1 DataStructure

1.1 BIT

```cpp
#define lowbit(x) x&-x

int arr[N]; //紀錄前綴和
int bit[N];

void conv(int a[], int n)  //離散化
{
    vector<int> tmp;
    for(int i = 1; i <= n; i++) tmp.push_back(a[i]);
    sort(tmp.begin(), tmp.end());
    for(int i = 1; i <= n; i++) a[i] = lower_bound(tmp.begin(), tmp.end(), a[i]) - tmp.begin() + 1;
}

void buildbit() //每個bit[x]紀錄[x-lowbit(x)+1, x]的總和
{
    for(int i = 0; i < n; i++) bit[i] = arr[i]-arr[i-lowbit(i)];
}

int sum(int x) //查詢[1,x]的總和
{
    int rtn = 0;
    for(;x;x-=lowbit(x)) rtn += bit[x];
    return rtn;
}

void modify(int x, int d) //把位置x的東西加上d
{
    for(;x<=n;x+=lowbit(x)) bit[x] += d;
}
```

1.2 2d_st_tag

```cpp
//二維陣列單點查詢區間加值
class St1d
{
private:
    ll st[4*N];

public:
    void build();
    void modify(int l, int r, int idx, int L, int R, ll v);
    ll query(int l, int r, int idx, int x);
    void down(int idx);
};

void St1d::build()
{
    memset(st, 0, sizeof(st));
}

void St1d::modify(int l, int r, int idx, int L, int R, ll v)
{
    if(r < L || R < l) return;
    if(L <= l && r <= R)
    {
        st[idx] += v;
        return;
    }
    assert(l != r);
    down(idx);
    int mid = (l+r)/2;
    modify(l, mid, idx*2, L, R, v);
    modify(mid+1, r, idx*2+1, L, R, v);
}

ll St1d::query(int l, int r, int idx, int x)
{
    if(x < l || r < x) return 0;
    if(l == x && r == x) return st[idx];
    down(idx);
    int mid = (l+r)/2;
    ll left = query(l, mid, idx*2, x);
    ll right = query(mid+1, r, idx*2+1, x);
    return left+right;
}

void St1d::down(int idx)
{
    st[idx*2] += st[idx], st[idx*2+1] += st[idx];
    st[idx] = 0;
}

//////////////////////

class St2d
{
private:
    St1d st[4*N];

public:
    void build(int il, int ir, int idx);
    void modify(int il, int ir, int jl, int jr, int idx, int iL, int iR, int jL, int jR, ll v);
    ll query(int il, int ir, int jl, int jr, int idx, int i, int j);
};

void St2d::build(int il, int ir, int idx)
{
    st[idx].build();
    if(il == ir) return;
    int mid = (il+ir)/2;
    build(il, mid, idx*2);
    build(mid+1, ir, idx*2+1);
}

void St2d::modify(int il, int ir, int jl, int jr, int idx, int iL, int iR, int jL, int jR, ll v)
{
    if(ir < iL || iR < il) return;
    if(iL <= il && ir <= iR)
    {
        st[idx].modify(jl, jr, 1, jL, jR, v); return;
    }
    int mid = (il+ir)/2;
    modify(il, mid, jl, jr, idx*2, iL, iR, jL, jR, v);
    modify(mid+1, ir, jl, jr, idx*2+1, iL, iR, jL, jR, v);
}

ll St2d::query(int il, int ir, int jl, int jr, int idx, int i, int j)
{
    ll tot = 0;
    if(i < il || ir < i) return 0;
    if(il <= i && i <= ir) tot += st[idx].query(jl, jr, 1, j);
    if(il == i && ir == i) return tot;
    int mid = (il+ir)/2;
    tot += query(il, mid, jl, jr, idx*2, i, j);
    tot += query(mid+1, ir, jl, jr, idx*2+1, i, j);
    return tot;
}
```

1.3 undo_disjoint_set

```cpp
struct DisjointSet {
  // save() is like recursive
  // undo() is like return
  int n, fa[MXN], sz[MXN];
  vector<pair<int*,int>> h;
  vector<int> sp;
  void init(int tn) {
    n=tn;
    for (int i=0; i<n; i++) sz[fa[i]=i]=1;
    sp.clear(); h.clear();
  }
  void assign(int *k, int v) {
    h.PB({k, *k});
    *k=v;
  }
  void save() { sp.PB(SZ(h)); }
  void undo() {
    assert(!sp.empty());
    int last=sp.back(); sp.pop_back();
    while (SZ(h)!=last) {
      auto x=h.back(); h.pop_back();
      *x.F=x.S;
    }
  }
  int f(int x) {
    while (fa[x]!=x) x=fa[x];
    return x;
  }
  void uni(int x, int y) {
    x=f(x); y=f(y);
    if (x==y) return ;
    if (sz[x]<sz[y]) swap(x, y);
    assign(&sz[x], sz[x]+sz[y]);
    assign(&fa[y], x);
  }
}djs;
```

1.4 disjoint_set

```cpp
// path compression
int f[N];

int findrt(int x)
{
```

```
    if(f[x] == x) return x;
    else return f[x] = findrt(f[x]);
}

int same(int x, int y)
{
    return findrt(x) == findrt(y);
}

void uni(int x, int y)
{
    f[findrt(y)] = findrt(x);
}

void init()
{
  for(int i = 0; i < N; i++) f[i] = i;
}

//union by rank
int f[N];  //disjoint set
int rk[N]; //union by rank

int findrt(int x)
{
    if(f[x] == x) return x;
    else return f[x] = findrt(f[x]);
}

bool same(int x, int y)
{
    return findrt(x) == findrt(y);
}

void uni(int x, int y)
{
    x = findrt(x), y = findrt(y);
    if(x == y) return;
    if(rk[x] < rk[y]) f[x] = y;
    else if(rk[x] == rk[y]) f[x] = y, rk[y]++;
    else f[y] = x;
}

void init()
{
  for(int i = 0; i < N; i++) f[i] = i, rank[i] = 0;
}
```

