

1 Computational_Geometry

1.1 Geometry.cpp

```

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};
33template<typename T>
34struct 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 cross(const point<T> &p)const{//點和有向
45        //直線的關係，>0左邊、=0在線上、<0右邊
46        return (p2-p1).cross(p-p1);
47    }
48    bool point_on_segment(const point<T>&p)
49        const{//點是否線段上
50        return cross(p)==0&&(p1-p).dot(p2-p)<=0;
51    }
52    T dis2(const point<T> &p,bool is_segment
53        =0)const{//點跟直線/線段的距離平方
54    point<T> v=p2-p1,v1=p-p1;
55    if(is_segment){
56        point<T> v2=p-p2;
57        if(v.dot(v1)<=0)return v1.abs2();
58        if(v.dot(v2)>=0)return v2.abs2();
59    }
60    T tmp=v.cross(v1);
61    return tmp*tmp/v.abs2();
62}
63
64T seg_dis2(const line<T> &l)const{//兩線段
65    //距離平方
66    return min({dis2(l.p1,l),dis2(l.p2,l),l.
67        dis2(p1,l),l.dis2(p2,l)});
68}
69
70point<T> projection(const point<T> &p)
71    const{//點對直線的投影
72    point<T> n=(p2-p1).normal();
73    return p-n*(p-p1).dot(n)/n.abs2();
74}
75
76point<T> mirror(const point<T> &p)const{
77    //點對直線的鏡射，要先呼叫pton轉成一般式
78    point<T> R;
79    T d=a*b+b*b;
80    R.x=(b*b*p.x-a*a*p.x-2*a*b*p.y-2*a*c)/d;
81    R.y=(a*a*p.y-b*b*p.y-2*a*b*p.x-2*b*c)/d;
82    return R;
83}
84
85bool equal(const line &l)const{//直線相等
86    return cross(l.p1)==0&&cross(l.p2)==0;
87}
88
89bool parallel(const line &l)const{
90    return (p1-p2).cross(l.p1-l.p2)==0;
91}
92
93bool cross_seg(const line &l)const{
94    return (p2-p1).cross(l.p1-p1)*(p2-p1).
95        cross(l.p2-p1)<=0;//直線是否交線段
96}
97
98char line_intersect(const line &l)const{//
99    //直線相交情況，-1無限多點、1交於一點、0
100    //不相交
101    return parallel(l)?(cross(l.p1)==0?-1:0):
102        1;
103}
104
105char seg_intersect(const line &l)const{//
106    //線段相交情況，-1無限多點、1交於一點、0
107    //不相交
108    T c1=(p2-p1).cross(l.p1-p1);
109    T c2=(p2-p1).cross(l.p2-p1);
110    T c3=(l.p2-l.p1).cross(p1-l.p1);
111    T c4=(l.p2-l.p1).cross(p2-l.p1);
112    if(c1==0&&c2==0){
113        if(p1==l.p1&&(p2-p1).dot(l.p2)<=0)
114            return 1;
115        if(p1==l.p2&&(p2-p1).dot(l.p1)<=0)
116            return 1;
117        if(p2==l.p1&&(p1-p2).dot(l.p2)<=0)
118            return 1;
119        if(p2==l.p2&&(p1-p2).dot(l.p1)<=0)
120            return 1;
121        return -1;
122    }else if(c1*c2<=0&&c3*c4<=0)return 1;
123    return 0;
124}
125
126point<T> line_intersection(const line &l)
127    const{//直線交點
128    while(l<r){ //點是否在凸多邊形內，是的話
129        //回傳1、在邊上回傳-1、否則回傳0
130        int mid=(l+r)/2;
131        T a1=(p[mid]-p[0]).cross(x-p[0]);
132        T a2=(p[mid+1]-p[0]).cross(x-p[0]);
133        if(a1>=0&&a2<=0){
134            T res=(p[mid+1]-p[mid]).cross(x-p[
135                mid]);
136            return res>0?1:(res>=0?-1:0);
137        }else if(a1<0)r=mid-1;
138        else l=mid+1;
139    }
140    return 0;
141}
142
143vector<T> getA()const{//凸包邊對x軸的夾角
144    vector<T>res;//一定是遞增的
145    for(size_t i=0;i<p.size();++i)
146        res.push_back((p[(i+1)%p.size()]-p[i]).
147            getA());
148    return res;
149}
150
151bool line_intersect(const vector<T>&A,
152    const line<T> &l)const{//O(LogN)
153    int f1=upper_bound(A.begin(),A.end(),(l.
154        p1-l.p2).getA())-A.begin();
155    int f2=upper_bound(A.begin(),A.end(),(l.
156        p2-l.p1).getA())-A.begin();
157    return l.cross_seg(line<T>(p[f1],p[f2]))
158        ;
159}
160
161polygon cut(const line<T> &l)const{//凸包
162    //對直線切割，得到直線L左側的凸包
163    polygon ans;
164    for(int n=p.size(),i=n-1,j=0;j<n;i=j++){
165        if(1.cross(p[i])>=0){
166            ans.p.push_back(p[i]);
167            if(1.cross(p[j])<0)
168                ans.p.push_back(l.
169                    line_intersection(line<T>(p[i]
170                        ],p[j])));
171        }else if(1.cross(p[j])>0)
172            ans.p.push_back(l.line_intersection(
173                line<T>(p[i],p[j])));
174    }
175    return ans;
176}
177
178static bool graham_cmp(const point<T>&a,
179    const point<T>&b){ //凸包排序函數
180    return (a.x<b.x)||((a.x==b.x&&a.y<b.y));
181}
182
183void graham(vector<point<T>> &s){ //凸包
184    sort(s.begin(),s.end(),graham_cmp);
185    p.resize(s.size()+1);
186    int m=0;
187    for(size_t i=0;i<s.size();++i){
188        while(m>2&&(p[m-1]-p[m-2]).cross(s[i]
189            ]-p[m-2])<=0)--m;
190        p[m++]=s[i];
191    }
192    for(int i=s.size()-2,t=m+1;i>=0;--i){
193        while(m>2&&(p[m-1]-p[m-2]).cross(s[i]
194            ]-p[m-2])<=0)--m;
195        p[m++]=s[i];
196    }
197    T diam()const{//直徑
198        T a1=(p[mid]-p[0]).cross(x-p[0]);
199        T a2=(p[mid+1]-p[0]).cross(x-p[0]);
200        if(a1>=0&&a2<=0){
201            T res=(p[mid+1]-p[mid]).cross(x-p[
202                mid]);
203            return res>0?1:(res>=0?-1:0);
204        }else if(a1<0)r=mid-1;
205        else l=mid+1;
206    }
207    return 0;
208}
209
210vector<point<T>> p; //逆時針順序
211T area()const{//面積
212    T ans=0;
213    for(int i=p.size()-1,j=0;j<(int)p.size()
214        ;i=j++){
215        ans+=p[i].cross(p[j]);
216    }
217    return ans/2;
218}
219
220point<T> center_of_mass()const{//重心
221    T cx=0,cy=0,w=0;
222    for(int i=p.size()-1,j=0;j<(int)p.size()
223        ;i=j++){
224        T a=p[i].cross(p[j]);
225        cx+=(p[i].x+p[j].x)*a;
226        cy+=(p[i].y+p[j].y)*a;
227        w+=a;
228    }
229    return point<T>(cx/3/w,cy/3/w);
230}
231
232char ahas(const point<T>&t)const{//點是否
233    //在簡單多邊形內，是的話回傳1、在邊上回
234    //傳-1、否則回傳0
235    bool c=0;
236    for(int i=0,j=p.size()-1;i<p.size();i=
237        ++j)
238        if(line<T>(p[i],p[j]).point_on_segment
239            (t))return -1;
240    else if((p[i].y>t.y)!=p[j].y>t.y)&&
241        t.x<(p[j].x-p[i].x)*(t.y-p[i].y)/(p[j]
242            .y-p[i].y)+p[i].x)
243        c=!c;
244    return c;
245}
246
247char point_in_convex(const point<T>&x)
248    const{
249    int l=1,r=(int)p.size()-2;
250    while(l<r){ //點是否在凸多邊形內，是的話
251        //回傳1、在邊上回傳-1、否則回傳0
252        int mid=(l+r)/2;
253        T a1=(p[mid]-p[0]).cross(x-p[0]);
254        T a2=(p[mid+1]-p[0]).cross(x-p[0]);
255        if(a1>=0&&a2<=0){
256            T res=(p[mid+1]-p[mid]).cross(x-p[
257                mid]);
258            return res>0?1:(res>=0?-1:0);
259        }else if(a1<0)r=mid-1;
260        else l=mid+1;
261    }
262    return 0;
263}
264
265T diam()const{//直徑
266    T a1=(p[mid]-p[0]).cross(x-p[0]);
267    T a2=(p[mid+1]-p[0]).cross(x-p[0]);
268    if(a1>=0&&a2<=0){
269        T res=(p[mid+1]-p[mid]).cross(x-p[
270            mid]);
271        return res>0?1:(res>=0?-1:0);
272    }else if(a1<0)r=mid-1;
273    else l=mid+1;
274}
275
276vector<T> getA()const{//凸包邊對x軸的夾角
277    vector<T>res;//一定是遞增的
278    for(size_t i=0;i<p.size();++i)
279        res.push_back((p[(i+1)%p.size()]-p[i]).
280            getA());
281    return res;
282}
283
284bool line_intersect(const vector<T>&A,
285    const line<T> &l)const{//O(LogN)
286    int f1=upper_bound(A.begin(),A.end(),(l.
287        p1-l.p2).getA())-A.begin();
288    int f2=upper_bound(A.begin(),A.end(),(l.
289        p2-l.p1).getA())-A.begin();
290    return l.cross_seg(line<T>(p[f1],p[f2]))
291        ;
292}
293
294polygon cut(const line<T> &l)const{//凸包
295    //對直線切割，得到直線L左側的凸包
296    polygon ans;
297    for(int n=p.size(),i=n-1,j=0;j<n;i=j++){
298        if(1.cross(p[i])>=0){
299            ans.p.push_back(p[i]);
300            if(1.cross(p[j])<0)
301                ans.p.push_back(l.
302                    line_intersection(line<T>(p[i]
303                        ],p[j])));
304        }else if(1.cross(p[j])>0)
305            ans.p.push_back(l.line_intersection(
306                line<T>(p[i],p[j])));
307    }
308    return ans;
309}
310
311static bool graham_cmp(const point<T>&a,
312    const point<T>&b){ //凸包排序函數
313    return (a.x<b.x)||((a.x==b.x&&a.y<b.y));
314}
315
316void graham(vector<point<T>> &s){ //凸包
317    sort(s.begin(),s.end(),graham_cmp);
318    p.resize(s.size()+1);
319    int m=0;
320    for(size_t i=0;i<s.size();++i){
321        while(m>2&&(p[m-1]-p[m-2]).cross(s[i]
322            ]-p[m-2])<=0)--m;
323        p[m++]=s[i];
324    }
325    for(int i=s.size()-2,t=m+1;i>=0;--i){
326        while(m>2&&(p[m-1]-p[m-2]).cross(s[i]
327            ]-p[m-2])<=0)--m;
328        p[m++]=s[i];
329    }
330    T diam()const{//直徑
331        T a1=(p[mid]-p[0]).cross(x-p[0]);
332        T a2=(p[mid+1]-p[0]).cross(x-p[0]);
333        if(a1>=0&&a2<=0){
334            T res=(p[mid+1]-p[mid]).cross(x-p[
335                mid]);
336            return res>0?1:(res>=0?-1:0);
337        }else if(a1<0)r=mid-1;
338        else l=mid+1;
339    }
340    return 0;
341}
342
343vector<point<T>> p; //逆時針順序
344T area()const{//面積
345    T ans=0;
346    for(int i=p.size()-1,j=0;j<(int)p.size()
347        ;i=j++){
348        ans+=p[i].cross(p[j]);
349    }
350    return ans/2;
351}
352
353point<T> center_of_mass()const{//重心
354    T cx=0,cy=0,w=0;
355    for(int i=p.size()-1,j=0;j<(int)p.size()
356        ;i=j++){
357        T a=p[i].cross(p[j]);
358        cx+=(p[i].x+p[j].x)*a;
359        cy+=(p[i].y+p[j].y)*a;
360        w+=a;
361    }
362    return point<T>(cx/3/w,cy/3/w);
363}
364
365char ahas(const point<T>&t)const{//點是否
366    //在簡單多邊形內，是的話回傳1、在邊上回
367    //傳-1、否則回傳0
368    bool c=0;
369    for(int i=0,j=p.size()-1;i<p.size();i=
370        ++j)
371        if(line<T>(p[i],p[j]).point_on_segment
372            (t))return -1;
373    else if((p[i].y>t.y)!=p[j].y>t.y)&&
374        t.x<(p[j].x-p[i].x)*(t.y-p[i].y)/(p[j]
375            .y-p[i].y)+p[i].x)
376        c=!c;
377    return c;
378}
379
380char point_in_convex(const point<T>&x)
381    const{
382    int l=1,r=(int)p.size()-2;
383    while(l<r){ //點是否在凸多邊形內，是的話
384        //回傳1、在邊上回傳-1、否則回傳0
385        int mid=(l+r)/2;
386        T a1=(p[mid]-p[0]).cross(x-p[0]);
387        T a2=(p[mid+1]-p[0]).cross(x-p[0]);
388        if(a1>=0&&a2<=0){
389            T res=(p[mid+1]-p[mid]).cross(x-p[
390                mid]);
391            return res>0?1:(res>=0?-1:0);
392        }else if(a1<0)r=mid-1;
393        else l=mid+1;
394    }
395    return 0;
396}
397
398T diam()const{//直徑
399    T a1=(p[mid]-p[0]).cross(x-p[0]);
400    T a2=(p[mid+1]-p[0]).cross(x-p[0]);
401    if(a1>=0&&a2<=0){
402        T res=(p[mid+1]-p[mid]).cross(x-p[
403            mid]);
404        return res>0?1:(res>=0?-1:0);
405    }else if(a1<0)r=mid-1;
406    else l=mid+1;
407}
408
409vector<T> getA()const{//凸包邊對x軸的夾角
410    vector<T>res;//一定是遞增的
411    for(size_t i=0;i<p.size();++i)
412        res.push_back((p[(i+1)%p.size()]-p[i]).
413            getA());
414    return res;
415}
416
417bool line_intersect(const vector<T>&A,
418    const line<T> &l)const{//O(LogN)
419    int f1=upper_bound(A.begin(),A.end(),(l.
420        p1-l.p2).getA())-A.begin();
421    int f2=upper_bound(A.begin(),A.end(),(l.
422        p2-l.p1).getA())-A.begin();
423    return l.cross_seg(line<T>(p[f1],p[f2]))
424        ;
425}
426
427polygon cut(const line<T> &l)const{//凸包
428    //對直線切割，得到直線L左側的凸包
429    polygon ans;
430    for(int n=p.size(),i=n-1,j=0;j<n;i=j++){
431        if(1.cross(p[i])>=0){
432            ans.p.push_back(p[i]);
433            if(1.cross(p[j])<0)
434                ans.p.push_back(l.
435                    line_intersection(line<T>(p[i]
436                        ],p[j])));
437        }else if(1.cross(p[j])>0)
438            ans.p.push_back(l.line_intersection(
439                line<T>(p[i],p[j])));
440    }
441    return ans;
442}
443
444static bool graham_cmp(const point<T>&a,
445    const point<T>&b){ //凸包排序函數
446    return (a.x<b.x)||((a.x==b.x&&a.y<b.y));
447}
448
449void graham(vector<point<T>> &s){ //凸包
450    sort(s.begin(),s.end(),graham_cmp);
451    p.resize(s.size()+1);
452    int m=0;
453    for(size_t i=0;i<s.size();++i){
454        while(m>2&&(p[m-1]-p[m-2]).cross(s[i]
455            ]-p[m-2])<=0)--m;
456        p[m++]=s[i];
457    }
458    for(int i=s.size()-2,t=m+1;i>=0;--i){
459        while(m>2&&(p[m-1]-p[m-2]).cross(s[i]
460            ]-p[m-2])<=0)--m;
461        p[m++]=s[i];
462    }
463    T diam()const{//直徑
464        T a1=(p[mid]-p[0]).cross(x-p[0]);
465        T a2=(p[mid+1]-p[0]).cross(x-p[0]);
466        if(a1>=0&&a2<=0){
467            T res=(p[mid+1]-p[mid]).cross(x-p[
468                mid]);
469            return res>0?1:(res>=0?-1:0);
470        }else if(a1<0)r=mid-1;
471        else l=mid+1;
472    }
473    return 0;
474}
475
476vector<point<T>> p; //逆時針順序
477T area()const{//面積
478    T ans=0;
479    for(int i=p.size()-1,j=0;j<(int)p.size()
480        ;i=j++){
481        ans+=p[i].cross(p[j]);
482    }
483    return ans/2;
484}
485
486point<T> center_of_mass()const{//重心
487    T cx=0,cy=0,w=0;
488    for(int i=p.size()-1,j=0;j<(int)p.size()
489        ;i=j++){
490        T a=p[i].cross(p[j]);
491        cx+=(p[i].x+p[j].x)*a;
492        cy+=(p[i].y+p[j].y)*a;
493        w+=a;
494    }
495    return point<T>(cx/3/w,cy/3/w);
496}
497
498char ahas(const point<T>&t)const{//點是否
499    //在簡單多邊形內，是的話回傳1、在邊上回
500    //傳-1、否則回傳0
501    bool c=0;
502    for(int i=0,j=p.size()-1;i<p.size();i=
503        ++j)
504        if(line<T>(p[i],p[j]).point_on_segment
505            (t))return -1;
506    else if((p[i].y>t.y)!=p[j].y>t.y)&&
507        t.x<(p[j].x-p[i].x)*(t.y-p[i].y)/(p[j]
508            .y-p[i].y)+p[i].x)
509        c=!c;
510    return c;
511}
512
513char point_in_convex(const point<T>&x)
514    const{
515    int l=1,r=(int)p.size()-2;
516    while(l<r){ //點是否在凸多邊形內，是的話
517        //回傳1、在邊上回傳-1、否則回傳0
518        int mid=(l+r)/2;
519        T a1=(p[mid]-p[0]).cross(x-p[0]);
520        T a2=(p[mid+1]-p[0]).cross(x-p[0]);
521        if(a1>=0&&a2<=0){
522            T res=(p[mid+1]-p[mid]).cross(x-p[
523                mid]);
524            return res>0?1:(res>=0?-1:0);
525        }else if(a1<0)r=mid-1;
526        else l=mid+1;
527    }
528    return 0;
529}
530
531T diam()const{//直徑
532    T a1=(p[mid]-p[0]).cross(x-p[0]);
533    T a2=(p[mid+1]-p[0]).cross(x-p[0]);
534    if(a1>=0&&a2<=0){
535        T res=(p[mid+1]-p[mid]).cross(x-p[
536            mid]);
537        return res>0?1:(res>=0?-1:0);
538    }else if(a1<0)r=mid-1;
539    else l=mid+1;
540}
541
542vector<T> getA()const{//凸包邊對x軸的夾角
543    vector<T>res;//一定是遞增的
544    for(size_t i=0;i<p.size();++i)
545        res.push_back((p[(i+1)%p.size()]-p[i]).
546            getA());
547    return res;
548}
549
550bool line_intersect(const vector<T>&A,
551    const line<T> &l)const{//O(LogN)
552    int f1=upper_bound(A.begin(),A.end(),(l.
553        p1-l.p2).getA())-A.begin();
554    int f2=upper_bound(A.begin(),A.end(),(l.
555        p2-l.p1).getA())-A.begin();
556    return l.cross_seg(line<T>(p[f1],p[f2]))
557        ;
558}
559
560polygon cut(const line<T> &l)const{//凸包
561    //對直線切割，得到直線L左側的凸包
562    polygon ans;
563    for(int n=p.size(),i=n-1,j=0;j<n;i=j++){
564        if(1.cross(p[i])>=0){
565            ans.p.push_back(p[i]);
566            if(1.cross(p[j])<0)
567                ans.p.push_back(l.
568                    line_intersection(line<T>(p[i]
569                        ],p[j])));
570        }else if(1.cross(p[j])>0)
571            ans.p.push_back(l.line_intersection(
572                line<T>(p[i],p[j])));
573    }
574    return ans;
575}
576
577static bool graham_cmp(const point<T>&a,
578    const point<T>&b){ //凸包排序函數
579    return (a.x<b.x)||((a.x==b.x&&a.y<b.y));
580}
581
582void graham(vector<point<T>> &s){ //凸包
583    sort(s.begin(),s.end(),graham_cmp);
584    p.resize(s.size()+1);
585    int m=0;
586    for(size_t i=0;i<s.size();++i){
587        while(m>2&&(p[m-1]-p[m-2]).cross(s[i]
588            ]-p[m-2])<=0)--m;
589        p[m++]=s[i];
590    }
591    for(int i=s.size()-2,t=m+1;i>=0;--i){
592        while(m>2&&(p[m-1]-p[m-2]).cross(s[i]
