ACM ICPC
Regional 2014

Contents

1. 2-DBinaryIndexedTree

```
#define MAXN 1033

int nx,ny,s[MAXN][MAXN];
int lowbit(int x){ return x&(-x); }

int getSum(int x, int y){
        int ret = 0;
        for( int i= x ; i>0 ; i-=lowbit(i) )
                for( int j=y ; j>0 ; j-=lowbit(j) )
                        ans+=s[i][j];
        return ans;
}
int add(int x,int y,int d){
        for( int i=x ; i<nx ; i+=low_bit(i) )
                for( int j=y ; j<ny ; j+=low_bit(j) )
                        s[i][j]+=d;
}
```

2. Articulation

```
// graph[0][1]=graph[1][0]=true; 無向圖建邊
// dfnlow(0,0); 呼叫=> 點root
int
      dfn[MAX],low[MAX],answer[MAX],depth=1,ansc=0;
bool graph[MAX][MAX];
void dfnlow(int u,int v)
{
  int w;
  bool yes=0;
  int child=0;
  dfn[u]=low[u]=depth++;
  for(w=0;w<MAX;w++)
    if(graph[u][w])
    {
      if(dfn[w]<0)
      {
        dfnlow(w,u);
        child++;
        low[u]=(low[u]<low[w])?low[u]:low[w];
        if(low[w]>=dfn[u])
        {
          yes=1;
          /*if(low[w]!=dfn[u])cout<<u<<"-"<<w<<endl;找
                cut edge u-w */
        }
      }
      else if(w!=v)
        low[u]=(low[u]<dfn[w])?low[u]:dfn[w];
    }
  if( (u==v&&child>1 ) || (u!=v&&yes) )
    answer[ansc++]=u;
}
```

3. bellman

```
int bellman(int n,int edg)
{
  int i,j,flag;
  for(i=1;i<=n;++i) dis[i]=INF;
  dis[1]=0;
  for(i=0;i<n-1;++i){
    for(j=flag=0;j<edg;++j){
      if(dis[edge[j].u]+edge[j].w<dis[edge[j].v])
dis[edge[j].v]=dis[edge[j].u]+edge[j].w;
      flag=1;
    }
    if(!flag) return 1; //沒有negative cycle
  }
  for(j=0;j<edg;++j) //檢查negative cycle
    if(dis[edge[j].u]+edge[j].w<dis[edge[j].v])
      return 0;
  return 1; //沒有negative cycle
}
```

4. Josephus

```
// 輸出: 人中每人殺一人，最後剩下的人的，nmid0~n-1
int killer(int n,int m){
  if(n==1) return 0;
  return (killer(n-1,m)+m)%n;
}
// 若要1~，輸出n就好+1
```

```
// 若要每次殺不同個數，傳參數的改掉，平移不動mm
```

5. KMP

```
char P[Plen]; // pattern string
char T[Tlen]; // target string
int pi[Plen]; // pi[i]: max len prefix == suffix in
      P[0..i]

void kmp_pre() {
    pi[0] = 0;
    for(int i = 1; i < Plen; i++) {
        pi[i] = pi[i - 1];
        while(pi[i] > 0 && P[i] != P[pi[i]])
            pi[i] = pi[pi[i] - 1];
        if(P[i] == P[pi[i]])
            pi[i]++;
    }
}

void kmp() {
    for(int i = 0, j = 0; i < Tlen; i++) {
        while(j > 0 && T[i] != P[j])
            j = pi[j - 1];
        if(T[i] == P[j]) j++;
        if(j == Plen) ; // match (i - Plen + 1)
    }
}
```

6. Dinic

```
// D_type 改成Flow 的Type
#include<cstdio>
#include<cstring>
#define VTEX 100
#define MAXE 100000
#define INF 10000000
#define min(a,b) ((a)<(b)?(a):(b))
#define rev(x) (x&1?(x-1):(x+1))
typedef int D_type;
using namespace std;
typedef struct{
  int v,next;
  D_type flow;
}EDGE;
int pre[VTEX],level[VTEX],que[VTEX];
EDGE edg[MAXE];
bool build(int source,int sink)
{
  int i,head=0,tail=0,now;
  memset(level,0,(sink+1)*sizeof(int));
  level[source]=1;
  que[tail++]=source;
  while(head<tail)
  {
    now=que[head%VTEX];
    ++head;
    for(i=pre[now];i!=-1;i=edg[i].next)
    {
      if(edg[i].flow&&!level[edg[i].v])
      {
        que[tail%VTEX]=edg[i].v;
        ++tail;
        level[edg[i].v]=level[now]+1;
        if(edg[i].v==sink) return true;
      }
    }
  }
  return false;
}
D_type findflow(int now,D_type beforemin,int
      sink)
{
  int i;
  D_type ans=0,flow;
  if(now==sink||!beforemin) return beforemin;
  for(i=pre[now];i!=-1;i=edg[i].next)
  {
    if(level[edg[i].v]==level[now]+1&& \
      (flow=findflow(edg[i].v,min(beforemin,edg[i].flow),sink)))
    {
      edg[i].flow-=flow;
      edg[rev(i)].flow+=flow;
      ans+=flow;
      beforemin-=flow;
      if(beforemin==0) break;
    }
  }
  return ans;
}
D_type dinic(int source,int sink)
{
  D_type ans=0;
  while(build(source,sink))
        ans+=findflow(source,INF,sink);
  return ans;
}
int main(void)
{
  int i,n,m,index,a,b;
  D_type w;
  scanf("%d%d",&n,&m);
  memset(pre,-1,n*sizeof(int));
  for(i=index=0;i<m;++i)
  {
    scanf("%d%d%d",&a,&b,&w);
    edg[index].v=b;
    edg[index].flow=w;
    edg[index].next=pre[a];
    pre[a]=index++;
    edg[index].v=a;
    edg[index].flow=0;
    edg[index].next=pre[b];
    pre[b]=index++;
  }
  printf("%d\n",dinic(0,n-1));
  return 0;
}
```

7. EulersPhi

```
#define MAXN 2000001
int euler[MAXN];
bool isprim[MAXN];
void phi(int n){
  memset(isprim,true,(n+1)*sizeof(bool));
  for(int i=1;i<=n;++i) euler[i]=i;
  isprim[0]=isprim[1]=false;
  for(int i=2;i<=n;++i){
    if(isprim[i]){
      euler[i]=i-1;
      for(j=i+i;j<=n;j+=i){
        isprim[j]=false;
        euler[j]-=euler[j]/i;
      }
    }
  }
}
```

8. extendGCD

```
// ax + by = gcd, |x|+|y| will be minimum
void exgcd(int a,int b,int &x,int &y, int &gcd){
        if(b==0)
                gcd=a,x=1,y=0;
        else{
                exgcd(b, a%b, y, x , gcd);
                y = y - (a/b) * x;
        }
}
```

