26     TP augment(int u,TP CF){
27         if(u==T||!CF)return ans+=PIS*CF,CF;
28         vis[u]=1;
29         TP r=CF,d;
30         for(int i=g[u];~i;i=e[i].pre){
31             if(e[i].r&&!e[i].cost&&!vis[e[i].v]){
32                 d=augment(e[i].v,min(r,e[i].r));
33                 e[i].r-=d;
34                 e[i^1].r+=d;
35                 if(!(r-=d))break;
36             }
37         }
38         return CF-r;
39     }
40     bool modlabel(){
41         for(int u=0;u<n;++u)dis[u]=INF;
42         static deque<int>q;
43         dis[T]=0,q.push_back(T);
44         while(q.size()){
45             int u=q.front();q.pop_front();
46             TP dt;
47             for(int i=g[u];~i;i=e[i].pre){
48                 if(e[i^1].r&&(dt=dis[u]-e[i].cost)<
49                     dis[e[i].v]){
50                     if((dis[e[i].v]=dt)<=dis[q.size()])
51                         q.push_front(e[i].v);
52                     }else q.push_back(e[i].v);
53                 }
54             }
55         }
56         for(int u=0;u<n;++u)
57             for(int i=g[u];~i;i=e[i].pre)
58                 e[i].cost+=dis[e[i].v]-dis[u];
59         return PIS+=dis[S], dis[S]<INF;
60     }
61     TP mincost(int s,int t){
62         S=s,T=t;
63         PIS=ans=0;
64         while(modlabel()){
65             do memset(vis,0,sizeof(bool)*(n+1));
66             while(augment(S,INF));
67         }
68     };

```

## 3.5 dinic

```

1 template<typename T>
2 struct DINIC{
3     static const int MAXN=105;
4     static const T INF=INT_MAX;
5     int n, LV[MAXN], cur[MAXN];
6     struct edge{
7         int v,pre;
8         T cap,r;
9         edge(int v,int pre,T cap):v(v),pre(pre),
10             cap(cap),r(cap){}

```



```

11 int g[MAXN];
12 vector<edge> e;
13 void init(int _n){
14     memset(g,-1,sizeof(int)*((n=_n)+1));
15     e.clear();
16 }
17 void add_edge(int u,int v,T cap,bool
18     directed=false){
19     e.push_back(edge(v,g[u],cap));
20     g[u]=e.size()-1;
21     e.push_back(edge(u,g[v],directed?0:cap))
22     ;
23     g[v]=e.size()-1;
24 }
25 int bfs(int s,int t){
26     memset(LV,0,sizeof(int)*(n+1));
27     memcpy(cur,g,sizeof(int)*(n+1));
28     queue<int> q;
29     q.push(s);
30     LV[s]=1;
31     while(q.size()){
32         int u=q.front();q.pop();
33         for(int i=g[u];~i;i=e[i].pre){
34             if(!LV[e[i].v]&&e[i].r){
35                 LV[e[i].v]=LV[u]+1;
36                 q.push(e[i].v);
37                 if(e[i].v==t)return 1;
38             }
39         }
40     }
41     return 0;
42 }
43 T dfs(int u,int t,T CF=INF){
44     if(u==t)return CF;
45     T df;
46     for(int &i=cur[u];~i;i=e[i].pre){
47         if(LV[e[i].v]==LV[u]+1&&e[i].r){
48             if(df=dfs(e[i].v,t,min(CF,e[i].r))){
49                 e[i].r-=df;
50                 e[i^1].r+=df;
51                 return df;
52             }
53         }
54     }
55     return LV[u]=0;
56 }
57 T dinic(int s,int t,bool clean=true){
58     if(clean)for(size_t i=0;i<e.size();++i)
59         e[i].r=e[i].cap;
60     T ans=0, f=0;
61     while(bfs(s,t))while(f=dfs(s,t))ans+=f;
62     return ans;
63 };
64
65 #include <iostream>
66 #include <vector>
67
68 #define INT_MAX 10000000
69
70 using namespace std;
71
72 void DijkstrasTest();
73
74 int main() {
75     DijkstrasTest();
76     return 0;
77 }
78
79 class Node;
80 class Edge;
81
82 void Dijkstras();
83 vector<Node*>* AdjacentRemainingNodes(Node*
84     node);
85 Node* ExtractSmallest(vector<Node*>& nodes);
86 int Distance(Node* node1, Node* node2);
87 bool Contains(vector<Node*>& nodes, Node*
88     node);
89 void PrintShortestRouteTo(Node* destination)
90     ;
91
92 vector<Node*> nodes;
93 vector<Edge*> edges;
94
95 class Node {
96 public:
97     Node(char id)
98         : id(id), previous(NULL),
99           distanceFromStart(INT_MAX) {
100         nodes.push_back(this);
101     }
102     public:
103     char id;
104     Node* previous;
105     int distanceFromStart;
106 };
107
108 class Edge {
109 public:
110     Edge(Node* node1, Node* node2, int
111         distance)
112         : node1(node1), node2(node2), distance(
113             distance) {
114         edges.push_back(this);
115     }
116     bool Connects(Node* node1, Node* node2) {
117         return (
118             (node1 == this->node1 &&
119              node2 == this->node2) ||
120             (node1 == this->node2 &&
121              node2 == this->node1));
122     }
123     public:
124     Node* node1;
125     Node* node2;
126     int distance;
127 };
128
129 void DijkstrasTest() {
130     Node* a = new Node('a');
131     Node* b = new Node('b');
132     Node* c = new Node('c');
133     Node* d = new Node('d');
134     Node* e = new Node('e');
135     Node* f = new Node('f');
136     Node* g = new Node('g');
137
138     Edge* e1 = new Edge(a, c, 1);
139     Edge* e2 = new Edge(a, d, 2);
140     Edge* e3 = new Edge(b, c, 2);
141     Edge* e4 = new Edge(c, d, 1);
142     Edge* e5 = new Edge(b, f, 3);
143     Edge* e6 = new Edge(c, e, 3);
144     Edge* e7 = new Edge(e, f, 2);
145     Edge* e8 = new Edge(d, g, 1);
146     Edge* e9 = new Edge(g, f, 1);
147
148     a->distanceFromStart = 0; // set start
149     node
150     Dijkstras();
151     PrintShortestRouteTo(f);
152 }
153
154 void Dijkstras() {
155     while (nodes.size() > 0) {
156         Node* smallest = ExtractSmallest(nodes);
157         vector<Node*>* adjacentNodes =
158             AdjacentRemainingNodes(smallest);
159
160         const int size = adjacentNodes->size();
161         for (int i = 0; i < size; ++i) {
162             Node* adjacent = adjacentNodes->at(i);
163             int distance = Distance(smallest,
164                 adjacent) +
165                 smallest->distanceFromStart;
166
167             if (distance < adjacent->
168                 distanceFromStart) {
169                 adjacent->distanceFromStart =
170                     distance;
171                 adjacent->previous = smallest;
172             }
173             delete adjacentNodes;
174         }
175     }
176
177     // Find the node with the smallest distance,
178     // remove it, and return it.
179     Node* ExtractSmallest(vector<Node*>& nodes)
180     {
181         int size = nodes.size();
182         if (size == 0) return NULL;
183         int smallestPosition = 0;
184         Node* smallest = nodes.at(0);
185         for (int i = 1; i < size; ++i) {
186             Node* current = nodes.at(i);
187             if (current->distanceFromStart <
188                 smallest->distanceFromStart) {
189                 smallest = current;
190                 smallestPosition = i;
191             }
192         }
193         nodes.erase(nodes.begin() +
194             smallestPosition);
195         return smallest;
196     }
197
198     // Return all nodes adjacent to 'node' which
199     // are still
200     // in the 'nodes' collection.
201     vector<Node*>* AdjacentRemainingNodes(Node*
202         node) {
203         vector<Node*>* adjacentNodes = new vector<
204             Node*>();
205         const int size = edges.size();
206         for (int i = 0; i < size; ++i) {
207             Edge* edge = edges.at(i);
208             Node* adjacent = NULL;
209             if (edge->node1 == node) {
210                 adjacent = edge->node2;
211             } else if (edge->node2 == node) {
212                 adjacent = edge->node1;
213             }
214             if (adjacent && Contains(nodes, adjacent
215                 )) {
216                 adjacentNodes->push_back(adjacent);
217             }
218         }
219         return adjacentNodes;
220     }
221
222     // Return distance between two connected
223     // nodes
224     int Distance(Node* node1, Node* node2) {
225         const int size = edges.size();
226         for (int i = 0; i < size; ++i) {
227             Edge* edge = edges.at(i);
228             if (edge->Connects(node1, node2)) {
229                 return edge->distance;
230             }
231         }
232         return -1; // should never happen
233     }
234
235     // Does the 'nodes' vector contain 'node'
236     bool Contains(vector<Node*>& nodes, Node*
237         node) {
238         const int size = nodes.size();
239         for (int i = 0; i < size; ++i) {
240             if (node == nodes.at(i)) {
241                 return true;
242             }
243         }
244         return false;
245     }
246
247     void PrintShortestRouteTo(Node* destination)
248     {
249         Node* previous = destination;
250         cout << "Distance from start: "
251             << destination->distanceFromStart <<
252             endl;
253         while (previous) {
254             cout << previous->id << " ";
255         }
256     }
257 }

```

## 4 Graph

### 4.1 Dijkstra

// Dijkstra's Algorithm in C++