593            ]-p[m-2])<=0)--m;
594        p[m++]=s[i];
595    }
596    T diam()const{//直徑
597        T a1=(p[mid]-p[0]).cross(x-p[0]);
598        T a2=(p[mid+1]-p[0]).cross(x-p[0]);
599        if(a1>=0&&a2<=0){
600            T res=(p[mid+1]-p[mid]).cross(x-p[
601                mid]);
602            return res>0?1:(res>=0?-1:0);
603        }else if(a1<0)r=mid-1;
604        else l=mid+1;
605    }
606    return 0;
607}
608
609vector<point<T>> p; //逆時針順序
610T area()const{//面積
611    T ans=0;
612    for(int i=p.size()-1,j=0;j<(int)p.size()
613        ;i=j++){
614        ans+=p[i].cross(p[j]);
615    }
616    return ans/2;
617}
618
619point<T> center_of_mass()const{//重心
620    T cx=0,cy=0,w=0;
621    for(int i=p.size()-1,j=0;j<(int)p.size()
622        ;i=j++){
623        T a=p[i].cross(p[j]);
624        cx+=(p[i].x+p[j].x)*a;
625        cy+=(p[i].y+p[j].y)*a;
626        w+=a;
627    }
628    return point<T>(cx/3/w,cy/3/w);
629}
630
631char ahas(const point<T>&t)const{//點是否
632    //在簡單多邊形內，是的話回傳1、在邊上回
633    //傳-1、否則回傳0
634    bool c=0;
635    for(int i=0,j=p.size()-1;i<p.size();i=
636        ++j)
637        if(line<T>(p[i],p[j]).point_on_segment
638            (t))return -1;
639    else if((p[i].y>t.y)!=p[j].y>t.y)&&
640        t.x<(p[j].x-p[i].x)*(t.y-p[i].y)/(p[j]
641            .y-p[i].y)+p[i].x)
642        c=!c;
643    return c;
644}
645
646char point_in_convex(const point<T>&x)
647    const{
648    int l=1,r=(int)p.size()-2;
649    while(l<r){ //點是否在凸多邊形內，是的話
650        //回傳1、在邊上回傳-1、否則回傳0
651        int mid=(l+r)/2;
652        T a1=(p[mid]-p[0]).cross(x-p[0]);
653        T a2=(p[mid+1]-p[0]).cross(x-p[0]);
654        if(a1>=0&&a2<=0){
655            T res=(p[mid+1]-p[mid]).cross(x-p[
656                mid]);
657            return res>0?1:(res>=0?-1:0);
658        }else if(a1<0)r=mid-1;
659        else l=mid+1;
660    }
661    return 0;
662}
663
664T diam()const{//直徑
665    T a1=(p[mid]-p[0]).cross(x-p[0]);
666    T a2=(p[mid+1]-p[0]).cross(x-p[0]);
667    if(a1>=0&&a2<=0){
668        T res=(p[mid+1]-p[mid]).cross(x-p[
669            mid]);
670        return res>0?1:(res>=0?-1:0);
671    }else if(a1<0)r=mid-1;
672    else l=mid+1;
673}
674
675vector<T> getA()const{//凸包邊對x軸的夾角
676    vector<T>res;//一定是遞增的
677    for(size_t i=0;i<p.size();++i)
678        res.push_back((p[(i+1)%p.size()]-p[i]).
679            getA());
680    return res;
681}
682
683bool line_intersect(const vector<T>&A,
684    const line<T> &l)const{//O(LogN)
685    int f1=upper_bound(A.begin(),A.end(),(l.
686        p1-l.p2).getA())-A.begin();
687    int f2=upper_bound(A.begin(),A.end(),(l.
688        p2-l.p1).getA())-A.begin();
689    return l.cross_seg(line<T>(p[f1],p[f2]))
690        ;
691}
692
693polygon cut(const line<T> &l)const{//凸包
694    //對直線切割，得到直線L左側的凸包
695    polygon ans;
696    for(int n=p.size(),i=n-1,j=0;j<n;i=j++){
697        if(1.cross(p[i])>=0){
698            ans.p.push_back(p[i]);
699            if(1.cross(p[j])<0)
700                ans.p.push_back(l.
701                    line_intersection(line<T>(p[i]
702                        ],p[j])));
703        }else if(1.cross(p[j])>0)
704            ans.p.push_back(l.line_intersection(
705                line<T>(p[i],p[j])));
706    }
707    return ans;
708}
709
710static bool graham_cmp(const point<T>&a,
711    const point<T>&b){ //凸包排序函數
712    return (a.x<b.x)||((a.x==b.x&&a.y<b.y));
713}
714
715void graham(vector<point<T>> &s){ //凸包
716    sort(s.begin(),s.end(),graham_cmp);
717    p.resize(s.size()+1);
718    int m=0;
719    for(size_t i=0;i<s.size();++i){
720        while(m>2&&(p[m-1]-p[m-2]).cross(s[i]
721            ]-p[m-2])<=0)--m;
722        p[m++]=s[i];
723    }
724    for(int i=s.size()-2,t=m+1;i>=0;--i){
725        while(m>2&&(p[m-1]-p[m-2]).cross(s[i]
726            ]-p[m-2])<=0)--m;
727        p[m++]=s[i];
728    }
729    T diam()const{//直徑
730        T a1=(p[mid]-p[0]).cross(x-p[0]);
731        T a2=(p[mid+1]-p[0]).cross(x-p[0]);
732        if(a1>=0&&a2<=0){
733            T res=(p[mid+1]-p[mid]).cross(x-p[
734                mid]);
735            return res>0?1:(res>=0?-1:0);
736        }else if(a1<0)r=mid-1;
737        else l=mid+1;
738    }
739    return 0;
740}
741
742vector<point<T>> p; //逆時針順序
743T area()const{//面積
744    T ans=0;
745    for(int i=p.size()-1,j=0;j<(int)p.size()
746        ;i=j++){
747        ans+=p[i].cross(p[j]);
748    }
749    return ans/2;
750}
751
752point<T> center_of_mass()const{//重心
753    T cx=0,cy=0,w=0;
754    for(int i=p.size()-1,j=0;j<(int)p.size()
755        ;i=j++){
756        T a=p[i].cross(p[j]);
757        cx+=(p[i].x+p[j].x)*a;
758        cy+=(p[i].y+p[j].y)*a;
759        w+=a;
760    }
761    return point<T>(cx/3/w,cy/3/w);
762}
763
764char ahas(const point<T>&t)const{//點是否
765    //在簡單多邊形內，是的話回傳1、在邊上回
766    //傳-1、否則回傳0
767    bool c=0;
768    for(int i=0,j=p.size()-1;i<p.size();i=
769        ++j)
770        if(line<T>(p[i],p[j]).point_on_segment
771            (t))return -1;
772    else if((p[i].y>t.y)!=p[j].y>t.y)&&
773        t.x<(p[j].x-p[i].x)*(t.y-p[i].y)/(p[j]
774            .y-p[i].y)+p[i].x)
775        c=!c;
776    return c;
777}
778
779char point_in_convex(const point<T>&x)
780    const{
781    int l=1,r=(int)p.size()-2;
782    while(l<r){ //點是否在凸多邊形內，是的話
783        //回傳1、在邊上回傳-1、否則回傳0
784        int mid=(l+r)/2;
785        T a1=(p[mid]-p[0]).cross(x-p[0]);
786        T a2=(p[mid+1]-p[0]).cross(x-p[0]);
787        if(a1>=0&&a2<=0){
788            T res=(p[mid+1]-p[mid]).cross(x-p[
789                mid]);
790            return res>0?1:(res>=0?-1:0);
791        }else if(a1<0)r=mid-1;
792        else l=mid+1;
793    }
794    return 0;
795}
796
797T diam()const{//直徑
798    T a1=(p[mid]-p[0]).cross(x-p[0]);
799    T a2=(p[mid+1]-p[0]).cross(x-p[0]);
800    if(a1>=0&&a2<=0){
801        T res=(p[mid+1]-p[mid]).cross(x-p[
802            mid]);
803        return res>0?1:(res>=0?-1:0);
804    }else if(a1<0)r=mid-1;
805    else l=mid+1;
806}
807
808vector<T> getA()const{//凸包邊對x軸的夾角
809    vector<T>res;//一定是遞增的
810    for(size_t i=0;i<p.size();++i)
811        res.push_back((p[(i+1)%p.size()]-p[i]).
812            getA());
813    return res;
814}
815
816bool line_intersect(const vector<T>&A,
817    const line<T> &l)const{//O(LogN)
818    int f1=upper_bound(A.begin(),A.end(),(l.
819        p1-l.p2).getA())-A.begin();
820    int f2=upper_bound(A.begin(),A.end(),(l.
821        p2-l.p1).getA())-A.begin();
822    return l.cross_seg(line<T>(p[f1],p[f2]))
823        ;
824}
825
826polygon cut(const line<T> &l)const{//凸包
827    //對直線切割，得到直線L左側的凸包
828    polygon ans;
829    for(int n=p.size(),i=n-1,j=0;j<n;i=j++){
830        if(1.cross(p[i])>=0){
831            ans.p.push_back(p[i]);
832            if(1.cross(p[j])<0)
833                ans.p.push_back(l.
834                    line_intersection(line<T>(p[i]
835                        ],p[j])));
836        }else if(1.cross(p[j])>0)
837            ans.p.push_back(l.line_intersection(
838                line<T>(p[i],p[j])));
839    }
840    return ans;
841}
842
843static bool graham_cmp(const point<T>&a,
844    const point<T>&b){ //凸包排序函數
845    return (a.x<b.x)||((a.x==b.x&&a.y<b.y));
846}
```

```

205 int n=p.size(),t=1;
206 T ans=0;p.push_back(p[0]);
207 for(int i=0;i<n;i++){
208     point<T> now=p[i+1]-p[i];
209     while(now.cross(p[t+1]-p[i])>now.cross
210         (p[t]-p[i]))t=(t+1)%n;
211     ans=max(ans,max((p[i]-p[t]).abs2()),(p[
212         i+1]-p[t+1]).abs2()));
213 }
214 return p.pop_back(),ans;
215 }
216 min_cover_rectangle(){//最小覆蓋矩形
217 int n=p.size(),t=1,r=1,l;
218 if(n<3)return 0;//也可以做最小周長矩形
219 T ans=1e99;p.push_back(p[0]);
220 for(int i=0;i<n;i++){
221     point<T> now=p[i+1]-p[i];
222     while(now.cross(p[t+1]-p[i])>now.cross
223         (p[t]-p[i]))t=(t+1)%n;
224     while(now.dot(p[r+1]-p[i])>now.dot(p[r
225         ]-p[i]))r=(r+1)%n;
226     if(!l)l=r;
227     while(now.dot(p[l+1]-p[i])<now.dot(p[
228         l]-p[i]))l=(l+1)%n;
229     T d=now.abs2();
230     T tmp=now.cross(p[t]-p[i])*(now.dot(p[
231         r]-p[i])-now.dot(p[l]-p[i]))/d;
232     ans=min(ans,tmp);
233 }
234 return p.pop_back(),ans;
235 }
236 max_triangle(){//最大內接三角形
237 int n=p.size(),a=1,b=2;
238 if(n<3)return 0;
239 T ans=0,tmp;p.push_back(p[0]);
240 for(int i=0;i<n;++i){
241     while((p[a]-p[i]).cross(p[b+1]-p[i])>
242         tmp=(p[a]-p[i]).cross(p[b]-p[i]))
243         b=(b+1)%n;
244     ans=max(ans,tmp);
245     while((p[a+1]-p[i]).cross(p[b]-p[i])>
246         tmp=(p[a]-p[i]).cross(p[b]-p[i]))
247         a=(a+1)%n;
248     ans=max(ans,tmp);
249 }
250 return p.pop_back(),ans/2;
251 }
252 dis2(polygon &p1){//凸包最近距離平方
253 vector<point<T>> &P=p,&Q=p1.p;
254 int n=P.size(),m=Q.size(),l=0,r=0;
255 for(int i=0;i<n;++i)if(P[i].y<P[l].y)l=i;
256 for(int i=0;i<m;++i)if(Q[i].y<Q[r].y)r=i;
257 P.push_back(P[0]),Q.push_back(Q[0]);
258 T ans=1e99;
259 for(int i=0;i<n;++i){
260     while((P[l]-P[l+1]).cross(Q[r+1]-Q[r])
261         <0)r=(r+1)%m;
262     ans=min(ans,line<T>(P[l],P[l+1]).
263         seg_dis2(line<T>(Q[r],Q[r+1])));
264     l=(l+1)%n;
265 }
266 return P.pop_back(),Q.pop_back(),ans;
267 }
268 static char sign(const point<T>&t){
269     return (t.y==0?t.x:t.y)<0;
270 }
271 static bool angle_cmp(const line<T>& A,
272     const line<T>& B){
273     point<T> a=A.p2-A.p1,b=B.p2-B.p1;
274     return sign(a)<sign(b)||((sign(a)==sign(b)
275         )&&a.cross(b)>0);
276 }
277 int halfplane_intersection(vector<line<T>
278     > &s){//半平面交
279     sort(s.begin(),s.end(),angle_cmp);//線段
280     //左側為該線段半平面
281     int L,R,n=s.size();
282     vector<point<T>> > px(n);
283     vector<line<T>> > q(n);
284     q[L=R=0]=s[0];
285     for(int i=1;i<n;++i){
286         while(L<R&&s[i].cross(px[R-1])<=0)--R;
287         while(L<R&&s[i].cross(px[L])<=0)++L;
288         q[++R]=s[i];
289         if(q[R].parallel(q[R-1])){
290             --R;
291             if(q[R].cross(s[i].p1)>0)q[R]=s[i];
292         }
293         if(L<R)px[R-1]=q[R-1].
294             line_intersection(q[R]);
295     }
296     while(L<R&&q[L].cross(px[R-1])<=0)--R;
297     p.clear();
298     if(R-L<=1)return 0;
299     px[R]=q[R].line_intersection(q[L]);
300     for(int i=L;i<R;++i)p.push_back(px[i]);
301     return R-L+1;
302 }
303 template<typename T>
304 struct triangle{
305     point<T> a,b,c;
306     triangle(){
307         triangle(const point<T> &a,const point<T>
308             &b,const point<T> &c):a(a),b(b),c(c){}
309     }
310     T area()const{
311         T t=(b-a).cross(c-a)/2;
312         return t>0?t:-t;
313     }
314     point<T> barycenter()const{//重心
315         return (a+b+c)/3;
316     }
317     point<T> circumcenter()const{//外心
318         static line<T> u,v;
319         u.p1=(a+b)/2;
320         u.p2=point<T>(u.p1.x-a.y+b.y,u.p1.y+a.x-
321             b.x);
322         v.p1=(a+c)/2;
323         v.p2=point<T>(v.p1.x-a.y+c.y,v.p1.y+a.x-
324             c.x);
325         return u.line_intersection(v);
326     }
327     point<T> incenter()const{//內心
328         T A=sqrt((b-c).abs2()),B=sqrt((a-c).abs2
329             ()),C=sqrt((a-b).abs2());
330         return point<T>(A*a.x+B*b.x+C*c.x,A*a.y+
331             B*b.y+C*c.y)/(A+B+C);
332     }
333     point<T> perpencer()const{//垂心
334         return barycenter()*3-circumcenter()*2;
335     }
336 }
337 template<typename T>
338 struct point3D{
339     T x,y,z;
340     point3D(){
341         point3D(const T&x,const T&y,const T&z):x(x
342             ),y(y),z(z){}
343     }
344     point3D operator+(const point3D &b)const{
345         return point3D(x+b.x,y+b.y,z+b.z);
346     }
347     point3D operator-(const point3D &b)const{
348         return point3D(x-b.x,y-b.y,z-b.z);
349     }
350     point3D operator*(const T &b)const{
351         return point3D(x*b,y*b,z*b);
352     }
353     point3D operator/(const T &b)const{
354         return point3D(x/b,y/b,z/b);
355     }
356     bool operator==(const point3D &b)const{
357         return x==b.x&&y==b.y&&z==b.z;
358     }
359     T dot(const point3D &b)const{
360         return x*b.x+y*b.y+z*b.z;
361     }
362     point3D cross(const point3D &b)const{
363         return point3D(y*b.z-z*b.y,z*b.x-x*b.z,x
364             *b.y-y*b.x);
365     }
366     T abs2()const{//向量长度的平方
367         return dot(*this);
368     }
369     T area2(const point3D &b)const{//和b、原點
370         //圍成面積的平方
371         return cross(b).abs2()/4;
372     }
373 };
374 template<typename T>
375 struct line3D{
376     point3D<T> p1,p2;
377     line3D(){
378         line3D(const point3D<T> &p1,const point3D<
379             T> &p2):p1(p1),p2(p2){}
380     }
381     T dis2(const point3D<T> &p,bool is_segment
382         =0)const{//點跟直線/線段的距離平方
383         point3D<T> v=p2-p1,v1=p-p1;
384         if(is_segment){
385             point3D<T> v2=p-p2;
386             if(v.dot(v1)<=0)return v1.abs2();
387             if(v.dot(v2)>=0)return v2.abs2();
388         }
389         point3D<T> tmp=v.cross(v1);
390         return tmp.abs2()/v.abs2();
391     }
392     pair<point3D<T>,point3D<T>> closest_pair(
393         const line3D<T> &l1)const{
394         point3D<T> v1=(p1-p2),v2=(l1.p1-l1.p2);
395         point3D<T> N=v1.cross(v2),ab(p1-l1.p1);
396         //if(N.abs2()==0)return NULL;平行或重合
397         T tmp=N.dot(ab),ans=tmp*tmp/N.abs2();//
398         //最近點距離
399         point3D<T> d1=p2-p1,d2=l1.p1-l1.p1,D=d1.
400             cross(d2),G=l1.p1-p1;
401         T t1=(G.cross(d2)).dot(D)/D.abs2();
402         T t2=(G.cross(d1)).dot(D)/D.abs2();
403         return make_pair(p1+d1*t1,l1.p1+d2*t2);
404     }
405     bool same_side(const point3D<T> &a,const
406         point3D<T> &b)const{
407         return (p2-p1).cross(a-p1).dot((p2-p1).
408             cross(b-p1))>0;
409     }
410 };
411 template<typename T>
412 struct plane{
413     point3D<T> p0,n;//平面上的點和法向量
414     plane(){
415         plane(const point3D<T> &p0,const point3D<T>
416             &n):p0(p0),n(n){}
417     }
418     T dis2(const point3D<T> &p)const{//點到平
419         //面距離的平方
420     }
421     T tmp=(p-p0).dot(n);
422     return tmp*tmp/n.abs2();
423 }
424 point3D<T> projection(const point3D<T> &p)
425     const{
426     return p-n*(p-p0).dot(n)/n.abs2();
427 }
428 point3D<T> line_intersection(const line3D<T>
429     &l1,const
430     T tmp=n.dot(l1.p2-l1.p1);//等於0表示平行或
431     //重合該平面
432     return l1.p1+(l1.p2-l1.p1)*(n.dot(p0-l1.p1)/
433         tmp);
434 }
435 line3D<T> plane_intersection(const plane &
436     pl1)const{
437     point3D<T> e=n.cross(pl1.n),v=n.cross(e);
438     T tmp=pl1.n.dot(v);//等於0表示平行或重合
439     //該平面
440     point3D<T> q=p0+(v*(pl1.n.dot(pl1.p0-p0))/
441         tmp);
442     return line3D<T>(q,q+e);
443 }
444 template<typename T>
445 struct triangle3D{
446     point3D<T> a,b,c;
447     triangle3D(){
448         triangle3D(const point3D<T> &a,const
449             point3D<T> &b,const point3D<T> &c):a(a
450             ),b(b),c(c){}
451     }
452     bool point_in(const point3D<T> &p)const{//
453         //點在該平面上的投影在三角形中
454         return line3D<T>(b,c).same_side(p,a)&&
455             line3D<T>(a,c).same_side(p,b)&&
456             line3D<T>(a,b).same_side(p,c);
457     }
458 }
459 };
460 template<typename T>
461 struct tetrahedron{//四面體
462     point3D<T> a,b,c,d;
463     tetrahedron(){
464         tetrahedron(const point3D<T> &a,const
465             point3D<T> &b,const point3D<T> &c,
466             const point3D<T> &d):a(a),b(b),c(c),d(
467             d){}
468     }
469     T volume6()const{//體積的六倍
470         return (d-a).dot((b-a).cross(c-a));
471     }
472     point3D<T> centroid()const{
473         return (a+b+c+d)/4;
474     }
475     bool point_in(const point3D<T> &p)const{
476         return triangle3D<T>(a,b,c).point_in(p)
477             &&triangle3D<T>(c,d,a).point_in(p);
478     }
479 }