9. Maxflow-EK

```
#define MAXN 1033
#define inf ((int)1e9)
#define MAXQ 1033

int e[ MAXN ][ MAXN ] , e_num[ MAXN ];
int cap[ MAXN ][ MAXN ] , flo[ MAXN ][ MAXN
    ];
int vis[ MAXN ] , pat[ MAXN ];
int q[ MAXQ ],tail,head;

int bfs(int s,int t){
    int i , u , v ;
    head = 0 , tail = 1;
    q[0] = s;
    while( head < tail ){
        u = q[head%MAXQ];
        ++ head;
        vis[u] = 1;
        if( u == t ) return 1;
        for(i=0;i<e_num[u];++i){
            v = e[u][i];
            if(vis[v]) continue;
            if( cap[u][v] - flo[u][v] >0 || flo[v][u] > 0)
                pat[v] = u , q[(tail++)%MAXQ] =
                    v;
        }
    }
    return 0;
}
int f_flow(int s,int t){
    int i,pre,f=inf,tmp;
    for(i = t ; i!=s ; i = pre){
        pre=pat[i];
        tmp=cap[pre][i]-flo[pre][i];
        if(tmp>0){
            if(tmp<f)f=tmp;
        }else{
            if(flo[i][pre]<f)f=flo[i][pre];
        }
    }
    for(i = t ; i!=s ; i = pre){
        pre = pat[i];
        tmp = cap[pre][i] - flo[pre][i];
        if(tmp > 0)  flo[pre][i] += f;
        else  flo[i][pre] -= f;
    }
    return f;
}
int Edmonds_karp(int s,int t){
    int ret=0;
    while(1){
        memset(vis,0,sizeof(vis));
        if(bfs(s,t)==0)break;
        ret+=f_flow(s,t);
    }
    return ret;
}
```

10. LISnlgn

```
#define MAXN 100
using namespace std;
int
    pos[MAXN],lis[MAXN],seq[MAXN],ans[MAXN];
int main(void)
{
  int i,n,len,l,r,mid,now;
  while(scanf("%d",&n)!=EOF)
  {
    for(i=0;i<n;i++) scanf("%d",&seq[i]);

    // 初始化
    len=0;
    lis[len++]=seq[0];
    pos[0]=1;

    // 嚴格遞增
    for(i=1;i<n;i++)
    {
      // Append
      if(lis[len-1]<seq[i]) // 若是非遞減: 改成<=
      {
        lis[len++]=seq[i];
        pos[i]=len;
      }
      else
      {
        // Binary Search
        l=0,r=len-1;
        while(l<r)
        {
          mid=l+(r-l)/2;
          if(lis[mid]<seq[i]) l=mid+1; //非遞減: 改
              成<=
          else r=mid;
        }

        //選擇正確的地方插入
        lis[r]=seq[i];
        pos[i]=r+1;
      }
    }

    // 答案LIS
    printf("%d\n",len);

    //其中一組解
    for(i=n-1,now=len;i>=0&&now>0;i--)
      if(pos[i]==now)
        ans[--now]=seq[i];
    for(i=0;i<len;i++) printf(" %d",ans[i]);
    printf("\n");
  }
  return 0;
}
```

11. MCMF

```
// 加單向邊addEdge(u,v,cost,capacity);
// 初始化tot=0; memset(prev,-1,sizeof(prev));
#define MAXE 10010  // 邊個數
#define MAXN 102  // 點個數
#define min(a,b) ((a)<(b)?(a):(b))
#define INF (int)1e9
```

```
using namespace std;

typedef struct{
  int u,v,next;
  int flow,cost;
}EDGE;

int
      prev[MAXN],p[MAXN],tot,que[MAXN],dis[MAXN];
bool inque[MAXN];
EDGE edg[MAXE*2];

void add(int u,int v,int cost,int flow){
  edg[tot].u=u;
  edg[tot].v=v;
  edg[tot].flow=flow;
  edg[tot].cost=cost;
  edg[tot].next=prev[u];
  prev[u]=tot++;
}

void addEdge(int u,int v,int cost,int flow){
  add(u,v,cost,flow);
  add(v,u,-cost,0);
}

bool spfa(int s,int t,int n){
  int i,head,tail,now,next;
  for(i=0;i<n;++i)
  {
    dis[i]=INF;
    inque[i]=false;
    p[i]=-1;
  }
  dis[s]=0;
  head=tail=0;
  que[tail++]=s;
  inque[s]=true;
  while(head<tail)
  {
    now=que[head%MAXN];
    inque[now]=false;
    ++head;
    for(i=prev[now];i!=-1;i=edg[i].next)
    {
      next=edg[i].v;
      if(edg[i].flow&&dis[now]+edg[i].cost<dis[next])
      {
        dis[next]=dis[now]+edg[i].cost;
        p[next]=i;
        if(!inque[next])
        {
          que[tail%MAXN]=next;
          inque[next]=true;
          ++tail;
        }
      }
    }
  }
  return dis[t]!=INF;
}

void MCMF(int s,int t,int n)
{
  int i,MF=0,mc=0,ff;

  while(spfa(s,t,n)){
    ff=INF;
    for(i=p[t];i!=-1;i=p[edg[i].u])
      ff=min(ff,edg[i].flow);

    for(i=p[t];i!=-1;i=p[edg[i].u]){
      edg[i].flow-=ff;
      edg[i^1].flow+=ff;
    }

    MF+=ff;
    mc+=ff*dis[t];
  }
  // MF -> MaxFlow
  // mc -> minimum cost
}
```