```

181     previous = previous->previous;
182 }
183 cout << endl;
184 }
185 // these two not needed
186 vector<Edge*>* AdjacentEdges(vector<Edge*>&
187     Edges, Node* node);
188 void RemoveEdge(vector<Edge*>& Edges, Edge*
189     edge);
190 vector<Edge*>* AdjacentEdges(vector<Edge*>&
191     edges, Node* node) {
192     vector<Edge*>* adjacentEdges = new vector<
193         Edge*>();
194     const int size = edges.size();
195     for (int i = 0; i < size; ++i) {
196         Edge* edge = edges.at(i);
197         if (edge->node1 == node) {
198             cout << "adjacent: " << edge->node2->
199                 id << endl;
200             adjacentEdges->push_back(edge);
201         } else if (edge->node2 == node) {
202             cout << "adjacent: " << edge->node1->
203                 id << endl;
204             adjacentEdges->push_back(edge);
205         }
206     }
207     return adjacentEdges;
208 }
209 void RemoveEdge(vector<Edge*>& edges, Edge*
210     edge) {
211     vector<Edge*>::iterator it;
212     for (it = edges.begin(); it < edges.end();
213         ++it) {
214         if (*it == edge) {
215             edges.erase(it);
216             return;
217         }
218     }
219 }

```

## 4.2 FloydWarshall

```

1 // Floyd-Warshall
2 // by. MiohitoKiri5474
3 #include<bits/stdc++.h>
4 #define maxN 5005
5 typedef long long LL;
6
7 using namespace std;
8
9 LL graph[maxN][maxN];
10
11 int main(){
12     ios::sync_with_stdio ( false );
13     cin.tie ( 0 );
14     cout.tie ( 0 );
15
16     int n, m, u, v, w, q;
17     cin >> n >> m;

```

```

18     memset ( graph, 0x3f3f3f, sizeof graph );
19     while ( m-- ){
20         cin >> u >> v >> w;
21         graph[u][v] = graph[v][u] = w;
22     }
23     for ( int i = 0 ; i < n ; i++ )
24         graph[i][i] = 0;
25     for ( int k = 0 ; k < n ; k++ )
26         for ( int i = 0 ; i < n ; i++ )
27             for ( int j = 0 ; j < n ; j++ )
28                 if ( graph[i][j] > graph[i][k] +
29                     graph[k][j] )
30                     graph[i][j] = graph[i][k] + graph[
31                         k][j];
32     cin >> q;
33     while ( q-- ){
34         cin >> u >> v;
35         cout << graph[u][v] << '\n';
36     }
37 }

```

## 4.3 全局最小割

```

1 const int INF=0x3f3f3f3f;
2 template<typename T>
3 struct stoer_wagner{// 0-base
4     static const int MAXN=150;
5     T g[MAXN][MAXN],dis[MAXN];
6     int nd[MAXN],n,s,t;
7     void init(int _n){
8         n=_n;
9         for(int i=0;i<n;++i)
10             for(int j=0;j<n;++j)g[i][j]=0;
11     }
12     void add_edge(int u,int v,T w){
13         g[u][v]=g[v][u]+=w;
14     }
15     T min_cut(){
16         T ans=INF;
17         for(int i=0;i<n;++i)nd[i]=i;
18         for(int ind,tn=n;tn>1;--tn){
19             for(int i=1;i<tn;++i)dis[ind[i]]=0;
20             for(int i=1;i<tn;++i){
21                 ind=i;
22                 for(int j=i;j<tn;++j){
23                     dis[ind[j]]+=g[ind[i-1]][nd[j]];
24                     if(dis[ind[j]]<dis[ind[i]])ind=j;
25                 }
26                 swap(nd[ind],nd[i]);
27             }
28             if(ans>dis[ind[i]])ans=dis[t=nd[ind
29                 ]],s=nd[ind-1];
30             for(int i=0;i<tn;++i)
31                 g[ind[ind-1]][nd[i]]=g[ind[i]][nd[ind
32                     -1]]+g[ind[i]][nd[ind]];
33             return ans;
34         }
35     }
36 };

```

## 4.4 dsu

```

1 int p[100005], rk[100005];
2 int find(int n){
3     if(p[n] == n)return n;
4     return find(p[n]);
5 }
6 void union_set(int x, int y){
7     int rx = find(x);
8     int ry = find(y);
9     if(rx != ry){
10         if(rk[rx] > rk[ry]){
11             p[ry] = rx;
12         } else if(rk[ry] > rk[rx]){
13             p[rx] = ry;
14         } else {
15             p[rx] = ry;
16             rk[ry] += 1;
17         }
18     }
19 }
20 }
21 }

```

## 4.5 spfa

```

1 #include<bits/stdc++.h>
2
3 using namespace std;
4
5 #define maxN 100005
6 #define pb push_back
7 typedef pair < int, int > pii;
8
9 vector < pii > edges[maxN];
10 int dis[maxN];
11 bool inQueue[maxN]; // 紀錄是否已經在queue中
12
13 inline void SPFA ( int start ){
14     memset ( dis, 0x3f3f3f, sizeof dis );
15     dis[start]=0;
16     queue < int > q;
17     q.push ( start );
18     inQueue[start] = true;
19     while ( !q.empty() ){
20         int now = q.front();
21         q.pop();
22         inQueue[now] = false; // 紀錄已經取出
23         random_shuffle ( edges.begin(), edges.
24             end() );//打亂順序 有些測資會故意讓
25             算法爆炸
26         for ( auto i: edges[now] ){ // 跑過所有
27             可以被now連結到的點
28             if ( dis[i.first] > dis[now] + i.
29                 second ){
30                 dis[i.first] = dis[now] + i.second;
31                 if ( !inQueue[i.first] ){
32                     // 如果點沒有在queue中 · 再加入
33                     queue · 並紀錄已經在queue中
34                     inQueue[i.first] = true;
35                     q.push ( i.first );
36                 }
37             }
38         }
39     }
40 }

```

```

31     }
32 }
33 }
34 }
35 }

```

## 4.6 kruskal

```

1 // Kruskal
2 // by. MiohitoKiri5474
3 #include<bits/stdc++.h>
4
5 using namespace std;
6
7 #define maxN 10005
8 #define pb push_back
9 typedef pair < int, int > pii;
10
11 int dis[maxN];
12
13 inline void init ( void ){
14     for ( int i = 0 ; i < maxN ; i++ )
15         dis[i] = i;
16 }
17
18 inline int find ( int a ){
19     return dis[a] == a ? a : dis[a] = find (
20         dis[a] );
21 }
22
23 inline void Union ( int a, int b ){
24     dis[find ( a )] = find ( b );
25 }
26
27 inline bool same ( int a, int b ){
28     return find ( a ) == find ( b );
29 }
30
31 struct node{
32     int u, v, w;
33 };
34
35 inline bool cmp ( node a, node b ){
36     return a.w < b.w;
37 }
38
39 vector < node > edges;
40 vector < pii > mst[maxN];
41
42 inline void Kruskal ( void ){
43     sort ( edges.begin(), edges.end(), cmp );
44     for ( auto i: edges ){ // C++ 11寫法 · 不懂
45         再來問
46         if ( same ( i.u, i.v ) )
47             continue;
48         Union ( i.u, i.v );
49         mst[i.u].pb ( pii ( i.v, i.w ) );
50         mst[i.v].pb ( pii ( i.u, i.w ) );
51     }
52 }

```

## 4.7 bellmanford

```

1 #include<bits/stdc++.h>
2
3 using namespace std;
4
5 #define maxN 100005
6 #define pb push_back
7 typedef pair< int, int > pii;
8
9 vector< pii > edges[maxN];
10 int dis[maxN];
11
12 inline void BellmanFord ( int start ){
13     memset ( dis, 0x3f3f3f, sizeof dis );
14     dis[start]=0;
15     queue< int > q;
16     q.push ( start );
17     while ( !q.empty() ){
18         int now = q.front();
19         q.pop();
20         for ( auto i: edges[now] ){
21             if ( dis[i.first] > dis[now] + i.
                second ){ // 檢查是否能更新
22                 q.push ( i.first );
23                 dis[i.first] = dis[now] + i.second;
24             }
25         }
26     }
27 }

```

## 4.8 kruskalMST

```

1 // C++ program for Kruskal's algorithm to
  // find Minimum
2 // Spanning Tree of a given connected,
  // undirected and
3 // weighted graph
4 #include<bits/stdc++.h>
5 using namespace std;
6
7 // Creating shortcut for an integer pair
8 typedef pair<int, int> iPair;
9
10 // Structure to represent a graph
11 struct Graph
12 {
13     int V, E;
14     vector< pair<int, iPair> > edges;
15
16     // Constructor
17     Graph(int V, int E)
18     {
19         this->V = V;
20         this->E = E;
21     }
22
23     // Utility function to add an edge
24     void addEdge(int u, int v, int w)
25     {
26         edges.push_back({w, {u, v}});
27     }

```