1.5 1d_segTree_tag

```
//線段樹懶人標記：一維陣列區間加值區間乘值區間查詢總和
struct Node //data = data*mul+add;
{
    ll data, mul, add;
};

ll getval(int l, int r, int idx)
{
    return (st[idx].data*st[idx].mul%MD+(r-l+1)*st[idx].add%
        MD)%MD;
}

void up(int l, int r, int idx)
{
    int mid = l+(r-l)/2;
    st[idx].data = (getval(l, mid, idx*2)+getval(mid+1, r,
        idx*2+1))%MD;
}

void down(int l, int r, int idx)
{
    st[idx].data = getval(l, r, idx);
    int lson = idx*2, rson = idx*2+1;
    if(l != r)
    {
        st[lson].mul = st[lson].mul*st[idx].mul%MD;
        st[lson].add = (st[lson].add*st[idx].mul+st[idx].add)
            %MD;
        st[rson].mul = st[rson].mul*st[idx].mul%MD;
        st[rson].add = (st[rson].add*st[idx].mul+st[idx].add)
            %MD;
    }
    st[idx].mul = 1, st[idx].add = 0;
}

void buildst(int l, int r, int idx)
{
    st[idx].mul = 1, st[idx].add = 0;
    if(l == r)
    {
        st[idx].data = arr[l];
        return;
    }
    int mid = l+(r-l)/2;
    buildst(l, mid, idx*2);
    buildst(mid+1, r, idx*2+1);
    up(l, r, idx);
}

void add(int l, int r, int idx, int L, int R, int v) //操作L,
    R
{
    if(r < L || R < l) return;
    if(L <= l && r <= R)
    {
        st[idx].add = (st[idx].add+v)%MD;
        return;
    }
    down(l, r, idx);
    int mid = l+(r-l)/2;
    add(l, mid, idx*2, L, R, v);
    add(mid+1, r, idx*2+1, L, R, v);
    up(l, r, idx);
}

void mul(int l, int r, int idx, int L, int R, int v)
{
    if(r < L || R < l) return;
    if(L <= l && r <= R)
    {
        st[idx].add = st[idx].add*v%MD;
        st[idx].mul = st[idx].mul*v%MD;
        return;
    }
    down(l, r, idx);
    int mid = l+(r-l)/2;
    mul(l, mid, idx*2, L, R, v);
    mul(mid+1, r, idx*2+1, L, R, v);
    up(l, r, idx);
}

ll query(int l, int r, int idx, int L, int R)
{
    if(r < L || R < l) return 0;
    if(L <= l && r <= R)
    {
        return getval(l, r, idx);
    }
    down(l, r, idx);
    int mid = l+(r-l)/2;
    return (query(l, mid, idx*2, L, R)+query(mid+1, r, idx
        *2+1, L, R))%MD;
}
```

1.6 Matrix

```
ll SZ,MOD;
const int MAXSZ=105;

struct Mat
{
    ll m[MAXSZ][MAXSZ];
    Mat(){memset(m, 0, sizeof(m));}
};

Mat matMul(const Mat &A, const Mat &B)
{
    Mat rtn;
    for(int i = 0; i < SZ; i++)
        for(int k = 0; k < SZ; k++)
            if(A.m[i][k])for(int j = 0; j < SZ; j++)
            {
                rtn.m[i][j]+=(A.m[i][k]*B.m[k][j]);
            }
    return rtn;
}
//B is of size SZ
vector<ll> matMul(const Mat &A, const vector<ll> &B)
{
    vector<ll> rtn(SZ,0);
    for(int i = 0; i < SZ; i++)
        for(int j = 0; j < SZ; j++)
            rtn[i]=(rtn[i]+A.m[i][j]*B[j]);

    return rtn;
}

Mat matPow(Mat& M, ll p)
{
    if(p == 0)
    {
        Mat iden;
        for(int i=0;i<SZ;i++)iden.m[i][i]=1;
        return iden;
    }
    if(p == 1)return M;
    Mat rtn = matPow(M, p/2);
    if(p&1)return matMul(matMul(rtn, rtn), M);
    else return matMul(rtn, rtn);
}
```

1.7 treap

```
struct Treap
{
    int pri, sz;
    int rev;
    ll data, sum;   // tag: make-same
    Treap *lchild, *rchild;
    Treap(ll d):pri(rand()), sz(1), rev(0), data(d), sum(d),
        lchild(NULL), rchild(NULL)
    {
    }
    inline void up();
    inline void down();
};

inline int size(Treap *t) { return t? t->sz:0; }
inline ll get_data(Treap *t) { return t? t->data:0; }
inline ll get_sum(Treap *t) { return t? t->sum:0; }

inline void Treap::up()
{
    if(lchild) lchild->down();
    if(rchild) rchild->down();
    sz = 1+size(lchild)+size(rchild);
    sum = get_sum(lchild) + data + get_sum(rchild);
}

inline void Treap::down()
{
    if(rev)
    {
        swap(mxpre, mxpost);
        swap(lchild, rchild);
        if(lchild) lchild->rev ^= 1;
        if(rchild) rchild->rev ^= 1;
        rev ^= 1;
    }
}

Treap *merge(Treap *a, Treap *b)
{
    if(!a || !b) return (a? a:b);
    if(a->pri < b->pri)
    {
        a->down();
        a->rchild = merge(a->rchild, b);
        a->up();
        return a;
    }
    else
    {
        b->down();
        b->lchild = merge(a, b->lchild);
        b->up();
        return b;
    }
}

void split(Treap *o, Treap *&a, Treap *&b, int k)
{
    if(!o) a = b = NULL;
    else
    {
        o->down();
        if(k >= size(o->lchild)+1)
        {
            a = o;
            split(o->rchild, a->rchild, b, k-size(o->lchild)
                -1);
        }
        else
        {
            b = o;
            split(o->lchild, a, b->lchild, k);
        }
        o->up();
    }
}
```