12. MinimumEnclosingCircle

```
#include<cstdio>
#include<cmath>
#define dis(a,b)
      sqrt((a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y))
#define MAXN 102
using namespace std;

// (x,y)
typedef struct{
  double x,y;
}POINT;
typedef struct{ // ax+by+c=0
  double a,b,c;
}LINE;

// Circle
typedef struct{
  double r;
  POINT cen;
}CIRCLE;

// Find ax+by+c=0
LINE getL(POINT p1, POINT p2)
{
  LINE l;
  l.a=p2.y-p1.y;
  l.b=p1.x-p2.x;
  l.c=(p2.x-p1.x)*p1.y+(p1.y-p2.y)*p1.x;
  return l;
}

// Find Mid Vertical Line
LINE getMVL(POINT p1, POINT p2)
{
  LINE l,org;
  org=getL(p1,p2);
  l.a=org.b;
  l.b=-org.a;
  l.c=(-l.a)*(p1.x+p2.x)/2+(-l.b)*(p1.y+p2.y)/2;
  return l;
}

// Find Intersect
POINT getP(LINE l1,LINE l2)
{
  double bse,dx,dy;
  POINT p;
  bse=l1.a*l2.b-l2.a*l1.b;
  dx=(-l1.c)*l2.b-(-l2.c)*l1.b;
  dy=l1.a*(-l2.c)-l2.a*(-l1.c);
  p.x=dx/bse;
  p.y=dy/bse;
  return p;
```

```
}

// Find Circle
CIRCLE getCIRCLE(POINT p1,POINT
      p2,POINT p3)
{
  LINE l1,l2;
  CIRCLE c;
  l1=getMVL(p1,p2);
  l2=getMVL(p2,p3);
  c.cen=getP(l1,l2);
  c.r=sqrt( (p1.x-c.cen.x)*(p1.x-c.cen.x) \
          +(p1.y-c.cen.y)*(p1.y-c.cen.y) \
         );
  return c;
}

// n points
POINT pt[MAXN];

// check the third point
void third_check(CIRCLE &c,int id,int id2)
{
  int i;
  c.r=dis(pt[id],pt[id2])/2;
  c.cen.x=(pt[id].x+pt[id2].x)/2;
  c.cen.y=(pt[id].y+pt[id2].y)/2;

  // check the points between two points
  for(i=0;i<id;i++)
    if(dis(c.cen,pt[i])>c.r)
      c=getCIRCLE(pt[id],pt[id2],pt[i]);
}

// check the second point
void second_check(CIRCLE &c,int id)
{
  int i;
  c.r=dis(pt[0],pt[id])/2;
  c.cen.x=(pt[0].x+pt[id].x)/2;
  c.cen.y=(pt[0].y+pt[id].y)/2;

  // check the points between two points
  for(i=1;i<id;i++)
    if(dis(c.cen,pt[i])>c.r)
      third_check(c,i,id);
}

// find the minimum radius of n points
void min_enclosing_circle(CIRCLE &c,int n)
{
  int i;

  // only one point
  if(n==1)
  {
    c.cen=pt[0];
    c.r=0.0;
    return;
  }

  // radius for two points
  c.cen.x=(pt[0].x+pt[1].x)/2;
  c.cen.y=(pt[0].y+pt[1].y)/2;
  c.r=dis(pt[0],pt[1])/2;
  if(n==2) return;

  // check the third point
  for(i=2;i<n;i++)
    if(dis(c.cen,pt[i])>c.r)
      second_check(c,i);
}

int main(void)
{
  int i,n;
  CIRCLE c;

  while(scanf("%d",&n)==1)
  {
    // Read n points
    for(i=0;i<n;i++)
      scanf("%lf%lf",&pt[i].x,&pt[i].y);

    // Find the radius enclosing all points
    min_enclosing_circle(c,n);
    printf("%lf %lf %lf\n",c.cen.x,c.cen.y,c.r);
  }

  return 0;
}
```

13. MinimumVertexCover

```
#include<cstdio>
#include<cstring>
#define MAXN 16
#define INF 1e9
#define min(a,b) ((a)<(b)?(a):(b))
using namespace std;

// Adj. Matrix
int edg[MAXN][MAXN];

// DP
int dp[1<<MAXN][MAXN];
bool used[1<<MAXN][MAXN];

// Initialize
void ini(void)
{
  int i,j;
  memset(used,false,sizeof(used));
}

// TSP
int TSP(int sts,int now,int n)
{
  int i,prev;

  // Basic - start at 0
  if(sts==1<<now)
    return edg[0][now];

  // Exist
  if(used[sts][now])
    return dp[sts][now];

  dp[sts][now]=INF;
  used[sts][now]=true;
  prev=sts-(1<<now);
  for(i=0;i<n;i++)
    if(prev&(1<<i))
      dp[sts][now]=min(dp[sts][now], \
            TSP(prev,i,n)+edg[i][now]);

  return dp[sts][now];
}
```

```
45
46  // Main
47  int main(void)
48  {
49    int n,i,j;
50
51    // Read Input
52    while(scanf("%d",&n)==1)
53    {
54      if(!n)
55      {
56        printf("0\n");
57        continue;
58      }
59      ini();
60      for(i=0;i<n;i++)
61        for(j=0;j<n;j++)
62          scanf("%d",&edg[i][j]);
63      printf("%d\n",TSP((1<<n)-1,0,n)); // 起點永遠
            是0
64  /* 若不需要回到0,則
65  ans=min(ans,TSP((1<<n)-1,i,n)), i=0~n-1 */
66    }
67    return 0;
68  }
```