```

28
29     // Function to find MST using Kruskal's
30     // MST algorithm
31     int kruskalMST();
32 };
33
34 // To represent Disjoint Sets
35 struct DisjointSets
36 {
37     int *parent, *rnk;
38     int n;
39
40     // Constructor.
41     DisjointSets(int n)
42     {
43         // Allocate memory
44         this->n = n;
45         parent = new int[n+1];
46         rnk = new int[n+1];
47
48         // Initially, all vertices are in
49         // different sets and have rank 0.
50         for (int i = 0; i <= n; i++)
51         {
52             rnk[i] = 0;
53
54             //every element is parent of itself
55             parent[i] = i;
56         }
57     }
58
59     // Find the parent of a node 'u'
60     // Path Compression
61     int find(int u)
62     {
63         /* Make the parent of the nodes in the
64            path
65            from u--> parent[u] point to parent[u]
66            */
67         if (u != parent[u])
68             parent[u] = find(parent[u]);
69         return parent[u];
70     }
71
72     // Union by rank
73     void merge(int x, int y)
74     {
75         x = find(x), y = find(y);
76
77         /* Make tree with smaller height
78            a subtree of the other tree */
79         if (rnk[x] > rnk[y])
80             parent[y] = x;
81         else // If rnk[x] <= rnk[y]
82             parent[x] = y;
83
84         if (rnk[x] == rnk[y])
85             rnk[y]++;
86     }
87
88     /* Functions returns weight of the MST*/
89     int Graph::kruskalMST()
90     {
91         int mst_wt = 0; // Initialize result

```

```

92
93     // Sort edges in increasing order on basis
94     // of cost
95     sort(edges.begin(), edges.end());
96
97     // Create disjoint sets
98     DisjointSets ds(V);
99
100     // Iterate through all sorted edges
101     vector< pair<int, iPair> >::iterator it;
102     for (it=edges.begin(); it!=edges.end(); it
        ++){
103         int u = it->second.first;
104         int v = it->second.second;
105
106         int set_u = ds.find(u);
107         int set_v = ds.find(v);
108
109         // Check if the selected edge is
110         // creating
111         // a cycle or not (Cycle is created if u
112         // and v belong to same set)
113         if (set_u != set_v)
114         {
115             // Current edge will be in the MST
116             // so print it
117             cout << u << " - " << v << endl;
118
119             // Update MST weight
120             mst_wt += it->first;
121
122             // Merge two sets
123             ds.merge(set_u, set_v);
124         }
125     }
126
127     return mst_wt;
128 }
129
130 // Driver program to test above functions
131 int main()
132 {
133     /* Let us create above shown weighted
134        and undirected graph */
135     int V = 9, E = 14;
136     Graph g(V, E);
137
138     // making above shown graph
139     g.addEdge(0, 1, 4);
140     g.addEdge(0, 7, 8);
141     g.addEdge(1, 2, 8);
142     g.addEdge(1, 7, 11);
143     g.addEdge(2, 3, 7);
144     g.addEdge(2, 8, 2);
145     g.addEdge(2, 5, 4);
146     g.addEdge(3, 4, 9);
147     g.addEdge(3, 5, 14);
148     g.addEdge(4, 5, 10);
149     g.addEdge(5, 6, 2);
150     g.addEdge(6, 7, 1);
151     g.addEdge(6, 8, 6);
152     g.addEdge(7, 8, 7);
153
154     cout << "Edges of MST are \n";
155     int mst_wt = g.kruskalMST();

```

```

156     cout << "\nWeight of MST is " << mst_wt;
157
158     return 0;
159 }

```

## 4.9 PrimeMST

```

1 // Prim
2 // by. MiohitoKiri5474
3 #include<bits/stdc++.h>
4
5 using namespace std;
6
7 #define pb push_back
8 #define F first
9 #define S second
10 typedef pair< int, int > pii;
11 typedef pair< int, pii > pipii;
12 const int maxN = 100005;
13
14 int dis[maxN], sz[maxN];
15
16 inline void init ( void ){
17     for ( int i = 0 ; i < maxN ; i++ )
18         dis[i] = i, sz[i] = 1;
19 }
20
21 int find ( int n ){
22     return dis[n] == n ? n : dis[n] = find (
        dis[n] );
23 }
24
25 inline void Union ( int a, int b ){
26     a = find ( a ), b = find ( b );
27     dis[a] = b;
28     sz[b] += sz[a];
29 }
30
31 inline bool same ( int a, int b ){
32     return find ( a ) == find ( b );
33 }
34
35 vector< pii > edges[maxN], mst[maxN];
36 bool pushed[maxN];
37
38 inline void Pirm ( int n ){
39     priority_queue< pipii, vector< pipii >,
        greater< pipii > > pq;
40     for ( auto i: edges[0] )
41         pq.push ( pipii ( i.S, pii ( i.F, 0 ) )
            );
42     pushed[0] = true;
43     init();
44     while ( sz[find ( 0 )] != n ){
45         pipii top = pq.top();
46         pq.pop();
47         while ( same ( 0, top.S.F ) ){
48             top = pq.top();
49             pq.pop();
50         }
51         int u = top.S.F, v = top.S.S;

```

```

53 mst[u].pb ( pii ( v, top.F ) );
54 mst[v].pb ( pii ( u, top.F ) );
55 Union ( u, v );
56
57 if ( !pushed[u] ){
58     for ( auto i: edges[u] )
59         pq.push ( pii ( i.S, pii ( i.F, u
60             ) ) );
61     pushed[u] = true;
62 }
63 if ( !pushed[v] ){
64     for ( auto i: edges[v] )
65         pq.push ( pii ( i.S, pii ( i.F, v
66             ) ) );
67     pushed[v] = true;
68 }

```

## 4.10 treeISO

```

1 const int MAXN=100005;
2 const long long X=12327,P=0xdefaced;
3 vector<int> g[MAXN];
4 bool vis[MAXN];
5 long long dfs(int u){//hash ver
6     vis[u]=1;
7     vector<long long> tmp;
8     for(auto v:g[u])if(!vis[v])tmp.pb(dfs(v));
9     if(tmp.empty())return 177;
10    long long ret=4931;
11    sort(tmp.begin(),tmp.end());
12    for(auto v:tmp)ret=((ret*X)^v)%P;
13    return ret;
14 }
15 //-----
16 string dfs(int x,int p){
17     vector<string> c;
18     for(int y:g[x])
19         if(y!=p)c.emplace_back(dfs(y,x));
20     sort(c.begin(),c.end());
21     string ret("");
22     for(auto &s:c)ret+=s;
23     ret+="";
24     return ret;
25 }

```

## 4.11 MaximumClique

```

1 struct MaxClique{
2     static const int MAXN=105;
3     int N,ans;
4     int g[MAXN][MAXN],dp[MAXN],stk[MAXN][MAXN];
5     int sol[MAXN],tmp[MAXN];//sol[0~ans-1]為答案
6     void init(int n){
7         N=n;//0-base
8         memset(g,0,sizeof(g));

```

```

9     }
10    void add_edge(int u,int v){
11        g[u][v]=g[v][u]=1;
12    }
13    int dfs(int ns,int dep){
14        if(!ns){
15            if(dep>ans){
16                ans=dep;
17                memcpy(sol,tmp,sizeof tmp);
18                return 1;
19            }else return 0;
20        }
21        for(int i=0;i<ns;++i){
22            if(dep+ns-i<=ans)return 0;
23            int u=stk[dep][i],cnt=0;
24            if(dep+dp[u]<=ans)return 0;
25            for(int j=i+1;j<ns;++j){
26                int v=stk[dep][j];
27                if(g[u][v])stk[dep+1][cnt++]=v;
28            }
29            tmp[dep]=u;
30            if(dfs(cnt,dep+1))return 1;
31        }
32        return 0;
33    }
34    int clique(){
35        int u,v,ns;
36        for(ans=0,u=N-1;u>=0;--u){
37            for(ns=0,tmp[0]=u,v=u+1;v<N;++v)
38                if(g[u][v])stk[1][ns++]=v;
39            dfs(ns,1),dp[u]=ans;
40        }
41        return ans;
42    }
43 }

```

## 5 Linear Programming

### 5.1 simplex

```

1 /*target:
2     max \sum_{j=1}^n A_{0,j}*x_j
3     condition:
4     \sum_{j=1}^n A_{i,j}*x_j <= A_{i,0} | i=1~m
5     x_j >= 0 | j=1~n
6     VDB = vector<double>*/
7     template<class VDB>
8     VDB simplex(int m,int n,vector<VDB> a){
9         vector<int> left(m+1), up(n+1);
10        iota(left.begin(), left.end(), n);
11        iota(up.begin(), up.end(), 0);
12        auto pivot = [&](int x, int y){
13            swap(left[x], up[y]);
14            auto k = a[x][y]; a[x][y] = 1;
15            vector<int> pos;
16            for(int j = 0; j <= n; ++j){
17                a[x][j] /= k;
18                if(a[x][j] != 0) pos.push_back(j);
19            }
20            for(int i = 0; i <= m; ++i){
21                if(a[i][y]==0 || i == x) continue;

```

