

1 HDU 2063 二分图匹配_HK

```
2  #include<bits/stdc++.h>
3  using namespace std;
4  //#include<ext/rope>
5  //using namespace __gnu_cxx
6  //#include<ext/pb_ds/priority_queue.hpp>
7  //using namespace __gnu_pbds;
8  #define lowbit(x) (x&-x)
9  #define pb(x) push_back(x)
10 #define all(x) (x).begin(),(x).end()
11 #define clr(a,b) memset(a,b,sizeof(a))
12 #define caze(T) for(scanf("%d",&T);T;T--)
13 #define debug cout<<"???"<<endl
14 #define inf (1<<30)
15 #define endl ('\n')
16 #define ll long long
17 #define pii pair<int,int>
18 #define ull unsigned long long
19 #define IOS ios::sync_with_stdio(0),cin.tie(0),cout.tie(0)
20 const int maxp=5e5+7;
21 const int maxn=4e3+7;
22 struct EDGE{int v,nxt;}edge[1000010];
23 int tot;int head[maxn];
24 void AE(int u,int v){edge[tot]={v,head[u]},head[u]=tot++;}
25 int mx[maxn],my[maxn];
26 int dx[maxn],dy[maxn];
27 bool vis[maxn];
28 int nx,ny;
29 int dis;
30 bool bfs()
31 {
32     queue<int>q;
33     dis=inf;
34     clr(dx,-1);
35     clr(dy,-1);
36     for(int i=1;i<=nx;++i)
37         if(mx[i]==-1)
38             q.push(i),dx[i]=0;
39     while(!q.empty())
40     {
41         int u=q.front();q.pop();
42         if(dx[u]>dis) break;
```

```

43         for(int i=head[u],v;~i;i=edge[i].nxt)
44         {
45             v=edge[i].v;
46             if(dy[v]==-1)
47             {
48                 dy[v]=dx[u]+1;
49                 if(my[v]==-1)
50                     dis=dy[v];
51                 else
52                 {
53                     dx[my[v]]=dy[v]+1;
54                     q.push(my[v]);
55                 }
56             }
57         }
58     }
59     return dis!=inf;
60 }
61 bool dfs(int u)
62 {
63     for(int i=head[u],v;~i;i=edge[i].nxt)
64     {
65         v=edge[i].v;
66         if(vis[v]||dy[v]!=dx[u]+1) continue;
67         vis[v]=1;
68         if(my[v]!=-1&&dy[v]==dis) continue;
69         if(my[v]==-1||dfs(my[v]))
70         {
71             my[v]=u;
72             mx[u]=v;
73             return 1;
74         }
75     }
76     return 0;
77 }
78 int HK()
79 {
80     int ret=0;
81     clr(mx,-1);
82     clr(my,-1);
83     while(bfs())
84     {
85         clr(vis,0);
86         for(int i=1;i<=nx;++i)

```

```

87             ret+=(mx[i]==-1&&dfs(i));
88         }
89         return ret;
90     }
91     int main()
92     {
93         int k;
94         while(~scanf("%d",&k)&&k)
95         {
96             scanf("%d%d",&nx,&ny);
97             tot=0;clr(head,-1);
98             for(int i=0,u,v;i<k;++i)
99             {
100                 scanf("%d%d",&u,&v);
101                 AE(u,v);
102             }
103             printf("%d\n",HK());
104         }
105     }
106
107

```

108

109

110

111

112

113

114

115

```

116 二分图匹配_西方算法(匈牙利)
117 const int maxn=xxx;
118 vector<int>v[maxn];
119 bool used[maxn];
120 int lef[maxn];
121 void ini()
122 {
123     for(int i=1;i<=H;++i)
124         v[i].clear();
125     clr(lef,-1);
126 }
127 bool dfs(int x)
128 {
129     for(auto c:v[x])
130     {
131         if(used[c]==0)
132         {
133             used[c]=1;
134             if(lef[c]==-1||dfs(lef[c]))
135             {
136                 lef[c]=x;
137                 return 1;
138             }
139         }
140     }
141     return 0;
142 }
143 int solve(int n)
144 {
145     int ret=0;
146     for(int i=1;i<=n;++i)
147     {
148         clr(used,-1);
149         ret+=dfs(i);
150     }
151     return ret;
152 }
153
154
155

```

156 奔小康赚大钱,KM 裸

```
157 #include<bits/stdc++.h>
158 using namespace std;
159 #define clr(a,b) memset(a,b,sizeof(a))
160 #define ll long long
161 #define ull unsigned long long
162 #define lowbit(x) (x&-x)
163 #define pb(x) push_back(x)
164 #define IOS ios::sync_with_stdio(0),cin.tie(0),cout.tie(0)
165 #define inf (1<<30)
166 #define Endl ('\n')
167
168 const int N=333;
169
170 int n,nx,ny;
171 int link[N],lx[N],ly[N],slack[N];
172 int visx[N],visy[N],w[N][N];
173 bool dfs(int x)
174 {
175     visx[x]=1;
176     for(int y=0;y<ny;++y)
177     {
178         if(visy[y])
179             continue;
180         int tp=lx[x]+ly[y]-w[x][y];
181         if(tp==0)
182         {
183             visy[y]=1;
184             if(link[y]==-1||dfs(link[y]))
185             {
186                 link[y]=x;
187                 return 1;
188             }
189         }
190         else if(slack[y]>tp)
191             slack[y]=tp;
192     }
193     return 0;
194 }
195 int KM()
196 {
197     clr(link,-1);
198     clr(ly,0);
```

```

199     for(int i=0;i<nx;++i)
200     {
201         lx[i]=-inf;
202         for(int j=0;j<ny;++j)
203             if(w[i][j]>lx[i])
204                 lx[i]=w[i][j];
205     }
206     for(int x=0;x<nx;++x)
207     {
208         for(int i=0;i<ny;++i)
209             slack[i]=inf;
210         while(1)
211         {
212             clr(visx,0);
213             clr(visy,0);
214             if(dfs(x))
215                 break;
216             int d=inf;
217             for(int i=0;i<ny;++i)
218                 if(!visy[i]&&d>slack[i])
219                     d=slack[i];
220             for(int i=0;i<nx;++i)
221                 if(visx[i])
222                     lx[i]-=d;
223             for(int i=0;i<ny;++i)
224                 if(visy[i])
225                     ly[i]+=d;
226             else
227                 slack[i]-=d;
228         }
229     }
230     int ret=0;
231     for(int i=0;i<ny;++i)
232         if(link[i]!=-1)
233             ret+=w[link[i]][i];
234     return ret;
235 }
236 void solve(int n)
237 {
238     for(int i=0;i<n;++i)
239         for(int j=0;j<n;++j)
240             scanf("%d",&w[i][j]);
241     nx=ny=n;
242     printf("%d\n",KM());