14. GEOMETRY

```
1   #define eps 1e-7
2   #define MAXPN 1003
3   #define MAXLN 1003
4   int dcmp(double x){
5   if(fabs(x)<eps)return 0;
6     else return x<0? -1:1;
7   }
8
9   typedef struct Point{
10    double x,y;
11    Point(double x=0,double y=0):x(x),y(y){}
12    Point operator/(double r)const{ return
            Point(x/r , y/r); }
13
14    double operator*(const Point &p)const{ // 內積
15    return x*p.x + y*p.y;
16    }
17    double operator^(const Point &p)const{ // 外積
18      return x*p.y - y*p.x;
19    }
20
21    bool operator< (const Point &p)const{ // 小於
22      return dcmp(x-p.x)< 0 ||
23  (dcmp(x-p.x)==0 && dcmp(y-p.y) < 0);
24    }
25
26      double len() { return sqrt(x*x+y*y); } // 長度
27      double len2(){ return x*x+y*y; }
28      double angle(){ return atan2(y,x); } // 向量極
            角
29  }Point;
30  typedef Point Vector;
31
32  Vector rot(Vector v,double a){ // 向量逆時針旋轉a
33      return Vector( v.x * cos(a) - v.y * sin(a),
34                     v.x * sin(a) + v.y * cos(a));
35  }
36
37  typedef struct Line_f{ // ax+by+c=0
38      double a,b,c;
39      Line_f(double a=0,double b=0,double c=0):
40  a(a),b(b),c(c){}
41      Line_f(Point A,Point B){
42          a = A.y - B.y;
43          b = - (A.x - B.x);
44          c = - a*A.x - b*A.y;
45      }
46  }Line_f;
47
48  int get_inter_point(L_f L1, L_f L2, Pt &ret){
        //兩直線交
        點
49      double delta_x=(L2.c*L1.b-L2.b*L1.c); // 1:
            交一
            點
50      double delta_y=(L1.c*L2.a-L1.a*L2.c); // 0: 平
            行
51
52    if(dcmp(delta)==0){
53      if(dcmp(delta_x)==0 &&
            dcmp(delta_y)==0)return 2;
54      else return 0;
55    }
56      ret = Point(delta_x/delta , delta_y/delta);
57      return 1;
58  } // return 2:same line, 1:point, 0:parallel
59
60  int ori(Point A,Point B,Point C){ // △有向面積正
        負ABC
61      return dcmp( (B-A) ^ (C-A) );
62  }
63  double area(Point p[],int n){ // 多邊形有向面積
64      double ret=0.0;
65      for(int i=0;i<n;i++) ret += p[i] ^ p[(i+1)%n ];
66      return ret/2.0;
67  }
68
69  bool btw(Point tar, Point B, Point C){ // 是否在之
        間ABC
70      return dcmp( (B-tar)*(C-tar) ) < 0; // eg: B
            A C
71  }
72
73  //是否在線段上tarBC不含端點()
74  bool isPointOnSegment(Pt tar, Pt B,Pt C){
75      return dcmp((B-tar)^(C-tar))==0 &&
76  dcmp((B-tar)*(C-tar))<0;
77  }
78  //是否在多邊形內tar
79  int isPointInPolygon(Point tar, Point p[], int pn){
80    int wn = 0;
81    for(int i=0;i<pn;i++){
82      if(isPointOnSegment(tar,p[i],p[(i+1)%pn]))return
            -1;
83  if( tar == p[i] || tar == p[(i+1)%pn] )return -1;
84      int k = dcmp((p[(i+1)%pn]-p[i])^(tar-p[i]));
85      int d1= dcmp(p[i].y - tar.y);
86      int d2= dcmp(p[(i+1)%pn].y - tar.y);
87      if(k>0 && d1 <= 0 && d2 > 0) wn++;
88      if(k<0 && d2 <= 0 && d1 > 0) wn--;
89      }
90      if(wn!=0)return 1; // 在內部
91      else return 0; // 在外部
92  }
93  // 是否在凸多邊形內tar
94  int isPointInConvex(Point tar,Point p[],int pn){
95  for(int i=0;i<pn;i++){
96  if(isPointOnSegment(tar,p[i],p[(i+1)%pn]))return
        -1;
97  if( tar == p[i] || tar == p[(i+1)%pn] )return -1;
```

```
98  // 逆時針不再左側或線段上
99  if(dcmp((p[i]-tar)^(p[(i+1)%pn]-tar))<=0)return 0;
100 }
101 return 1;
102 }
103 // 是否在凸多邊形內、上tar
104 int isPointInConvex_NlogN(Point tar,Point p[],int
        n){
105 if(n<3)return 0;
106 if( dcmp((tar-p[0])^(p[ 1]-p[0]))>0 )return 0;
107 if( dcmp((tar-p[0])^(p[n-1]-p[0]))<0 )return 0;
108 int L=2,R=n-1,M,line=-1;
109 while(L<=R){
110 M=(L+R)/2;
111 if( dcmp( (tar-p[0])^(p[M]-p[0]) )>=0 )line=M,
        R=M-1;
112 else L=M+1;
113 }
114 return dcmp( (p[line]-p[line-1])^(tar-p[line-1])
        )>=0;
115 }
116
117 bool segmentIntersection(Pt p1, Pt p2, Pt p3, Pt
        p4){
118     int a123 = ori(p1,p2,p3); /* 線段是否相交*/
119     int a124 = ori(p1,p2,p4); // 含端點
120     int a341 = ori(p3,p4,p1);
121     int a342 = ori(p3,p4,p2);
122     if(a123 == 0 && a124 ==0){ // 共線
123         return btw(p1,p3,p4) || btw(p2,p3,p4) ||
                btw(p3,p1,p2) || btw(p4,p1,p2);
124     }else if(a123*a124<=0 &&
            a341*a342<=0)return true;
125     return false;
126 }
127
128 /* 如果不希望在凸包的邊上有輸入點，把<= 改成< */
129 void getConvexHull(Point p[], int pn, Point ch[],
        int &m){
130     sort(p,p+pn);
131     m=0;
132 for(int i=0;i<pn;i++){
133 while(m>1 &&
        dcmp((ch[m-1]-ch[m-2])^(p[i]-ch[m-2]))<=0)m--;
134       ch[m++] = p[i];
135     }
136     int k=m;
137     for(int i=pn-2;i>=0;i--){
138         while(m>k && dcmp(
                (ch[m-1]-ch[m-2])^(p[i]-ch[m-2])
                )<=0)m--;
139         ch[m++] = p[i];
140     }
141     if(pn>1)m--;
142 }
143
144 /* 有向直線，左邊為對應半平面*/
145 typedef struct Line{
146     Point p; // 直線上任一點
147     Vector v; // 方向向量
148     double angle; // 極角
149
150     Line(Point p=Point(0.0,0.0), Vector
            v=Vector(0.0,0.0)):p(p),v(v){
151         angle=atan2(v.y , v.x);
152     }
153     bool operator <(const Line &L)const{
154         return angle<L.angle;
155     }
156 }Line;
157
158 /* 用有向直線A->B 切割，如果退化，可能會變成一點或
        線段in */
159 void cutPolygonBy2Point(Polygon in,Polygon
        &ou, Point A,Point B){
160     ou.n=0;
161     for(int i=0;i<in.n;i++){
162         Point C = in.p[i];
163         Point D = in.p[(i+1)&in.n];
164         if(dcmp( (B-A)^(C-A)
                )>=0)ou.p[ou.n++]=C;
165         if(dcmp( (B-A)^(C-D) ) !=0 ){
166             Point ip;
167             int ret=get_inter_point(Line_f(A,B),
                    Line_f(C,D), ip);
168             if(isPointOnSegment(ip,C,D))ou.p[ou.n++]=ip;
169         }
170     }
171 }
172 // 是否在有向直線的左側tarL線上不算()
173 bool isPointOnLeft(Line L,Point tar){
174     return dcmp( L.v^(tar-L.p) )>0;
175 }
176 // 兩直線交點，假定交點唯一存在
177 Point getIntersection(Line a,Line b){
178     Vector u = a.p - b.p;
179     double t = ( b.v^u )/( a.v^b.v );
180     return a.p+a.v*t;
181 }
182
183 /* 計算半平面交O(NlogN) */
184 void halfPlaneIntersection(Line L[],int Ln, Point
        poly[], int &polyn){
185     sort(L,L+Ln); //按極角排序
186     int first=0,last=0;
187     Point p[MAXLN]; // p[i是]deq[i和]deq[i的交
            點+1]
188     Line deq[MAXLN];
189     deq[0]=L[0]; // 初始化只有一個半平面
190     for(int i=1;i<Ln;i++){
191         while(first<last &&
                !isPointOnLeft(L[i],p[last-1]))last--;
192         while(first<last &&
                !isPointOnLeft(L[i],p[first]))first++;
193         deq[++last] = L[i];
194         if(dcmp( deq[last].v ^ deq[last-1].v )==0){
195             // 兩向量平行且同向，取內側的
196             last--;
197             if(isPointOnLeft(deq[last],L[i].p))deq[last]=L[i];
198         }
199         if(first<last)p[last-1]=getIntersection(deq[last-1],deq[last]);
200     }
201     while(first<last &&
            !isPointOnLeft(deq[first],p[last-1]))last--;
202     // 刪除無用平面(*)
203     if(last - first<=1){ // 空集(* *)
204         polyn=0;
205         return;
206     }
207     p[last] = getIntersection(deq[last],deq[first]);
208 // 計算首尾兩個平面的交點
209
210     polyn=0;
211     for(int i=first;i<=last;i++)poly[polyn++]=p[i];
212 }
```