```

22        k = a[i][y], a[i][y] = 0;
23        for(int j : pos) a[i][j] -= k*a[x][j];
24    }
25 }
26 for(int x,y;;){
27     for(int i=x+1; i <= m; ++i)
28         if(a[i][0]<a[x][0]) x = i;
29     if(a[x][0]>=0) break;
30     for(int j=y+1; j <= n; ++j)
31         if(a[x][j]<a[x][y]) y = j;
32     if(a[x][y]>=0) return VDB();//infeasible
33     pivot(x, y);
34 }
35 for(int x,y;;){
36     for(int j=y+1; j <= n; ++j)
37         if(a[0][j] > a[0][y]) y = j;
38     if(a[0][y]<=0) break;
39     x = -1;
40     for(int i=1; i<=m; ++i) if(a[i][y] > 0)
41         if(x == -1 || a[i][0]/a[i][y]
42             < a[x][0]/a[x][y]) x = i;
43     if(x == -1) return VDB();//unbounded
44     pivot(x, y);
45 }
46 VDB ans(n + 1);
47 for(int i = 1; i <= m; ++i)
48     if(left[i] <= n) ans[left[i]] = a[i][0];
49 ans[0] = -a[0][0];
50 return ans;
51 }

```

## 6 Number Theory

### 6.1 bit set

```

1 void sub_set(int S){
2     int sub=S;
3     do{
4         //對某集合的子集合的處理
5         sub=(sub-1)&S;
6     }while(sub!=S);
7 }
8 void k_sub_set(int k,int n){
9     int comb=(1<<k)-1,S=1<<n;
10    while(comb<S){
11        //對大小為k的子集合的處理
12        int x=comb&-comb,y=comb+x;
13        comb=((comb~y)/x>>1)|y;
14    }
15 }

```

### 6.2 FFT

```

1 template<typename T,typename VT=vector<
2     complex<T> > >
3 struct FFT{
4     const T pi;

```

```

4     FFT(const T pi=acos((T)-1)):pi(pi){}
5     unsigned bit_reverse(unsigned a,int len){
6         a=((a&0x55555555U)<<1)|((a&0xAAAAAAAAU)>>1);
7         a=((a&0x33333333U)<<2)|((a&0xCCCCCCCCU)>>2);
8         a=((a&0xF0F0F0F0U)<<4)|((a&0x0F0F0F0F0U)>>4);
9         a=((a&0xFF0F0F0F0U)<<8)|((a&0xFF0F0F0F0U)>>8);
10        a=((a&0x0000FFFFU)<<16)|((a&0xFFFF0000U)
11            >>16);
12        return a>>(32-len);
13    }
14    void fft(bool is_inv,VT &in,VT &out,int N)
15    {
16        int bitlen=__lg(N),num=is_inv?-1:1;
17        for(int i=0;i<N;++i)out[bit_reverse(i,
18            bitlen)]=in[i];
19        for(int step=2;step<=N;step<=(1)){
20            const int mh=step>>1;
21            for(int i=0;i<mh;++i){
22                complex<T> wi=exp(complex<T>(0,i*num
23                    *pi/mh));
24                for(int j=i;j<N;j+=step){
25                    int k=j+mh;
26                    complex<T> u=out[j],t=wi*out[k];
27                    out[j]=u+t;
28                    out[k]=u-t;
29                }
30            }
31            if(is_inv)for(int i=0;i<N;++i)out[i]/=N;
32        }
33    }

```

### 6.3 質因數分解

```

1 LL func(const LL n,const LL mod,const int c)
2 {
3     return (LLmul(n,n,mod)+c+mod)%mod;
4 }
5 LL pollrho(const LL n, const int c) {//循環長度
6     LL a=1, b=1;
7     a=func(a,n,c)%n;
8     b=func(b,n,c)%n; b=func(b,n,c)%n;
9     while(gcd(abs(a-b),n)==1) {
10        a=func(a,n,c)%n;
11        b=func(b,n,c)%n; b=func(b,n,c)%n;
12    }
13    return gcd(abs(a-b),n);
14 }
15
16 void prefactor(LL &n, vector<LL> &v) {
17     for(int i=0;i<12;++i) {
18         while(n%prime[i]==0) {
19             v.push_back(prime[i]);
20             n/=prime[i];
21         }
22     }
23 }
24
25 void smallfactor(LL n, vector<LL> &v) {
26     if(n<MAXPRIME) {

```



```

27 while(isp[(int)n]) {
28     v.push_back(isp[(int)n]);
29     n/=isp[(int)n];
30 }
31 v.push_back(n);
32 } else {
33     for(int i=0;i<primecnt&&prime[i]*prime[i]
34         ]<=n;++i) {
35         while(n%prime[i]==0) {
36             v.push_back(prime[i]);
37             n/=prime[i];
38         }
39     }
40     if(n!=1) v.push_back(n);
41 }
42
43 void comfactor(const LL &n, vector<LL> &v) {
44     if(n<1e9) {
45         smallfactor(n,v);
46         return;
47     }
48     if(Isprime(n)) {
49         v.push_back(n);
50         return;
51     }
52     LL d;
53     for(int c=3;;++c) {
54         d = pollorrho(n,c);
55         if(d!=n) break;
56     }
57     comfactor(d,v);
58     comfactor(n/d,v);
59 }
60
61 void Factor(const LL &x, vector<LL> &v) {
62     LL n = x;
63     if(n==1) { puts("Factor 1"); return; }
64     prefactor(n,v);
65     if(n==1) return;
66     comfactor(n,v);
67     sort(v.begin(),v.end());
68 }
69
70 void AllFactor(const LL &n,vector<LL> &v) {
71     vector<LL> tmp;
72     Factor(n,tmp);
73     v.clear();
74     v.push_back(1);
75     int len;
76     LL now=1;
77     for(int i=0;i<tmp.size();++i) {
78         if(i==0 || tmp[i]!=tmp[i-1]) {
79             len = v.size();
80             now = 1;
81         }
82         now*=tmp[i];
83         for(int j=0;j<len;++j)
84             v.push_back(v[j]*now);
85     }
86 }

```

## 6.4 Matrix

```

1 template<typename T>
2 struct Matrix{
3     using rt = std::vector<T>;
4     using mt = std::vector<rt>;
5     using matrix = Matrix<T>;
6     int r,c;
7     mt m;
8     Matrix(int r,int c):r(r),c(c),m(r,rt(c)){
9         rt& operator[](int i){return m[i];}
10        matrix operator+(const matrix &a){
11            matrix rev(r,c);
12            for(int i=0;i<r;++i)
13                for(int j=0;j<c;++j)
14                    rev[i][j]=m[i][j]+a.m[i][j];
15            return rev;
16        }
17        matrix operator-(const matrix &a){
18            matrix rev(r,c);
19            for(int i=0;i<r;++i)
20                for(int j=0;j<c;++j)
21                    rev[i][j]=m[i][j]-a.m[i][j];
22            return rev;
23        }
24        matrix operator*(const matrix &a){
25            matrix rev(r,a.c);
26            matrix tmp(a.c,a.r);
27            for(int i=0;i<a.r;++i)
28                for(int j=0;j<a.c;++j)
29                    tmp[j][i]=a.m[i][j];
30            for(int i=0;i<r;++i)
31                for(int j=0;j<a.c;++j)
32                    for(int k=0;k<c;++k)
33                        rev.m[i][j]+=m[i][k]*tmp[j][k];
34            return rev;
35        }
36        bool inverse(){
37            Matrix t(r,r+c);
38            for(int y=0;y<r;y++){
39                t.m[y][c+y] = 1;
40                for(int x=0;x<c;++x)
41                    t.m[y][x]=m[y][x];
42            }
43            if( !t.gas() )
44                return false;
45            for(int y=0;y<r;y++){
46                for(int x=0;x<c;++x)
47                    m[y][x]=t.m[y][c+x]/t.m[y][y];
48            }
49            return true;
50        }
51        T gas(){
52            vector<T> lazy(r,1);
53            bool sign=false;
54            for(int i=0;i<r;++i){
55                if( m[i][i]==0 ){
56                    int j=i+1;
57                    while(j<r&&!m[j][i])j++;
58                    if(j==r)continue;
59                    m[i].swap(m[j]);
60                    sign=!sign;
61                }
62                for(int j=0;j<r;++j){
63                    if(i==j)continue;
64                    lazy[j]=lazy[j]*m[i][i];

```

```

64         T mx=m[j][i];
65         for(int k=0;k<c;++k)
66             m[j][k]=m[j][k]*m[i][i]-m[i][k]*mx;
67     }
68 }
69 T det=sign?-1:1;
70 for(int i=0;i<r;++i){
71     det = det*m[i][i];
72     det = det/lazy[i];
73     for(auto &j:m[i])j/=lazy[i];
74 }
75 return det;
76 }
77 };

```

## 7 String

### 7.1 reverseBWT

```

1 const int MAXN = 305, MAXC = 'Z';
2 int ranks[MAXN], tots[MAXC], first[MAXC];
3 void rankBWT(const string &bw){
4     memset(ranks,0,sizeof(int)*bw.size());
5     memset(tots,0,sizeof(tots));
6     for(size_t i=0;i<bw.size();++i)
7         ranks[i] = tots[int(bw[i])];
8 }
9 void firstCol(){
10     memset(first,0,sizeof(first));
11     int totc = 0;
12     for(int c='A';c<='Z';++c){
13         if(!tots[c]) continue;
14         first[c] = totc;
15         totc += tots[c];
16     }
17 }
18 string reverseBwt(string bw,int begin){
19     rankBWT(bw, firstCol());
20     int i = begin; //原字串最後一個元素的位置
21     string res;
22     do{
23         char c = bw[i];
24         res = c + res;
25         i = first[int(c)] + ranks[i];
26     }while( i != begin );
27     return res;
28 }

```

### 7.2 suffix array lcp