```

```

413 }
414 };
415 template<typename T>
416 struct convexhull3D{
417     static const int MAXN=1005;
418     struct face{
419         int a,b,c;
420         face(int a,int b,int c):a(a),b(b),c(c){}
421     };
422     vector<point3D<T>> pt;
423     vector<face> ans;
424     int fid[MAXN][MAXN];
425     void build(){
426         int n=pt.size();
427         ans.clear();
428         memset(fid,0,sizeof(fid));
429         ans.emplace_back(0,1,2);
430         ans.emplace_back(2,1,0);
431         int ftop = 0;
432         for(int i=3, ftop=1; i<n; ++i,++ftop){
433             vector<face> next;
434             for(auto &f:ans){
435                 T d=(pt[i]-pt[f.a]).dot((pt[f.b]-pt[f.a]).cross(pt[f.c]-pt[f.a]));
436                 if(d==0) next.push_back(f);
437                 int ff=0;
438                 if(d>0) ff=ftop;
439                 else if(d<0) ff=-ftop;
440                 fid[f.a][f.b]=fid[f.b][f.c]=fid[f.c][f.a]=ff;
441             }
442             for(auto &f:ans){
443                 if(fid[f.a][f.b]>0 && fid[f.a][f.b]!=fid[f.b][f.a])
444                     next.emplace_back(f.a,f.b,i);
445                 if(fid[f.b][f.c]>0 && fid[f.b][f.c]!=fid[f.c][f.b])
446                     next.emplace_back(f.b,f.c,i);
447                 if(fid[f.c][f.a]>0 && fid[f.c][f.a]!=fid[f.a][f.c])
448                     next.emplace_back(f.c,f.a,i);
449             }
450             ans=next;
451         }
452     }
453     point3D<T> centroid()const{
454         point3D<T> res(0,0,0);
455         T vol=0;
456         for(auto &f:ans){
457             T tmp=pt[f.a].dot(pt[f.b].cross(pt[f.c]));
458             res=res+(pt[f.a]+pt[f.b]+pt[f.c])*tmp;
459             vol+=tmp;
460         }
461         return res/(vol*4);
462     }
463 };

```

1.2 SmallestCircle.cpp

```

1 #include "Geometry.cpp"
2 struct Circle{
3     typedef point<double> p;

```

```

4     typedef const point<double> cp;
5     p x;
6     double r2;
7     bool incircle(cp &c)const{return (x-c).abs2()<=r2;};
8 };
9 Circle TwoPointCircle(Circle::cp &a, Circle::cp &b) {
10     Circle::p m=(a+b)/2;
11     return (Circle){m,(a-m).abs2()};
12 }
13 Circle outcircle(Circle::p a, Circle::p b, Circle::p c) {
14     if(TwoPointCircle(a,b).incircle(c))
15         return TwoPointCircle(a,b);
16     if(TwoPointCircle(b,c).incircle(a))
17         return TwoPointCircle(b,c);
18     if(TwoPointCircle(c,a).incircle(b))
19         return TwoPointCircle(c,a);
20     Circle::p ret;
21     double a1=b.x-a.x, b1=b.y-a.y, c1=(a1*a1+b1*b1)/2;
22     double a2=c.x-a.x, b2=c.y-a.y, c2=(a2*a2+b2*b2)/2;
23     double d = a1*b2 - a2*b1;
24     ret.x=a.x+(c1*b2-c2*b1)/d;
25     ret.y=a.y+(a1*c2-a2*c1)/d;
26     return (Circle){ret,(ret-a).abs2()};
27 }
28 Circle SmallestCircle(std::vector<Circle::p> &p){
29     int n=p.size();
30     if(n==1) return (Circle){p[0],0.0};
31     if(n==2) return TwoPointCircle(p[0],p[1]);
32     random_shuffle(p.begin(),p.end());
33     Circle c = {p[0],0.0};
34     for(int i=0;i<n;++i){
35         if(c.incircle(p[i])) continue;
36         c=Circle{p[i],0.0};
37         for(int j=0;j<i;++j){
38             if(c.incircle(p[j])) continue;
39             c=TwoPointCircle(p[i],p[j]);
40             for(int k=0;k<j;++k){
41                 if(c.incircle(p[k])) continue;
42                 c=outcircle(p[i],p[j],p[k]);
43             }
44         }
45     }
46     return c;
47 }

```

1.3 最近點對.cpp

```

1 #define INF LLONG_MAX
2 template<typename T>
3 T closest_pair(vector<point<T>> &v,vector<point<T>> &t,int l,int r){
4     T dis=INF, tmd;

```

```

5     if(l==r)return dis;
6     int mid=(l+r)/2;
7     if((tmd=closest_pair(v,t,l,mid)<dis)dis=tmd;
8     if((tmd=closest_pair(v,t,mid+1,r)<dis)dis=tmd;
9     t.clear();
10    for(int i=l;i<=r;++i)
11        if((v[i].x-v[mid].x)*(v[i].x-v[mid].x)<dis)t.push_back(v[i]);
12    sort(t.begin(),t.end(),point<T>::y_cmp);
13    // 如果用merge_sort的方式可以O(n)
14    for(size_t i=0;i<t.size();++i)
15        for(size_t j=1;j<=3&&i+j<t.size();++j)
16            if((tmd=(t[i]-t[i+j]).abs2())<dis)dis=tmd;
17    return dis;
18 }
19 template<typename T>
20 inline T closest_pair(vector<point<T>> &v){
21     vector<point<T>> t;
22     sort(v.begin(),v.end(),point<T>::x_cmp);
23     return closest_pair(v,t,0,v.size()-1);
24 }
25 // 最近點對距離

```

2 Data_Structure

2.1 DLX.cpp

```

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]; // 每個節點代表的列跟行
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
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())>=ansdreturn 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())>=ansdreturn 0;
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 };

```

2.2 Dynamic_KD_tree.cpp

```

1 template<typename T,size_t kd>//有kd個維度
2 struct kd_tree{
3     struct point{
4         T d[kd];
5         T dist(const point &x)const{
6             T ret=0;
7             for(size_t i=0;i<kd;++i)ret+=std::abs(
8                 d[i]-x.d[i]);
9             return ret;
10        }
11        bool operator==(const point &p){
12            for(size_t i=0;i<kd;++i)
13                if(d[i]!=p.d[i])return 0;
14            return 1;
15        }
16        bool operator<(const point &b)const{
17            return d[0]<b.d[0];
18        }
19    private:
20        struct node{
21            node *l,*r;
22            point pid;
23            int s;
24            node(const point &p):l(0),r(0),pid(p),s
25                (1){}
26            ~node(){delete l;delete r;}
27            void up(){s=(l?l->s:0)+1+(r?r->s:0);}
28        }*root;
29        const double alpha,loga;
30        const T INF;//記得要給INF·表示極大值
31        int maxn;
32        struct __cmp{
33            int sort_id;
34            bool operator()(const node*x,const node*
35                y)const{
36                return operator()(x->pid,y->pid);
37            }
38            bool operator()(const point &x,const
39                point &y)const{
40                if(x.d[sort_id]!=y.d[sort_id])
41                    return x.d[sort_id]<y.d[sort_id];
42                for(size_t i=0;i<kd;++i)
43                    if(x.d[i]!=y.d[i])return x.d[i]<y.d[
44                        i];
45                return 0;
46            }
47        }cmp;
48        int size(node *o){return o?o->s:0;}
49        std::vector<node*> A;

```

```

46 node* build(int k,int l,int r){
47     if(l>r) return 0;
48     if(k==kd) k=0;
49     int mid=(l+r)/2;
50     cmp.sort_id = k;
51     std::nth_element(A.begin()+l,A.begin()+
52         mid,A.begin()+r+1,cmp);
53     node *ret=A[mid];
54     ret->l = build(k+1,l,mid-1);
55     ret->r = build(k+1,mid+1,r);
56     ret->up();
57     return ret;
58 }
59 bool isbad(node*o){
60     return size(o->l)>alpha*o->s||size(o->r)
61         >alpha*o->s;
62 }
63 void flatten(node *u,typename std::vector<
64     node*>::iterator &it){
65     if(!u)return;
66     flatten(u->l,it);
67     *it=u;
68     flatten(u->r,++it);
69 }
70 void rebuild(node*&u,int k){
71     if((int)A.size()<u->s)A.resize(u->s);
72     typename std::vector<node*>::iterator it
73         =A.begin();
74     flatten(u,it);
75     u=build(k,0,u->s-1);
76 }
77 bool insert(node*&u,int k,const point &x,
78     int dep){
79     if(!u) return u=new node(x), dep<=0;
80     ++u->s;
81     cmp.sort_id=k;
82     if(insert(cmp(x,u->pid)?u->l:u->r,(k+1)%
83         kd,x,dep-1)){
84         if(!isbad(u))return 1;
85         rebuild(u,k);
86     }
87     return 0;
88 }
89 node *findmin(node*o,int k){
90     if(!o)return 0;
91     if(cmp.sort_id==k)return o->l?findmin(o
92         ->l,(k+1)%kd):o;
93     node *l=findmin(o->l,(k+1)%kd);
94     node *r=findmin(o->r,(k+1)%kd);
95     if(l&&!r)return cmp(l,o)?l:o;
96     if(!l&&r)return cmp(r,o)?r:o;
97     if(!l&&!r)return 0;
98     if(cmp(l,r))return cmp(l,o)?l:o;
99     return cmp(r,o)?r:o;
100 }
101 bool erase(node *&u,int k,const point &x){
102     if(!u)return 0;
103     if(u->pid==x){
104         if(u->r){
105             u->l = u->r;
106             u->r = 0;
107             return 1;
108         }
109         else{
110             delete u;
111             return u=0, 1;
112         }
113     }
114     --u->s;
115     cmp.sort_id=k;

```

```

105     u->pid=findmin(u->r,(k+1)%kd)->pid;
106     return erase(u->r,(k+1)%kd,u->pid);
107 }
108 cmp.sort_id=k;
109 if(erase(cmp(x,u->pid)?u->l:u->r,(k+1)%
110     kd,x)){
111     return --u->s, 1;
112     return 0;
113 }
114 T heuristic(const T h[])const{
115     T ret=0;
116     for(size_t i=0;i<kd;++i)ret+=h[i];
117     return ret;
118 }
119 int qM;
120 std::priority_queue<std::pair<T,point > >
121     pQ;
122 void nearest(node *u,int k,const point &x,
123     T *h,T &mndist){
124     if(u==0||heuristic(h)>mndist)return;
125     T dist=u->pid.dist(x,old=h[k]);
126     /*mndist=std::min(mndist,dist);*/
127     if(dist<mndist){
128         pQ.push(std::make_pair(dist,u->pid));
129         if((int)pQ.size()==qM+1)
130             mndist=pQ.top().first,pQ.pop();
131     }
132     if(x.d[k]<u->pid.d[k]){
133         nearest(u->l,(k+1)%kd,x,h,mndist);
134         h[k]=std::abs(x.d[k]-u->pid.d[k]);
135         nearest(u->r,(k+1)%kd,x,h,mndist);
136     }
137     else{
138         nearest(u->r,(k+1)%kd,x,h,mndist);
139         h[k]=std::abs(x.d[k]-u->pid.d[k]);
140         nearest(u->l,(k+1)%kd,x,h,mndist);
141     }
142     h[k]=old;
143 }
144 std::vector<point>in_range;
145 void range(node *u,int k,const point&mi,
146     const point&ma){
147     if(!u)return;
148     bool is=1;
149     for(int i=0;i<kd;++i)
150         if(u->pid.d[i]<mi.d[i]||ma.d[i]<u->pid
151             .d[i]){
152             is=0;break;
153         }
154     if(is)in_range.push_back(u->pid);
155     if(mi.d[k]<u->pid.d[k])range(u->l,(k+1)
156         %kd,mi,ma);
157     if(ma.d[k]>u->pid.d[k])range(u->r,(k+1)
158         %kd,mi,ma);
159 }
160 public:
161 kd_tree(const T &INF,double a=0.75):root
162     (0),alpha(a),loga(log2(1.0/a)),INF(INF
163     ),maxn(1){}
164 ~kd_tree(){delete root;}
165 void clear(){delete root;root=0,maxn=1;}
166 void build(int n,const point *p){
167     delete root,A.resize(maxn=n);
168     for(int i=0;i<n;++i)A[i]=new node(p[i]);
169     root=build(0,0,n-1);
170 }
171 void insert(const point &x){

```

2.3 kd_tree_replace_segment_tr

```

1 /*kd樹代替高維線段樹*/
2 struct node{
3     node *l,*r;
4     point pid,mi,ma;
5     int s;
6     int data;
7     node(const point &p,int d):l(0),r(0),pid(p
8         ),mi(p),ma(p),s(1),data(d),dmin(d),
9         dmax(d){}
10 void up(){
11     mi=ma=pid;
12     s=1;
13     if(l){
14         for(int i=0;i<kd;++i){
15             mi.d[i]=min(mi.d[i],l->mi.d[i]);
16             ma.d[i]=max(ma.d[i],l->ma.d[i]);
17         }
18     }
19     if(r){
20         for(int i=0;i<kd;++i){
21             mi.d[i]=min(mi.d[i],r->mi.d[i]);
22             ma.d[i]=max(ma.d[i],r->ma.d[i]);
23         }
24     }
25     s+=r->s;
26 }
27 void up2(){
28     //其他懶惰標記向上更新

```



```

28 }
29 void down(){
30     //其他懶惰標記下推
31 }
32 }*root;
33
34 /*檢查區間包含用的函數*/
35 inline bool range_include(node *o,const
    point &L,const point &R){
36     for(int i=0;i<kd;++i){
37         if(L.d[i]>o->ma.d[i]||R.d[i]<o->mi.d[i])
38             return 0;
39     }//只要(L,R)區間有和o的區間有交集就回傳
40     true
41     return 1;
42 }
43 inline bool range_in_range(node *o,const
    point &L,const point &R){
44     for(int i=0;i<kd;++i){
45         if(L.d[i]>o->mi.d[i]||o->ma.d[i]>R.d[i])
46             return 0;
47     }//如果(L,R)區間完全包含o的區間就回傳true
48     return 1;
49 }
50 inline bool point_in_range(node *o,const
    point &L,const point &R){
51     for(int i=0;i<kd;++i){
52         if(L.d[i]>o->pid.d[i]||R.d[i]<o->pid.d[i])
53             return 0;
54     }//如果(L,R)區間完全包含o->pid這個點就回傳
55     true
56     return 1;
57 }
58
59 /*單點修改 · 以單點改值為例*/
60 void update(node *u,const point &x,int data,
    int k=0){
61     if(!u)return;
62     u->down();
63     if(u->pid==x){
64         u->data=data;
65         u->up2();
66         return;
67     }
68     cmp.sort_id=k;
69     update(cmp(x,u->pid)?u->l:u->r,x,data,(k
        +1)%kd);
70     u->up2();
71 }
72
73 /*區間修改*/
74 void update(node *o,const point &L,const
    point &R,int data){
75     if(!o)return;
76     o->down();
77     if(range_in_range(o,L,R)){
78         //區間懶惰標記修改
79         o->down();
80         return;
81     }
82     if(point_in_range(o,L,R)){
83         //這個點在(L,R)區間 · 但是他的左右子樹不
84         一定在區間中
85         //單點懶惰標記修改