```

```
243 }
244 int main()
245 {
246     int n;
247     while(~scanf("%d",&n))
248         solve(n);
249 }
```

250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286

计算几何

287

288 圆和多边形面积交:HDU-5462 给出个圆,给个多边形,求交

289 集面积.输入是线,这题要判断线的方向.

```
290 #include<bits/stdc++.h>
291 #include<ext/rope>
292 using namespace std;
293 #define clr(a,b) memset(a,b,sizeof(a))
294 #define ll long long
295 #define ull unsigned long long
296 #define lowbit(x) (x&-x)
297 #define pb(x) push_back(x)
298 #define IOS ios::sync_with_stdio(0),cin.tie(0),cout.tie(0)
299 #define inf (1<<30)
300 #define caze(T) for(scanf("%d",&T);T;T--)
301 #define Endl ('\n')
302 const double pi=acos(-1.0);
303 const double eps=1e-8;
304 int dcmp(double x){return fabs(x)<=eps?0:(x<0?-1:1);}
305 double sqr(double x){return x*x;}
306 struct point
307 {
308     double x,y,id;
309     point(){}
310     point(double x,double y,int id=-1):x(x),y(y),id(id) {}
311     point operator-(const point w)const {return point(x-
312 w.x,y-w.y);}
313     point operator+(const point w)const {return
314 point(x+w.x,y+w.y);}
315     double operator*(const point& w)const {return
316 x*w.x+y*w.y;}
317     point operator*(double a) {return point(x*a,y*a);}
318     double operator^(const point& w)const {return x*w.y-
319 y*w.x;}
320     point operator/(double a) {return point(x/a,y/a);}
321     friend ostream &operator<<(ostream& out,const point& w)
322 {out<<'('<<w.x<<','<<w.y<<')';return out;}
323     void input(){scanf("%lf%lf",&x,&y);}
324     double len2(){return x*x+y*y;}
325     double len(){return sqrt(x*x+y*y);}
326     point change_len(double r)
```



```

327     {
328         double l=len();
329         if(dcmp(l)==0) return *this;
330         r/=l;
331         return point(x*r,y*r);
332     }
333 };
334 inline double cross(const point& A,const point& B){return
335 A.x*B.y-B.x*A.y;}
336 inline double dot(const point& q,const point& w){return
337 q.x*w.x+q.y*w.y;}
338 inline double Xmul(const point& A,const point& B,const point&
339 C){return cross(C-A,B-A);}
340 inline double dis(const point& q,const point& w){return
341 sqrt(dot(q-w,q-w));}
342 inline double rad(const point& A,const point& B){return
343 fabs(atan2(fabs(cross(A,B)),dot(A,B)));}
344 int Andrew(int n,point *st,point *ed)
345 {
346     sort(st,st+n,[](const point& A,const point&
347 B)->bool{return A.x==B.x?A.y<B.y:A.x<B.x;});
348     int tot=0;
349     for (int i = 0; i < n; ++i)
350     {
351         while(tot>1&&cross(ed[tot-1]-ed[tot-2],st[i]-
352 ed[tot-2])<=0) tot--;
353         ed[tot++]=st[i];
354     }
355     int tp=tot;
356     for (int i = n - 2; ~i; --i)
357     {
358         while(tot>tp&&cross(ed[tot-1]-ed[tot-2],st[i]-
359 ed[tot-2])<=0) tot--;
360         ed[tot++]=st[i];
361     }
362     tot--(n>1);
363     return tot;
364 }
365 double Area(int n,point *p)
366 {
367     double S=0;
368     for (int i = 1; i < n - 1; ++i)
369         S+=fabs(Xmul(p[0],p[i],p[i+1]));
370     return S/2;

```

```

371 }
372 struct Line
373 {
374     point u,v;
375     double k;
376     Line(){}
377     Line(point u,point v):u(u),v(v){k=atan2(v.y-u.y,v.x-
378 u.x);}
379     Line(point u,double k):u(u),k(k){v=u+(dcmp(k-
380 pi/2)?point(1,tan(k)):point(0,1));}
381     void input(){u.input();v.input();get_angle();}
382     void get_angle(){k=atan2(v.y-u.y,v.x-u.x);}
383     double len(){return dis(u,v);}
384     double pdis(point w) {return fabs(cross(w-u,v-
385 u)/len());}
386     point operator&(const Line& b)const
387     {
388         point ret=u;
389         double t=(cross(u-b.u,b.u-b.v))/cross(u-v,b.u-
390 b.v);
391         ret.x+=(v.x-u.x)*t;
392         ret.y+=(v.y-u.y)*t;
393         return ret;
394     }
395     point project(const point& w)const{return u+(((v-u)*((v-
396 u)*(w-u)))/(v-u).len2());}
397     friend ostream &operator<<(ostream &out,const Line&
398 w){out<<w.u<<"->"<<w.v;return out;}
399 };
400 Line Q[100010];
401 void Hpi(int n,Line *line,point *res,int &resn)
402 {
403     for (int i = 0; i < n; ++i) line[i].get_angle();
404     int tot=n;
405     sort(line,line+n,[](const Line& A,const Line&
406 B)->bool{return fabs(A.k-B.k)>eps?A.k<B.k:cross(A.u-B.u,B.v-
407 B.u)<0;});
408     tot=1;
409     for (int i = 1; i < n; ++i)
410         if(fabs(line[i].k-line[i-1].k)>eps)
411             line[tot++]=line[i];
412     int head=0,tail=1;
413     Q[0]=line[0];
414     Q[1]=line[1];