1.8 1d_segTree

```
void buildst(int l, int r, int idx)  //l, r是st的區間
{
    if(l == r)
    {
        st[idx] = arr[l];
        return;
    }
    int mid = (l+r)/2;
    buildst(l, mid, idx*2);
    buildst(mid+1, r, idx*2+1);
    st[idx] = max(st[idx*2], st[idx*2+1]);
}

ll query(int l, int r, int idx, int L, int R)  //L,R是操作的
    區間
{
    if(r < L || R < l) return -INF;
    if(L <= l && r <= R) return st[idx];
    int mid = (l+r)/2;
    return max(query(l, mid, idx*2, L, R), query(mid+1, r,
        idx*2+1, L, R));
}

void modify(int l, int r, int idx, int x, int v)
{
    if(r < x || x < l) return;
    if(l == r)
    {
        st[idx] += v; return;
    }
    int mid = (l+r)/2;
    modify(l, mid, idx*2, x, v);
    modify(mid+1, r, idx*2+1, x, v);
    st[idx] = max(st[idx*2], st[idx*2+1]);
}
```

2 Flow

2.1 MaxDensitySubgraph

```
#include<stdio.h>
#include<string.h>
const int N=1500;
const double inf=0x3fffffff;
const double eps=1e-8;
int gap[N],dis[N],start,end,ans,sum,head[N],num,dep[N],n,m;
bool vis[N];
struct edge
{
  int st,ed,next;
  double flow;
}e[80*N];
struct node
{
  int x,y;
}P[1100];
void addedge(int x,int y,double w)
{
  e[num].st=x;e[num].ed=y;e[num].flow=w;e[num].next=head[x];
      head[x]=num++;
  e[num].st=y;e[num].ed=x;e[num].flow=0;e[num].next=head[y];
      head[y]=num++;
}
void makemap(double g)
{
  int i;
  memset(head,-1,sizeof(head));
  num=0;
  for(i=1;i<=n;i++)
    addedge(i,end,g);
  for(i=0;i<m;i++)
  {
    addedge(n+i+1,P[i].y,inf);
    addedge(n+i+1,P[i].x,inf);
    addedge(start,n+i+1,1.0);
  }
}
double dfs(int u,double minflow)
{
    if(u==end)return minflow;
    int i,v;
    double f,flow=0.0;
    for(i=head[u];i!=-1;i=e[i].next)
    {
        v=e[i].ed;
        if(e[i].flow>0)
        {
            if(dis[v]+1==dis[u])
            {
                f=dfs(v,e[i].flow>minflow-flow?minflow-flow:e
                    [i].flow);
                flow+=f;
                e[i].flow-=f;
                e[i^1].flow+=f;
                if(minflow-flow<=1e-8)return flow;
                if(dis[start]>=ans)return flow;
            }
        }
    }
    if(--gap[dis[u]]==0)
        dis[start]=ans;
    dis[u]++;
    gap[dis[u]]++;
    return flow;
}
double isap()
```

```
{
    double maxflow=0.0;
    memset(gap,0,sizeof(gap));
    memset(dis,0,sizeof(dis));
    gap[0]=ans;
    while(dis[start]<ans)
        maxflow+=dfs(start,inf);
    return 1.0*m-maxflow;
}
void dfs1(int u)
{
  vis[u]=true;
  if(u>=1&&u<=n)
  sum++;
  for(int i=head[u];i!=-1;i=e[i].next)
  {
    int v=e[i].ed;
    if(vis[v]==false&&e[i].flow>0)
      dfs1(v);
  }
}
int main()
{
  int i;
  double Left,Right,mid,flow;
  while(scanf("%d%d",&n,&m)!=-1)
  {
    if(m==0){printf("1\n1\n");continue;}
    start=0,end=n+m+1,ans=end+1;
    for(i=0;i<m;i++)
    {
      scanf("%d%d",&P[i].x,&P[i].y);
    }
    Left=0;Right=m;
    while(Right-Left>=1.0/n/n)//胡伯涛的论文给出了证明,不同解
        之间误差的精度不超过1/(n*n)
    {
      mid=(Left+Right)/2;
      makemap(mid);
      flow=isap();//求出最大权值闭合图
      if(flow<eps)//如果小于0，g值太大
        Right=mid;
      else Left=mid;
    }
    makemap(Left);//最大密度建图
    isap();
    memset(vis,false,sizeof(vis));
    sum=0;
    dfs1(start);
    printf("%d\n",sum);
    for(i=1;i<=n;i++)
      if(vis[i]==true)//残留网络中源点能到达的点
        printf("%d\n",i);
  }
  return 0;
}
```