```

```

80 }
81 if(o->l&&range_include(o->l,L,R))update(o
    ->l,L,R,data);
82 if(o->r&&range_include(o->r,L,R))update(o
    ->r,L,R,data);
83 o->up2();
84 }
85
86 /*區間查詢 · 以總和為例*/
87 int query(node *o,const point &L,const point
    &R){
88     if(!o)return 0;
89     o->down();
90     if(range_in_range(o,L,R))return o->sum;
91     int ans=0;
92     if(point_in_range(o,L,R))ans+=o->data;
93     if(o->l&&range_include(o->l,L,R))ans+=
        query(o->l,L,R);
94     if(o->r&&range_include(o->r,L,R))ans+=
        query(o->r,L,R);
95     return ans;
96 }

```

2.4 reference_point.cpp

```

1 template<typename T>
2 struct _RefC{
3     T data;
4     int ref;
5     _RefC(const T&d=0):data(d),ref(0){}
6 };
7 template<typename T>
8 struct _rp{
9     _RefC<T> *p;
10    T *operator->(){return &p->data;}
11    T &operator*(){return p->data;}
12    operator _RefC<T>*(){return p;}
13    _rp &operator=(const _rp &t){
14        if(p&&!--p->ref)delete p;
15        p=t.p,p&&+p->ref;
16        return *this;
17    }
18    _rp(_RefC<T> *t=0):p(t){p&&+p->ref;}
19    _rp(const _rp &t):p(t.p){p&&+p->ref;}
20    ~_rp(){if(p&&!--p->ref)delete p;}
21 };
22 template<typename T>
23 inline _rp<T> new_rp(const T&nd){
24     return _rp<T>(new _RefC<T>(nd));
25 }

```

2.5 skew_heap.cpp

```

1 node *merge(node *a,node *b){
2     if(!a||!b) return a?a:b;
3     if(b->data<a->data) swap(a,b);
4     swap(a->l,a->r);
5     a->l=merge(b,a->l);
6     return a;
7 }

```

2.6 undo_disjoint_set.cpp

```

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

```

2.7 整體二分.cpp

```

1 void totBS(int L, int R, vector<Item> M){
2     if(Q.empty()) return; //維護全域B陣列
3     if(L==R) 整個M的答案=r, return;
4     int mid = (L+R)/2;
5     vector<Item> mL, mR;
6     do_modify_B_with_divide(mid,M);
7     //讓B陣列在遞迴的時候只會保留[L~mid]的資訊
8     undo_modify_B(mid,M);
9     totBS(L,mid,mL);
10    totBS(mid+1,R,mR);
11 }

```

3 default

3.1 debug.cpp

```

1 //volatile
2 #ifdef DEBUG
3 #define dbg(...) {\
4     fprintf(stderr,"%s - %d : (%s) = ",
5         PRETTY_FUNCTION__,__LINE__,#
6         _VA_ARGS__); \
7     _DO(__VA_ARGS__); \
8 }
9 template<typename I> void _DO(I&&x){cerr<<x
10    <<endl;}
11 template<typename I,typename...T> void _DO(I
12    &&x,T&&...tail){cerr<<x<<" ";_DO(tail
13    ...);}
14
15 #else
16 #define dbg(...)
17 #endif

```

3.2 ext.cpp

```

1 #include<bits/extc++.h>
2 #include<ext/pd_ds/assoc_container.hpp>
3 #include<ext/pd_ds/tree_policy.hpp>
4 using namespace __gnu_cxx;
5 using namespace __gnu_pbds;
6 template<typename T>
7 using pbds_set = tree<T,null_type,less<T>,
8     rb_tree_tag,
9     tree_order_statistics_node_update>;
10 template<typename T,typename U>
11 using pbds_map = tree<T,U,less<T>,
12     rb_tree_tag,
13     tree_order_statistics_node_update>;
14 using heap=__gnu_pbds::priority_queue<int>;
15 //s.find_by_order(1); //0 base
16 //s.order_of_key(1);

```

3.3 IncStack.cpp

```

1 //Magic
2 #pragma GCC optimize "Ofast"
3 //stack resize,change esp to rsp if 64-bit
4 system
5 asm("mov %0,%esp\n" :: "g"(mem+1000000));
6 -Wl,--stack,214748364 -trigraphs
7 //linux stack resize
8 #include<sys/resource.h>
9 void increase_stack(){
10    const rlim_t ks=64*1024*1024;
11    struct rlimit rl;
12    int res=getrlimit(RLIMIT_STACK,&rl);
13    if(!res&&rl.rlim_cur<ks){
14        rl.rlim_cur=ks;
15        res=setrlimit(RLIMIT_STACK,&rl);
16    }
17 }

```

3.4 input.cpp

```

1 inline int read(){
2     int x=0; bool f=0; char c=getchar();
3     while(ch<'0' || '9'<ch) f|=ch=='-' ,ch=getchar
4     ();
5     while('0'<=ch&&ch<='9') x=x*10-'0'+ch,ch=
6     getchar();
7     return f?-x:x;
8 }
9 // #!/bin/bash
10 // g++ -std=c++11 -O2 -Wall -Wextra -Wno-
11 // unused-result -DDEBUG $1 && ./a.out
12 // -fsanitize=address -fsanitize=undefined
13 // -fsanitize=return

```

4 Flow

4.1 dinic.cpp

```

1 template<typename T>
2 struct DINIC{
3     static const int MAXN=105;
4     static const T INF=INT_MAX;
5     int n, level[MAXN], cur[MAXN];
6     struct edge{
7         int v,pre;
8         T cap,flow,r;
9         edge(int v,int pre,T cap):v(v),pre(pre),
10         cap(cap),flow(0),r(cap){}
11 };
12 int g[MAXN];
13 vector<edge> e;
14 void init(int _n){
15     memset(g,-1,sizeof(int)*((n=_n)+1));
16     e.clear();
17 }
18 void add_edge(int u,int v,T cap,bool
19     directed=false){
20     e.push_back(edge(v,g[u],cap));
21     g[u]=e.size()-1;
22     e.push_back(edge(u,g[v],directed?0:cap));
23     g[v]=e.size()-1;
24 }
25 int bfs(int s,int t){
26     memset(level,0,sizeof(int)*(n+1));
27     memcpy(cur,g,sizeof(int)*(n+1));
28     queue<int> q;
29     q.push(s);
30     level[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(!level[e[i].v]&&e[i].r){
35                 level[e[i].v]=level[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 cur_flow=INF){
44     if(u==t) return cur_flow;
45     T df=0;
46     for(int &i=cur[u];~i;i=e[i].pre){
47         if(level[e[i].v]==level[u]+1&&e[i].r){
48             if(df=dfs(e[i].v,t,min(cur_flow,e[i]
49             ].r))){
50                 e[i].flow+=df;
51                 e[i^1].flow-=df;
52                 e[i].r-=df;
53                 e[i^1].r+=df;
54                 return df;
55             }
56         }
57     }
58     return level[u]=0;
59 }
60 T dinic(int s,int t,bool clean=true){
61     if(clean){
62         for(size_t i=0;i<e.size();i++){
63             e[i].flow=0;
64             e[i].r=e[i].cap;
65         }
66     }
67     T ans=0, mf=0;
68     while(bfs(s,t)) while(mf=dfs(s,t)) ans+=mf;
69     return ans;
70 }

```

4.2 ISAP_with_cut.cpp

```

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,flow,r;
10        edge(int v,int pre,T cap):v(v),pre(pre),
11        cap(cap),flow(0),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     g[v]=e.size()-1;
25 }
26 T dfs(int u,int s,int t,T cur_flow=INF){
27     if(u==t) return cur_flow;
28     T df=0;
29     for(int &i=cur[u];~i;i=e[i].pre){
30         if(level[e[i].v]==level[u]+1&&e[i].r){
31             if(df=dfs(e[i].v,t,min(cur_flow,e[i]
32             ].r))){
33                 e[i].flow+=df;
34                 e[i^1].flow-=df;
35                 e[i].r-=df;
36                 e[i^1].r+=df;
37                 return df;
38             }
39         }
40     }
41     return level[u]=0;
42 }
43 T isap(int s,int t,bool clean=true){
44     memset(d,0,sizeof(int)*(n+1));
45     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].flow=0;
49         e[i].r=e[i].cap;
50     }
51     T max_flow=0;
52     for(gap[0]=n;d[s]<n;d[s]<max_flow+=dfs(s,s,t)
53     );
54     return max_flow;
55 }
56 vector<int> cut_e; //最小割邊集
57 bool vis[MAXN];
58 void dfs_cut(int u){
59     vis[u]=1; //表示u屬於source的最小割集
60     for(int i=g[u];~i;i=e[i].pre){
61         if(e[i].flow<e[i].cap&&vis[e[i].v]){
62             dfs_cut(e[i].v);
63         }
64     }
65     T min_cut(int s,int t){
66         T ans=isap(s,t);
67         memset(vis,0,sizeof(bool)*(n+1));
68         dfs_cut(s);
69         for(int u=0;u<n;u++){
70             if(vis[u]) for(int i=g[u];~i;i=e[i].pre)
71             if(!vis[e[i].v]) cut_e.push_back(i);
72         }
73     }
74 }

```

4.3 MinCostMaxFlow.cpp

```

1 template<typename _T>
2 struct MCMF{
3     static const int MAXN=440;
4     static const _T INF=999999999;
5     struct edge{

```

```

6         int v,pre;
7         _T cap,cost;
8         edge(int v,int pre,_T cap,_T cost):v(v),
9         pre(pre),cap(cap),cost(cost){}
10 };
11 int n,s,t;
12 _T 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,_T cap,_T cost,
21     bool directed=false){
22     e.push_back(edge(v,g[u],cap,cost));
23     g[u]=e.size()-1;
24     e.push_back(edge(u,g[v],directed?0:cap,-
25     cost));
26     g[v]=e.size()-1;
27 }
28 _T augment(int u,_T cur_flow){
29     if(u==t || !cur_flow) return ans+=piS*
30     cur_flow,cur_flow;
31     vis[u]=1;
32     _T r=cur_flow,d;
33     for(int i=g[u];~i;i=e[i].pre){
34         if(e[i].cap&&!e[i].cost&&vis[e[i].v]){
35             d=augment(e[i].v,min(r,e[i].cap));
36             e[i].cap-=d;
37             e[i^1].cap+=d;
38             if(!r==d) break;
39         }
40     }
41     return cur_flow-r;
42 }
43 bool modlabel(){
44     for(int u=0;u<n;u++) dis[u]=INF;
45     static deque<int> q;
46     dis[t]=0,q.push_back(t);
47     while(q.size()){
48         int u=q.front();q.pop_front();
49         _T dt;
50         for(int i=g[u];~i;i=e[i].pre){
51             if(e[i^1].cap&&(dt=dis[u]-e[i].cost)
52             <dis[e[i].v]){
53                 if((dis[e[i].v]=dt)<=dis[q.size()])
54                 q.front():S){}
55                 q.push_front(e[i].v);
56             }else q.push_back(e[i].v);
57         }
58     }
59 }
60 for(int u=0;u<n;u++){
61     for(int i=g[u];~i;i=e[i].pre)
62     e[i].cost+=dis[e[i].v]-dis[u];
63     return piS+=dis[S], dis[S]<INF;
64 }

```

```

6         int v,pre;
7         _T cap,cost;
8         edge(int v,int pre,_T cap,_T cost):v(v),
9         pre(pre),cap(cap),cost(cost){}
10 };
11 int n,s,t;
12 _T 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,_T cap,_T cost,
21     bool directed=false){
22     e.push_back(edge(v,g[u],cap,cost));
23     g[u]=e.size()-1;
24     e.push_back(edge(u,g[v],directed?0:cap,-
25     cost));
26     g[v]=e.size()-1;
27 }
28 _T augment(int u,_T cur_flow){
29     if(u==t || !cur_flow) return ans+=piS*
30     cur_flow,cur_flow;
31     vis[u]=1;
32     _T r=cur_flow,d;
33     for(int i=g[u];~i;i=e[i].pre){
34         if(e[i].cap&&!e[i].cost&&vis[e[i].v]){
35             d=augment(e[i].v,min(r,e[i].cap));
36             e[i].cap-=d;
37             e[i^1].cap+=d;
38             if(!r==d) break;
39         }
40     }
41     return cur_flow-r;
42 }
43 bool modlabel(){
44     for(int u=0;u<n;u++) dis[u]=INF;
45     static deque<int> q;
46     dis[t]=0,q.push_back(t);
47     while(q.size()){
48         int u=q.front();q.pop_front();
49         _T dt;
50         for(int i=g[u];~i;i=e[i].pre){
51             if(e[i^1].cap&&(dt=dis[u]-e[i].cost)
52             <dis[e[i].v]){
53                 if((dis[e[i].v]=dt)<=dis[q.size()])
54                 q.front():S){}
55                 q.push_front(e[i].v);
56             }else q.push_back(e[i].v);
57         }
58     }
59 }
60 for(int u=0;u<n;u++){
61     for(int i=g[u];~i;i=e[i].pre)
62     e[i].cost+=dis[e[i].v]-dis[u];
63     return piS+=dis[S], dis[S]<INF;
64 }

```

```

65     }return ans;
66 }
67 };

```

5 Graph

5.1 Augmenting_Path.cpp

```

1 #define MAXN1 505
2 #define MAXN2 505
3 int n1,n2;//n1個點連向n2個點
4 int match[MAXN2]; //屬於n2的點匹配了哪個點
5 vector<int> g[MAXN1]; //圖
6 bool vis[MAXN2]; //是否走訪過
7 bool dfs(int u){
8     for(size_t i=0;i<g[u].size();++i){
9         int v=g[u][i];
10        if(vis[v])continue;
11        vis[v]=1;
12        if(match[v]==-1||dfs(match[v]))
13            return match[v]=u, 1;
14    }
15    return 0;
16 }
17 inline int max_match(){
18     int ans=0;
19     memset(match,-1,sizeof(int)*n2);
20     for(int i=0;i<n1;++i){
21         memset(vis,0,sizeof(bool)*n2);
22         if(dfs(i))++ans;
23     }
24     return ans;
25 }

```

5.2 Augmenting_Path_multiple

```

1 #define MAXN1 1005
2 #define MAXN2 505
3 int n1,n2;//n1個點連向n2個點，其中n2個點可以
4     匹配很多邊
5 vector<int> g[MAXN1]; //圖
6 int c[MAXN2]; //每個屬於n2點最多可以接受幾條
7     匹配邊
8 vector<int> match_list[MAXN2]; //每個屬於n2的
9     點匹配了那些點
10 bool vis[MAXN2]; //是否走訪過
11 bool dfs(int u){
12     for(size_t i=0;i<g[u].size();++i){
13         int v=g[u][i];
14         if(vis[v])continue;
15         vis[v]=true;
16         if((int)match_list[v].size()<c[v]){
17             return match_list[v].push_back(u),
18                 true;
19         }else{

```

```

16         for(size_t j=0;j<match_list[v].size()
17             ;++j){
18             int next_u=match_list[v][j];
19             if(dfs(next_u))
20                 return match_list[v][j]=u, true;
21         }
22     }
23     return false;
24 }
25 inline int max_match(){
26     for(int i=0;i<n2;++i)match_list[i].clear();
27     int cnt=0;
28     for(int u=0;u<n1;++u){
29         memset(vis,0,sizeof(bool)*n2);
30         if(dfs(u))++cnt;
31     }
32     return cnt;
33 }

```

5.3 blossom_matching.cpp

```

1 #define MAXN 505
2 vector<int> g[MAXN];
3 int pa[MAXN],match[MAXN],st[MAXN],S[MAXN],v[
4     MAXN];
5 int t,n;
6 int lca(int x,int y){
7     for(++t;swap(x,y)){
8         if(x==0)continue;
9         if(v[x]==t)return x;
10        v[x]=t;
11        x=st[pa[match[x]]];
12    }
13 }
14 #define qpush(x) q.push(x),S[x]=0
15 void flower(int x,int y,int l,queue<int> &q)
16 {
17     while(st[x]!=1){
18         pa[x]=y;
19         if(S[y==match[x]]==1)qpush(y);
20         st[x]=st[y]=1, x=pa[y];
21     }
22 }
23 bool bfs(int x){
24     for(int i=1;i<n;++i)st[i]=i;
25     memset(S+1,-1,sizeof(int)*n);
26     queue<int> q; qpush(x);
27     while(q.size()){
28         x=q.front(),q.pop();
29         for(size_t i=0;i<g[x].size();++i){
30             int y=g[x][i];
31             if(S[y]==-1){
32                 pa[y]=x,S[y]=1;
33                 if(!match[y]){
34                     for(int lst=x;y=lst,x=pa[y])
35                         lst=match[x],match[x]=y,match[y]
36                             =x;
37                     return 1;
38                 }
39                 qpush(match[y]);
40             }else if(!S[y]&&st[y]!=st[x]){

```

```

38         int l=lca(y,x);
39         flower(y,x,l,q),flower(x,y,l,q);
40     }
41 }
42 }
43 return 0;
44 }
45 inline blossom(){
46     int ans=0;
47     for(int i=1;i<n;++i)
48         if(!match[i]&&bfs(i))++ans;
49     return ans;
50 }

```

5.4 graphISO.cpp

```

1 const int MAXN=1005,K=30;//K要夠大
2 const long long A=3,B=11,C=2,D=19,P=0
3     xdefaced;
4 long long f[K+1][MAXN];
5 vector<int> g[MAXN],rg[MAXN];
6 int n;
7 void init(){
8     for(int i=0;i<n;++i){
9         f[0][i]=1;
10        g[i].clear(), rg[i].clear();
11    }
12 }
13 void add_edge(int u,int v){
14     g[u].push_back(v), rg[v].push_back(u);
15 }
16 long long point_hash(int u){//O(N)
17     for(int t=1;t<=K;++t){
18         for(int i=0;i<n;++i){
19             f[t][i]=f[t-1][i]*A%P;
20             for(int j:g[i])f[t][i]=(f[t][i]+f[t-1][j]*B%P)%P;
21             for(int j:rg[i])f[t][i]=(f[t][i]+f[t-1][j]*C%P)%P;
22             if(i==u)f[t][i]+=D;//如果圖太大的話，
23                 把這行刪掉，執行一次後f[K]就會是所
24                 有點的答案
25             f[t][i]%=P;
26         }
27     }
28     return f[K][u];
29 }
30 vector<long long> graph_hash(){
31     vector<long long> ans;
32     for(int i=0;i<n;++i)ans.push_back(
33         point_hash(i)); //O(N^2)
34     sort(ans.begin(),ans.end());
35     return ans;
36 }