```

1 #define radix_sort(x,y){\
2     for(i=0;i<A;++i)c[i]=0;\
3     for(i=0;i<n;++i)c[x[y[i]]]++; \
4     for(i=1;i<A;++i)c[i]+=c[i-1]; \
5     for(i=n-1;i--i)sa[--c[x[y[i]]]]=y[i]; \
6 }
7 #define AC(r,a,b)\

```

```

8     r[a]!=(r[b]||a+k>n||r[a+k]!=r[b+k])
9 void suffix_array(const char *s,int n,int *
10     sa,int *rank,int *tmp,int *c){
11     int A='z'+1,i,k,id=0;
12     for(i=0;i<n;++i)rank[tmp[i]=i]=s[i];
13     radix_sort(rank,tmp);
14     for(k=1;id<n-1;k<=1){
15         for(id=0,i=n-k;i<n;++i)tmp[id++]=i;
16         for(i=0;i<n;++i)
17             if(sa[i]>k)tmp[id++]=sa[i]-k;
18         radix_sort(rank,tmp);
19         swap(rank,tmp);
20         for(rank[sa[0]]=id=0,i=1;i<n;++i)
21             rank[sa[i]]=id+=AC(tmp,sa[i-1],sa[i]);
22         A=id+1;
23     }
24     //h:高度數組 sa:後綴數組 rank:排名
25 void suffix_array_lcp(const char *s,int len,
26     int *h,int *sa,int *rank){
27     for(int i=0;i<len;++i)rank[sa[i]]=i;
28     for(int i=0,k=0;i<len;++i){
29         if(rank[i]==0)continue;
30         if(k--<0)
31             while(s[i+k]==s[sa[rank[i]-1]+k])++k;
32         h[rank[i]]=k;
33     }
34 }

```

### 7.3 kmp

```

1 #include<bits/stdc++.h>
2 using namespace std;
3 // #define int long long
4 #define IOS ios::sync_with_stdio(0);cin.tie
5     (0);
6 vector<int> v(1005);
7 void build_next(string T){
8     v[0] = 0;
9     int pre_len = 0;
10    int i = 1;
11    while(i<T.length()){
12        if(T[pre_len] == T[i]){
13            pre_len++;
14            v.push_back(pre_len);
15            i++;
16        } else {
17            if(pre_len > 0){
18                pre_len = v[pre_len-1];
19            }
20            if(pre_len == 0){
21                v.push_back(0);
22                i++;
23            }
24        }
25    }
26 }
27 bool KMP(string S, string T){
28     build_next(T);
29     int i=0,j=0;
30     while(i<S.length()){
31         if(S[i] == T[j]){

```

```

31     i++;
32     j++;
33 } else if(j>0){
34     j = v[j-1];
35 } else {
36     i++;
37 }
38 if(j == T.length()){
39     return 1;
40 }
41 }
42 return 0;
43 }
44 }
45
46 int32_t main(){
47     int k;
48     cin>>k;
49     while(k--){
50         int q;
51         string s;
52         cin>>s;
53         cin>>q;
54         while(q--){
55             string t;
56             cin>>t;
57             if(KMP(s,t) == 1)cout<<"y"<<endl;
58             else cout<<"n"<<endl;
59         }
60     }
61 }
62 return 0;
63 }
64 }

```

## 7.4 hash

```

1 #define MAXN 1000000
2 #define mod 1073676287
3 /*mod 必須要是質數*/
4 typedef long long T;
5 char s[MAXN+5];
6 T h[MAXN+5];/*hash陣列*/
7 T h_base[MAXN+5];/*h_base[n]=(prime^n)%mod*/
8 void hash_init(int len,T prime){
9     h_base[0]=1;
10    for(int i=1;i<=len;++i){
11        h[i]=(h[i-1]*prime+s[i-1])%mod;
12        h_base[i]=(h_base[i-1]*prime)%mod;
13    }
14 }
15 T get_hash(int l,int r){/*閉區間寫法，設編號
    為 0 ~ len-1*/
16     return (h[r+1]-(h[l]*h_base[r-l+1])%mod+
17         mod)%mod;

```

## 7.5 AC 自動機

```

1 template<char L='a',char R='z'>
2 class ac_automaton{
3     struct joe{
4         int next[R-L+1],fail,efl,ed,cnt_dp,vis;
5         joe():ed(0),cnt_dp(0),vis(0){
6             for(int i=0;i<=R-L;++i)next[i]=0;
7         }
8     };
9     public:
10    std::vector<joe> S;
11    std::vector<int> q;
12    int qs,qe,vt;
13    ac_automaton():S(1),qs(0),qe(0),vt(0){
14        void clear(){
15            q.clear();
16            S.resize(1);
17            for(int i=0;i<=R-L;++i)S[0].next[i]=0;
18            S[0].cnt_dp=S[0].vis=qs=qe=vt=0;
19        }
20        void insert(const char *s){
21            int o=0;
22            for(int i=0,id;s[i];++i){
23                id=s[i]-L;
24                if(!S[o].next[id]){
25                    S.push_back(joe());
26                    S[o].next[id]=S.size()-1;
27                }
28                o=S[o].next[id];
29            }
30            ++S[o].ed;
31        }
32        void build_fail(){
33            S[0].fail=S[0].efl=-1;
34            q.clear();
35            q.push_back(0);
36            ++qe;
37            while(qs!=qe){
38                int pa=q[qs++],id,t;
39                for(int i=0;i<=R-L;++i){
40                    t=S[pa].next[i];
41                    if(!t)continue;
42                    id=S[pa].fail;
43                    while(~id&&!S[id].next[i])id=S[id].fail;
44                    S[t].fail=~id?S[id].next[i]:0;
45                    S[t].efl=S[t].fail.ed+S[t].fail.S[t].fail.efl;
46                    q.push_back(t);
47                    ++qe;
48                }
49            }
50        }
51        /*DP出每個前綴在字串s出現的次數並傳回所有
        字串被s匹配成功的次數O(N*M)*/
52        int match_0(const char *s){
53            int ans=0,id,p=0,t;
54            for(i=0;s[i];++i){
55                id=s[i]-L;
56                while(!S[p].next[id]&&p)S[p].fail;
57                if(!S[p].next[id])continue;
58                p=S[p].next[id];
59                ++S[p].cnt_dp;/*匹配成功則它所有後綴都
        可以被匹配(DP計算)*/
60            }
61            for(i=qe-1;i>=0;--i){

```

```

62            ans+=S[q[i]].cnt_dp*S[q[i]].ed;
63            if(~S[q[i]].fail)S[q[i]].fail.
                cnt_dp+=S[q[i]].cnt_dp;
64        }
65        return ans;
66    }
67    /*多串匹配走efl邊並傳回所有字串被s匹配成功
        的次數O(N*M^1.5)*/
68    int match_1(const char *s)const{
69        int ans=0,id,p=0,t;
70        for(int i=0;s[i];++i){
71            id=s[i]-L;
72            while(!S[p].next[id]&&p)S[p].fail;
73            if(!S[p].next[id])continue;
74            p=S[p].next[id];
75            if(S[p].ed)ans+=S[p].ed;
76            for(t=S[p].efl;~t;t=S[t].efl){
77                ans+=S[t].ed;/*因為都走efl邊所以保證
        匹配成功*/
78            }
79        }
80        return ans;
81    }
82    /*枚舉(s的子字串na)的所有相異字串各恰一次
        並傳回次數O(N*M^(1/3))*/
83    int match_2(const char *s){
84        int ans=0,id,p=0,t;
85        ++vt;
86        /*把戳記vt+=1，只要vt沒溢位，所有S[p].
        vis==vt就會變成false
        這種利用vt的方法可以O(1)歸零vis陣列*/
87        for(int i=0;s[i];++i){
88            id=s[i]-L;
89            while(!S[p].next[id]&&p)S[p].fail;
90            if(!S[p].next[id])continue;
91            p=S[p].next[id];
92            if(S[p].ed&&S[p].vis!=vt){
93                S[p].vis=vt;
94                ans+=S[p].ed;
95            }
96            for(t=S[p].efl;~t&&S[t].vis!=vt;t=S[t].efl){
97                S[t].vis=vt;
98                ans+=S[t].ed;/*因為都走efl邊所以保證
        匹配成功*/
99            }
100        }
101        return ans;
102    }
103    /*把AC自動機變成真的自動機*/
104    void evolution(){
105        for(qs=1;qs!=qe;){
106            int p=q[qs++];
107            for(int i=0;i<=R-L;++i)
108                if(S[p].next[i]==0)S[p].next[i]=S[S[p].fail].next[i];
109        }
110    }
111 }
112 }

```

## 7.6 minimal string rotation

```

1 int min_string_rotation(const string &s){
2     int n=s.size(),i=0,j=1,k=0;
3     while(i<n&&j<n&&k<n){
4         int t=s[(i+k)%n]-s[(j+k)%n];
5         ++k;
6         if(t>0){
7             if(t>0)i+=k;
8             else j+=k;
9             if(i==j)++j;
10            k=0;
11        }
12    }
13    return min(i,j);/*最小循環表示法起始位置
14 }

```

## 7.7 Z