2.2 dinic

```
template<typename T>
struct DINIC{
  static const int MAXN=105;
  static const T INF=INT_MAX;
  int n, level[MAXN], cur[MAXN];
  struct edge{
    int v,pre;
    T cap,flow,r;
    edge(int v,int pre,T cap):v(v),pre(pre),cap(cap),flow(0),
        r(cap){}
  };
  int g[MAXN];
  vector<edge> e;
  void init(int _n){
    memset(g,-1,sizeof(int)*((n=_n)+1));
    e.clear();
  }
  void add_edge(int u,int v,T cap,bool directed=false){
    e.push_back(edge(v,g[u],cap));
    g[u]=e.size()-1;
    e.push_back(edge(u,g[v],directed?0:cap));
    g[v]=e.size()-1;
  }
  int bfs(int s,int t){
    memset(level,0,sizeof(int)*(n+1));
    memcpy(cur,g,sizeof(int)*(n+1));
    queue<int> q;
    q.push(s);
    level[s]=1;
    while(q.size()){
      int u=q.front();q.pop();
      for(int i=g[u];~i;i=e[i].pre){
        if(!level[e[i].v]&&e[i].r){
          level[e[i].v]=level[u]+1;
          q.push(e[i].v);
          if(e[i].v==t)return 1;
        }
      }
    }
    return 0;
  }
  T dfs(int u,int t,T cur_flow=INF){
    if(u==t)return cur_flow;
    T df;
    for(int &i=cur[u];~i;i=e[i].pre){
      if(level[e[i].v]==level[u]+1&&e[i].r){
        if(df=dfs(e[i].v,t,min(cur_flow,e[i].r))){
          e[i].flow+=df;
          e[i^1].flow-=df;
          e[i].r-=df;
          e[i^1].r+=df;
          return df;
        }
      }
    }
    return level[u]=0;
  }
  T dinic(int s,int t,bool clean=true){
    if(clean){
      for(size_t i=0;i<e.size();++i){
        e[i].flow=0;
        e[i].r=e[i].cap;
      }
    }
    T ans=0, mf=0;
    while(bfs(s,t))while(mf=dfs(s,t))ans+=mf;
    return ans;
  }
};
```

2.3 MinCostMaxFlow

```
template<typename TP>
struct MCMF{
  static const int MAXN=440;
  static const TP INF=999999999;
  struct edge{
    int v,pre;
    TP r,cost;
    edge(int v,int pre,TP r,TP cost):v(v),pre(pre),r(r),cost(
        cost){}
  };
  int n,S,T;
  TP dis[MAXN],PIS,ans;
  bool vis[MAXN];
  vector<edge> e;
  int g[MAXN];
  void init(int _n){
    memset(g,-1,sizeof(int)*((n=_n)+1));
    e.clear();
  }
  void add_edge(int u,int v,TP r,TP cost,bool directed=false)
      {
    e.push_back(edge(v,g[u],r,cost));
    g[u]=e.size()-1;
    e.push_back(
    edge(u,g[v],directed?0:r,-cost));
    g[v]=e.size()-1;
  }
  TP augment(int u,TP CF){
    if(u==T||!CF)return ans+=PIS*CF,CF;
    vis[u]=1;
    TP r=CF,d;
    for(int i=g[u];~i;i=e[i].pre){
      if(e[i].r&&!e[i].cost&&!vis[e[i].v]){
        d=augment(e[i].v,min(r,e[i].r));
        e[i].r-=d;
        e[i^1].r+=d;
        if(!(r-=d))break;
      }
    }
    return CF-r;
  }
  bool modlabel(){
    for(int u=0;u<=n;++u)dis[u]=INF;
    static deque<int>q;
    dis[T]=0,q.push_back(T);
    while(q.size()){
      int u=q.front();q.pop_front();
      TP dt;
      for(int i=g[u];~i;i=e[i].pre){
        if(e[i^1].r&&(dt=dis[u]-e[i].cost)<dis[e[i].v]){
          if((dis[e[i].v]=dt)<=dis[q.size()?q.front():S]){
            q.push_front(e[i].v);
          }else q.push_back(e[i].v);
        }
      }
    }
    for(int u=0;u<=n;++u)
      for(int i=g[u];~i;i=e[i].pre)
        e[i].cost+=dis[e[i].v]-dis[u];
    return PIS+=dis[S], dis[S]<INF;
  }
  TP mincost(int s,int t){
    S=s,T=t;
```

```
    PIS=ans=0;
    while(modlabel()){
      do memset(vis,0,sizeof(bool)*(n+1));
      while(augment(S,INF));
    }return ans;
  }
};
```

3 Geometry

3.1 intercircle

```
vector<Point> interCir(Point o1, double r1, Point o2, double
    r2)
{
  double d=sqrt((o1-o2).abs2());
  double c=(r1*r1 + d*d - r2*r2)/2.0/r1/d;
  double s=sqrt(1.0-c*c);
  Point v=(o2-o1)*r1/d;
  // case 0 intersections
  if(d>r1+r2||d<fabs(r1-r2)) return{};
  // case 1 intersection
  if(d-eps<=r1+r2&&r1+r2<=d+eps) return{o1+v};
  if(d-eps<=fabs(r1-r2)&&fabs(r1-r2)<=d+eps) return{o1-v};
  // case 2 intersections
  Point v_up=(Point){v.x*c-v.y*s,v.x*s+v.y*c};
  Point v_down=(Point){v.x*c+v.y*s,-v.x*s+v.y*c};
  return {o1+v_up,o1+v_down};
} // 求兩圓交點
```

3.2 point

```
const double eps = 5e-8;
struct Point{
  double x,y;
  Point(){}
  Point(double x,double y):x(x),y(y){}
  Point operator+(Point b)const{
    return Point(x+b.x,y+b.y);
  }
  Point operator-(Point b)const{
    return Point(x-b.x,y-b.y);
  }
  Point operator*(double b)const{
    return Point(x*b,y*b);
  }
  Point operator/(double b)const{
    return Point(x/b,y/b);
  }
  bool operator==(Point b)const{
    return (fabs(x-b.x)<=eps&&fabs(y-b.y)<=eps);
  }
  double dot(Point b)const{
    return x*b.x+y*b.y;
  }
  double cross(Point b)const{
    return x*b.y-y*b.x;
  }
  Point normal()const{
    return Point(-y,x);
  } // 求法向量
  double abs2()const{
    return dot(*this);
  } // 向量長度的平方
  double rad(const Point b)const{
    return fabs(atan2(fabs(cross(b)),dot(b)));
  } // 兩向量的弧度
};
```