```

5.5 KM.cpp

```

1 #define MAXN 405
2 #define INF 0x3f3f3f3f

```

```

3 int n;// 1-base，0表示沒有匹配
4 int g[MAXN][MAXN],lx[MAXN],ly[MAXN],pa[MAXN],
5     slack_y[MAXN];
6 int match_y[MAXN],match_x[MAXN];
7 bool vx[MAXN],vy[MAXN];
8 void augment(int y){
9     for(int x,z;y=y=z){
10        x=pa[y],z=match_x[x];
11        match_y[y]=x,match_x[x]=y;
12    }
13 }
14 void bfs(int st){
15     for(int i=1;i<n;++i)slack_y[i]=INF,vx[i]=
16         vy[i]=0;
17     queue<int> q;q.push(st);
18     for(;;){
19         while(q.size()){
20             int x=q.front(),q.pop();
21             vx[x]=1;
22             for(int y=1;y<n;++y)if(!vy[y]){
23                 int t=lx[x]+ly[y]-g[x][y];
24                 if(t==0){
25                     pa[y]=x;
26                     if(!match_y[y]){augment(y);return}
27                     vy[y]=1,q.push(match_y[y]);
28                 }else if(slack_y[y]>t)pa[y]=x,
29                     slack_y[y]=t;
30             }
31             int cut=INF;
32             for(int y=1;y<n;++y){
33                 if(!vy[y]&&cut>slack_y[y])cut=slack_y[
34                     y];
35             }
36             for(int y=1;y<n;++y){
37                 if(!vy[y]&&slack_y[y]==0){
38                     if(!match_y[y]){augment(y);return;}
39                     vy[y]=1,q.push(match_y[y]);
40                 }
41             }
42         }
43     }
44 }
45 }
46 long long KM(){
47     memset(match_y,0,sizeof(int)*(n+1));
48     memset(ly,0,sizeof(int)*(n+1));
49     for(int x=1;x<n;++x){
50         lx[x]=-INF;
51         for(int y=1;y<n;++y)
52             lx[x]=max(lx[x],g[x][y]);
53     }
54     for(int x=1;x<n;++x)bfs(x);
55     long long ans=0;
56     for(int y=1;y<n;++y)ans+=g[match_y[y]][y];
57     return ans;
58 }

```

5.6 MaximumClique.cpp

```

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 };

```

5.7 MinimumMeanCycle.cpp

```

1 #include<cstdio>//for DBL_MAX
2 int dp[maxN+1][maxN+1];
3 double mnc(int n){
4     int u,v,w;
5     const int inf=0x7f7f7f7f;
6     memset(dp,0x7f,sizeof(dp));
7     memset(dp[0],0,sizeof(dp[0]));
8     for(int i=0;i<n;++i){
9         for(auto e:E){
10             tie(u,v,w)=e;
11             if(dp[i][u]!=inf)

```

```

12         dp[i+1][v]=min(dp[i+1][v],dp[i][u]+w);
13     }
14     double res = DBL_MAX;
15     for(int i=1;i<n;++i){
16         double val = DBL_MIN;
17         for(int j=0;j<n;++j)
18             val=max(val,double(dp[n][i]-dp[i][j]))/(n-j);
19         res=min(res,val);
20     }
21     return res;
22 }
23 }

```

5.8 Rectilinear_MST.cpp

```

1 //平面曼哈頓最小生成樹構造圖(去除非必要邊)
2 #define T int
3 #define INF 0x3f3f3f3f
4 struct point{
5     T x,y;
6     int id;//從0開始編號
7     point(){}
8     T dist(const point &p)const{
9         return abs(x-p.x)+abs(y-p.y);
10    }
11 };
12 bool cmpx(const point &a,const point &b){
13     return a.x<b.x||(a.x==b.x&&a.y<b.y);
14 }
15 struct edge{
16     int u,v;
17     T cost;
18     edge(int u,int v,T c):u(u),v(v),cost(c){}
19     bool operator<(const edge&e)const{
20         return cost<e.cost;
21     }
22 };
23 struct bit_node{
24     T mi;
25     int id;
26     bit_node(const T&mi=INF,int id=-1):mi(mi),id(id){}
27 };
28 vector<bit_node> bit;
29 void bit_update(int i,const T&data,int id){
30     for(;i=i&(-i)){
31         if(data<bit[i].mi)bit[i]=bit_node(data,id);
32     }
33 }
34 int bit_find(int i,int m){
35     bit_node x;
36     for(;i<m;i+=i&(-i)) if(bit[i].mi<x.mi)x=bit[i];
37     return x.id;
38 }
39 vector<edge> build_graph(int n,point p[]){
40     vector<edge> e;//edge for MST
41     for(int dir=0;dir<4;+dir){//4種座標變換
42         if(dir%2) for(int i=0;i<n;++i) swap(p[i].x,p[i].y);

```

```

43     else if(dir==2) for(int i=0;i<n;++i) p[i].x=-p[i].x;
44     sort(p,p+n,cmpx);
45     vector<T> ga(n), gb;
46     for(int i=0;i<n;++i)ga[i]=p[i].y-p[i].x;
47     gb=ga, sort(gb.begin(),gb.end());
48     gb.erase(unique(gb.begin(),gb.end()),gb.end());
49     int m=gb.size();
50     bit=vector<bit_node>(m+1);
51     for(int i=n-1;i>=0;--i){
52         int pos=lower_bound(gb.begin(),gb.end(),ga[i])-gb.begin()+1;
53         int ans=bit_find(pos,m);
54         if(~ans)e.push_back(edge(p[i].id,p[ans].id,p[i].dist(p[ans])));
55         bit_update(pos,p[i].x+p[i].y,i);
56     }
57     return e;
58 }
59 }

```

5.9 treeISO.cpp

```

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 }

```

5.10 一般圖最小權完美匹配.cpp

```

1 struct Graph {
2     // Minimum General Weighted Matching (
3     // Perfect Match) 0-base
4     static const int MXN = 105;
5     int n, edge[MXN][MXN];
6     int match[MXN],dis[MXN],onstk[MXN];
7     vector<int> stk;
8     void init(int _n) {

```

```

9         n = _n;
10        for (int i=0; i<n; i++)
11            for (int j=0; j<n; j++)
12                edge[i][j] = 0;
13    }
14    void add_edge(int u, int v, int w) {
15        edge[u][v] = edge[v][u] = w;
16    }
17    bool SPFA(int u){
18        if (onstk[u]) return true;
19        stk.push_back(u);
20        onstk[u] = 1;
21        for (int v=0; v<n; v++){
22            if (u != v && match[u] != v && !onstk[v]){
23                int m = match[v];
24                if (dis[m] > dis[u] - edge[v][m] + edge[u][v]){
25                    dis[m] = dis[u] - edge[v][m] + edge[u][v];
26                    onstk[v] = 1;
27                    stk.push_back(v);
28                    if (SPFA(m)) return true;
29                    stk.pop_back();
30                    onstk[v] = 0;
31                }
32            }
33        }
34        onstk[u] = 0;
35        stk.pop_back();
36        return false;
37    }
38    int solve() {
39        // find a match
40        for (int i=0; i<n; i+=2){
41            match[i] = i+1, match[i+1] = i;
42        }
43        for(;;){
44            int found = 0;
45            for (int i=0; i<n; i++) dis[i] = onstk[i] = 0;
46            for (int i=0; i<n; i++){
47                stk.clear();
48                if (!onstk[i] && SPFA(i)){
49                    found = 1;
50                    while (stk.size()>=2){
51                        int u = stk.back(); stk.pop_back();
52                        int v = stk.back(); stk.pop_back();
53                        match[u] = v;
54                        match[v] = u;
55                    }
56                }
57            }
58            if (!found) break;
59        }
60        int ret = 0;
61        for (int i=0; i<n; i++)
62            ret += edge[i][match[i]];
63        ret /= 2;
64        return ret;
65    }
66 }graph;

```


5.11 全局最小割.cpp

```

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[nd[i]]=0;
20             for(int i=1;i<tn;++i){
21                 ind=i;
22                 for(int j=i;j<tn;++j){
23                     dis[nd[j]]+=g[nd[i-1]][nd[j]];
24                     if(dis[nd[ind]]<dis[nd[j]])ind=j;
25                 }
26                 swap(nd[ind],nd[i]);
27             }
28             if(ans>dis[nd[ind]])ans=dis[t=nd[ind]];
29             for(int i=0;i<tn;++i)
30                 g[nd[ind-1]][nd[i]]=g[nd[i]][nd[ind-1]]+g[nd[i]][nd[ind]];
31         }
32         return ans;
33     }
34 };

```

5.12 平面圖判定.cpp

```

1 static const int MAXN = 20;
2 struct Edge{
3     int u, v;
4     Edge(int s, int d) : u(s), v(d) {}
5 };
6 bool isK33(int n, int degree[]){
7     int t = 0, z = 0;
8     for(int i=0;i<n;++i){
9         if(degree[i] == 3)++t;
10        else if(degree[i] == 0)++z;
11        else return false;
12    }
13    return t == 6 && t + z == n;
14 }
15 bool isK5(int n, int degree[]){
16     int f = 0, z = 0;
17     for(int i=0;i<n;++i){
18         if(degree[i] == 4)++f;
19         else if(degree[i] == 0)++z;
20         else return false;
21     }

```

```

22     return f == 5 && f + z == n;
23 }
24 // it judge a given graph is Homeomorphic
25     with K33 or K5
26 bool isHomeomorphic(bool G[MAXN][MAXN],
27     const int n){
28     for(;;){
29         int cnt = 0;
30         for(int i=0;i<n;++i){
31             vector<Edge> E;
32             for(int j=0;j<n&&E.size()<3;++j)
33                 if(G[i][j] && i != j)
34                     E.push_back(Edge(i, j));
35             if(E.size() == 1){
36                 G[i][E[0].v] = G[E[0].v][i] = false;
37             }else if(E.size() == 2){
38                 G[i][E[0].v] = G[E[0].v][i] = false;
39                 G[i][E[1].v] = G[E[1].v][i] = false;
40                 G[E[0].v][E[1].v] = G[E[1].v][E[0].v] = true;
41                 ++cnt;
42             }
43             if(cnt == 0)break;
44         }
45         static int degree[MAXN];
46         fill(degree, degree + n, 0);
47         for(int i=0;i<n;++i){
48             for(int j=i+1; j<n; ++j){
49                 if(!G[i][j])continue;
50                 ++degree[i];
51                 ++degree[j];
52             }
53         }
54         return !(isK33(n, degree) || isK5(n, degree));

```

5.13 弦圖完美消除序列.cpp

```

1 struct chordal{
2     static const int MAXN=1005;
3     int n;// 0-base
4     vector<int>G[MAXN];
5     int rank[MAXN],label[MAXN];
6     bool mark[MAXN];
7     void init(int _n){n=_n;
8         for(int i=0;i<n;++i)G[i].clear();
9     }
10    void add_edge(int u,int v){
11        G[u].push_back(v);
12        G[v].push_back(u);
13    }
14    vector<int> MCS(){
15        memset(rank,-1,sizeof(int)*n);
16        memset(label,0,sizeof(int)*n);
17        priority_queue<pair<int,int>> pq;
18        for(int i=0;i<n;++i)pq.push(make_pair(0,i));
19        for(int i=n-1;i>=0;--i)for(;;){
20            int u=pq.top().second;pq.pop();
21            if(~rank[u])continue;
22            rank[u]=i;

```

```

23         for(auto v:G[u])if(rank[v]==-1){
24             pq.push(make_pair(++label[v],v));
25         }
26         break;
27     }
28     vector<int> res(n);
29     for(int i=0;i<n;++i)res[rank[i]]=i;
30     return res;
31 }
32 bool check(vector<int> ord){//弦圖判定
33     for(int i=0;i<n;++i)rank[ord[i]]=i;
34     memset(mark,0,sizeof(bool)*n);
35     for(int i=0;i<n;++i){
36         vector<pair<int,int>> tmp;
37         for(auto u:G[ord[i]])if(!mark[u])
38             tmp.push_back(make_pair(rank[u],u));
39         sort(tmp.begin(),tmp.end());
40         if(tmp.size()){
41             int u=tmp[0].second;
42             set<int> S;
43             for(auto v:G[u])S.insert(v);
44             for(size_t j=1;j<tmp.size();++j)
45                 if(!S.count(tmp[j].second))return 0;
46         }
47         mark[ord[i]]=1;
48     }
49     return 1;
50 }
51 };

```

5.14 最小斯坦納樹 DP.cpp

```

1 //n個點，其中r個要構成斯坦納樹
2 //答案在max(dp[(1<=r)-1][k]) k=0~n-1
3 //p表示要構成斯坦納樹的點集
4 //O( n^3 + n^3*r + n^2*2^r )
5 #define REP(i,n) for(int i=0;i<(int)n;++i)
6 const int MAXN=30,MAXM=8;// 0-base
7 const int INF=0x3f3f3f3f;
8 int dp[1<MAXM][MAXN];
9 int g[MAXN][MAXN];
10 void init(){memset(g,0,sizeof(g));}
11 void add_edge(int u,int v,int w){
12     g[u][v]=g[v][u]=min(g[v][u],w);
13 }
14 void steiner(int n,int r,int *p){
15     REP(k,n)REP(i,n)REP(j,n)
16         g[i][j]=min(g[i][j],g[i][k]+g[k][j]);
17     REP(i,n)g[i][i]=0;
18     REP(i,r)REP(j,n)dp[1<=i][j]=g[p[i]][j];
19     for(int i=1;i<(1<=r);++i){
20         if(!i&(i-1))continue;
21         REP(j,n)dp[i][j]=INF;
22         REP(j,n){
23             int tmp=INF;
24             for(int s=i&(i-1);s>=i&(i-1)-1;--s)
25                 tmp=min(tmp,dp[s][j]+dp[i&s][j]);
26             REP(k,n)dp[i][k]=min(dp[i][k],g[j][k]+tmp);
27         }
28     }

```

5.15 最小樹形圖 __ 朱劉.cpp

```

1 template<typename T>
2 struct zhu_liu{
3     static const int MAXN=110,MAXM=10005;
4     struct node{
5         int u,v;
6         T w,tag;
7         node *l,*r;
8         node(int u=0,int v=0,T w=0):u(u),v(v),w(w),tag(0),l(0),r(0){}
9     }
10    void down(){
11        w+=tag;
12        if(l)l->tag+=tag;
13        if(r)r->tag+=tag;
14        tag=0;
15    }
16    mem[MAXN];
17    node *pq[MAXN*2],*E[MAXN*2];
18    int st[MAXN*2],id[MAXN*2],m;
19    void init(int n){
20        for(int i=1;i<n;++i){
21            pq[i]=E[i]=0, st[i]=id[i]=i;
22            m=0;
23        }
24        node *merge(node *a,node *b){//skew heap
25            if(!a||!b)return a?a:b;
26            a->down(),b->down();
27            if(b->w<a->w)return merge(b,a);
28            swap(a->l,a->r);
29            a->l=merge(b,a->l);
30            return a;
31        }
32        void add_edge(int u,int v,T w){
33            if(u!=v)pq[v]=merge(pq[v],&(mem[m++]=node(u,v,w)));
34        }
35        int find(int x,int *st){
36            return st[x]==x?x:st[x]=find(st[x],st);
37        }
38        T build(int root,int n){
39            T ans=0;int N=n,all=n;
40            for(int i=1;i<N;++i){
41                if(i==root||!pq[i])continue;
42                while(pq[i]){
43                    pq[i]->down(),E[i]=pq[i];
44                    pq[i]=merge(pq[i]->l,pq[i]->r);
45                    if(find(E[i]->u,id)!=find(i,id))break;
46                }
47                if(find(E[i]->u,id)==find(i,id))continue;
48                ans+=E[i]->w;
49                if(find(E[i]->u,st)!=find(i,st)){
50                    if(pq[i])pq[i]->tag-=E[i]->w;
51                    pq[i]=pq[i];id[N]=N;
52                    for(int u=find(E[i]->u,id);u!=i;u=find(E[u]->u,id)){
53                        if(pq[u])pq[u]->tag-=E[u]->w;
54                        id[find(u,id)]=N;
55                        pq[N]=merge(pq[N],pq[u]);

```

```

55     }
56     st[N]=find(i,st);
57     id[find(i,id)]=N;
58     }else st[find(i,st)]=find(E[i]->u,st)
59     ,--all;
60     return all==1?ans:-INT_MAX;//圖不連通就
61     無解
62 }

```

5.16 穩定婚姻模板.cpp

```

1 queue<int> Q;
2 for ( i : 所有考生 ) {
3     設定在第0志願;
4     Q.push(考生i);
5 }
6 while(Q.size()){
7     當前考生=Q.front();Q.pop();
8     while ( 此考生未分發 ) {
9         指標移到下一志願;
10        if ( 已經沒有志願 or 超出志願總數 )
11            break;
12        計算該考生在該科系加權後的總分;
13        if ( 不符合科系需求 ) continue;
14        if ( 目前科系有餘額 ) {
15            依加權後分數高低順序將考生id加入科系錄
16            取名單中;
17        }
18        if ( 目前科系已額滿 ) {
19            if ( 此考生成績比最低分數還高 ) {
20                依加權後分數高低順序將考生id加入科系
21                錄取名單;
22                Q.push(被踢出的考生);
23            }
24        }
25    }
26 }

```

6 language

6.1 CNF.cpp

```

1 #define MAXN 55
2 struct CNF{
3     int s,x,y;//s->xy | s->x, if y==1
4     int cost;
5     CNF(){}
6     CNF(int s,int x,int y,int c):s(s),x(x),y(y)
7     ,cost(c){}
8 int state;//規則數量

```

```

9 map<char,int> rule;//每個字元對應到的規則·
10 小寫字母為終端字符
11 vector<CNF> cnf;
12 void init(){
13     state=0;
14     rule.clear();
15     cnf.clear();
16 }
17 void add_to_cnf(char s,const string &p,int
18     cost){
19     //加入一個s -> <p>的文法·代價為cost
20     if(rule.find(s)==rule.end())rule[s]=state
21     ++;
22     for(auto c:p)if(rule.find(c)==rule.end())
23     rule[c]=state++;
24     if(p.size()==1){
25         cnf.push_back(CNF(rule[s],rule[p[0]],-1,
26             cost));
27     }else{
28         int left=rule[s];
29         int sz=p.size();
30         for(int i=0;i<sz-2;++i){
31             cnf.push_back(CNF(left,rule[p[i]],
32                 state,0));
33             left=state++;
34         }
35         cnf.push_back(CNF(left,rule[p[sz-2]],
36             rule[p[sz-1]],cost));
37     }
38 }
39 vector<long long> dp[MAXN][MAXN];
40 vector<bool> neg_INF[MAXN][MAXN];//如果花費
41 是真的可能會有無限小的情形
42 void relax(int l,int r,const CNF &c,long
43     long cost,bool neg_c=0){
44     if(!neg_INF[l][r][c.s]&&(neg_INF[l][r][c.x]
45         ||cost<dp[l][r][c.s])){
46         if(neg_c||neg_INF[l][r][c.x]){
47             dp[l][r][c.s]=0;
48             neg_INF[l][r][c.s]=true;
49         }else dp[l][r][c.s]=cost;
50     }
51 }
52 void bellman(int l,int r,int n){
53     for(int k=1;k<=state;++k)
54     for(auto c:cnf)
55     if(c.y==1)relax(l,r,c,dp[l][r][c.x]+c
56     .cost,k==n);
57 }
58 void cyk(const vector<int> &tok){
59     for(int i=0;i<(int)tok.size();++i){
60         for(int j=0;j<(int)tok.size();++j){
61             dp[i][j]=vector<long long>(state+1,
62                 INT_MAX);
63             neg_INF[i][j]=vector<bool>(state+1,
64                 false);
65         }
66         dp[i][i][tok[i]]=0;
67         bellman(i,i,tok.size());
68     }
69     for(int r=1;r<(int)tok.size();++r){
70         for(int l=r-1;l>=0;--l){
71             for(int k=1;k<r;++k)
72             for(auto c:cnf)