```

1 void z_alg(char *s,int len,int *z){
2     int l=0,r=0;
3     z[0]=len;
4     for(int i=1;i<len;++i){
5         z[i]=i>r?(i-l+z[i-l]<z[l]?z[i-l]:r-i+1);
6         while(i+z[i]<len&&s[i+z[i]]==s[z[i]])++z[i];
7         if(i+z[i]-1>r)r=i+z[i]-1,l=i;
8     }
9 }

```

## 8 Tarjan

### 8.1 橋連通分量

```

1 #define N 1005
2 struct edge{
3     int u,v;
4     bool is_bridge;
5     edge(int u=0,int v=0):u(u),v(v),is_bridge(0){}
6 };
7 vector<edge> E;
8 vector<int> G[N];/* 1-base
9 int low[N],vis[N],Time;
10 int bcc_id[N],bridge_cnt,bcc_cnt;/* 1-base
11 int st[N],top;/*BCC用
12 void add_edge(int u,int v){
13     G[u].push_back(E.size());
14     E.emplace_back(u,v);
15     G[v].push_back(E.size());
16     E.emplace_back(v,u);
17 }
18 void dfs(int u,int re=-1){/*u當前點，re為u連
        接前一個點的邊
19     int v;
20     low[u]=vis[u]=++Time;
21     st[top++]=u;
22     for(int e:G[u]){

```

```

23 | v=E[e].v;
24 | if(!vis[v]){
25 |     dfs(v,e^1);//e^1反向邊
26 |     low[u]=min(low[u],low[v]);
27 |     if(vis[u]<low[v]){
28 |         E[e].is_bridge=E[e^1].is_bridge=1;
29 |         ++bridge_cnt;
30 |     }
31 | }else if(vis[v]<vis[u]&&e!=re)
32 |     low[u]=min(low[u],vis[v]);
33 | }
34 | if(vis[u]==low[u]){//處理BCC
35 |     ++bcc_cnt;//1-base
36 |     do bcc_id[v=st[--top]]=bcc_cnt;//每個點
37 |         所在的BCC
38 |     while(v!=u);
39 | }
40 | void bcc_init(int n){
41 |     Time=bcc_cnt=bridge_cnt=top=0;
42 |     E.clear();
43 |     for(int i=1;i<=n;++i){
44 |         G[i].clear();
45 |         vis[i]=bcc_id[i]=0;
46 |     }
47 | }

```

## 9 Tree Problem

### 9.1 LCA

```

1 | const int MAXN=100000; // 1-base
2 | const int MLG=17; //Log2(MAXN)+1;
3 | int pa[MLG+2][MAXN+5];
4 | int dep[MAXN+5];
5 | vector<int> G[MAXN+5];
6 | void dfs(int x,int p=0){//dfs(root);
7 |     pa[0][x]=p;
8 |     for(int i=0;i<=MLG;++i)
9 |         pa[i+1][x]=pa[i][pa[i][x]];
10 |     for(auto &i:G[x]){
11 |         if(i==p)continue;
12 |         dep[i]=dep[x]+1;
13 |         dfs(i,x);
14 |     }
15 | }
16 | inline int jump(int x,int d){
17 |     for(int i=0;i<=MLG;++i)
18 |         if((d>>i)&1) x=pa[i][x];
19 |     return x;
20 | }
21 | inline int find_lca(int a,int b){
22 |     if(dep[a]>dep[b])swap(a,b);
23 |     b=jump(b,dep[b]-dep[a]);
24 |     if(a==b)return a;
25 |     for(int i=MLG;i>=0;--i){
26 |         if(pa[i][a]!=pa[i][b]){
27 |             a=pa[i][a];
28 |             b=pa[i][b];
29 |         }

```

### 9.2 link cut tree

```

1 | struct splay_tree{
2 |     int ch[2],pa; //子節點跟父母
3 |     bool rev; //反轉的懶惰標記
4 |     splay_tree():pa(0),rev(0){ch[0]=ch[1]=0;}
5 | };
6 | vector<splay_tree> nd;
7 | //有的時候用vector會TLE，要注意
8 | //這邊以node[0]作為null節點
9 | bool isroot(int x){//判斷是否為這棵splay
10 |     tree的根
11 |     return nd[nd[x].pa].ch[0]!=x&&nd[nd[x].pa]
12 |         .ch[1]!=x;
13 | }
14 | void down(int x){//懶惰標記下推
15 |     if(nd[x].rev){
16 |         if(nd[x].ch[0])nd[nd[x].ch[0]].rev^=1;
17 |         if(nd[x].ch[1])nd[nd[x].ch[1]].rev^=1;
18 |         swap(nd[x].ch[0],nd[x].ch[1]);
19 |         nd[x].rev=0;
20 |     }
21 | }
22 | void push_down(int x){//所有祖先懶惰標記下推
23 |     if(!isroot(x))push_down(nd[x].pa);
24 |     down(x);
25 | }
26 | void up(int x){//將子節點的資訊向上更新
27 | }
28 | void rotate(int x){//旋轉，會自行判斷轉的方
29 |     向
30 |     int y=nd[x].pa,z=nd[y].pa,d=(nd[y].ch[1]==
31 |         x);
32 |     nd[x].pa=z;
33 |     if(!isroot(y))nd[z].ch[nd[z].ch[1]==y]=x;
34 |     nd[y].ch[d]=nd[x].ch[d^1];
35 |     nd[nd[y].ch[d]].pa=y;
36 |     nd[y].pa=x,nd[x].ch[d^1]=y;
37 |     up(y),up(x);
38 | }
39 | void splay(int x){//將x伸展到splay tree的根
40 |     push_down(x);
41 |     while(!isroot(x)){
42 |         int y=nd[x].pa;
43 |         if(!isroot(y)){
44 |             int z=nd[y].pa;
45 |             if((nd[z].ch[0]==y)^(nd[y].ch[0]==x))
46 |                 rotate(y);
47 |             else rotate(x);
48 |         }
49 |         rotate(x);
50 |     }

```

```

51 |     up(x);
52 |     last=x;
53 |     x=nd[x].pa;
54 | }
55 | return last;//access後splay tree的根
56 | }
57 | void access(int x,bool is=0){//is=0就是一般
58 |     的access
59 |     int last=0;
60 |     while(x){
61 |         splay(x);
62 |         if(is&&!nd[x].pa){
63 |             //printf("%d\n",max(nd[Last].ma,nd[nd[
64 |                 x].ch[1]].ma));
65 |         }
66 |         nd[x].ch[1]=last;
67 |         up(x);
68 |         last=x;
69 |         x=nd[x].pa;
70 |     }
71 | }
72 | void query_edge(int u,int v){
73 |     access(u);
74 |     access(v,1);
75 | }
76 | void make_root(int x){
77 |     access(x),splay(x);
78 |     nd[x].rev^=1;
79 | }
80 | void make_root(int x){
81 |     nd[access(x)].rev^=1;
82 |     splay(x);
83 | }
84 | void cut(int x,int y){
85 |     make_root(x);
86 |     access(y);
87 |     splay(y);
88 |     nd[y].ch[0]=0;
89 |     nd[x].pa=0;
90 | }
91 | void cut_parents(int x){
92 |     access(x);
93 |     splay(x);
94 |     nd[nd[x].ch[0]].pa=0;
95 |     nd[x].ch[0]=0;
96 | }
97 | void link(int x,int y){
98 |     make_root(x);
99 |     nd[x].pa=y;
100 | }
101 | int find_root(int x){
102 |     x=access(x);
103 |     while(nd[x].ch[0])x=nd[x].ch[0];
104 |     splay(x);
105 |     return x;
106 | }
107 | int query(int u,int v){
108 |     //傳回uv路徑splay tree的根結點
109 |     //這種寫法無法求LCA
110 |     make_root(u);
111 |     return access(v);

```

```

112 | //假設求鏈上點權的總和，sum是子樹的權重和，
113 |     data是節點的權重
114 |     access(u);
115 |     int lca=access(v);
116 |     splay(u);
117 |     if(u==lca){
118 |         //return nd[lca].data+nd[nd[lca].ch[1]].
119 |             sum
120 |     }else{
121 |         //return nd[lca].data+nd[nd[lca].ch[1]].
122 |             sum+nd[u].sum
123 |     }
124 | }
125 | struct EDGE{
126 |     int a,b,w;
127 | }e[10005];
128 | int n;
129 | vector<pair<int,int>> G[10005];
130 | //first表示子節點，second表示邊的編號
131 | int pa[10005],edge_node[10005];
132 | //pa是父母節點，暫存用的，edge_node是每個編
133 |     被存在哪個點裡面的陣列
134 | void bfs(int root){
135 |     //在建構的時候把每個點都設成一個splay tree
136 |     queue<int> q;
137 |     for(int i=1;i<=n;++i)pa[i]=0;
138 |     q.push(root);
139 |     while(q.size()){
140 |         int u=q.front();
141 |         q.pop();
142 |         for(auto P:G[u]){
143 |             int v=P.first;
144 |             if(v!=pa[u]){
145 |                 pa[v]=u;
146 |                 nd[v].pa=u;
147 |                 nd[v].data=e[P.second].w;
148 |                 edge_node[P.second]=v;
149 |                 up(v);
150 |                 q.push(v);
151 |             }
152 |         }
153 |     }
154 | }
155 | void change(int x,int b){
156 |     splay(x);
157 |     //nd[x].data=b;
158 |     up(x);

```

## 10 other

### 10.1 上下最大正方形