15. Dijkstra

```
#define INF 1000000
#define MAX 100
using namespace std;
typedef struct{
  int val,id;
}NODE;
typedef struct{
  int v,w;
}EDGE;
int dis[MAX];
bool find[MAX];
vector<EDGE>edg[MAX];
priority_queue<NODE>pque;
bool operator < (const NODE& a, const NODE&
      b){
  return a.val > b.val;
}
void dijkstra(int start,int n){
  int i,j,now;
  NODE tmp;
  for(i=0;i<n;++i)
  {
    dis[i]=INF;
    find[i]=false;
  }
  dis[start]=0;
  tmp.id=start;
  tmp.val=0;
  pque.push(tmp);
  for(i=0;i<n-1;++i)
  {
    if(pque.empty()) break;
    while(!pque.empty()&&find[pque.top().id])
          pque.pop();
    if(pque.empty()) break;
    now=pque.top().id;
    pque.pop();
    find[now]=true;
    for(j=0;j<(int)edg[now].size();++j)
    {
      if(!find[edg[now][j].v])
      {
        if(dis[now]+edg[now][j].w<dis[edg[now][j].v])
        {
          dis[edg[now][j].v]=dis[now]+edg[now][j].w;
          tmp.id=edg[now][j].v;
          tmp.val=dis[edg[now][j].v];
          pque.push(tmp);
        }
      }
    }
  }
}
```

16. MSS

```
// 若整個都是負的，輸出最大的就好aryentry
#define HM 30 //1D
#define RM 30 //2D
#define CM 30 //3D
int MSS1D(int n , int ary[]){
  int i , sum , M;
  M = sum = 0;
  for(i = 0 ; i<n ; ++i )
  {
    sum += ary[i];
    if(sum < 0 ) sum = 0;
    if(sum > M ) M = sum;
  }
  return M;
}
int MSS2D(int R,int C,int ary[RM][CM])
{
  int left,width,sum,M,i,tmp[RM];
  M=0;
  for(left=0;left<C;++left)
  {
    for(i=0;i<R;++i) tmp[i]=0;
    for(width=0;width+left<C;++width)
    {
      for(i=0;i<R;++i) tmp[i]+=ary[i][left+width];
      sum=MSS1D(R,tmp);
      if(sum>M) M=sum;
    }
  }
  return M;
}
int MSS3D(int H,int R,int C,int
      ary[HM][RM][CM])
{
  int start,width,sum,M,i,j,tmp[RM][CM];
  M=0;
  for(start=0;start<H;++start)
  {
    for(i=0;i<R;++i)
      for(j=0;j<C;++j)
        tmp[i][j]=0;
    for(width=0;width+start<H;++width)
    {
      for(i=0;i<R;++i)
        for(j=0;j<C;++j)
          tmp[i][j]+=ary[width+start][i][j];
      sum=MSS2D(R,C,tmp);
      if(sum>M) M=sum;
    }
  }
  return M;
}
```

17. SCC

```
void dfs_fin(int U) {
    for(Each V in edge(U,V))
        if(!vis[V])
            dfs_fin(V);
    fin_stk.push(U);
}

void dfs_scc(int U) {
    sccV[scc_cnt] = U;
    for(Each V in edge(V, U)) //reverse edge
        if(!vis[V])
            dfs_scc(V);
}

void scc() {
    for(Each V in G)
        if(!vis[V]) dfs_fin(V);
    for(V = stk.top, stk.pop)// foreach V in decre
          fin time
        if(!vis[V]) dfs_scc(V), scc_cnt++;
    for(Each (U,V) in G) {
        if(sccV[U] != sccV[V])
            // (sccV[U], sccv[V]) = true;
    }
}
```

18. SPFA

```
typedef struct node{
  int next,w;
}EDGE;
int count[VMAX],dis[VMAX]; //點個數VMAX
bool inqueue[VMAX];
queue<int>que;
vector<EDGE>edge[VMAX];
bool SPFA(int start,int n){
  int i,now,next;
  for(i=0;i<n;++i){
    count[i]=0;
    dis[i]=INF;
    inqueue[i]=false;
  }
  que.push(start);
  dis[start]=0;
  count[start]=1;
  while(!que.empty())
  {
    now=que.front();
    que.pop();
    inqueue[now]=false;
    for(i=0;i<(int)edge[now].size();++i)
    {
      next=edge[now][i].next;
      if(dis[next]>dis[now]+edge[now][i].w)
      {
        dis[next]=dis[now]+edge[now][i].w;
        if(!inqueue[next])
        {
          que.push(next);
          inqueue[next]=true;
          count[next]++;
        }
        if(count[next]>=n)return true;
      }
    }
  }
  return false;
}
```