```

```

60         if(~c.y)relax(l,r,c,dp[l][k][c.x]+
61             dp[k+1][r][c.y]+c.cost);
62         bellman(l,r,tok.size());
63     }
64 }

```

7 Linear_Programming

7.1 最大密度子圖.cpp

```

1 typedef double T;//POJ 3155
2 const int MAXN=105;
3 struct edge{
4     int u,v;
5     T w;
6     edge(int u=0,int v=0,T w=0):u(u),v(v),w(w)
7     {}
8 }
9 vector<edge> E;
10 int n,m;// 1-base
11 T de[MAXN],pv[MAXN];//每個點的邊權和和點權(
12 有些題目會給)
13 void init(){
14     E.clear();
15     for(int i=1;i<=n;++i)de[i]=pv[i]=0;
16 }
17 void add_edge(int u,int v,T w){
18     E.push_back(edge(u,v,w));
19     de[u]+=w,de[v]+=w;
20 }
21 T U;//二分搜的最大值
22 void get_U(){
23     U=0;
24     for(int i=1;i<=n;++i)U+=2*pv[i];
25     for(size_t i=0;i<E.size();++i)U+=E[i].w;
26 }
27 ISAP<T> isap;//網路流
28 int s,t;//原匯點
29 void build(T L){
30     isap.init(n+2);
31     for(size_t i=0;i<E.size();++i)
32     isap.add_edge(E[i].u,E[i].v,E[i].w);
33     for(int v=1;v<=n;++v){
34         isap.add_edge(s,v,U);
35         isap.add_edge(v,t,U+2*L-de[v]-2*pv[v]);
36     }
37 }
38 int main(){
39     while(~scanf("%d%d",&n,&m)){
40         if(!m){
41             puts("1\n1");
42             continue;
43         }
44         init();
45         int u,v;
46         for(int i=0;i<m;++i){
47             scanf("%d%d",&u,&v);
48             add_edge(u,v,1);
49         }

```

```

48     get_U();
49     s=n+1,t=n+2;
50     T l=0,r=U,k=1.0/(n*n);
51     while(r-l>k){//二分搜最大值
52         T mid=(l+r)/2;
53         build(mid);
54         T res=(U*n-isap.isap(s,t))/2;
55         if(res>0)l=mid;
56         else r=mid;
57     }
58     build(l);
59     isap.min_cut(s,t);
60     vector<int> ans;
61     for(int i=1;i<=n;++i)
62     if(isap.vis[i])ans.push_back(i);
63     printf("%d\n",ans.size());
64     for(size_t i=0;i<ans.size();++i)
65     printf("%d\n",ans[i]);
66 }
67 return 0;
68 }

```

8 Number_Theory

8.1 basic.cpp

```

1 template<typename T>
2 void gcd(const T &a,const T &b,T &d,T &x,T &
3     y){
4     if(!b) d=a,x=1,y=0;
5     else gcd(b,a%b,d,y,x), y=-x*(a/b);
6 }
7 long long int phi[N+1];
8 void phiTable(){
9     for(int i=1;i<=N;++i)phi[i]=i;
10    for(int i=1;i<=N;++i)for(x=i*2;x<=N;x+=i)
11    phi[x]-=phi[i];
12 }
13 void all_divdown(const LL &n){ // all n/x
14     for(LL a=1;a<=n;a=n/(n/(a+1)))
15     // dosomething;
16 }
17 const int MAXPRIME = 1000000;
18 int iscom[MAXPRIME], prime[MAXPRIME],
19     primecnt;
20 phi[MAXPRIME], mu[MAXPRIME];
21 void sieve(void){
22     memset(iscom,0,sizeof(iscom));
23     primecnt = 0;
24     phi[1] = mu[1] = 1;
25     for(int i=2;i<MAXPRIME;++i) {
26         if(!iscom[i]) {
27             prime[primecnt++] = i;
28             mu[i] = -1;
29             phi[i] = i-1;
30         }
31         for(int j=0;j<primecnt;++j) {
32             int k = i * prime[j];
33             if(k>MAXPRIME) break;
34             iscom[k] = prime[j];

```

```

33     if(i%prime[j]==0) {
34         mu[k] = 0;
35         phi[k] = phi[i] * prime[j];
36         break;
37     } else {
38         mu[k] = -mu[i];
39         phi[k] = phi[i] * (prime[j]-1);
40     }
41 }
42 }
43 }
44 }
45 bool g_test(const LL &g, const LL &p, const
    vector<LL> &v) {
46     for(int i=0;i<v.size();++i)
47         if(modexp(g,(p-1)/v[i],p)==1)
48             return false;
49     return true;
50 }
51 LL primitive_root(const LL &p) {
52     if(p==2) return 1;
53     vector<LL> v;
54     Factor(p-1,v);
55     v.erase(unique(v.begin(), v.end()), v.end
        ());
56     for(LL g=2;g<p;++g)
57         if(g_test(g,p,v))
58             return g;
59     puts("primitive_root NOT FOUND");
60     return -1;
61 }
62 int Legendre(const LL &a, const LL &p) {
63     return modexp(a%p,(p-1)/2,p); }
64 LL inv(const LL &a, const LL &n) {
65     LL d,x,y;
66     gcd(a,n,d,x,y);
67     return d==1 ? (x+n)%n : -1;
68 }
69
70 int inv[maxN];
71 LL invtable(int n,LL P){
72     inv[1]=1;
73     for(int i=2;i<n;++i)
74         inv[i]=(P-(P/i))*inv[P%i]%P;
75 }
76
77 LL log_mod(const LL &a, const LL &b, const
    LL &p) {
78     // a ^ x = b ( mod p )
79     int m=sqrt(p+.5), e=1;
80     LL v=inv(modexp(a,m,p), p);
81     map<LL,int> x;
82     x[1]=0;
83     for(int i=1;i<m;++i) {
84         e = Llmul(e,a,p);
85         if(!x.count(e)) x[e] = i;
86     }
87     for(int i=0;i<m;++i) {
88         if(x.count(b)) return i*m + x[b];
89         b = Llmul(b,v,p);
90     }
91     return -1;
92 }
93 }

```

```

94 LL Tonelli_Shanks(const LL &n, const LL &p)
    {
95     // x^2 = n ( mod p )
96     if(n==0) return 0;
97     if(Legendre(n,p)!=1) while(1) { puts("SQRT
        ROOT does not exist"); }
98     int S = 0;
99     LL Q = p-1;
100     while( !(Q&1) ) { Q>>=1; ++S; }
101     if(S==1) return modexp(n%p,(p+1)/4,p);
102     LL z = 2;
103     for(; Legendre(z,p)!=-1; ++z)
104         LL c = modexp(z,Q,p);
105         LL R = modexp(n%p,(Q+1)/2,p), t = modexp(n
            %p,Q,p);
106         int M = S;
107         while(1) {
108             if(t==1) return R;
109             LL b = modexp(c,1L<<(M-i-1),p);
110             R = Llmul(R,b,p);
111             t = Llmul(Llmul(b,b,p), t, p);
112             c = Llmul(b,b,p);
113             M = i;
114         }
115         return -1;
116     }
117
118 template<typename T>
119 T Euler(T n){
120     T ans=n;
121     for(T i=2;i*i<=n;++i){
122         if(n%i==0){
123             ans=ans/i*(i-1);
124             while(n%i==0)n/=i;
125         }
126     }
127     if(n>1)ans=ans/n*(n-1);
128     return ans;
129 }
130
131 //Chinese_remainder_theorem
132 template<typename T>
133 T pow_mod(T n,T k,T m){
134     T ans=1;
135     for(n=(n>=m?n%m:n);k>=1){
136         if(k&1)ans=ans*n%m;
137         n=n*n%m;
138     }
139     return ans;
140 }
141
142 template<typename T>
143 T crt(vector<T> &m,vector<T> &a){
144     T M=1,tM,ans=0;
145     for(int i=0;i<(int)m.size();++i)M*=m[i];
146     for(int i=0;i<(int)a.size();++i){
147         tM=M/m[i];
148         ans=(ans+(a[i]*tM%M)*pow_mod(tM,Euler(m
            [i])-1,m[i])%M)%M;
149     }
150     //如果m[i]是質數 · Euler(m[i])-1=m[i]-2 ·
        就不用算Euler了*/
151     return ans;
152 }
153 //java code

```

```

154 //求sqrt(N)的連分數
155 public static void Pell(int n){
156     BigInteger N,p1,p2,q1,q2,a0,a1,a2,g1,g2,h1
        ,h2,p,q;
157     g1=q2=p1=BigInteger.ZERO;
158     h1=q1=p2=BigInteger.ONE;
159     a0=a1=BigInteger.valueOf((int)Math.sqrt
        (1.0*n));
160     BigInteger ans=a0.multiply(a0);
161     if(ans.equals(BigInteger.valueOf(n))){
162         System.out.println("No solution!");
163         return ;
164     }
165     while(true){
166         g2=a1.multiply(h1).subtract(g1);
167         h2=N.subtract(g2.pow(2)).divide(h1);
168         a2=g2.add(a0).divide(h2);
169         p=a1.multiply(p2).add(p1);
170         q=a1.multiply(q2).add(q1);
171         if(p.pow(2).subtract(N.multiply(q.pow
            (2))).compareTo(BigInteger.ONE)==0)
            break;
172         g1=g2;h1=h2;a1=a2;
173         p1=p2;p2=p;
174         q1=q2;q2=q;
175     }
176     System.out.println(p+" "+q);
177 }

```

```

10     for(int j=i+1;j<n;++j)
11         if(s[j]<s[i])++t;
12     res+=t*factorial[n-i-1];
13 }
14 return res;
15 }
16 vector<int> decode(int a,int n){
17     vector<int> res;
18     vector<bool> vis(n,0);
19     for(int i=n-1;i>=0;--i){
20         int t=a/factorial[i],j;
21         for(j=0;j<n;++j)
22             if(!vis[j]){
23                 if(t==0)break;
24                 --t;
25             }
26         res.push_back(j);
27         vis[j]=1;
28         a%=factorial[i];
29     }
30     return res;
31 }

```

8.4 FFT.cpp

```

1 template<typename T,typename VT=vector<
    complex<T> > >
2 struct FFT{
3     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&0xF0F0F0F0U)
            >>4);
9         a=((a&0xFF0FF0FFU)<<8)|((a&0xFF0FF0FFU)
            >>8);
10        a=((a&0x000FFFFFU)<<16)|((a&0xFFFF0000U)
            >>16);
11        return a>>(32-len);
12    }
13    void fft(bool is_inv,VT &in,VT &out,int N)
        {
14        int bitlen=__lg(N),num=is_inv?-1:1;
15        for(int i=0;i<N;++i)out[bit_reverse(i,
            bitlen)]=in[i];
16        for(int step=2;step<=N;step<<=1){
17            const int mh=step>>1;
18            for(int i=0;i<mh;++i){
19                complex<T> wi=exp(complex<T>(0,i*num
                    *pi/mh));
20                for(int j=i;j<N;j+=step){
21                    int k=j+mh;
22                    complex<T> u=out[j],t=wi*out[k];
23                    out[j]=u+t;
24                    out[k]=u-t;
25                }
26            }
27        }
28        if(is_inv)for(int i=0;i<N;++i)out[i]/=N;
29    }

```

8.2 bit_set.cpp

```

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 }

```

8.3 cantor_expansion.cpp

```

1 int factorial[MAXN];
2 void init(){
3     factorial[0]=1;
4     for(int i=1;i<=MAXN;++i)factorial[i]=
        factorial[i-1]*i;
5 }
6 int encode(const vector<int> &s){
7     int n=s.size(),res=0;
8     for(int i=0;i<n;++i){
9         int t=0;

```

30| };

8.6 FWT.cpp

8.5 find_real_root.cpp

```

1 // an*x^n + ... + a1x + a0 = 0;
2 int sign(double x){
3     return x < -eps ? -1 : x > eps;
4 }
5
6 double get(const vector<double>&coef, double
7     x){
8     double e = 1, s = 0;
9     for(auto i : coef) s += i*e, e *= x;
10    return s;
11 }
12
13 double find(const vector<double>&coef, int n
14     , double lo, double hi){
15     double sign_lo, sign_hi;
16     if( !(sign_lo = sign(get(coef,lo))) )
17         return lo;
18     if( !(sign_hi = sign(get(coef,hi))) )
19         return hi;
20     if(sign_lo * sign_hi > 0) return INF;
21     for(int stp = 0; stp < 100 && hi - lo >
22         eps; ++stp){
23         double m = (lo+hi)/2.0;
24         int sign_mid = sign(get(coef,m));
25         if(!sign_mid) return m;
26         if(sign_lo*sign_mid < 0) hi = m;
27         else lo = m;
28     }
29     return (lo+hi)/2.0;
30 }
31
32 vector<double> cal(vector<double>coef, int n
33 ){
34     vector<double>res;
35     if(n == 1){
36         if(sign(coef[1])) res.pb(-coef[0]/coef
37             [1]);
38         return res;
39     }
40     vector<double>dcoef(n);
41     for(int i = 0; i < n; ++i) dcoef[i] = coef
42         [i+1]*(i+1);
43     vector<double>droot = cal(dcoef, n-1);
44     droot.insert(droot.begin(), -INF);
45     droot.pb(INF);
46     for(int i = 0; i+1 < droot.size(); ++i){
47         double tmp = find(coef, n, droot[i],
48             droot[i+1]);
49         if(tmp < INF) res.pb(tmp);
50     }
51     return res;
52 }
53
54 int main () {
55     vector<double>ve;
56     vector<double>ans = cal(ve, n);
57     // 視情況把答案 +eps，避免 -0
58 }

```

```

1 vector<int> F_OR_T(vector<int> f, bool
2     inverse){
3     for(int i=0; (2<<i)<=f.size(); ++i)
4         for(int j=0; j<f.size(); j+=2<<i)
5             for(int k=0; k<(1<<i); ++k)
6                 f[j+k+(1<<i)] += f[j+k]*(inverse
7                     ?-1:1);
8     return f;
9 }
10 vector<int> rev(vector<int> A) {
11     for(int i=0; i<A.size(); i+=2)
12         swap(A[i],A[i^(A.size()-1)]);
13     return A;
14 }
15 vector<int> F_AND_T(vector<int> f, bool
16     inverse){
17     return rev(F_OR_T(rev(f), inverse));
18 }
19 vector<int> F_XOR_T(vector<int> f, bool
20     inverse){
21     for(int i=0; (2<<i)<=f.size(); ++i)
22         for(int j=0; j<f.size(); j+=2<<i)
23             for(int k=0; k<(1<<i); ++k){
24                 int u=f[j+k], v=f[j+k+(1<<i)];
25                 f[j+k+(1<<i)] = u-v, f[j+k] = u+v;
26             }
27     if(inverse) for(auto &a:f) a/=f.size();
28     return f;
29 }

```

8.7 LinearCongruence.cpp

```

1 pair<LL,LL> LinearCongruence(LL a[],LL b[],
2     LL m[],int n) {
3     // a[i]*x = b[i] ( mod m[i] )
4     for(int i=0;i<n;++i) {
5         LL x, y, d = extgcd(a[i],m[i],x,y);
6         if(b[i]%d!=0) return make_pair(-1LL,0LL);
7         m[i] /= d;
8         b[i] = LLmul(b[i]/d,x,m[i]);
9     }
10    LL lastb = b[0], lastm = m[0];
11    for(int i=1;i<n;++i) {
12        LL x, y, d = extgcd(m[i],lastm,x,y);
13        if((lastb-b[i])%d!=0) return make_pair
14            (-1LL,0LL);
15        lastb = LLmul((lastb-b[i])/d,x,(lastm/d)
16            )*m[i];
17        lastm = (lastm/d)*m[i];
18        lastb = (lastb+b[i])%lastm;
19    }
20    return make_pair(lastb<0?lastb+lastm:lastb
21        ,lastm);
22 }

```

8.8 Lucas.cpp

```

1 int mod_fact(int n,int &e){
2     e=0;
3     if(n==0)return 1;
4     int res=mod_fact(n/P,e);
5     e += n/P;
6     if((n/P)%2==0)return res*fact[n%P]%P;
7     return res*(P-fact[n%P])%P;
8 }
9 int Cmod(int n,int m){
10    int a1,a2,a3,e1,e2,e3;
11    a1=mod_fact(n,e1);
12    a2=mod_fact(m,e2);
13    a3=mod_fact(n-m,e3);
14    if(e1>e2+e3)return 0;
15    return a1*inv(a2*a3%P,P)%P;
16 }

```

8.9 Matrix.cpp

```

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        return true;
49    }
50    T gas(){
51        vector<T> lazy(r,1);
52        bool sign=false;
53        for(int i=0;i<r;++i){
54            if( m[i][i]==0 ){
55                int j=i+1;
56                while(j<r&&!m[j][i])j++;
57                if(j==r)continue;
58                m[i].swap(m[j]);
59                sign=!sign;
60            }
61            for(int j=0;j<r;j++){
62                if(i==j)continue;
63                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 };

```

8.10 MillerRobin.cpp

```

1 LL LLmul(LL a, LL b, const LL &mod) {
2     LL ans=0;
3     while(b) {
4         if(b&1) {
5             ans+=a;
6             if(ans>=mod) ans-=mod;
7         }
8         a<<=1, b>>=1;
9         if(a>=mod) a-=mod;
10    }
11    return ans;
12 }
13 LL mod_mul(LL a,LL b,LL m){
14    a%=m,b%=m; /* fast for m < 2^58 */
15    LL y=(LL)((double)a*b/m+.5);
16    LL r=(a*b-y*m)%m;
17    return r<0?r+m:r;
18 }
19 template<typename T>
20 T pow(T a,T b,T mod){/*a^b%mod
21    T ans=1;
22    for(;b;a=mod_mul(a,a,mod),b>>=1)
23        if(b&1)ans=mod_mul(ans,a,mod);
24    return ans;
25 }
26 int sprp[3]={2,7,61}; //int範圍可解