```

1 | void solve(int n,int a[],int b[]){// 1-base
2 |     int ans=0;
3 |     deque<int> da,db;
4 |     for(int l=1,r=1;r<=n;++r){
5 |         while(da.size()&&a[da.back()]>=a[r]){
6 |             da.pop_back();
7 |         }

```

```

8 |     da.push_back(r);
9 |     while(db.size() && b[db.back()] >= b[r]){
10 |         db.pop_back();
11 |     }
12 |     db.push_back(r);
13 |     for(int d=a[da.front()+b[db.front()]]; r-
14 |         1+1>d; ++1){
15 |         if(da.front()==1) da.pop_front();
16 |         if(db.front()==1) db.pop_front();
17 |         if(da.size() && db.size()){
18 |             d=a[da.front()+b[db.front()]];
19 |         }
20 |     }
21 |     ans=max(ans, r-1+1);
22 | }
23 | printf("%d\n", ans);

```

## 10.2 WhatDay

```

1 | int whatday(int y, int m, int d){
2 |     if(m<=2) m+=12, --y;
3 |     if(y<1752 || y==1752 && m<9 || y==1752 && m==9 && d<3)
4 |         return (d+2*m+3*(m+1)/5+y+y/4+5)%7;
5 |     return (d+2*m+3*(m+1)/5+y+y/4-y/100+y/400)%7;
6 | }

```

## 10.3 最大矩形

```

1 | LL max_rectangle(vector<int> s){
2 |     stack<pair<int, int> > st;
3 |     st.push(make_pair(-1, 0));
4 |     s.push_back(0);
5 |     LL ans=0;
6 |     for(size_t i=0; i<s.size(); ++i){
7 |         int h=s[i];
8 |         pair<int, int> now=make_pair(h, i);
9 |         while(h<st.top().first){
10 |             now=st.top();
11 |             st.pop();
12 |             ans=max(ans, (LL)(i-now.second)*now.first);
13 |         }
14 |         if(h>st.top().first){
15 |             st.push(make_pair(h, now.second));
16 |         }
17 |     }
18 |     return ans;
19 | }

```

## 11 zformula

### 11.1 formula

#### 11.1.1 Pick 公式

給定頂點坐標均是整點的簡單多邊形，面積 = 內部格點數 + 邊上格點數/2 - 1

#### 11.1.2 圖論

- 對於平面圖： $F = E - V + C + 1$ ，C 是連通分量數
- 對於平面圖： $E \leq 3V - 6$
- 對於連通圖 G，最大獨立點集的大小設為  $I(G)$ ，最大匹配大小設為  $M(G)$ ，最小點覆蓋設為  $C_v(G)$ ，最小邊覆蓋設為  $C_e(G)$ 。對於任意連通圖：

- $I(G) + C_v(G) = |V|$
- $M(G) + C_e(G) = |V|$

- 對於連通二分圖：

- $I(G) = C_v(G)$
- $M(G) = C_e(G)$

- 最大權閉合圖：

- $C(u, v) = \infty, (u, v) \in E$
- $C(S, v) = W_v, W_v > 0$
- $C(v, T) = -W_v, W_v < 0$
- $ans = \sum_{W_v > 0} W_v - flow(S, T)$

- 最大密度子圖：

- 求  $\max \left( \frac{W_e + W_v}{|V|} \right), e \in E', v \in V'$
- $U = \sum_{v \in V} 2W_v + \sum_{e \in E} W_e$
- $C(u, v) = W_{(u, v)}, (u, v) \in E$ ，雙向邊
- $C(S, v) = U, v \in V$
- $D_u = \sum_{(u, v) \in E} W_{(u, v)}$
- $C(v, T) = U + 2g - D_v - 2W_v, v \in V$
- 二分搜  $g$ ：  
 $l = 0, r = U, eps = 1/n^2$   
 if $((U \times |V| - flow(S, T))/2 > 0) l = mid$   
 else  $r = mid$
- $ans = min\_cut(S, T)$
- $|E| = 0$  要特殊判斷

- 弦圖：

- 點數大於 3 的環都要有一條弦
- 完美消除序列從後往前依次給每個點染色，給每個點染上可以染的最小顏色
- 最大團大小 = 色數
- 最大獨立集：完美消除序列從前往後能選就選
- 最小團覆蓋：最大獨立集的點和他延伸的邊構成
- 區間圖是弦圖
- 區間圖的完美消除序列：將區間按造又端點由小到大排序
- 區間圖染色：用線段樹做

#### 11.1.3 dinic 特殊圖複雜度

- 單位流： $O \left( \min(V^{3/2}, E^{1/2}) E \right)$
- 二分圖： $O(V^{1/2} E)$

#### 11.1.4 0-1 分數規劃

$x_i = \{0, 1\}$ ， $x_i$  可能會有其他限制，求  $\max \left( \frac{\sum B_i x_i}{\sum C_i x_i} \right)$

- $D(i, g) = B_i - g \times C_i$
- $f(g) = \sum D(i, g) x_i$
- $f(g) = 0$  時  $g$  為最佳解， $f(g) < 0$  沒有意義
- 因為  $f(g)$  單調可以二分搜  $g$
- 或用 Dinkelbach 通常比較快