19. COINCHANGE

```
// 分別代表錢幣幣值，數量如果有限()
int val[3]={2,4,5}, ,num[3]={2,1,3};

// 無限換錢-能否湊成某價位
memset(dp,false,sizeof(dp));
dp[0]=true;
// 每種錢幣都試試看
for(i=0;i<3;++i){
  // 每種幣值都試試看
  for(j=val[i];j<=100;++j)
    dp[j]|=dp[j-val[i]];
}

// 無限換錢-湊成某價位有幾種
memset(dp,0,sizeof(dp));
dp[0]=1;

// 每種錢幣都試試看
for(i=0;i<3;++i){
// 每種幣值都試試看
  for(j=val[i];j<=100;++j)
    dp[j]+=dp[j-val[i]];
}

// 無限換錢-湊成某價位最少要幾個硬幣
dp[0]=0;
for(i=1;i<=100;++i) dp[i]=INF;
// 每種錢幣都試試看
for(i=0;i<3;++i)
{
  // 每種幣值都試試看
  for(j=val[i];j<=100;++j)
    dp[j]=min(dp[j],dp[j-val[i]]+1);
 }

// 無限換錢-湊成某價位可以用幾個硬幣(Bit Mask)
memset(dp,0,sizeof(dp));
// 每種錢幣都試試看
for(i=0;i<3;++i)
{
// 必可以用個構成該錢幣本身1()
  dp[val[i]]|=1;

// 每種幣值都試試看
  for(j=val[i];j<=100;++j)
    dp[j]|=(dp[j-val[i]]<<1);
}
// ex: 塊是否可以由個硬幣構成63?
if(dp[6]&(1<<(3-1))) printf("Yes\n");
else printf("No");

// 有限換錢-能否湊成某價位
memset(dp,false,(total+1)*sizeof(bool));
dp[0]=true;
for(i=0;i<n;i++){
  memset(used,0,(total+1)*sizeof(int));
  for(j=val[i];j<=total;j++){
    if(!dp[j]&&dp[j-val[i]]&&used[j-val[i]]<num[i]){
      dp[j]=true;
      used[j]=used[j-val[i]]+1;
    }
  }
}


// 有限換錢-湊成某價位有幾種
memset(dp,0,(total+1)*sizeof(int));
dp[0]=1;

// 嘗試每一種錢幣
for(i=0;i<n;++i){
  // 由後往前嘗試每一種存在的幣值
  for(j=total-val[i];j>=0;--j){
    // 如果該幣值已可構成才需要更新
    if(dp[j]){
      // 由後往前嘗試不同數量
      for(k=num[i];k>0;--k)
      {
        // 超出要求的範圍
        if(j+k*val[i]>total) continue;
        dp[j+k*vak[i]]+=dp[j];
      }
    }
  }
}

// 有限換錢-湊成某價位最少要幾個硬幣
dp[0]=0;
for(i=1;i<=100;++i) dp[i]=INF;
// 每種錢幣都試試看
for(i=0;i<3;++i){
  // 跑num[i次]
  for(k=0;k<num[i];++k){
    // 每種幣值都試試看
    for(j=100;j>=val[i];--j)
```

```
96          dp[j]=min(dp[j],dp[j-val[i]]+1);
97      }
98  }
99
100 // 有限換錢-湊成某價位可以用幾個硬幣(Bit Mask) [陣
        列用DPlong long]
101 memset(dp,0,sizeof(dp));
102 // 每種錢幣都試試看
103 for(i=0;i<3;++i){
104   // 跑num[i次]
105   for(k=0;k<num[i];++k){
106     // 每種幣值都試試看
107     for(j=100;j>=val[i];--j)
108       dp[j]|=(dp[j-val[i]]<<1LL);
109     // 必定可以用個構成該幣值本身1()
110     dp[val[i]]|=1LL;
111   }
112 }
```