3.3 SegmentGeometry

```
double EPS = 1e-10;

double add(double a, double b)
{
    if(abs(a+b)<EPS*(abs(a)+abs(b)))return 0;
    else return a+b;
}

struct P//struct for 2d vector/point
{
    double x,y;
    P(){}
    P(double x, double y):x(x),y(y){}
    P operator+(P p){return P(add(x,p.x), add(y,p.y));}
    P operator-(P p){return P(add(x,-p.x), add(y,-p.y));}
    P operator*(double d){return P(x*d,y*d);}
    double dot(P p){return add( x*p.x, y*p.y );}
    double det(P p){return add( x*p.y, -y*p.x );}
};

//is point q on p1p2
bool on_seg(P p1, P p2, P q){return (p1-q).det(p2-q)==0&&(p1-
    q).dot(p2-q)<=0;}

P intersection(P p1, P p2, P q1, P q2)//p and q Must not be
    parallel
{return p1 + (p2-p1)*((q2-q1).det(q1-p1)/(q2-q1).det(p2-p1))
    ;}

bool par(P p1, P p2, P p3, P p4){return (p2-p1).det(p4-p3)
    ==0;}

bool operator<(const P& lhs, const P& rhs)
{return (lhs.x==rhs.x)?lhs.y<rhs.y:lhs.x<rhs.x;}

bool operator==(const P& lhs, const P& rhs)
{return lhs.x==rhs.x&&lhs.y==rhs.y;}

double len(P vec)
{return sqrt(add(vec.x*vec.x, vec.y*vec.y));}

double dis(P p1, P p2)
{return len(p2-p1);}

struct seg
{
    seg(){}
    seg(P _p1, P _p2)
    {
        p[0]=_p1;
        p[1]=_p2;
        if(p[1]<p[0])swap(p[0],p[1]);
    }
    P p[2];
};

bool par(seg& lhs, seg& rhs)
{return par(lhs.p[0],lhs.p[1],rhs.p[0],rhs.p[1]);}

P intersection(seg& lhs, seg& rhs)//p and q Must not be
    parallel
{return intersection(lhs.p[0],lhs.p[1],rhs.p[0],rhs.p[1]);}

bool on_seg(seg& sg, P q)
{return on_seg(sg.p[0],sg.p[1],q);}

bool overlap(seg s1, seg s2)
{
    return par(s1,s2)&&
    ( on_seg(s1,s2.p[0])||on_seg(s1,s2.p[1])||
    on_seg(s2,s1.p[0])||on_seg(s2,s1.p[1]) );
}

bool is_intersect(seg s1, seg s2)
{
    if(par(s1,s2))return false;
    P p0 = intersection(s1,s2);
    return on_seg(s1,p0)&&on_seg(s2,p0);
}

//make sure the vec is not vertical
double interpolate(seg& vec, double X)
{
    double y0=vec.p[0].y,y1=vec.p[1].y,
    x0=vec.p[0].x,x1=vec.p[1].x;
    return y0+(y1-y0)*(X-x0)/(x1-x0);
}

//pts in clockwise order, p[N]=p[0]
bool in_poly(P* pol,int N,P pt)
{
    double X = pt.x,Y=pt.y;
    int pas=0;
    for(int i=0;i<N;i++)
    {
        if(pol[i].x==pol[i+1].x)continue;
        seg s0(pol[i],pol[i+1]);
        //up or down?
        double Y1 = interpolate(s0,X);
        if(Y1<Y-EPS)continue;
        double xl=min(pol[i].x,pol[i+1].x),xr=max(pol[i].x,
            pol[i+1].x);
        if(xl<X-EPS&&xr>=X-EPS)pas++;
    }
    return pas&1;
}

double dpseg(P p, P p1, P p2)//p to p1p2, p1!=p2
{
    P v=p2-p1, v1=p-p1, v2=p-p2;
    if( v.dot(v1) < EPS )return dis(p,p1);
    if( v.dot(v2) > EPS )return dis(p,p2);
    return fabs((p-p1).det(v))/len(v);
}
```

```

double dpseg(P p, seg s1)
{
    return dpseg(p,s1.p[0],s1.p[1]);
}

double dsegseg(P p1, P p2, P p3, P p4)
{
    if( is_intersect( seg(p1,p2), seg(p3,p4) ) )return 0;
    return min( min( dpseg(p1,p3,p4),dpseg(p2,p3,p4) ), min(
        dpseg(p3,p1,p2),dpseg(p4,p1,p2) ) );
}

double dsegseg(seg s1, seg s2)
{
    return dsegseg( s1.p[0],s1.p[1],s2.p[0],s2.p[1] );
}
```

3.4 nearestDist

```
bool cmp_y(P a, P b)
{
    return a.y < b.y;
}

bool cmp_x(P a, P b)
{
    return a.x < b.x;
}

double dc(P *arr, int n)
{
    if(n == 1) return INF;
    int mid = n/2;
    double cx = arr[mid].x;
    double dist = min( dc(arr, mid), dc(arr+mid, n-mid) );
    inplace_merge(arr, arr+mid, arr+n, cmp_y);
    static vector<P> brr; brr.clear();
    for(int i = 0; i < n; i++)
    {
        if(fabs(arr[i].x)-cx >= dist) continue;
        for(int j = brr.size()-1; j >= 0; j--)
        {
            double dx = brr[j].x-arr[i].x;
            double dy = brr[j].y-arr[i].y;
            if(fabs(dy) >= dist) break;
            dist = min(dist, sqrt(dx*dx+dy*dy));
        }
        brr.push_back(arr[i]);
    }
    return dist;
}

double nearestDist(P *arr, int n)
{
    sort(arr, arr+n, cmp_x);
    return dc(arr, n);
}
```