```

```

415         resn=0;
416         for (int i = 2; i < tot; ++i)
417         {
418             if(fabs(cross(Q[tail].v-Q[tail].u,Q[tail-1].v-
419 Q[tail-1].u))<eps||fabs(cross(Q[head].v-Q[head].u,Q[head+1].v-
420 Q[head+1].u))<eps)
421                 return;
422             while(head<tail&&(cross((Q[tail]&Q[tail-1])-
423 line[i].u,line[i].v-line[i].u))>eps) tail--;
424             while(head<tail&&(cross((Q[head]&Q[head+1])-
425 line[i].u,line[i].v-line[i].u))>eps) head++;
426             Q[++tail]=line[i];
427         }
428         while(head<tail&&(cross(((Q[tail]&Q[tail-1])-
429 Q[head].u),Q[head].v-Q[head].u))>eps) tail--;
430         while(head<tail&&(cross(((Q[head]&Q[head+1])-
431 Q[tail].u),Q[tail].v-Q[tail].v))>eps) head++;
432         if(tail<=head+1)
433             return;
434         for (int i = head; i < tail; ++i)
435             res[resn++]=Q[i]&Q[i+1];
436         if(head<tail-1)
437             res[resn++]=Q[head]&Q[tail];
438     }
439     struct Circle
440     {
441         point o;
442         double r;
443         Circle(){}
444         Circle(point o,double r):o(o),r(r){}
445     };
446     int relation(point w,Line l)
447     {
448         //1:左侧 2:右侧 3:线上
449         int c=dcmp(cross(w-l.u,l.v-l.u));
450         return c<0?1:(c==0?3:2);
451     }
452     int relation(point p,Circle a)
453     {
454         //0:圆外,1:圆上,2:圆内
455         double d=dis(p,a.o)-a.r;
456         if(dcmp(d)==0) return 1;
457         return (dcmp(d)<0?2:0);
458     }

```

```

459 int relation(Line a,Circle b)
460 {
461     //0: 相离,1: 相切,2: 相交
462     double p=a.pdis(b.o);
463     if (dcmp (p-b.r) == 0) return 1;
464     return (dcmp (p-b.r) < 0 ? 2 : 0);
465 }
466 int line_cirlce_intersection(Line l,Circle c,point& p1,point&
467 p2)
468 {
469     if(!relation(l,c))
470         return 0;
471     point a=l.project(c.o);
472     double d=l.pdis(c.o);
473     d=sqrt(c.r*c.r-d*d);
474     if(dcmp(d)==0)
475     {
476         p1=a,p2=a;
477         return 0;
478     }
479     p1=a+(l.v-l.u).change_len(d);
480     p2=a-(l.v-l.u).change_len(d);
481     return 2;
482 }
483 double circle_traingle_area(point a,point b,Circle c)
484 {
485     point p=c.o;double r=c.r;
486     if(dcmp(cross(p-a,p-b))==0)
487         return 0;
488     point q[6];
489     int len=0;
490     q[len++]=a;
491     Line l=Line(a,b);
492     if (line_cirlce_intersection (l, c, q[1], q[2]) == 2)
493     {
494         if (dcmp (dot (a-q[1], b-q[1])) < 0) q[len++] =
495 q[1];
496         if (dcmp (dot (a-q[2], b-q[2])) < 0) q[len++] =
497 q[2];
498     }
499     q[len++]=b;
500     if(len==4&&dcmp(dot (q[0]-q[1], q[2]-q[1])) > 0)
501         swap(q[1],q[2]);
502     double ans=0;

```

```

503     for (int i = 0; i < len - 1; ++i)
504     {
505         if(relation(q[i],c)==0||relation(q[i+1],c)==0)
506         {
507             double arg=rad(q[i]-p,q[i+1]-p);
508             ans+=r*r*arg/2.0;
509         }
510         else
511             ans+=fabs(cross (q[i]-p, q[i+1]-p))/2;
512     }
513     return ans;
514 }
515 double area_polygon_circle(Circle c,point* p,int n)
516 {
517     double ans=0;
518     p[n]=p[0];
519     for (int i = 0; i < n; ++i)
520     {
521         if(dcmp(cross(p[i+1]-c.o,p[i]-c.o))>=0)
522             ans+=circle_traingle_area(p[i],p[i+1],c);
523         else
524             ans-=circle_traingle_area(p[i],p[i+1],c);
525     }
526     return fabs(ans);
527 }
528 point aa[105][2005];
529 point pa[200006];
530 double smx[105],smy[105],smqx[105],smqy[105];
531 Line hp[1000];
532 int main()
533 {
534     int T,n,cas=1,m;
535     caze(T)
536     {
537         scanf("%d%d",&n,&m);
538         for (int i = 0; i < n; ++i)
539         {
540             smx[i]=smy[i]=smqx[i]=smqy[i]=0;
541             for (int j = 0; j < m; ++j)
542             {
543                 aa[i][j].input();
544                 smx[i]+=aa[i][j].x;
545                 smy[i]+=aa[i][j].y;
546                 smqx[i]+=sqr(aa[i][j].x);

```

```

547             smqy[i]+=sqr(aa[i][j].y);
548         }
549     }
550     int tot,cnt;
551     printf("Case #d:",cas++);
552     for (int i = 0; i < n; ++i)
553     {
554         cnt=0;
555         hp[cnt++]=Line(point(0,0),point(4095,0));
556
557         hp[cnt++]=Line(point(4095,0),point(4095,4095));
558
559         hp[cnt++]=Line(point(4095,4095),point(0,4095));
560         hp[cnt++]=Line(point(0,4095),point(0,0));
561         double A=0,B=0,C=0;
562         bool f=0;
563         for (int j = 0; j < n; ++j)
564         {
565             if(i==j) continue;
566             A=-2.0*(smx[i]-smx[j]);
567             B=-2.0*(smy[i]-smy[j]);
568             C=smqx[i]+smqy[i]-smqx[j]-smqy[j];
569             point uu,vv;
570             if(dcmp(B)!=0)
571             {
572                 uu=point(0,C/-B);
573                 if(dcmp(A)!=0) vv=point(C/-
574 A,0);
575                 else vv=point(1,C/-B);
576             }
577             else
578             {
579                 if(dcmp(A)!=0) uu=point(C/-
580 A,1),vv=point(C/-A,0);
581                 else if(C>=0)
582                 {
583                     f=1;
584                     break;
585                 }
586                 else{cout<<1/0<<endl;}
587             }
588             int tp=dcmp((point(0,0)-uu)^(vv-
589 uu));
590             bool can=1;

```

```

591         if(tp>0&&C<=0) swap(uu,vv);
592         if(tp<0&&C>=0) swap(uu,vv);
593         if(tp==0)
594         {
595             tp=dcmp((point(0,4095)-
596 uu)^(vv-uu));
597             if(tp<0&&B*4095+C>=0)
598 swap(uu,vv);
599             if(tp>0&&B*4095+C<=0)
600 swap(uu,vv);
601             if(tp==0)
602             {
603
604 tp=dcmp((point(4095,0)-uu)^(vv-uu));
605                 if(tp<0&&A*4095+C>=0)
606 swap(uu,vv);
607                 if(tp>0&&A*4095+C<=0)
608 swap(uu,vv);
609                 if(tp==0)
610                     can=0;
611             }
612         }
613         if(can)
614             hp[cnt++]=Line(uu,vv);
615     }
616     if(!f)
617         Hpi(cnt,hp,pa,tot);
618     printf(" %d",f?0:(int)(Area(tot,pa)+0.5));
619 }
620 putchar('\n');
621 }
622 }
623
624
625
626
627
628
629
630
631
632
633
634