20. DancingLinkAlgorithmX

```
1   #include <stdio.h>
2   #define DLX_MAX_ROW 33
3   #define DLX_MAX_COL 333
4   #define DLX_MAX_NODE
        ((DLX_MAX_ROW)*(DLX_MAX_COL)+1)
5   #define DLX_HEAD 0
6   typedef struct DN
7   {
8       int row,col;
9       int L,R,U,D;    /* Left Right Up Down */
10  }DN;
11  DN Dn[ DLX_MAX_NODE ]; /* DLX node */
12  int Rh[ DLX_MAX_ROW ];  /* Row head */
13  int Cs[ DLX_MAX_COL ];  /* Column size */
14  int Ar[ DLX_MAX_ROW ];  /* Answer row */
15  int Row,Col,Nc,Dl;   /* Node cnt , dlx limit */
16
17  /* For RepeatCover */
18  int Cm[DLX_MAX_COL];/* Column mark */
19
20  void DLX_init(int dlx_row,int dlx_col)
21  {
22      int i;
23      Col=dlx_col;
24      Row=dlx_row;
25      Dn[DLX_HEAD].L=Col;
26      Dn[DLX_HEAD].R=1;
27      Dn[DLX_HEAD].U=DLX_HEAD;
28      Dn[DLX_HEAD].D=DLX_HEAD;
29      for(i=1;i<=Col;++i){
30          Dn[i].L=i-1;
31          Dn[i].R=i+1;
32          Dn[i].U=i;
33          Dn[i].D=i;
34          Dn[i].col=i;
35          Dn[i].row=0;
36          Cs[i]=0;
37      }
38      Dn[Col].R=DLX_HEAD;
39      Nc=Col+1;
40      for(i=1;i<=Row;++i)Rh[i]=-1;
41  }
42
43  void DLX_add_back(int dlx_row,int dlx_col)
44  {
45      Dn[Nc].col=dlx_col;
46      Dn[Nc].row=dlx_row;
47      if(Rh[dlx_row]==-1){
48          Rh[dlx_row]=Nc;
49          Dn[Nc].L=Nc;
50          Dn[Nc].R=Nc;
51          Dn[Nc].U=Dn[dlx_col].U;
52          Dn[Nc].D=dlx_col;
53          Dn[ Dn[dlx_col].U ].D=Nc;
54          Dn[dlx_col].U=Nc;
55      }else{
56          Dn[Nc].L = Dn[ Rh[dlx_row] ].L;
57          Dn[Nc].R = Rh[dlx_row];
58          Dn[Nc].U = Dn[dlx_col].U;
59          Dn[Nc].D = dlx_col;
60          Dn[ Dn[Nc].L ].R = Nc;
61          /* Dn[Dn[Nc].R].L=Nc; */
62          Dn[ Rh[dlx_row] ].L = Nc;
63          Dn[ Dn[dlx_col].U ].D = Nc;
64          Dn[dlx_col].U = Nc;
65      }
66      ++Nc;
67      ++Cs[dlx_col];
68  }
69
70  void DLX_EC_remove(int dlx_col)
71  {
72      int i,j;
73      Dn[ Dn[dlx_col].L ].R = Dn[dlx_col].R;
74      Dn[ Dn[dlx_col].R ].L = Dn[dlx_col].L;
75      for(i=Dn[dlx_col].D;i!=dlx_col;i=Dn[i].D){
76          for(j=Dn[i].R;j!=i;j=Dn[j].R){
77              Dn[ Dn[j].D ].U = Dn[j].U;
78              Dn[ Dn[j].U ].D = Dn[j].D;
79              --Cs[ Dn[j].col ];
80          }
81      }
82  }
83  void DLX_EC_resume(int dlx_col)
84  {
85      int i,j;
86      for(i=Dn[dlx_col].U;i!=dlx_col;i=Dn[i].U){
87          for(j=Dn[i].L;j!=i;j=Dn[j].L){
88              Dn[ Dn[j].D ].U=j;
89              Dn[ Dn[j].U ].D=j;
90              ++Cs[ Dn[j].col ];
91          }
92      }
93      Dn[ Dn[dlx_col].L ].R=dlx_col;
94      Dn[ Dn[dlx_col].R ].L=dlx_col;
95  }
96  int DLX_EC_search(int dlx_k)
97  {
98      int dlx_choose_col=-1;
99      int dlx_chosen_size=-1;
100     int i,j,ret;
101     if(Dn[DLX_HEAD].R==DLX_HEAD){
102         Dl=dlx_k;
103         return 1;
104     }
105     for(i=Dn[DLX_HEAD].R;i!=DLX_HEAD;i=Dn[i].R){
106         if(dlx_chosen_size==-1 ||
                Cs[i]<dlx_chosen_size){
107             dlx_chosen_size=Cs[i];
108             dlx_choose_col=i;
109         }
110     }
111     DLX_EC_remove(dlx_choose_col);
112     for(i=Dn[dlx_choose_col].D; \
113         i!=dlx_choose_col; \
114         i=Dn[i].D){
115
```

```
        Ar[dlx_k]=Dn[i].row;
        for(j=Dn[i].R;j!=i;j=Dn[j].R)
            DLX_EC_remove(Dn[j].col);
        ret=DLX_EC_search(dlx_k+1);
        if(ret==1)return 1;
        for(j=Dn[i].L;j!=i;j=Dn[j].L)
            DLX_EC_resume(Dn[j].col);
    }
    DLX_EC_resume(dlx_choose_col);
    return 0;
}

void DLX_RC_remove(int dlx_col)
{
    int i;
    for(i=Dn[dlx_col].D;i!=dlx_col;i=Dn[i].D){
        Dn[ Dn[i].R ].L = Dn[i].L;
        Dn[ Dn[i].L ].R = Dn[i].R;
    }
}
void DLX_RC_resume(int dlx_col)
{
    int i;
    for(i=Dn[dlx_col].U;i!=dlx_col;i=Dn[i].U){
        Dn[ Dn[i].R ].L=i;
        Dn[ Dn[i].L ].R=i;
    }
}
int DLX_RC_h()
{
    int i,j,k;
    int ret=0;
    for(i=1;i<=Col;++i)Cm[i]=0;
    for(k=Dn[DLX_HEAD].R;k!=DLX_HEAD;k=Dn[k].R){
        if(Cm[k]==0){
            ++ret;
            Cm[k]=1;
            for(i=Dn[k].D;i!=k;i=Dn[i].D)
                for(j=Dn[i].R;j!=i;j=Dn[j].R)
                    Cm[Dn[j].col]=1;
        }
    }
    return ret;
}
int DLX_RC_search_up(int dlx_k)
{
    int dlx_choose_col=-1;
    int dlx_chosen_size=-1;
    int i,j,ret;
    int dlx_h=DLX_RC_h();
    if(dlx_k+dlx_h>=Dl)return 0;
        /* down should update Dl limit and
           answer row here */
    if(Dn[DLX_HEAD].R==DLX_HEAD){
        Dl=dlx_k;
        return 1;
    }
    for(i=Dn[DLX_HEAD].R;i!=DLX_HEAD;i=Dn[i].R){
        if(dlx_chosen_size==-1 ||
           Cs[i]<dlx_chosen_size){
            dlx_chosen_size=Cs[i];
            dlx_choose_col=i;
        }
    }
    for(i=Dn[dlx_choose_col].D; \
        i!=dlx_choose_col; \
        i=Dn[i].D){

        DLX_RC_remove(i);
        for(j=Dn[i].R;j!=i;j=Dn[j].R)
            DLX_RC_remove(j);
        Ar[dlx_k]=Dn[i].row;
        ret=DLX_RC_search_up(dlx_k+1);
        /* up return */
            if(ret==1)return 1;
        /* down dont return */
        for(j=Dn[i].L;j!=i;j=Dn[j].L)
            DLX_RC_resume(j);
        DLX_RC_resume(i);
    }
    return 0;
}
int DLX_RC_IDAstar()
{
    Dl=DLX_RC_h();
    while(DLX_RC_search_up(0)==0){
        if(Dl>Row)return -1;
        ++Dl;
    }
    return Dl;
}
int main()
{
    int
        map[DLX_MAX_ROW][DLX_MAX_COL],N,M,i,j,cc;
    while(scanf("%d%d",&M,&N)!=EOF){
        for(i=1;i<=M;i++)
            for(j=1;j<=N;j++)
                scanf("%d",&map[i][j]);
        DLX_init(M,N);
        for(i=1;i<=M;i++)
            for(j=1;j<=N;j++)
                if(map[i][j]==1)
                    DLX_add_back(i,j);
        puts("-----test beg-----");
                /*** RepeatCover ***/
        cc=DLX_RC_IDAstar();
        printf("%d\n",Dl);
        for(i=0;i<cc;++i)
            printf("row %d\n",Ar[i]);

        /*** ExactCover ***/
        cc=DLX_EC_search(0);
        if(cc==1){
            puts("Yes");
            printf("deep %d\n",Dl);
            for(i=0;i<Dl;i++)
                printf("%d ",Ar[i]);
            putchar('\n');
        }else{
            puts("No");
        } /* End */
    }
    return 0;
}
```

21. BipartiteMatching

```
// 左邊跟右邊的最多數node
#define MAXL 102
#define MAXR 102
int link[MAXR];
bool used[MAXR];
vector<int>edg[MAXL];
bool DFS(int now)
{
  int i,next;
  for(i=0;i<(int)edg[now].size();++i)
```

```
  {
    next=edg[now][i];
    if(!used[next])
    {
      used[next]=true;
      if(!link[next]||DFS(link[next]))
      {
        link[next]=now;
        return true;
      }
    }
  }
  return false;
}
int Bipartite(int nL,int nR)
{
  int i,ans=0;
  memset(link,0,(nR+1)*sizeof(int));
  for(i=1;i<=nL;++i)
  {
    memset(used,false,(nR+1)*sizeof(bool));
    if(DFS(i)) ++ans;
  }
  return ans;
}
int main(void)
{
  int n,nL,nR,i,a,b;
  // 代表幾個，跟分別是左右有幾個點npairnLnRset
  while(scanf("%d%d%d",&n,&nL,&nR)==3)
  {
    // ID 1~nL/nR
    for(i=0;i<n;++i)
    {
      scanf("%d%d",&a,&b);
      edg[a].push_back(b);
    }
    printf("%d\n",Bipartite(nL,nR));
    for(i=1;i<=nL;++i) edg[i].clear();
  }
  return 0;
}
```