4 Graph

4.1 StronglyConnectedComponent

```
int V, E;
vector<int> G[MAXV];
vector<int> rG[MAXV];
vector<int> vs;//postorder
bool used[MAXV];
int comp[MAXV];//scc id, topologically ordered

void add_edge(int from, int to)
{
    G[from].pb(to);
    rG[to].pb(from);
}

void dfs(int u)//get postorder
{
    used[u] = true;
    for(int i=0;i<G[u].size();i++)
        if(!used[G[u][i]])dfs(G[u][i]);
    vs.pb(u);
}

void rdfs(int u, int k)
{
    used[u]=true;
    comp[u]=k;
    for(int i=0;i<rG[u].size();i++)
        if(!used[rG[u][i]])rdfs(rG[u][i],k);
}

int scc()//return scc cnt
{
    memset(used,0,sizeof(used));
    vs.clear();
    FOR(v,1,V)if(!used[v])dfs(v);
    memset(used,0,sizeof(used));
    int k = 0;//sccID
    FORD(i,V-1,0)
        if(!used[ vs[i] ])
        {
            rdfs(vs[i],k);
            k++;
        }
    return k;
}
```

4.2 bellman_Ford

```
struct edge{ int from, to, cost; };
#define INF 2147483647

edge es[100];

int d[100]; //min distance
int V, E, s, f;

bool bellman_ford()  // return true if there is negative loop
{
    for(int i = 0; i < V; i++) d[i] = INF;
    d[s] = 0;

    for(int i = 0; i < V; i++)
    {
        for(int j = 0; j < E; j++)
        {
            edge e = es[j];
            if(d[e.from] != INF && d[e.to] > d[e.from] + e.
                cost)
            {
                d[e.to] = d[e.from] + e.cost;
                if(i == V - 1) return true; //got neg loop
            }
            if(d[e.to] != INF && d[e.from] > d[e.to] + e.cost
                )
            {
                d[e.from] = d[e.to] + e.cost;
                if(i == V - 1) return true; //got neg loop
            }
        }
    }
    return false;
}
```

4.3 MaxMatching

```
#define FZ(x) memset(x,0,sizeof(x))
struct GenMatch   // 1-base
{
    static const int MAXN = 250;
    int V;
    bool el[MAXN][MAXN];
    int pr[MAXN];
    bool inq[MAXN],inp[MAXN],inb[MAXN];
    queue<int> qe;
    int st,ed;
    int nb;
    int bk[MAXN],djs[MAXN];
    int ans;
    void init(int _V)
    {
        V = _V;
        FZ(el);
        FZ(pr);
        FZ(inq);
        FZ(inp);
        FZ(inb);
        FZ(bk);
        FZ(djs);
        ans = 0;
    }
    void add_edge(int u, int v)
    {
        el[u][v] = el[v][u] = 1;
    }
    int lca(int u,int v)
    {
        memset(inp,0,sizeof(inp));
        while(1)
        {
            u = djs[u];
```

```
            inp[u] = true;
            if(u == st) break;
            u = bk[pr[u]];
        }
        while(1)
        {
            v = djs[v];
            if(inp[v]) return v;
            v = bk[pr[v]];
        }
        return v;
    }
    void upd(int u)
    {
        int v;
        while(djs[u] != nb)
        {
            v = pr[u];
            inb[djs[u]] = inb[djs[v]] = true;
            u = bk[v];
            if(djs[u] != nb) bk[u] = v;
        }
    }
    void blo(int u,int v)
    {
        nb = lca(u,v);
        memset(inb,0,sizeof(inb));
        upd(u);
                upd(v);
        if(djs[u] != nb) bk[u] = v;
        if(djs[v] != nb) bk[v] = u;
        for(int tu = 1; tu <= V; tu++)
            if(inb[djs[tu]])
            {
                djs[tu] = nb;
                if(!inq[tu])
                {
                    qe.push(tu);
                    inq[tu] = 1;
                }
            }
    }
    void flow()
    {
        memset(inq,false,sizeof(inq));
        memset(bk,0,sizeof(bk));
        for(int i = 1; i <= V; i++)
            djs[i] = i;

        while(qe.size()) qe.pop();
        qe.push(st);
        inq[st] = 1;
        ed = 0;
        while(qe.size())
        {
            int u = qe.front();
            qe.pop();
            for(int v = 1; v <= V; v++)
                if(el[u][v] && (djs[u] != djs[v]) && (pr[u]
                    != v))
                {
                    if((v == st) || ((pr[v] > 0) && bk[pr[v]]
                        > 0))
                        blo(u,v);
                    else if(bk[v] == 0)
                    {
                        bk[v] = u;
                        if(pr[v] > 0)
                        {
                            if(!inq[pr[v]]) qe.push(pr[v]);
                        }
                        else
                        {
                            ed = v;
                            return;
                        }
                    }
                }
        }
    }
    void aug()
    {
        int u,v,w;
        u = ed;
        while(u > 0)
        {
            v = bk[u];
            w = pr[v];
            pr[v] = u;
            pr[u] = v;
            u = w;
        }
    }
    int solve()
    {
        memset(pr,0,sizeof(pr));
        for(int u = 1; u <= V; u++)
            if(pr[u] == 0)
            {
                st = u;
                flow();
                if(ed > 0)
                {
                    aug();
                    ans ++;
                }
            }
        return ans;
    }
} gm;
```