```

1 | binary_search(){
2 |     while(r-l>eps){
3 |         g=(l+r)/2;
4 |         for(i:所有元素) D[i]=B[i]-g*C[i]; //D(i,g)
5 |         找出一組合法x[i]使f(g)最大;
6 |         if(f(g)>0) l=g;
7 |         else r=g;
8 |     }
9 |     Ans = r;
10 | }
11 | Dinkelbach(){
12 |     g=任意狀態(通常設為0);
13 |     do{
14 |         Ans=g;
15 |         for(i:所有元素) D[i]=B[i]-g*C[i]; //D(i,g)
16 |         找出一組合法x[i]使f(g)最大;
17 |         p=0, q=0;
18 |         for(i:所有元素)
19 |             if(x[i]*p+B[i], q+C[i]);
20 |         g=p/q; //更新解，注意q=0的情況
21 |     } while(abs(Ans-g)>EPS);
22 |     return Ans;
23 | }

```

#### 11.1.5 學長公式

- $\sum_{d|n} \phi(n) = n$
- $g(n) = \sum_{d|n} f(d) \Rightarrow f(n) = \sum_{d|n} \mu(d) \times g(n/d)$
- Harmonic series  $H_n = \ln(n) + \gamma + 1/(2n) - 1/(12n^2) + 1/(120n^4)$
- $\gamma = 0.57721566490153286060651209008240243104215$
- 格雷碼： $n \oplus (n >> 1)$
- $SG(A+B) = SG(A) \oplus SG(B)$
- 選轉矩陣  $M(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$

#### 11.1.6 基本數論

- $\sum_{d|n} \mu(n) = [n == 1]$
- $g(m) = \sum_{d|m} f(d) \Leftrightarrow f(m) = \sum_{d|m} \mu(d) \times g(m/d)$
- $\sum_{i=1}^n \sum_{j=1}^m \text{互質數量} = \sum \mu(d) \lfloor \frac{n}{d} \rfloor \lfloor \frac{m}{d} \rfloor$
- $\sum_{i=1}^n \sum_{j=1}^n lcm(i, j) = n \sum_{d|n} d \times \phi(d)$

#### 11.1.7 排組公式

- k 卡特蘭  $\frac{C_k^{kn}}{n(k-1)+1} \cdot C_m^n = \frac{n!}{m!(n-m)!}$
- $H(n, m) \cong x_1 + x_2 + \dots + x_n = k, num = C_k^{n+k-1}$
- Stirling number of  $2^{nd}$ ,  $n$  人分  $k$  組方法數目
  - $S(0, 0) = S(n, n) = 1$
  - $S(n, 0) = 0$
  - $S(n, k) = kS(n-1, k) + S(n-1, k-1)$
- Bell number,  $n$  人分任意多組方法數目
  - $B_0 = 1$
  - $B_n = \sum_{i=0}^n S(n, i)$
  - $B_{n+1} = \sum_{k=0}^n C_k^n B_k$
  - $B_{p^n} \equiv B_n + B_{n+1} \pmod{p}$ ,  $p$  is prime
  - $B_p^{p^n} \equiv mB_n + B_{n+1} \pmod{p}$ ,  $p$  is prime
  - From  $B_0 : 1, 1, 2, 5, 15, 52, 203, 877, 4140, 21147, 115975$
- Derangement, 錯排，沒有人在自己位置上
  - $D_n = n!(1 - \frac{1}{1!} + \frac{1}{2!} - \frac{1}{3!} + \dots + (-1)^n \frac{1}{n!})$
  - $D_n = (n-1)(D_{n-1} + D_{n-2}), D_0 = 1, D_1 = 0$
  - From  $D_0 : 1, 0, 1, 2, 9, 44, 265, 1854, 14833, 133496$

#### 6. Binomial Equality

- $\sum_k \binom{r}{m+k} \binom{s}{n-k} = \binom{r+s}{m+n}$
- $\sum_k \binom{m+k}{l} \binom{s}{n-k} = \binom{l+s}{l-m+n}$
- $\sum_k \binom{m+k}{l} \binom{s+k}{n} (-1)^k = (-1)^{l+m} \binom{s-m}{n-l}$
- $\sum_{k \leq l} \binom{l-k}{m} \binom{s}{k-n} (-1)^k = (-1)^{l+m} \binom{s-m-1}{l-n-m}$
- $\sum_{0 \leq k \leq l} \binom{l-k}{m} \binom{q+k}{n} = \binom{l+q+1}{m+n+1}$
- $\binom{r}{k} = (-1)^k \binom{k-r-1}{k}$
- $\binom{r}{m} \binom{m}{k} = \binom{r}{k} \binom{m-k}{m-n}$
- $\sum_{k \leq n} \binom{r+k}{n} = \binom{r+n+1}{m+1}$
- $\sum_{0 \leq k \leq n} \binom{k}{m} = \binom{n+1}{m+1}$
- $\sum_{k \leq m} \binom{m+r}{k} x^k y^{m-k} = \sum_{k \leq m} \binom{-r}{k} (-x)^k (x+y)^{m-k}$

#### 11.1.8 冪次, 冪次和

- $a^{b \% P} = a^{b \% \varphi(P) + \varphi(P)}, b \geq \varphi(P)$
- $1^3 + 2^3 + 3^3 + \dots + n^3 = \frac{n^4}{4} + \frac{n^3}{2} + \frac{n^2}{4}$
- $1^4 + 2^4 + 3^4 + \dots + n^4 = \frac{n^5}{5} + \frac{n^4}{2} + \frac{n^3}{3} - \frac{n}{30}$
- $1^5 + 2^5 + 3^5 + \dots + n^5 = \frac{n^6}{6} + \frac{n^5}{2} + \frac{5n^4}{12} - \frac{n^2}{12}$
- $0^k + 1^k + 2^k + \dots + n^k = \frac{(n+1)^{k+1} - \sum_{i=0}^{k-1} C_i^{k+1} P(i)}{k+1}, P(0) = n+1$
- $\sum_{k=0}^{m-1} k^n = \frac{1}{n+1} \sum_{k=0}^n C_k^{n+1} B_k m^{n+1-k}$
- $\sum_{j=0}^m C_j^{m+1} B_j = 0, B_0 = 1$
- 除了  $B_1 = -1/2$ ，剩下的奇數項都是 0
- $B_2 = 1/6, B_4 = -1/30, B_6 = 1/42, B_8 = -1/30, B_{10} = 5/66, B_{12} = -691/2730, B_{14} = 7/6, B_{16} = -3617/510, B_{18} = 43867/798, B_{20} = -174611/330,$



26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38

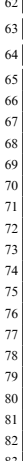
1.  $|X/G| = \frac{1}{|G|} \sum_{g \in G} |X^g|$
2.  $X^g = t^{c(g)}$
3.  $G$  表示有幾種轉法,  $X^g$  表示在那種轉法下, 有幾種是會保持對稱的,  $t$  是顏色數,  $c(g)$  是循環節不動的面數。
4. 正立方體塗三顏色, 轉 0 有  $3^6$  個元素不變, 轉 90 有 6 種, 每種有  $3^3$  不變, 180 有  $3 \times 3^4$ , 120(角) 有  $8 \times 3^2$ , 180(邊) 有  $6 \times 3^3$ , 全部  $\frac{1}{24} (3^6 + 6 \times 3^3 + 3 \times 3^4 + 8 \times 3^2 + 6 \times 3^3) = 57$

40  
41

1. Rooted tree:  $s_{n+1} = \frac{1}{n} \sum_{i=1}^n (i \times a_i \times \sum_{j=1}^{\lfloor n/i \rfloor} a_{n+1-i \times j})$
2. Unrooted tree:
  - (a) Odd:  $a_n - \sum_{i=1}^{n/2} a_i a_{n-i}$
  - (b) Even:  $Odd + \frac{1}{2} a_{n/2} (a_{n/2} + 1)$
3. Spanning Tree
  - (a) 完全圖  $n^n - 2$
  - (b) 一般圖 (Kirchhoff's theorem)  $M[i][i] = \text{degree}(V_i), M[i][j] = -1, \text{ if have } E(i, j), 0 \text{ if no edge. delete any one row and col in } A, \text{ ans} = \det(A)$

57  
58

## 60



# ACM ICPC Team Reference - NCUnknown

## Contents

<b>1 Computational Geometry</b>	<b>1</b>	<b>3 Flow</b>	<b>3</b>	<b>6 Number Theory</b>	<b>8</b>	<b>10 other</b>	<b>11</b>
1.1 Geometry	1	3.1 Gomory Hu	3	6.1 bit set	8	10.1 上下最大正方形	11
1.2 SmallestCircle	3	3.2 ISAP with cut	4	6.2 FFT	8	10.2 WhatDay	12
1.3 最近點對	3	3.3 maxFlow 拷貝	4	6.3 質因數分解	8	10.3 最大矩形	12
<b>2 Data Structure</b>	<b>3</b>	3.4 MinCostMaxFlow	4	6.4 Matrix	9	<b>11 zformula</b>	<b>12</b>
2.1 undo disjoint set	3	3.5 dinic	4	<b>7 String</b>	<b>9</b>	11.1 formula	12
2.2 DLX	3	<b>4 Graph</b>	<b>5</b>	7.1 reverseBWT	9	11.1.1 Pick 公式	12
		4.1 Dijkstra	5	7.2 suffix array lcp	9	11.1.2 圖論	12
		4.2 FloydWarshall	6	7.3 kmp	9	11.1.3 dinic 特殊圖複雜度	12
		4.3 全局最小割	6	7.4 hash	10	11.1.4 0-1 分數規劃	12
		4.4 dsu	6	7.5 AC 自動機	10	11.1.5 學長公式	12
		4.5 spfa	6	7.6 minimal string rotation	10	11.1.6 基本數論	12
		4.6 kruskal	6	7.7 Z	10	11.1.7 排組公式	12
		4.7 bellmanford	7	<b>8 Tarjan</b>	<b>10</b>	11.1.8 冪次, 冪次和	12
		4.8 kruskalMST	7	8.1 橋連通分量	10	11.1.9 Burnside's lemma	13
		4.9 PrimeMST	7	<b>9 Tree Problem</b>	<b>11</b>	11.1.10 Count on a tree	13
		4.10 treeISO	8	9.1 LCA	11	<b>12 Интернационал</b>	<b>13</b>
		4.11 MaximumClique	8	9.2 link cut tree	11	12.1 保佑	13
		<b>5 Linear Programming</b>	<b>8</b>				
		5.1 simplex	8				

# ACM ICPC Judge Test - NCUnknown

## C++ Resource Test

```
1 #include <bits/stdc++.h>
2 using namespace std;
3
4 namespace system_test {
5
6     const size_t KB = 1024;
7     const size_t MB = KB * 1024;
8     const size_t GB = MB * 1024;
9
10    size_t block_size, bound;
11    void stack_size_dfs(size_t depth = 1) {
```

```
12        if (depth >= bound)
13            return;
14        int8_t ptr[block_size]; // 若無法編譯將
15            block_size 改成常數
16        memset(ptr, 'a', block_size);
17        cout << depth << endl;
18        stack_size_dfs(depth + 1);
19    }
20    void stack_size_and_runtime_error(size_t
21        block_size, size_t bound = 1024) {
22        system_test::block_size = block_size;
23        system_test::bound = bound;
24        stack_size_dfs();
25    }
26    double speed(int iter_num) {
27        const int block_size = 1024;
28        volatile int A[block_size];
29        auto begin = chrono::high_resolution_clock
30            ::now();
31        while (iter_num--)
32            for (int j = 0; j < block_size; ++j)
33                A[j] += j;
34        auto end = chrono::high_resolution_clock::
35            now();
36        chrono::duration<double> diff = end -
37            begin;
```

```
38        return diff.count();
39    }
40    void runtime_error_1() {
41        // Segmentation fault
42        int *ptr = nullptr;
43        *(ptr + 7122) = 7122;
44    }
45    void runtime_error_2() {
46        // Segmentation fault
47        int *ptr = (int *)memset;
48        *ptr = 7122;
49    }
50    void runtime_error_3() {
51        // munmap_chunk(): invalid pointer
52        int *ptr = (int *)memset;
53        delete ptr;
54    }
55    void runtime_error_4() {
56        // free(): invalid pointer
57        int *ptr = new int[7122];
58        ptr += 1;
59        delete[] ptr;
60    }
61    }
62
```

```
63    void runtime_error_5() {
64        // maybe illegal instruction
65        int a = 7122, b = 0;
66        cout << (a / b) << endl;
67    }
68    void runtime_error_6() {
69        // floating point exception
70        volatile int a = 7122, b = 0;
71        cout << (a / b) << endl;
72    }
73    void runtime_error_7() {
74        // call to abort.
75        assert(false);
76    }
77    } // namespace system_test
78
79    #include <sys/resource.h>
80    void print_stack_limit() { // only work in
81        Linux
82        struct rlimit l;
83        getrlimit(RLIMIT_STACK, &l);
84        cout << "stack_size = " << l.rlim_cur << "
85            byte" << endl;
86    }
87
```