22. CantorExpansion

```
int
    factorial[MAXN]={1,1,2,6,24,120,720,5040,40320};
int StN_CantorExpansion(int s[],int n){ //全排列對
    應1~N!
  int i,j,ret=0,cnt;
  for( i=0 ; i<n ; i++){
    for( j=i+1 ,cnt=0 ; j<n ; j++) if(s[j]<s[i])
        cnt++;
    ret += cnt * factorial[n-i-1];
  }
  return ret+1;
}
// 1~N對應全排列!
void NtS_CantorExpansion(int s[],int n,int idx){
  bool vis[MAXN]={false};
  int i , num , cnt;
  idx--;
  for( i=0 ; i<n ; i++){
    cnt = idx/factorial[n-i-1];
    for(num=1 ; num<=n ; num++)
      if(!vis[num]) if( cnt-- ==0)break;
    s[i] = num , vis[num] = true;
    idx %= factorial[n-i-1];
  }
}
```

23. Trick

[比賽技巧]
<Flow>
* 點只能使用 k 次 => 把點拆成兩個點,flow 為 k
=> 若點的 indeg 或 outdeg 只有一條邊，

則不用拆點，該點確定只能使用一次
* 邊只能使用 k 次 => 把邊的 flow 設成 k

<MCMF 使用時機>
基本款:
1. 給定每個邊的 cost 跟 flow
要求最大 flow 下的最小 cost
變形款:
1. Matching+minCost
2. 每條路只能走一次
從起點到終點的最小 cost
(至少兩條 path 以上)=> 一條 PATH
只要 SPFA 就好

<KMP 應用>
prefix function:
給字串 S(長度 L)
問可以用子字串 A 的 k 次方表示 k=?
=>
求出 pi 後
1. 若 L/(L-pi[L]) 不整除 => k=1
2. 否則 k 最大為 L/(L-pi[L])

<LinearProgramming 差分約束>
也叫做線性規劃、Difference Constraint
1. Xa - Xb <= c 等同於
Xb 到 Xa 的單向邊 權重為 c
2. Xa - Xb = c 轉換成
Xa - Xb <= c ,Xa - Xb >= c (也就是 Xb - Xa <= -c)
3.
建立一個 super node
從此點到其他所有點
建立單向邊 權重為 0
4.
進行 SSSP
最後更新出來的 dis[i] 值
代表其中的一組可行解
其距離原點的大小
5.
如果有負環出現
代表矛盾、無解

註:
SPFA 一個 node 最多更新 V-1 次,
當 count>=V 時，有負環；
Bellman Ford 更新 V-1 次 edge,
再更新一次，若可以 relax，有負環

< 樹的特性>
1. 任兩點存在一條 path, 且唯一
2. n 個 node 剛好有 n-1 條 edge

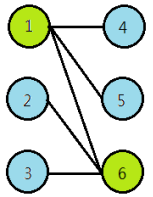
<Maximum Weight Bipartite Matching>
使用 flow 來 model:
左邊 n 個點，右邊 m 個點
0~ 左邊: cost 0, capacity 1
左邊 ~ 右邊: cost -weight, capacity 1
右邊 ~n+m+1: cost 0, capacity 1
求 MCMF
最後輸出-MCMF 即可

< 角度計算>
1. 旋轉點為 (0,0)，以 x 軸正向為轉軸:
=> (x,y) 的逆時針旋轉角度為 atan2(y,x)
// 為徑度，度 =360* 徑度/2pi
// 若 atan2 為負，代表順時針轉-atan2(y,x),
也代表逆時針轉 360.0+atan2(y,x) 度
2. 旋轉點不為 (0,0)，而是求兩向量的夾角:(旋轉點共點)
=> 以到旋轉點的距離當作轉軸 (等同法 1 的平移)

< 切木頭問題>
Q1: 給定木條總長度，某些位置要切，問最少 cost
// cost 為當次切時 (無論位置) 木條剩餘的長度
Sol: 定義 dp[i][j] 代表從 pos[i]~pos[j] 所需要的 min cost
則 dp[i][j]=(pos[j]-pos[i])+min(dp[i][k]+dp[k][j])
k 從 i+1~j-1，而 pos[0]=0，pos[最後]=len
=> DP
Q2: 給定木條總長度，要切成某些長度的木條,
問最少 cost
// cost 為當次切時木條剩餘長度
Sol: 由於不限位置，所以反向思考,
由最小的兩條開始拼裝
保持一個 min_heap,
每次找拼裝起來最小的兩條來拚
並加總 cost 後，丟進 min_heap,
直到 heap 剩下一個為止
=> Greedy

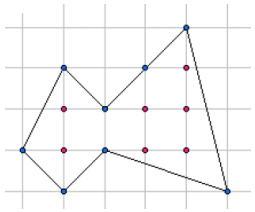
< 博弈理論>
1. 拿石頭問題
總共有 n 堆石頭，每堆有 ni 個石頭
A,B 兩玩家輪流拿石頭，每次只能從一堆選
至少拿一顆，最多整堆拿走,
拿走最後一顆石頭的獲勝
由 A 開始拿，問最後 A 是贏還是輸?
Sol:
把 n 個數字做 XOR，大於 0 就贏，等於 0 就輸

<Bipartite Matching 應用>
設左邊 A 右邊 B，Maximum Bipartite Matching 數 =k:
1. Maximum Independent Set
則答案為 |A|+|B|-k
2. Minimum Vertex Cover
則答案為 k
* 會在 Minimum Vertex Cover 出現的 Vertex
絕對不會出現在 Maximum Independent Set
且兩者互補!



1.6 為 Minimum Vertex Cover
其餘為 Maximum Independent Set
3. Minimum Edge Cover
先求出 k 後，對於所有未匹配的點 (共 h 個),
必可由以匹配點連過去，則答案為 k+h

<Pick's Theorem>



A=10, I=7, E=8
關係式: A = I + E/2 -1

< Chromatic polynomial >
完全圖 Kn: t(t-1)(t-2)⋯(t - (n-1))
n vertices tree: t * (t-1) ^ (n-1)
Cycle Cn: (t-1) ^ (n-1) + (-1) ^ (n)*(t-1)

G / uv : merge u , v
G - uv : removed uv
G + uv: add uv
P(G , k) = P(G - uv , k) - P(G / uv , k)

< 數學式>
直線 ax + by + c = 0 點 (x0 , y0) 到直線距離:

$$\frac{|ax_0 + by_0 + c|}{\sqrt{a^2 + b^2}}$$

< Euler characteristic 歐拉示性數>

$$V - E + F - C = 1$$

< maximal cliques >

```
BronKerbosch1(R, P, X):
    if P and X are both empty:
        report R as a maximal clique
    for each vertex v in P:
    BronKerbosch1(R ⋃ {v}, P ⋂ N(v), X ⋂ N(v))
        P := P \ {v}
        X := X ⋃ {v}
```

ACM ICPC Team Reference - Content

National Cheng Kung University - VIMers