4.4 ArticulationPoint

```
vector<int> G[MAXN];
int dfn[MAXN], low[MAXN], tim, ans;
void tarjan(int u, int p){ // p -> u
    int child=0, cut_node=0;
    dfn[u] = low[u] = ++tim;
    for(auto v:G[u]){ // u -> v
        if(!dfn[v]){ // tree edge
            tarjan(v, u); child++;
            low[u]= min(low[u], low[v]);
            if(low[v] >= dfn[u]) cut_node=1;
        }
        else if(v!=p)  //back edge
            low[u]= min(low[u], dfn[v]);
    }

    if(p!=-1 && cut_node) ans++;
    if(p==-1 && child>=2) ans++;
}
//how to call
for(int i=0; i<n; i++)
  if(!dnf[i])
    tarjan(i, -1);
```

4.5 TwoConnected

```
vector<int> G[MAXN];
vector<int> bcc[MAXN]; // bcc內的點

int dfn[MAXN], low[MAXN], tim, ans;
int st[MAXN], top;
int bccID[MAXN], bcc_cnt;// 每個點的bcc編號
bool is_cut[MAXN]; // 是否為割點, 割點的ID會被覆蓋

void tarjan(int u, int p){ // p->u
    int child= 0, w;
    dfn[u] = low[u] = ++tim;
    st[top++] = u;
    for(auto v:G[u]){ // u->v
        if(!dfn[v]){
            tarjan(v, u);  child++;
            low[u]= min(low[u], low[v]);
            if(low[v] >= dfn[u]){
                is_cut[u]= true;
             do{
                    w = st[--top];
                    bccID[w]= bcc_cnt;
                    bcc[bcc_cnt].push_back(w);
                }while(dfn[w] > dfn[v]);
                bccID[u]= bcc_cnt;
                bcc[bcc_cnt++].push_back(u);
            }
        }
        else if(v != p)
            low[u]= min(low[u], dfn[v]);
    }

    if(p == -1 && child<2)
        is_cut[u]= false;
}
```

4.6 TarjanBridge

```
void tarjan(int u,int p ){ // p -> u
    dfn[u] = low[u] = ++tim;

    for(auto v: G[u]){ //u->v
        if(!dfn[v]){
            tarjan(v, u);
            low[u] = min(low[u], low[v]);
            if(low[v] > dfn[u]){
                ans++;
                E.push_back(edge(u ,v));
            }
        }
        else if(v != p)
```

```
            low[u] = min(low[u], dfn[v]);
    }

}
struct edge{
    int from, to;
    edge(int u, int v):
         from(u), to(v){}
};
for(int i=0; i<n; i++)
  if(!dnf[i])
    tarjan(i, -1);
```

4.7 MaxWeightPerfectMatch

```
struct Graph {
  // Minimum General Weighted Matching (Perfect Match) 0-base
  static const int MXN = 105;
  int n, edge[MXN][MXN];
  int match[MXN],dis[MXN],onstk[MXN];
  vector<int> stk;
  void init(int _n) {
    n = _n;
    for (int i=0; i<n; i++)
      for (int j=0; j<n; j++)
        edge[i][j] = 0;
  }
  void add_edge(int u, int v, int w) {
    edge[u][v] = edge[v][u] = w;
  }
  bool SPFA(int u){
    if (onstk[u]) return true;
    stk.push_back(u);
    onstk[u] = 1;
    for (int v=0; v<n; v++){
      if (u != v && match[u] != v && !onstk[v]){
        int m = match[v];
        if (dis[m] > dis[u] - edge[v][m] + edge[u][v]){
          dis[m] = dis[u] - edge[v][m] + edge[u][v];
          onstk[v] = 1;
          stk.push_back(v);
          if (SPFA(m)) return true;
          stk.pop_back();
          onstk[v] = 0;
        }
      }
    }
    onstk[u] = 0;
    stk.pop_back();
    return false;
  }
  int solve() {
    // find a match
    for (int i=0; i<n; i+=2){
      match[i] = i+1, match[i+1] = i;
    }
    for(;;){
      int found = 0;
      for (int i=0; i<n; i++) dis[i] = onstk[i] = 0;
      for (int i=0; i<n; i++){
        stk.clear();
        if (!onstk[i] && SPFA(i)){
          found = 1;
          while (stk.size()>=2){
            int u = stk.back(); stk.pop_back();
            int v = stk.back(); stk.pop_back();
            match[u] = v;
            match[v] = u;
          }
        }
      }
      if (!found) break;
    }
    int ret = 0;
    for (int i=0; i<n; i++)
      ret += edge[i][match[i]];
    ret /= 2;
    return ret;
  }
}graph;
```

4.8 lca

```
const int MAXN=100000; // 1-base
const int MLG=17; //log2(MAXN)+1;
int pa[MLG+2][MAXN+5];
int dep[MAXN+5];
vector<int> G[MAXN+5];
void dfs(int x,int p=0){//dfs(root);
    pa[0][x]=p;
    for(int i=0;i<=MLG;++i)
        pa[i+1][x]=pa[i][pa[i][x]];
    for(auto &i:G[x]){
        if(i==p)continue;
        dep[i]=dep[x]+1;
        dfs(i,x);
    }
}
inline int jump(int x,int d){
    for(int i=0;i<=MLG;++i)
        if((d>>i)&1) x=pa[i][x];
    return x;
}
inline int find_lca(int a,int b){
    if(dep[a]>dep[b])swap(a,b);
    b=jump(b,dep[b]-dep[a]);
    if(a==b)return a;
    for(int i=MLG;i>=0;--i){
        if(pa[i][a]!=pa[i][b]){
            a=pa[i][a];
            b=pa[i][b];
        }
    }
    return pa[0][a];
}
```