```

635 裸凸包

```

636 #include<cstdio>
637 #include<cmath>
638 #include<cstring>
639 #include<algorithm>
640 using namespace std;
641 #define clr(a,b) memset(a,b,sizeof(a))
642 #define ll long long
643 #define ull unsigned long long
644 #define lowbit(x) (x&-x)
645 #define pb(x) push_back(x)
646 #define IOS ios::sync_with_stdio(0),cin.tie(0),cout.tie(0)
647 #define inf (1<<30)
648 #define caze(T) for(cin>>T;T;T--)
649 #define Endl ('\n')
650 struct point
651 {
652     double x,y;
653     point(){}
654     point(double x,double y):x(x),y(y){}
655     point operator-(const point w)const {return point(x-
656 w.x,y-w.y);}
657     bool operator<(const point& w)const {return
658 x==w.x?y<w.y:x<w.x;}
659 }a[2000007],p[2000007];
660 inline double cross(const point& A,const point& B){return
661 A.x*B.y-B.x*A.y;}
662 inline double dot(const point& q,const point& w){return
663 q.x*w.x+q.y*w.y;}
664 inline double Xmul(const point& A,const point& B,const point&
665 C){return (B.x-A.x)*(C.y-A.y)-(B.y-A.y)*(C.x-A.x);}
666 inline double dis(const point& q,const point& w){return
667 sqrt(dot(q-w,q-w));}
668 int n,tot;
669 void Andrew()
670 {
671     sort(a,a+n);
672     tot=0;
673     for (int i = 0; i < n; ++i)
674     {
675         while(tot>1&&cross(p[tot-1]-p[tot-2],a[i]-p[tot-
676 2])<=0) tot--;
677         p[tot++]=a[i];

```



```

678     }
679     int tp=tot;
680     for (int i = n - 2; ~i; --i)
681     {
682         while(tot>tp&&cross(p[tot-1]-p[tot-2],a[i]-p[tot-
683 2])<=0) tot--;
684         p[tot++]=a[i];
685     }
686     tot--=(n>1);
687 }
688 int main()
689 {
690     double R;
691     while(~scanf("%d%lf",&n,&R))
692     {
693         for (int i = 0; i < n; ++i)
694             scanf("%lf%lf",&a[i].x,&a[i].y);
695         Andrew();
696         double C=dis(p[0],p[tot-1]);
697         for (int i = 0; i < tot - 1; ++i)
698             C+=dis(p[i],p[i+1]);
699         printf("%d\n",(int)(C+2*acos(-1.0)*R+0.5));
700     }
701 }
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721

```

722 半平面交裸:求两多边形面积并减面积交

```
723 #include<bits/stdc++.h>
724 #include<ext/rope>
725 using namespace std;
726 #define clr(a,b) memset(a,b,sizeof(a))
727 #define ll long long
728 #define ull unsigned long long
729 #define lowbit(x) (x&-x)
730 #define pb(x) push_back(x)
731 #define IOS ios::sync_with_stdio(0),cin.tie(0),cout.tie(0)
732 #define inf (1<<30)
733 #define caze(T) for(scanf("%d",&T);T;T--)
734 #define Endl ('\n')
735 const double pi=acos(-1.0);
736 const double eps=1e-8;
737 int dcmp(double x){return fabs(x)<=1e-8?0:(x<0?-1:1);}
738 struct point
739 {
740     double x,y,id;
741     point(){}
742     point(double x,double y,int id=-1):x(x),y(y),id(id) {}
743     point operator-(const point w)const {return point(x-
744 w.x,y-w.y);}
745     point operator+(const point w)const {return
746 point(x+w.x,y+w.y);}
747     point operator*(double a) {return point(x*a,y*a);}
748     point operator/(double a) {return point(x/a,y/a);}
749     void input(){scanf("%lf%lf",&x,&y);}
750 };
751 point aa[200007],ab[200007];
752 point pa[200007],pb[200007];
753 inline double cross(const point& A,const point& B){return
754 A.x*B.y-B.x*A.y;}
755 inline double dot(const point& q,const point& w){return
756 q.x*w.x+q.y*w.y;}
757 inline double Xmul(const point& A,const point& B,const point&
758 C){return cross(C-A,B-A);}
759 inline double dis(const point& q,const point& w){return
760 sqrt(dot(q-w,q-w));}
761 inline double rad(const point& A,const point& B){return
762 fabs(atan2(fabs(cross(A,B)),dot(A,B)));}
763 int Andrew(int n,point *st,point *ed)
764 {
```

```

765         sort(st,st+n,[](const point& A,const point&
766 B)->bool{return A.x==B.x?A.y<B.y:A.x<B.x;});
767         int tot=0;
768         for (int i = 0; i < n; ++i)
769         {
770             while(tot>1&&cross(ed[tot-1]-ed[tot-2],st[i]-
771 ed[tot-2])<0) tot--;
772             ed[tot++]=st[i];
773         }
774         int tp=tot;
775         for (int i = n - 2; ~i; --i)
776         {
777             while(tot>tp&&cross(ed[tot-1]-ed[tot-2],st[i]-
778 ed[tot-2])<0) tot--;
779             ed[tot++]=st[i];
780         }
781         tot--(n>1);
782         return tot;
783     }
784     double Area(int n,point *p)
785     {
786         double S=0;
787         for (int i = 1; i < n - 1; ++i)
788             S+=fabs(Xmul(p[0],p[i],p[i+1]));
789         return S/2;
790     }
791     struct Line
792     {
793         point u,v;
794         double k;
795         Line(){}
796         Line(point u,point v):u(u),v(v){k=atan2(v.y-u.y,v.x-
797 u.x);}
798         Line(point u,double k):u(u),k(k){v=u+(dcmp(k-
799 pi/2)?point(1,tan(k)):point(0,1));}
800         void input(){u.input();v.input();}
801         double len(){return dis(u,v);}
802         point operator&(const Line& b)const
803         {
804             point ret=u;
805             double t=(cross(u-b.u,b.u-b.v))/cross(u-v,b.u-
806 b.v);
807             ret.x+=(v.x-u.x)*t;
808             ret.y+=(v.y-u.y)*t;