```



```

27 int llsprp
   [7]={2,325,9375,28178,450775,9780504,
28 1795265022}; //至少 unsigned long long 範圍
29 template<typename T>
30 bool isprime(T n,int *sprp,int num){
31     if(n==2)return 1;
32     if(n<2||n%2==0)return 0;
33     int t=0;
34     T u=n-1;
35     for(;u%2==0;++t)u>>=1;
36     for(int i=0;i<num;++i){
37         T a=sprp[i]%n;
38         if(a==0||a==1||a==n-1)continue;
39         T x=pow(a,u,n);
40         if(x==1||x==n-1)continue;
41         for(int j=0;j<t;++j){
42             x=mod_mul(x,x,n);
43             if(x==1)return 0;
44             if(x==n-1)break;
45         }
46         if(x==n-1)continue;
47         return 0;
48     }
49     return 1;
50 }

```

8.11 NTT.cpp

```

1 2615053605667*(2^18)+1,3
2 15*(2^27)+1,31
3 479*(2^21)+1,3
4 7*17*(2^23)+1,3
5 3*3*211*(2^19)+1,5
6 25*(2^22)+1,3
7 template<typename T,typename VT=vector<T> >
8 struct NTT{
9     const T P,G;
10     NTT(T p=(1<<23)*7*17+1,T g=3):P(p),G(g){}
11     unsigned bit_reverse(unsigned a,int len){
12         //Look FFT.cpp
13     }
14     T pow_mod(T n,T k,T m){
15         T ans=1;
16         for(n=(n>=m?n%m:n);k>>=1){
17             if(k&1)ans=ans*n%m;
18             n=n*n%m;
19         }
20         return ans;
21     }
22     void ntt(bool is_inv,VT &in,VT &out,int N)
23     {
24         int bitlen=__lg(N);
25         for(int i=0;i<N;++i)out[bit_reverse(i,
26             bitlen)]=in[i];
27         for(int step=2,id=1;step<=N;step<=1,++
28             id){
29             T wn=pow_mod(G,(P-1)>>id,P),wi=1,u,t;
30             const int mh=step>>1;
31             for(int i=0;i<mh;++i){
32                 for(int j=i;j<N;j+=step){
33                     u=out[j],t=wi*out[j+mh]%P;
34                     out[j]=u+t;
35                     out[j+mh]=u-t;

```

```

33         if(out[j]>=P)out[j]-=P;
34         if(out[j+mh]<0)out[j+mh]+=P;
35     }
36     wi=wi*wn%P;
37 }
38 }
39 if(is_inv){
40     for(int i=1;i<N/2;++i)swap(out[i],out[
41         N-i]);
42     T invn=pow_mod(N,P-2,P);
43     for(int i=0;i<N;++i)out[i]=out[i]*invn
44         %P;
45 }

```

8.12 Simpson.cpp

```

1 double simpson(double a,double b){
2     double c=a+(b-a)/2;
3     return (F(a)+4*F(c)+F(b))*(b-a)/6;
4 }
5 double asr(double a,double b,double eps,
6     double A){
7     double c=a+(b-a)/2;
8     double L=simpson(a,c),R=simpson(c,b);
9     if( abs(L+R-A)<15*eps )
10         return L+R+(L+R-A)/15.0;
11     return asr(a,c,eps/2,L)+asr(c,b,eps/2,R);
12 }
13 double asr(double a,double b,double eps){
14     return asr(a,b,eps,simpson(a,b));
15 }

```

8.13 外星模運算.cpp

```

1 //a[0]^(a[1]^a[2]^...)
2 #define maxn 1000000
3 int euler[maxn+5];
4 bool is_prime[maxn+5];
5 void init_euler(){
6     is_prime[1]=1; //一不是質數
7     for(int i=1;i<=maxn;i++)euler[i]=i;
8     for(int i=2;i<=maxn;i++){
9         if(!is_prime[i]){//是質數
10             euler[i]-=1;
11             for(int j=i<1;j<=maxn;j+=i){
12                 is_prime[j]=1;
13                 euler[j]=euler[j]/i*(i-1);
14             }
15         }
16     }
17 }
18 LL pow(LL a,LL b,LL mod){//a^b%mod
19     LL ans=1;
20     for(;b;a=a%mod,b>>=1)
21         if(b&1)ans=ans*a%mod;
22     return ans;
23 }

```

```

24 bool isless(LL *a,int n,int k){
25     if(*a==1)return k>1;
26     if(--n==0)return *a<k;
27     int next=0;
28     for(LL b=1;b<k;++next)
29         b*=a;
30     return isless(a+1,n,next);
31 }
32 LL high_pow(LL *a,int n,LL mod){
33     if(*a==1||--n==0)return *a%mod;
34     int k=0,r=euler[mod];
35     for(LL tma=1;tma!=pow(*a,k+r,mod);++k)
36         tma=tma*(a%mod);
37     if(isless(a+1,n,k))return pow(*a,high_pow(
38         a+1,n,k),mod);
39     int tmd=high_pow(a+1,n,r), t=(tmd-k+r)%r;
40     return pow(*a,k+t,mod);
41 }
42 LL a[1000005];
43 int t,mod;
44 int main(){
45     init_euler();
46     scanf("%d",&t);
47     #define n 4
48     while(t--){
49         for(int i=0;i<n;++i)scanf("%lld",&a[i]);
50         scanf("%d",&mod);
51         printf("%lld\n",high_pow(a,n,mod));
52     }
53     return 0;
54 }

```

8.14 質因數分解.cpp

```

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 pollorro(const LL n, const int c) { //循環
6     環節長度
7     LL a=1, b=1;
8     a=func(a,n,c)%n;
9     b=func(b,n,c)%n;
10    while(gcd(abs(a-b),n)==1) {
11        a=func(a,n,c)%n;
12        b=func(b,n,c)%n;
13    }
14    return gcd(abs(a-b),n);
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 void smallfactor(LL n, vector<LL> &v) {
25     if(n<MAXPRIME) {
26         while(isp[(int)n]) {
27             v.push_back(isp[(int)n]);
28             n/=isp[(int)n];
29         }
30         v.push_back(n);
31     } else {
32         for(int i=0;i<primecnt&&prime[i]*prime[i]
33             <=n;++i) {
34             while(n%prime[i]==0) {
35                 v.push_back(prime[i]);
36                 n/=prime[i];
37             }
38             if(n!=1) v.push_back(n);
39         }
40     }
41 }
42 void comfactor(const LL &n, vector<LL> &v) {
43     if(n<1e9) {
44         smallfactor(n,v);
45         return;
46     }
47     if(Isprime(n)) {
48         v.push_back(n);
49         return;
50     }
51     for(int i=0;i<n;++i)scanf("%lld",&a[i]);
52     scanf("%d",&mod);
53     printf("%lld\n",high_pow(a,n,mod));
54 }
55 }
56 comfactor(d,v);
57 comfactor(n/d,v);
58 }
59 }
60 void Factor(const LL &x, vector<LL> &v) {
61     LL n = x;
62     if(n==1) { puts("Factor 1"); return; }
63     prefactor(n,v);
64     if(n==1) return;
65     comfactor(n,v);
66     sort(v.begin(),v.end());
67 }
68 }
69 void AllFactor(const LL &n,vector<LL> &v) {
70     vector<LL> tmp;
71     Factor(n,tmp);
72     v.clear();
73     v.push_back(1);
74     int len;
75     LL now=1;
76     for(int i=0;i<tmp.size();++i) {
77         if(i==0 || tmp[i]!=tmp[i-1]) {
78             len = v.size();
79             now = 1;
80         }
81         now*=tmp[i];
82         for(int j=0;j<len;++j)
83             v.push_back(v[j]*now);
84     }
85 }
86 }

```

9 other

9.1 WhatDay.cpp

```
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
4         <3)
5         return (d+2*m+3*(m+1)/5+y+y/4+5)%7;
6     return (d+2*m+3*(m+1)/5+y+y/4-y/100+y/400)
7         %7;
8 }
```

9.2 上下最大正方形.cpp

```
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>d;d++){
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);
24 }
```

9.3 最大矩形.cpp

```
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.
13                first);
14        }
15        if(h>st.top().first){
```

```
15         st.push(make_pair(h,now.second));
16     }
17 }
18 return ans;
19 }
```

10 String

10.1 AC 自動機.cpp

```
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].
44                            fail;
45                        S[t].fail=~id?S[id].next[i]:0;
46                        S[t].efl=S[t].fail|.ed?S[t].fail:S
47                            [S[t].fail].efl;
48                        q.push_back(t);
49                    }
50                }
51            }
```

```
50 }
51 /*DP出每個前綴在字串s出現的次數並傳回所有
52 字串被s匹配成功的次數O(N*M)*/
53 int match_0(const char *s){
54     int ans=0,id,p=0,i;
55     for(i=0;s[i];++i){
56         id=s[i]-L;
57         while(!S[p].next[id]&&p)p=S[p].fail;
58         if(!S[p].next[id])continue;
59         p=S[p].next[id];
60         ++S[p].cnt_dp; /*匹配成功則它所有後綴都
61             可以被匹配(DP計算)*/
62     }
63     for(i=qe-1;i>=0;--i){
64         ans+=S[q[i]].cnt_dp*S[q[i]].ed;
65         if(~S[q[i]].fail)S[S[q[i]].fail].
66             cnt_dp+=S[q[i]].cnt_dp;
67     }
68     return ans;
69 }
70 /*多串匹配走efl邊並傳回所有字串被s匹配成功
71 的次數O(N*M^1.5)*/
72 int match_1(const char *s)const{
73     int ans=0,id,p=0,t;
74     for(int i=0;s[i];++i){
75         id=s[i]-L;
76         while(!S[p].next[id]&&p)p=S[p].fail;
77         if(!S[p].next[id])continue;
78         p=S[p].next[id];
79         if(S[p].ed)ans+=S[p].ed;
80         for(t=S[p].efl;~t;t=S[t].efl){
81             ans+=S[t].ed; /*因為都走efl邊所以保證
82                 匹配成功*/
83         }
84     }
85     return ans;
86 }
87 /*枚舉(s的子字串nA)的所有相異字串各恰一次
88 並傳回次數O(N*M^(1/3))*/
89 int match_2(const char *s){
90     int ans=0,id,p=0,t;
91     ++vt;
92     /*把戳記vt+=1，只要vt沒溢位，所有S[p].
93     vis==vt就會變成false
94     這種利用vt的方法可以O(1)歸零vis陣列*/
95     for(int i=0;s[i];++i){
96         id=s[i]-L;
97         while(!S[p].next[id]&&p)p=S[p].fail;
98         if(!S[p].next[id])continue;
99         p=S[p].next[id];
100         if(S[p].ed&&S[p].vis!=vt){
101             S[p].vis=vt;
102             ans+=S[p].ed;
103         }
104         for(t=S[p].efl;~t&&S[t].vis!=vt;t=S[t].
105             efl){
106             S[t].vis=vt;
107             ans+=S[t].ed; /*因為都走efl邊所以保證
108                 匹配成功*/
109         }
110     }
111     return ans;
112 }
```

```
104 /*把AC自動機變成真的自動機*/
105 void evolution(){
106     for(qs=1;qs!=qe;){
107         int p=q[qs++];
108         for(int i=0;i<=R-L;++i)
109             if(S[p].next[i]==0)S[p].next[i]=S[S[
110                 p].fail].next[i];
111     }
112 }
```

10.2 hash.cpp

```
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
16    為0 ~ Len-1*/
17    return (h[r+1]-(h[l]*h_base[r-l+1])%mod+
18        mod)%mod;
19 }
```

10.3 KMP.cpp

```
1 /*產生fail function*/
2 void kmp_fail(char *s,int len,int *fail){
3     int id=-1;
4     fail[0]=-1;
5     for(int i=1;i<len;++i){
6         while(~id&&s[id+1]!=s[i])id=fail[id];
7         if(s[id+1]==s[i])++id;
8         fail[i]=id;
9     }
10 }
11 /*以字串B匹配字串A，傳回匹配成功的數量(用B的
12 fail)*/
13 int kmp_match(char *A,int lenA,char *B,int
14     lenB,int *fail){
15     int id=-1,ans=0;
16     for(int i=0;i<lenA;++i){
17         while(~id&&B[id+1]!=A[i])id=fail[id];
18         if(B[id+1]==A[i])++id;
19         if(id==lenB-1){/*匹配成功*/
20             ++ans, id=fail[id];
21         }
22     }
23     return ans;
24 }
```

10.4 manacher.cpp

```

1 //原字串: asdsasdsa
2 //先把字串變成這樣: @#a#s#d#s#a#s#d#s#a#
3 inline void manacher(char *s,int len,int *z)
4 {
5     int l=0,r=0;
6     for(int i=1;i<len;++i){
7         z[i]=r>i?min(z[2*i-l],r-i):1;
8         while(s[i+z[i]]==s[i-z[i]]++)z[i];
9         if(z[i]+i>r)r=z[i]+i,l=i;
10    }

```

10.5 minimal_string_rotation.cpp

```

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){
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 }

```

10.6 reverseBWT.cpp

```

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])]+1;
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(const string &bw,int begin)
19 {
20     rankBWT(bw), firstCol();
21     int i = begin; //原本字串最後一個元素的位置
22     string res;
23     do{
24         char c = bw[i];

```

```

24     res = c + res;
25     i = first[int(c)] + ranks[i];
26 }while( i != begin );
27 return res;
28 }

```

10.7 suffix_array_lcp.cpp

```

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

```

10.8 Z.cpp

```

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?0:(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 }

```

11 Tarjan

11.1 dominator_tree.cpp

```

1 struct dominator_tree{
2     static const int MAXN=5005;
3     int n; // 1-base
4     vector<int> suc[MAXN],pre[MAXN];
5     int fa[MAXN],dfn[MAXN],id[MAXN],Time;
6     int semi[MAXN],idom[MAXN];
7     int anc[MAXN],best[MAXN]; //disjoint set
8     vector<int> dom[MAXN]; //dominator_tree
9     void init(int _n){
10        n=_n;
11        for(int i=1;i<=n;++i)suc[i].clear(),pre[i].clear();
12    }
13    void add_edge(int u,int v){
14        suc[u].push_back(v);
15        pre[v].push_back(u);
16    }
17    void dfs(int u){
18        dfn[u]=++Time,id[Time]=u;
19        for(auto v:suc[u]){
20            if(dfn[v])continue;
21            dfs(v),fa[dfn[v]]=dfn[u];
22        }
23    }
24    int find(int x){
25        if(x==anc[x])return x;
26        int y=find(anc[x]);
27        if(semi[best[x]]>semi[best[anc[x]]])best[x]=best[anc[x]];
28        return anc[x]=y;
29    }
30    void tarjan(int r){
31        Time=0;
32        for(int t=1;t<=n;++t){
33            dfn[t]=idom[t]=0; //u=r或是u無法到達r時
34            idom[id[u]]=0
35            dom[t].clear();
36            anc[t]=best[t]=semi[t]=t;
37        }
38        dfs(r);
39        for(int y=Time;y>=2;--y){
40            int x=fa[y],idy=id[y];
41            for(auto z:pre[idy]){
42                if(!dfn[z])continue;
43                find(z);
44                semi[y]=min(semi[y],semi[best[z]]);
45            }
46            dom[semi[y]].push_back(y);
47            anc[y]=x;
48            for(auto z:dom[x]){
49                find(z);
50                idom[z]=semi[best[z]]<x?best[z]:x;
51            }
52            dom[x].clear();
53        }
54        for(int u=2;u<=Time;++u){
55            if(idom[u]!=semi[u])idom[u]=idom[idom[u]];
56            dom[id[idom[u]]].push_back(id[u]);
57        }

```

```

56     }
57     }
58 }dom;

```

11.2 tnfsb017_2_sat.cpp

```

1 #include<bits/stdc++.h>
2 using namespace std;
3 #define MAXN 8001
4 #define MAXN2 MAXN*4
5 #define n(X) ((X)+2*N)
6 vector<int> v[MAXN2], rv[MAXN2], vis_t;
7 int N,M;
8 void addedge(int s,int e){
9     v[s].push_back(e);
10    rv[e].push_back(s);
11 }
12 int scc[MAXN2];
13 bool vis[MAXN2]={false};
14 void dfs(vector<int> *uv,int n,int k=-1){
15     vis[n]=true;
16     for(int i=0;i<uv[n].size();++i)
17         if(!vis[uv[n][i]])
18             dfs(uv,uv[n][i],k);
19     if(uv==v)vis_t.push_back(n);
20     scc[n]=k;
21 }
22 void solve(){
23     for(int i=1;i<=N;++i){
24         if(!vis[i])dfs(v,i);
25         if(!vis[n(i)])dfs(v,n(i));
26     }
27     memset(vis,0,sizeof(vis));
28     int c=0;
29     for(int i=vis_t.size()-1;i>=0;--i)
30         if(!vis[vis_t[i]])
31             dfs(rv,vis_t[i],c++);
32 }
33 int main(){
34     int a,b;
35     scanf("%d%d",&N,&M);
36     for(int i=1;i<=N;++i){
37         // (A or B)&(!A & !B) A^B
38         a=i*2-1;
39         b=i*2;
40         addedge(n(a),b);
41         addedge(n(b),a);
42         addedge(a,n(b));
43         addedge(b,n(a));
44     }
45     while(M--){
46         scanf("%d",&a,&b);
47         a = a>0?a*2-1:-a*2;
48         b = b>0?b*2-1:-b*2;
49         // A or B
50         addedge(n(a),b);
51         addedge(n(b),a);
52     }
53     solve();
54     bool check=true;
55     for(int i=1;i<=2*N;++i)
56         if(scc[i]==scc[n(i)])
57             check=false;

```

```

58 if(check){
59     printf("%d\n",N);
60     for(int i=1;i<=2*N;i+=2){
61         if(scc[i]>scc[i+2*N]) putchar('+');
62         else putchar('-');
63     }
64     puts("");
65 }else puts("0");
66 return 0;
67 }

```

11.3 橋連通分量.cpp

```

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 inline void add_edge(int u,int v){
13     G[u].push_back(E.size());
14     E.push_back(edge(u,v));
15     G[v].push_back(E.size());
16     E.push_back(edge(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(size_t i=0;i<G[u].size();++i){
23         int e=G[u][i];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
                所在的BCC
37             while(v!=u);
38         }
39     }
40     inline 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 }

```

11.4 雙連通分量 & 割點.cpp

```

1 #define N 1005
2 vector<int> G[N]; // 1-base
3 vector<int> bcc[N]; // 存每塊雙連通分量的點
4 int low[N],vis[N],Time;
5 int bcc_id[N],bcc_cnt; // 1-base
6 bool is_cut[N]; // 是否為割點
7 int st[N],top;
8 void dfs(int u,int pa=-1){ // u當前點, pa父親
9     int v,child=0;
10     low[u]=vis[u]=++Time;
11     st[top++]=u;
12     for(size_t i=0;i<G[u].size();++i){
13         if(!vis[v=G[u][i]]){
14             dfs(v,u),++child;
15             low[u]=min(low[u],low[v]);
16             if(vis[u]<=low[v]){
17                 is_cut[u]=1;
18                 bcc[++bcc_cnt].clear();
19                 int t;
20                 do{
21                     bcc_id[t=st[--top]]=bcc_cnt;
22                     bcc[bcc_cnt].push_back(t);
23                 }while(t!=v);
24                 bcc_id[u]=bcc_cnt;
25                 bcc[bcc_cnt].push_back(u);
26             }
27             }else if(vis[v]<vis[u]&&v!=pa){ // 反向邊
28                 low[u]=min(low[u],vis[v]);
29             }
30             if(pa!=-1&&child<2)is_cut[u]=0; // u是dfs樹
                的根要特判
31         }
32     inline void bcc_init(int n){
33         Time=bcc_cnt=top=0;
34         for(int i=1;i<=n;++i){
35             G[i].clear();
36             is_cut[i]=vis[i]=bcc_id[i]=0;
37         }
38     }