```

```

809         return ret;
810     }
811 };
812 Line ln[100010];
813 Line hp[100010];
814 void Hpi(int n, Line *line, point *res, int &resn)
815 {
816     int tot=1;
817     sort(line, line+n, [](const Line& A, const Line&
818 B)->bool{ return fabs(A.k-B.k)>eps?A.k<B.k:cross(A.u-B.u, B.v-
819 B.u)<0;});
820     for (int i = 1; i < n; ++i)
821         if(fabs(line[i].k-line[i-1].k)>eps)
822             line[tot++]=line[i];
823     int head=0, tail=1;
824     ln[0]=line[0];
825     ln[1]=line[1];
826     resn=0;
827     for (int i = 2; i < tot; ++i)
828     {
829         if(fabs(cross(ln[tail].v-ln[tail].u, ln[tail-1].v-
830 ln[tail-1].u))<eps || fabs(cross(ln[head].v-
831 ln[head].u, ln[head+1].v-ln[head+1].u))<eps)
832             return;
833         while(head<tail&&(cross((ln[tail]&ln[tail-1])-
834 line[i].u, line[i].v-line[i].u))>eps) tail--;
835         while(head<tail&&(cross((ln[head]&ln[head+1])-
836 line[i].u, line[i].v-line[i].u))>eps) head++;
837         ln[++tail]=line[i];
838     }
839     while(head<tail&&(cross(((ln[tail]&ln[tail-1])-
840 ln[head].u), ln[head].v-ln[head].u))>eps) tail--;
841     while(head<tail&&(cross(((ln[head]&ln[head+1])-
842 ln[tail].u), ln[tail].v-ln[tail].v))>eps) head++;
843     if(tail<=head+1)
844         return;
845     for (int i = head; i < tail; ++i)
846         res[resn++]=ln[i]&ln[i+1];
847     if(head<tail-1)
848         res[resn++]=ln[head]&ln[tail];
849 }
850 int main()
851 {
852     int T;

```

```

853     int n,m;
854     while(scanf("%d",&n)&&n)
855     {
856         for (int i = 0; i < n; ++i)
857             aa[i].input();
858         double Sa,Sb,Sc;
859         int tota=Andrew(n,aa,pa);
860         Sa=Area(tota,pa);
861         scanf("%d",&n);
862         for (int i = 0; i < n; ++i)
863             ab[i].input();
864         int totb=Andrew(n,ab,pb);
865         Sb=Area(totb,pb);
866         int cnt=0;
867         for (int i = 0; i < tota; ++i)
868             hp[cnt++]=Line(pa[i],pa[(i+1)%tota]);
869         for (int i = 0; i < totb; ++i)
870             hp[cnt++]=Line(pb[i],pb[(i+1)%totb]);
871         int totc=0;
872         Hpi(cnt,hp,aa,totc);
873         Sc=Area(totc,aa);
874         printf("%8.2f",Sa+Sb-Sc-Sc);
875     }
876     puts("");
877 }
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896

```

897 旋转卡壳求最远点对

```

898  const double pi=acos(-1.0);
899  const double eps=1e-8;
900  int dcmp(double x){return fabs(x)<=eps?0:(x<0?-1:1);}
901  double sqr(double x){return x*x;}
902  struct point
903  {
904      double x,y,id;
905      point(){}
906      point(double x,double y,int id=-1):x(x),y(y),id(id) {}
907      point operator-(const point w)const {return point(x-
908 w.x,y-w.y);}
909      point operator+(const point w)const {return
910 point(x+w.x,y+w.y);}
911      double operator*(const point& w)const {return
912 x*w.x+y*w.y;}
913      point operator*(double a) {return point(x*a,y*a);}
914      double operator^(const point& w)const {return x*w.y-
915 y*w.x;}
916      point operator/(double a) {return point(x/a,y/a);}
917      //friend ostream &operator<<(ostream& out,const point&
918 w) {out<<'('<<w.x<<','<<w.y<<')';return out;}
919      void input(){scanf("%lf%lf",&x,&y);}
920      double len2(){return x*x+y*y;}
921      double len(){return sqrt(x*x+y*y);}
922      bool operator<(const point& w)const{return
923 x==w.x?y<w.y:x<w.x;}
924      point change_len(double r)
925      {
926          double l=len();
927          if(dcmp(l)==0) return *this;
928          r/=l;
929          return point(x*r,y*r);
930      }
931  };
932  inline double cross(const point& A,const point& B){return
933 A.x*B.y-B.x*A.y;}
934  inline double dot(const point& q,const point& w){return
935 q.x*w.x+q.y*w.y;}
936  inline double Xmul(const point& A,const point& B,const point&
937 C){return cross(C-A,B-A);}
938  inline double dis(const point& q,const point& w){return
939 sqrt(dot(q-w,q-w));}

```

```

940 inline double rad(const point& A,const point& B){return
941 fabs(atan2(fabs(cross(A,B)),dot(A,B)));}
942 int Andrew(int n,point *st,point *ed)
943 {
944     sort(st,st+n);
945     int tot=0;
946     for (int i = 0; i < n; ++i)
947     {
948         while(tot>1&&cross(ed[tot-1]-ed[tot-2],st[i]-
949 ed[tot-2])<=0) tot--;
950         ed[tot++]=st[i];
951     }
952     int tp=tot;
953     for (int i = n - 2; ~i; --i)
954     {
955         while(tot>tp&&cross(ed[tot-1]-ed[tot-2],st[i]-
956 ed[tot-2])<=0) tot--;
957         ed[tot++]=st[i];
958     }
959     tot--=(n>1);
960     return tot;
961 }
962 double Area(int n,point *p)
963 {
964     double S=0;
965     for (int i = 1; i < n - 1; ++i)
966         S+=fabs(Xmul(p[0],p[i],p[i+1]));
967     return S/2;
968 }
969 int cal(int n,point *p)
970 {
971     int ret=0;
972     int tp=1;
973     for (int i = 0; i < n; ++i)
974     {
975         while(((p[(i+1)%n]-p[i])^(p[tp]-
976 p[i]))<((p[(i+1)%n]-p[i])^(p[(tp+1)%n]-p[i])))
977             (tp+=1)%=n;
978         ret=max(ret,(int)((p[tp]-p[i]).len2()+eps));
979     }
980     return ret;
981 }
982 point aa[50005];
983 point pa[50005];

```

```
984  int main()
985  {
986      int n;
987      scanf("%d",&n);
988      for (int i = 0; i < n; ++i)
989          aa[i].input();
990      int tot=Andrew(n,aa,pa);
991      printf("%d\n",cal(tot,pa));
992  }
```

993

994

995

996

997

998

999

1000

1001

1002

1003

1004

1005

1006

1007

1008

1009

1010

1011

1012

1013

1014

1015

1016

1017

1018

1019

1020

1021

1022

1023

1024

1025

1026

1027

1028 给定 n 个点求最大三角形面积,旋转卡壳:

```
1029 #include<iostream>
1030 #include<cstdio>
1031 #include<cstring>
1032 #include<cstdlib>
1033 #include<set>
1034 #include<ctime>
1035 #include<vector>
1036 #include<queue>
1037 #include<algorithm>
1038 #include<map>
1039 #include<cmath>
1040 using namespace std;
1041 #define clr(a,b) memset(a,b,sizeof(a))
1042 #define ll long long
1043 #define ull unsigned long long
1044 #define lowbit(x) (x&-x)
1045 #define pb(x) push_back(x)
1046 #define IOS ios::sync_with_stdio(0),cin.tie(0),cout.tie(0)
1047 #define inf (1<<30)
1048 #define caze(T) for(scanf("%d",&T);T;T--)
1049 #define Endl ('\n')
1050 const double pi=acos(-1.0);
1051 const double eps=1e-8;
1052 int dcmp(double x){return fabs(x)<=eps?0:(x<0?-1:1);}
1053 double sqr(double x){return x*x;}
1054 struct point
1055 {
1056     double x,y,id;
1057     point(){}
1058     point(double x,double y,int id=-1):x(x),y(y),id(id) {}
1059     point operator-(const point w)const {return point(x-
1060 w.x,y-w.y);}
1061     point operator+(const point w)const {return
1062 point(x+w.x,y+w.y);}
1063     double operator*(const point& w)const {return
1064 x*w.x+y*w.y;}
1065     point operator*(double a) {return point(x*a,y*a);}
1066     double operator^(const point& w)const {return x*w.y-
1067 y*w.x;}
1068     point operator/(double a) {return point(x/a,y/a);}
1069     friend ostream &operator<<(ostream& out,const point& w)
1070 {out<<'('<<w.x<<','<<w.y<<')';return out;}
```

```

1071     void input(){scanf("%lf%lf",&x,&y);}
1072     double len2(){return x*x+y*y;}
1073     double len(){return sqrt(x*x+y*y);}
1074     point change_len(double r)
1075     {
1076         double l=len();
1077         if(dcmp(l)==0) return *this;
1078         r/=l;
1079         return point(x*r,y*r);
1080     }
1081     bool operator<(const point& w)const {return
1082 x==w.x?y<w.y:x<w.x;}
1083 };
1084 inline double cross(const point& A,const point& B){return
1085 A.x*B.y-B.x*A.y;}
1086 inline double dot(const point& q,const point& w){return
1087 q.x*w.x+q.y*w.y;}
1088 inline double Xmul(const point& A,const point& B,const point&
1089 C){return cross(C-A,B-A);}
1090 inline double dis(const point& q,const point& w){return
1091 sqrt(dot(q-w,q-w));}
1092 inline double rad(const point& A,const point& B){return
1093 fabs(atan2(fabs(cross(A,B)),dot(A,B)));}
1094 int Andrew(int n,point *st,point *ed)
1095 {
1096     sort(st,st+n);
1097     //sort(st,st+n,[](const point& A,const point&
1098 B)->bool{return A.x==B.x?A.y<B.y:A.x<B.x;});
1099     int tot=0;
1100     for (int i = 0; i < n; ++i)
1101     {
1102         while(tot>1&&cross(ed[tot-1]-ed[tot-2],st[i]-
1103 ed[tot-2])<=0) tot--;
1104         ed[tot++]=st[i];
1105     }
1106     int tp=tot;
1107     for (int i = n - 2; ~i; --i)
1108     {
1109         while(tot>tp&&cross(ed[tot-1]-ed[tot-2],st[i]-
1110 ed[tot-2])<=0) tot--;
1111         ed[tot++]=st[i];
1112     }
1113     tot--(n>1);
1114     return tot;

```

```

1115 }
1116 double Area(int n,point *p)
1117 {
1118     double S=0;
1119     for (int i = 1; i < n - 1; ++i)
1120         S+=fabs(Xmul(p[0],p[i],p[i+1]));
1121     return S/2;
1122 }
1123 double cal(int n,point *p)
1124 {
1125     double ret=0;
1126     int t1=1,t2=2;
1127     for (int i = 0; i < n; ++i)
1128     {
1129         while(((p[t1]-p[i])^(p[t2]-p[i]))<((p[t1]-
1130 p[i])^(p[(t2+1)%n]-p[i])))
1131             (t2+=1)%=n;
1132         ret=max(ret,((p[t1]-p[i])^(p[t2]-p[i]))/2.0);
1133         while(((p[t1]-p[i])^(p[t2]-p[i]))<((p[(t1+1)%n]-
1134 p[i])^(p[t2]-p[i])))
1135             (t1+=1)%=n;
1136         ret=max(ret,((p[t1]-p[i])^(p[t2]-p[i]))/2.0);
1137     }
1138     return ret;
1139 }
1140 point aa[50005];
1141 point pa[50005];
1142 int main()
1143 {
1144     int n;
1145     while(scanf("%d",&n)&&(~n))
1146     {
1147         for (int i = 0; i < n; ++i)
1148             aa[i].input();
1149         int tot=Andrew(n,aa,pa);
1150         printf("%.2f\n",cal(tot,pa));
1151     }
1152 }
1153
1154
1155
1156
1157
1158