```

12 Tree_problem

12.1 HeavyLight.cpp

```

1 #include<vector>
2 #define MAXN 100005
3 int siz[MAXN],max_son[MAXN],pa[MAXN],dep[
    MAXN];
4 int link_top[MAXN],link[MAXN],cnt;
5 vector<int> G[MAXN];
6 void find_max_son(int u){
7     siz[u]=1;
8     max_son[u]=-1;

```

```

9     for(auto v:G[u]){
10         if(v==pa[u])continue;
11         pa[v]=u;
12         dep[v]=dep[u]+1;
13         find_max_son(v);
14         if(max_son[u]==-1||siz[v]>siz[max_son[u]]
            )max_son[u]=v;
15         siz[u]+=siz[v];
16     }
17 }
18 void build_link(int u,int top){
19     link[u]=++cnt;
20     link_top[u]=top;
21     if(max_son[u]==-1)return;
22     build_link(max_son[u],top);
23     for(auto v:G[u]){
24         if(v==max_son[u]||v==pa[u])continue;
25         build_link(v,v);
26     }
27 }
28 int find_lca(int a,int b){
29     // 求LCA, 可以在過程中對區間進行處理
30     int ta=link_top[a],tb=link_top[b];
31     while(ta!=tb){
32         if(dep[ta]<dep[tb]){
33             swap(ta,tb);
34             swap(a,b);
35         }
36         // 這裡可以對a所在的鏈做區間處理
37         // 區間為(Link[ta],Link[a])
38         ta=link_top[a=pa[ta]];
39     }
40     // 最後a,b會在同一條鏈, 若a!=b還要在進行一
        次區間處理
41     return dep[a]<dep[b]?a:b;
42 }

```

12.2 LCA.cpp

```

1 #define MAXN 100000
2 #define MAX_LOG 17
3 int pa[MAX_LOG+1][MAXN+5];
4 int dep[MAXN+5];
5 vector<int> G[MAXN+5];
6 void dfs(int x,int p){ // dfs(1,-1);
7     pa[0][x]=p;
8     for(int i=0;i+1<MAX_LOG;++i)pa[i+1][x]=pa[i]
        [pa[i][x]];
9     for(auto &i:G[x]){
10         if(i==p)continue;
11         dep[i]=dep[x]+1;
12         dfs(i,x);
13     }
14 }
15 inline int jump(int x,int d){
16     for(int i=0;i<d;++i)if((x>>i)&1)x=pa[i][x];
17     return x;
18 }
19 inline int find_lca(int a,int b){
20     if(dep[a]>dep[b])swap(a,b);
21     b=jump(b,dep[b]-dep[a]);
22     if(a==b)return a;

```

```

23     for(int i=MAX_LOG;i>=0;--i){
24         if(pa[i][a]!=pa[i][b]){
25             a=pa[i][a];
26             b=pa[i][b];
27         }
28     }
29     return pa[0][a];
30 }

```

12.3 link_cut_tree.cpp

```

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> node;
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
    tree的根
10     return node[node[x].pa].ch[0]!=x&&node[
        node[x].pa].ch[1]!=x;
11 }
12 void down(int x){ // 懶惰標記下推
13     if(node[x].rev){
14         if(node[x].ch[0])node[node[x].ch[0]].rev
            ^=1;
15         if(node[x].ch[1])node[node[x].ch[1]].rev
            ^=1;
16         std::swap(node[x].ch[0],node[x].ch[1]);
17         node[x].rev^=1;
18     }
19 }
20 void push_down(int x){ // 將所有祖先的懶惰標記
    下推
21     if(!isroot(x))push_down(node[x].pa);
22     down(x);
23 }
24 void up(int x){ // 將子節點的資訊向上更新
25     void rotate(int x){ // 旋轉, 會自行判斷轉的方
        向
26         int y=node[x].pa,z=node[y].pa,d=(node[y].
            ch[1]==x);
27         node[x].pa=z;
28         if(!isroot(y))node[z].ch[(node[z].ch[1]==y
            )?0:1]=x;
29         node[y].ch[d]=node[x].ch[d^1];
30         node[node[y].ch[d]].pa=y;
31         node[y].pa=x,node[x].ch[d^1]=y;
32         up(y),up(x);
33     }
34     void splay(int x){ // 將節點x伸展到所在splay
        tree的根
35         push_down(x);
36         while(!isroot(x)){
37             int y=node[x].pa;
38             if(!isroot(y)){
39                 int z=node[y].pa;
40                 if((node[z].ch[0]==y)^(node[y].ch[0]==
                    x))rotate(y);

```



```

41     else rotate(x);
42 }
43 rotate(x);
44 }
45 }
46 int access(int x){
47     int last=0;
48     while(x){
49         splay(x);
50         node[x].ch[1]=last;
51         up(x);
52         last=x;
53         x=node[x].pa;
54     }
55     return last; // 傳回access後splay tree的根
56 }
57 void access(int x, bool is=0){ // is=0就是一般
    的access
58     int last=0;
59     while(x){
60         splay(x);
61         if(is&&!node[x].pa){
62             // printf("%d\n", max(node[last].ma, node
63             [node[x].ch[1]].ma));
64         }
65         node[x].ch[1]=last;
66         up(x);
67         last=x;
68         x=node[x].pa;
69     }
70 void query_edge(int u, int v){
71     access(u);
72     access(v, 1);
73 }
74 void make_root(int x){
75     access(x), splay(x);
76     node[x].rev^=1;
77 }
78 void make_root(int x){
79     node[access(x)].rev^=1;
80     splay(x);
81 }
82 void cut(int x, int y){
83     make_root(x);
84     access(y);
85     splay(y);
86     node[y].ch[0]=0;
87     node[x].pa=0;
88 }
89 void cut_parents(int x){
90     access(x);
91     splay(x);
92     node[node[x].ch[0]].pa=0;
93     node[x].ch[0]=0;
94 }
95 void link(int x, int y){
96     make_root(x);
97     node[x].pa=y;
98 }
99 int find_root(int x){
100     x=access(x);
101     while(node[x].ch[0]) x=node[x].ch[0];
102     splay(x);
103     return x;

```

```

104 }
105 int query(int u, int v){
106     // 傳回uv路徑splay tree的根結點
107     // 這種寫法無法求LCA
108     make_root(u);
109     return access(v);
110 }
111 int query_lca(int u, int 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 node[lca].data+node[node[lca].
119         ch[1]].sum
120     }else{
121         // return node[lca].data+node[node[lca].
122         ch[1]].sum+node[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     不會壞掉
137     queue<int> q;
138     for(int i=1; i<=n; ++i) pa[i]=0;
139     q.push(root);
140     while(q.size()){
141         int u=q.front();
142         q.pop();
143         for(int i=0; i<(int)G[u].size(); ++i){
144             int v=G[u][i].first;
145             if(v!=pa[u]){
146                 pa[v]=u;
147                 node[v].pa=u;
148                 node[v].data=e[G[u][i].second].w;
149                 edge_node[G[u][i].second]=v;
150                 up(v);
151                 q.push(v);
152             }
153         }
154     }
155 }

```

12.4 POJ_tree.cpp

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

```

2 using namespace std;
3 #define MAXN 10005
4 int n, k;
5 vector<pair<int, int> >g[MAXN];
6 int size[MAXN];
7 bool vis[MAXN];
8 inline void init(){
9     for(int i=0; i<=n; ++i){
10         g[i].clear();
11         vis[i]=0;
12     }
13 }
14 void get_dis(vector<int> &dis, int u, int pa,
    int d){
15     dis.push_back(d);
16     for(size_t i=0; i<g[u].size(); ++i){
17         int v=g[u][i].first, w=g[u][i].second;
18         if(v!=pa&&!vis[v]) get_dis(dis, v, u, d+w);
19     }
20 }
21 vector<int> dis; // 這東西如果放在函數裡會TLE
22 int cal(int u, int d){
23     dis.clear();
24     get_dis(dis, u, -1, d);
25     sort(dis.begin(), dis.end());
26     int l=0, r=dis.size()-1, res=0;
27     while(l<r){
28         while(l<r&&dis[l]+dis[r]>k)--r;
29         res+=r-(l++);
30     }
31     return res;
32 }
33 pair<int, int> tree_centroid(int u, int pa,
    const int sz){
34     size[u]=1; // 找樹重心，second是重心
35     pair<int, int> res(INT_MAX, -1);
36     int ma=0;
37     for(size_t i=0; i<g[u].size(); ++i){
38         int v=g[u][i].first;
39         if(v==pa||vis[v]) continue;
40         res=min(res, tree_centroid(v, u, sz));
41         size[u]+=size[v];
42         ma=max(ma, size[v]);
43     }
44     ma=max(ma, sz-size[u]);
45     return min(res, make_pair(ma, u));
46 }
47 int tree_DC(int u, int sz){
48     int center=tree_centroid(u, -1, sz).second;
49     int ans=cal(center, 0);
50     vis[center]=1;
51     for(size_t i=0; i<g[center].size(); ++i){
52         int v=g[center][i].first, w=g[center][i].
53         second;
54         if(vis[v]) continue;
55         ans+=cal(v, w);
56         ans+=tree_DC(v, size[v]);
57     }
58     return ans;
59 }
60 int main(){
61     while(scanf("%d%d", &n, &k), n||k){
62         init();
63         for(int i=1; i<n; ++i){
64             int u, v, w;

```

```

64         scanf("%d%d%d", &u, &v, &w);
65         g[u].push_back(make_pair(v, w));
66         g[v].push_back(make_pair(u, w));
67     }
68     printf("%d\n", tree_DC(1, n));
69 }
70 return 0;
71 }

```

13 zformula

13.1 formula.tex

13.1.1 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

13.1.2 圖論

- $V - E + F = 2$
- 對於平面圖， $F = E - V + n + 1$ ， n 是連通分量
- 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：

- $C(u, V) = \infty, (u, v) \in E$
- $C(S, v) = W_v, W_v > 0$
- $C(v, T) = -W_v, W_v < 0$

- 最大密度子圖：

- $C(u, v) = 1, (u, v) \in E$
- $C(S, v) = U_v, v \in V$
- $C(v, T) = U + 2g - d_v, v \in V$

- 弦圖：

- 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的排序
- 區間圖染色：用線段樹做

```

1 double l=0,=m,stop=1.0/n/n;
2 while(r-l>=stop){
3     double(mid);
4     if((n*m-sol.maxFlow(s,t))/2>eps)l=mid;
5     else r=mid;
6 }
7 build(1);
8 sol.maxFlow(s,t);
9 vector<int> ans;
10 for(int i=1;i<=n;++i)
11     if(sol.vis[i])ans.push_back(i);

```

13.1.3 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 $M(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$

13.1.4 基本數論

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

13.1.5 排組公式

- k 卡特蘭 $\frac{C_n^{kn}}{n(k-1)+1} \cdot C_m^n = \frac{n!}{m!(n-m)!}$
- $H(n, m) \cong x_1 + x_2 \dots + x_n = k, num = C_k^{n+k-1}$
- 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
 - $B_{p^m+n} \equiv mB_n + B_{n+1} \pmod{p}$, 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
 - $D_n = n!(1 - \frac{1}{1!} + \frac{1}{2!} - \frac{1}{3!} \dots + (-1)^n \frac{1}{n!})$
 - $D_n = (n-1)(D_{n-1} + D_{n-2}), D_0 = 1, D_1 = 0$

(c) From $D_0 : 1, 0, 1, 2, 9, 44, 265, 1854, 14833, 133496$

6. Binomial Equality

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

13.1.6 幕次, 幕次和

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

13.1.7 Burnside's lemma

- $|X/G| = \frac{1}{|G|} \sum_{g \in G} |X^g|$
- $X^g = t^{c(g)}$
- 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 · $c(g)$ 是循環環不動的面數。
- 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 3×3^4 · 120(角) 有 8×3^2 · 180(邊) 有 6×3^3 · 全部 $\frac{1}{24} (3^6 + 6 \times 3^3 + 3 \times 3^4 + 8 \times 3^2 + 6 \times 3^3) = \frac{24}{57}$

13.1.8 Count on a tree

- Rooted tree: $s_{n+1} = \frac{1}{n} \sum_{i=1}^n (i \times a_i \times \sum_{j=1}^{\lfloor n/i \rfloor} a_{n+1-i \times j})$
- Unrooted tree:
 - Odd: $a_n - \sum_{i=1}^{n/2} a_i a_{n-i}$
 - Even: $Odd + \frac{1}{2} a_{n/2} (a_{n/2} + 1)$

3. Spanning Tree

- 完全圖 $n^n - 2$
- 一般圖 (Kirchhoff's theorem) $M[i][i] = \text{degree}(V_i), M[i][j] = -1, \text{if have } E(i, j), 0 \text{ if no edge. delete any one row and col in } A, \text{ans} = \det(A)$

13.2 java.tex

13.2.1 文件操作

```

1 import java.io.*;
2 import java.util.*;
3 import java.math.*;
4 import java.text.*;
5
6 public class Main{
7
8     public static void main(String args[]){
9         throws FileNotFoundException,
10         IOException
11         Scanner sc = new Scanner(new FileReader(
12             "a.in"));
13         PrintWriter pw = new PrintWriter(new
14             FileWriter("a.out"));
15         int n,m;
16         n=sc.nextInt();//读入下一个INT
17         m=sc.nextInt();
18
19         for(ci=1; ci<=c; ++ci){
20             pw.println("Case #"+ci+": easy for
21                 output");
22         }
23     }
24 }

```

13.2.2 优先队列

```

1 PriorityQueue queue = new PriorityQueue( 1,
2     new Comparator(){
3         public int compare( Point a, Point b ){
4             if( a.x < b.x || a.x == b.x && a.y < b.y )
5                 return -1;
6             else if( a.x == b.x && a.y == b.y )
7                 return 0;
8         }
9     });

```

13.2.3 Map

```

1 Map map = new HashMap();
2 map.put("sa", "dd");
3 String str = map.get("sa").toString();
4
5 for(Object obj : map.keySet()){
6     Object value = map.get(obj);
7 }

```

13.2.4 sort

```

1 static class cmp implements Comparator{
2     public int compare(Object o1, Object o2){
3         BigInteger b1=(BigInteger)o1;
4         BigInteger b2=(BigInteger)o2;
5         return b1.compareTo(b2);
6     }
7 }
8 public static void main(String[] args)
9     throws IOException{
10     Scanner cin = new Scanner(System.in);
11     int n;
12     n=cin.nextInt();
13     BigInteger[] seg = new BigInteger[n];
14     for (int i=0;i<n;i++)
15         seg[i]=cin.nextBigInteger();
16     Arrays.sort(seg, new cmp());
17 }

```

ACM ICPC TEAM REFERENCE - NTHU JINKELA

Contents

1 Computational_Geometry	1	3 default	5	7 Linear_Programming	10	10.7 suffix_array_lcp.cpp	15
1.1 Geometry.cpp	1	3.1 debug.cpp	5	7.1 最大密度子圖.cpp	10	10.8 Z.cpp	15
1.2 SmallestCircle.cpp	3	3.2 ext.cpp	5	8 Number_Theory	10		
1.3 最近點對.cpp	3	3.3 IncStack.cpp	5	8.1 basic.cpp	10		
2 Data_Structure	3	3.4 input.cpp	6	8.2 bit_set.cpp	11		
2.1 DLX.cpp	3	4 Flow	6	8.3 cantor_expansion.cpp	11		
2.2 Dynamic_KD_tree.cpp	4	4.1 dinic.cpp	6	8.4 FFT.cpp	11		
2.3 kd_tree_replace_segment_tree.cpp	4	4.2 ISAP_with_cut.cpp	6	8.5 find_real_root.cpp	12		
2.4 reference_point.cpp	5	4.3 MinCostMaxFlow.cpp	6	8.6 FWT.cpp	12		
2.5 skew_heap.cpp	5	5 Graph	7	8.7 LinearCongruence.cpp	12		
2.6 undo_disjoint_set.cpp	5	5.1 Augmenting_Path.cpp	7	8.8 Lucas.cpp	12		
2.7 整體二分.cpp	5	5.2 Augmenting_Path_multiple.cpp	7	8.9 Matrix.cpp	12		
		5.3 blossom_matching.cpp	7	8.10 MillerRobin.cpp	12		
		5.4 graphISO.cpp	7	8.11 NTT.cpp	13		
		5.5 KM.cpp	7	8.12 Simpson.cpp	13		
		5.6 MaximumClique.cpp	8	8.13 外星模運算.cpp	13		
		5.7 MinimumMeanCycle.cpp	8	8.14 質因數分解.cpp	13		
		5.8 Rectilinear_MST.cpp	8	9 other	14		
		5.9 treeISO.cpp	8	9.1 WhatDay.cpp	14		
		5.10 一般圖最小權完美匹配.cpp	8	9.2 上下最大正方形.cpp	14		
		5.11 全局最小割.cpp	9	9.3 最大矩形.cpp	14		
		5.12 平面圖判定.cpp	9	10 String	14		
		5.13 弦圖完美消除序列.cpp	9	10.1 AC 自動機.cpp	14		
		5.14 最小斯坦納樹 DP.cpp	9	10.2 hash.cpp	14		
		5.15 最小樹形圖 朱劉.cpp	9	10.3 KMP.cpp	14		
		5.16 穩定婚姻模板.cpp	10	10.4 manacher.cpp	15		
		6 language	10	10.5 minimal_string_rotation.cpp	15		
		6.1 CNF.cpp	10	10.6 reverseBWT.cpp	15		
						11 Tarjan	15
						11.1 dominator_tree.cpp	15
						11.2 tnfsb017_2_sat.cpp	15
						11.3 橋連通分量.cpp	16
						11.4 雙連通分量 & 割點.cpp	16
						12 Tree_problem	16
						12.1 HeavyLight.cpp	16
						12.2 LCA.cpp	16
						12.3 link_cut_tree.cpp	16
						12.4 POJ_tree.cpp	17
						13 zformula	17
						13.1 formula.tex	17
						13.1.1 Pick 公式	17
						13.1.2 圖論	17
						13.1.3 學長公式	18
						13.1.4 基本數論	18
						13.1.5 排組公式	18
						13.1.6 冪次, 冪次和	18
						13.1.7 Burnside's lemma	18
						13.1.8 Count on a tree	18
						13.2 java.tex	18
						13.2.1 文件操作	18
						13.2.2 优先队列	18
						13.2.3 Map	18
						13.2.4 sort	18