```

求两凸包最短距离

```
1159
1160 const double pi=acos(-1.0);
1161 const double eps=1e-8;
1162 int dcmp(double x){return fabs(x)<=eps?0:(x<0?-1:1);}
1163 double sqr(double x){return x*x;}
1164 struct point
1165 {
1166     double x,y,id;
1167     point(){}
1168     point(double x,double y,int id=-1):x(x),y(y),id(id) {}
1169     point operator-(const point w)const {return point(x-
1170 w.x,y-w.y);}
1171     point operator+(const point w)const {return
1172 point(x+w.x,y+w.y);}
1173     double operator*(const point& w)const {return
1174 x*w.x+y*w.y;}
1175     point operator*(double a) {return point(x*a,y*a);}
1176     double operator^(const point& w)const {return x*w.y-
1177 y*w.x;}
1178     point operator/(double a) {return point(x/a,y/a);}
1179     friend ostream &operator<<(ostream& out,const point& w)
1180 {out<<'('<<w.x<<','<<w.y<<')';return out;}
1181     void input(){scanf("%lf%lf",&x,&y);}
1182     double len2(){return x*x+y*y;}
1183     double len(){return sqrt(x*x+y*y);}
1184     point change_len(double r)
1185     {
1186         double l=len();
1187         if(dcmp(l)==0) return *this;
1188         r/=l;
1189         return point(x*r,y*r);
1190     }
1191     bool operator<(const point& w)const {return
1192 x==w.x?y<w.y:x<w.x;}
1193 };
1194 inline double cross(const point& A,const point& B){return
1195 A.x*B.y-B.x*A.y;}
1196 inline double dot(const point& q,const point& w){return
1197 q.x*w.x+q.y*w.y;}
1198 inline double Xmul(const point& A,const point& B,const point&
1199 C){return cross(C-A,B-A);}
1200 inline double dis(const point& q,const point& w){return
1201 sqrt(dot(q-w,q-w));}
```

```

1202 inline double rad(const point& A,const point& B){return
1203 fabs(atan2(fabs(cross(A,B)),dot(A,B)));}
1204 int Andrew(int n,point *st,point *ed)
1205 {
1206     sort(st,st+n);
1207     //sort(st,st+n,[](const point& A,const point&
1208 B)->bool{return A.x==B.x?A.y<B.y:A.x<B.x;});
1209     int tot=0;
1210     for (int i = 0; i < n; ++i)
1211     {
1212         while(tot>1&&cross(ed[tot-1]-ed[tot-2],st[i]-
1213 ed[tot-2]))<=0) tot--;
1214         ed[tot++]=st[i];
1215     }
1216     int tp=tot;
1217     for (int i = n - 2; ~i; --i)
1218     {
1219         while(tot>tp&&cross(ed[tot-1]-ed[tot-2],st[i]-
1220 ed[tot-2]))<=0) tot--;
1221         ed[tot++]=st[i];
1222     }
1223     tot--(n>1);
1224     return tot;
1225 }
1226 double Area(int n,point *p)
1227 {
1228     double S=0;
1229     for (int i = 1; i < n - 1; ++i)
1230         S+=fabs(Xmul(p[0],p[i],p[i+1]));
1231     return S/2;
1232 }
1233 point aa[10007],bb[10007];
1234 point pa[10007],pb[10007];
1235 double dist(point a,point b,point c)
1236 {
1237     double tp=fabs((b-a)^(c-a));
1238     point t1=b-a,t2=c-a,t3=c-b;
1239     if(dcmp(dot(t1,t2))<0) return t2.len();
1240     if(dcmp(dot(t1,t3))>0) return t3.len();
1241     return tp/dis(a,b);
1242 }
1243 double cal(point a,point b,point c,point d)
1244 {
1245     double t[4];

```

```

1246     int cnt=0;
1247     t[cnt++]=dist(a,b,c);
1248     t[cnt++]=dist(a,b,d);
1249     t[cnt++]=dist(c,d,a);
1250     t[cnt++]=dist(c,d,b);
1251     sort(t,t+cnt);
1252     return t[0];
1253 }
1254 double rot(point *p,point *q,int n,int m)
1255 {
1256     int mq=0,mp=0;
1257     p[n]=p[0],q[m]=q[0];
1258     for (int i = 1; i < n; ++i) mp=p[i].y<p[mp].y?i:mp;
1259     for (int i = 1; i < m; ++i) mq=q[i].y>q[mq].y?i:mq;
1260     double ans=dis(p[mp],q[mq]),t;
1261     for (int i = 0; i < n; ++i)
1262     {
1263         while(dcmp(t=((q[mq+1]-p[mp+1])^(p[mp]-p[mp+1]))-
1264 ((q[mq]-p[mp+1])^(p[mp]-p[mp+1])))==1)
1265             (mq+=1)%=m;
1266         if(dcmp(t)<0)
1267             ans=min(ans,dist(p[mp],p[mp+1],q[mq]));
1268         else
1269
1270             ans=min(ans,cal(q[mq],q[mq+1],p[mp],p[mp+1]));
1271             (mp+=1)%=n;
1272     }
1273     return ans;
1274 }
1275 int main()
1276 {
1277     int n,m;
1278     while(scanf("%d%d",&n,&m)&&n)
1279     {
1280         for (int i = 0; i < n; ++i)
1281             aa[i].input();
1282         for (int i = 0; i < m; ++i)
1283             bb[i].input();
1284         int ta=Andrew(n,aa,pa);
1285         int tb=Andrew(m,bb,pb);
1286
1287         printf("%.5f\n",min(rot(pa,pb,ta,tb),rot(pb,pa,tb,ta)));
1288     }
1289 }

```

Tarjan

1290

```
1291  const int maxn=50020;
1292  struct EDGE{int v,w,nxt;}edge[1000010];
1293  int tot;
1294  int head[maxn];
1295  void AE(int u,int v,int
1296  w){edge[tot]={v,w,head[u]},head[u]=tot++;}
1297  int n,m;
1298  int ttime,idx,col;
1299  int dfn[maxn];
1300  int low[maxn];
1301  int stk[maxn];
1302  bool vis[maxn];
1303  int belong[maxn];
1304  void init()
1305  {
1306      tot=ttime=idx=col=0;
1307      clr(dfn,0);
1308      clr(head,-1);
1309      clr(vis,0);
1310      clr(belong,0);
1311  }
1312  void tarjan(int u)
1313  {
1314      dfn[u]=low[u]=++ttime;
1315      vis[u]=1;
1316      stk[++idx]=u;
1317      int tp=0;
1318      for(int i=head[u];~i;i=edge[i].nxt)
1319      {
1320          int v=edge[i].v;
1321          if(!dfn[v])
1322          {
1323              tarjan(v);
1324              low[u]=min(low[u],low[v]);
1325          }
1326          else if(vis[v])
1327              low[u]=min(low[u],dfn[v]);
1328      }
1329      if(dfn[u]==low[u])
1330      {
1331          col++;
1332          do
```

```
1333         {
1334             vis[stk[idx]]=0;
1335             belong[stk[idx--]]=col;
1336         } while (vis[u]);
1337     }
1338 }
```

1339

1340

1341

1342

1343

1344

1345

1346

1347

1348

1349

1350

1351

1352

1353

1354

1355

1356

1357

1358

1359

1360

1361

1362

1363

1364

1365

1366

1367

1368

1369

1370

1371

1372

1373

1374

MCMF

1375

1376

dij:

1377

```
const int MAXN=222;
```

1378

```
struct EDGE{int to, cap, cost, flow, nxt;}edge[1<<22];
```

1379

```
int head[MAXN];
```

1380

```
int tot;
```

1381

```
void AE(int from, int to, int cap, int cost)
```

1382

```
{
```

1383

```
    edge[tot]={to, cap, cost, 0, head[from]}, head[from]=tot++;
```

1384

```
    edge[tot]={from, 0, -cost, 0, head[to]}, head[to]=tot++;
```

1385

```
}
```

1386

```
int cost, flow;
```

1387

```
int h[MAXN];
```

1388

```
int dist[MAXN], pre[MAXN];
```

1389

```
void min_cost_flow(int s, int t, int f, int N)
```

1390

```
{
```

1391

```
    fill(h, h+1+N, 0);
```

1392

```
    while(f>0)
```

1393

```
{
```

1394

```
        priority_queue<pii, vector<pii>, greater<pii> >q;
```

1395

```
        clr(dist, inf);
```

1396

```
        dist[s]=0, q.push(pii(0, s));
```

1397

```
        clr(pre, -1);
```

1398

```
        while(!q.empty())
```

1399

```
{
```

1400

```
            pii now=q.top();
```

1401

```
            q.pop();
```

1402

```
            if(dist[now.second]<now.first) continue;
```

1403

```
            int u=now.second;
```

1404

```
            for (int i = head[u]; ~i; i=edge[i].nxt)
```

1405

```
{
```

1406

```
                EDGE &e=edge[i];
```

1407

```
                if
```

1408

```
(e.cap>e.flow&&dist[e.to]>dist[u]+e.cost+h[u]-h[e.to])
```

1409

```
{
```

1410

```
            dist[e.to]=dist[u]+e.cost+h[u]-h[e.to];
```

1411

```
            pre[e.to]=i;
```

1412

1413

```
            q.push(pii(dist[e.to], e.to));
```

1414

```
}
```

1415

```

1416         }
1417     }
1418     if(dist[t]==inf) break;
1419     for (int i = 0; i <= N; ++i)
1420         h[i]+=dist[i];
1421     int d=f;
1422     for (int i = pre[t]; ~i; i=pre[edge[i^1].to])
1423         d=min(d,edge[i].cap-edge[i].flow);
1424     f-=d;flow+=d;
1425     cost+=d*h[t];
1426     for (int i = pre[t]; ~i; i=pre[edge[i^1].to])
1427     {
1428         edge[i].flow+=d;
1429         edge[i^1].flow-=d;
1430     }
1431 }
1432 }
1433 char mp[111][111];
1434 int xx[2][111],yy[2][111];
1435 int w[111][111];
1436 int nx,ny;
1437 int main()
1438 {
1439     int nn,mm;
1440     while(scanf("%d%d",&nn,&mm)&&nn)
1441     {
1442         nx=0,ny=0;
1443         for (int i = 0; i < nn; ++i)
1444         {
1445             getchar();
1446             for (int j = 0; j < mm; ++j)
1447             {
1448                 mp[i][j]=getchar();
1449                 if(mp[i][j]=='H')
1450                     xx[0][nx]=i,yy[0][nx++]=j;
1451                 if(mp[i][j]=='m')
1452                     xx[1][ny]=i,yy[1][ny++]=j;
1453             }
1454         }
1455         for (int i = 0; i < nx; ++i)
1456             for (int j = 0; j < ny; ++j)
1457                 w[i][j]=abs(xx[0][i]-
1458                     xx[1][j])+abs(yy[0][i]-yy[1][j]);
1459         int s=1,t=nx+ny+2,n=t;

```

```

1460         tot=0;
1461         clr(head,-1);
1462         flow=0,cost=0;
1463         for (int i = 0; i < nx; ++i)
1464             AE(s,i+2,1,0);
1465         for (int i = 0; i < nx; ++i)
1466             for (int j = 0; j < ny; ++j)
1467                 AE(i+2,j+nx+2,1,w[i][j]);
1468         for (int i = 0; i < ny; ++i)
1469             AE(i+nx+2,t,1,0);
1470         min_cost_flow(s,t,inf,n);
1471         printf("%d\n",cost);
1472     }
1473 }
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503

```

1504 SPFA(网上板子):

```
1505 const int N=1005;
1506 const int M=50000;
1507 const int inf=0x3f3f3f3f;
1508 queue<int> que;
1509 int n,m,ans=0;
1510 int
1511 first[50000],next[50000],go[50000],rest[50000],cost[50000],dis
1512 [1005],tot=1;
1513 bool visit[50000],work[50000];
1514 int src,des;
1515 void combin(int u,int v,int r,int w)
1516 {
1517
1518 next[++tot]=first[u],first[u]=tot,go[tot]=v,rest[tot]=r,cost[t
1519 ot]=w;
1520
1521 next[++tot]=first[v],first[v]=tot,go[tot]=u,rest[tot]=0,cost[t
1522 ot]=-w;
1523 }
1524 void init(int n,int m)
1525 {
1526 src=0,des=n+1;
1527 for(int i=1;i<=m;i++)
1528 {
1529 int u,v,w;
1530 scanf("%d%d%d",&u,&v,&w);
1531 combin(u,v,1,w);
1532 combin(v,u,1,w);
1533 }
1534 combin(src,1,2,0);
1535 combin(n,des,2,0);
1536 }
1537 bool spfa()
1538 {
1539 memset(dis,inf,sizeof(dis));
1540 memset(work,false,sizeof(work));
1541 int u;
1542 que.push(src),dis[src]=0;
1543 while(!que.empty())
1544 {
1545 u=que.front(),que.pop();
1546 visit[u]=false;
```

```

1547     for(int e=first[u];e;e=next[e])
1548     {
1549         int v=go[e];
1550         if(rest[e]&&dis[u]+cost[e]<dis[v])
1551         {
1552             dis[v]=dis[u]+cost[e];
1553             if(!visit[v])
1554             {
1555                 que.push(v);
1556                 visit[v]=true;
1557             }
1558         }
1559     }
1560 }
1561 return dis[des]<inf;
1562 }
1563 int dinic(int u,int flow)
1564 {
1565     if(u==des)
1566     {
1567         ans+=flow*dis[des];
1568         return flow;
1569     }
1570     work[u]=true;
1571     int res=0,temp,v,e;
1572     for(e=first[u];e;e=next[e])
1573     {
1574         if(!work[v=go[e]]&&rest[e]&&dis[v]==dis[u]+cost[e])
1575         {
1576             temp=dinic(v,min(rest[e],flow-res));
1577             if(temp)
1578             {
1579                 rest[e]-=temp,rest[e^1]+=temp;
1580                 res+=temp;
1581                 if(res==flow) break;
1582             }
1583         }
1584     }
1585     return res;
1586 }
1587 int maxflow()
1588 {
1589     while(spfa()) dinic(src,inf);
1590     return ans;

```

```
1591 }
1592 int main()
1593 {
1594     scanf("%d%d",&n,&m);
1595     init(n,m);
1596     cout<<maxflow()<<endl;
1597     return 0;
1598 }
```

```
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
```