4.9 MaximalClique

```
#define MAXN 32
int n, m, Max;
ll v[MAXN], deg[MAXN]; //neighbors

void update_maximum(ll R)
{
    int Size = 0;
    while(R)
    {
        if(R&1)Size++;
        R>>=1;
    }
    Max = max(Size, Max);
}

int pickPivot(ll P)
{
    int pivot = -1, Max = -1;
    memset(deg, 0, sizeof(deg));
    for(int i = 0; i < n; i++)
    {
        if(P&(1LL<<i)) //i is in P
        {
            if(pivot == -1) //i = default pivot
            {
                pivot = i;
                Max = deg[i];
            }
            for(int j = 0; j < i; j++)
            {
                if((P&(1LL<<j))&&(v[i]&(1LL<<j)))
                {
                    deg[i]++;
                    if(deg[i] > Max)
                    {
                        Max = deg[i];
                        pivot = i;
                    }
                    deg[j]++;
                    if(deg[j] > Max)
                    {
                        Max = deg[j];
                        pivot = j;
                    }
                }
            }
        }
    }
    return pivot;
}

void BronKerbosch(ll R, ll P, ll X)
{
    if(!P) //P is empty, no candidates left
    {
        if(!X)
        {
            //clique
             update_maximum(R);
        }
        return;
    }
    int u = pickPivot(P|X);
    for(int i = 0; i <= n-1; i++)
    {
        if(P&(~v[u])&(1LL<<i)) //vi is in P
        {
            BronKerbosch( R|(1LL<<i), P&v[i], X&v[i] );
            P&=(~(1LL<<i));
            X|=(1LL<<i);
        }
```

```
    }
}

int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0);
    while(cin >> n)
    {
        cin >> m;

        Max = 0;
        FOR(i,0,n-1)v[i] = 0;

        int a, b;
        FOR(i,1,m)
        {
            cin >> a >> b;
            v[a]|=(1LL<<b);
            v[b]|=(1LL<<a);
        }
        BronKerbosch(0, (1LL<<n)-1, 0);
        cout << Max << '\n';
    }
    return 0;
}
```

4.10 MinimumMeanCycle

```
#include<cfloat> //for DBL_MAX
int dp[MAXN][MAXN]; // 1-base,O(NM)
vector<tuple<int,int,int>> edge;
double mmc(int n){//allow negative weight
  const int INF=0x3f3f3f3f;
  for(int t=0;t<n;++t){
    memset(dp[t+1],0x3f,sizeof(dp[t+1]));
    for(const auto &e:edge){
      int u,v,w;
      tie(u,v,w) = e;
      dp[t+1][v]=min(dp[t+1][v],dp[t][u]+w);
    }
  }
  double res = DBL_MAX;
  for(int u=1;u<=n;++u){
    if(dp[n][u]==INF) continue;
    double val = -DBL_MAX;
    for(int t=0;t<n;++t)
      val=max(val,(dp[n][u]-dp[t][u])*1.0/(n-t));
    res=min(res,val);
  }
  return res;
}
```

4.11 匹配問題轉換

4.11.1 一般圖匹配問題轉換

1. | 最大匹配邊數 |+| 最小邊涵蓋 |=|V| (無孤立點)
2. | 最大獨立集 |+| 最小點涵蓋 |=|V|
3. 最大權匹配-> 最大權完美匹配: 用 0 邊補成完全圖
4. 最大權最大匹配-> 最大權匹配: 先把所有邊加上 | 最負邊權重 |+1, 得到新的圖 G'上沒有任何負邊, 然後所有邊再加上 G' 上所有邊權重和, 這樣最大權匹配就會 = 最大權最大匹配.

4.12 BridgeConnected

```
struct edge{
    int from, to;
    edge(int u, int v):from(u), to(v){}
};
vector<edge> bridge;
vector<int> G[MAXN];
int dfn[MAXN], low[MAXN], tim;
int bccID[MAXN], bccCNT;
int st[MAXN], top;

void tarjan(int u, int p){ // p->u
    dfn[u]= low[u]= ++tim;
    st[top++]= u;
    for(auto v:G[u]){ // u->v
        if(!dfn[v]){
            tarjan(v, u);
            low[u]= min(low[u], low[v]);
            if(low[v] > dfn[u])
               bridge.push_back(edge(u, v));
        }
        else if(v != p)
            low[u]= min(low[u], dfn[v]);
    }

    if(dfn[u] == low[u]){
        int w;
        do{
            w= st[--top];
            bccID[w]= bccCNT;
        }while(w != u);
        bccCNT++;
    }
}
```

4.13 HeavyLightDecomposition

```
const int MAX_N;
vector<int> link[MAX_N]; //edge

void dfs_build(int now, int fa, int *weight, int *depth, int
     *pa, int *son)
{
  weight[now]=1;
  son[now]=-1;
  pa[now]=fa;
  for(auto i:link[now])
  {
    if(i==fa) continue;
    depth[i]=depth[now]+1;
    dfs_build(i,now,weight,depth,pa,son);
    if(son[now]==-1||weight[son[now]]<weight[i]) son[now]=i;
    weight[now]+=weight[i];
  }
}
void build_top(int now, int top,int *pa, int *son, int *
     link_top)
{
  link_top[now]=top;
  if(son[now]==-1) return;
  build_top(son[now],top,pa,son,link_top);
  for(auto i:link[now])
  {
    if(i==son[now]||i==pa[now]) continue;
    build_top(i,i,pa,son,link_top);
  }
}
inline void HLD(int *weight, int *depth, int *pa, int *son,
     int *link_top)
{
  memset(son,-1,sizeof(int)*MAX_N);
  depth[1]=1; //set node(1) as root
  dfs_build(1,0,weight,depth,pa,son);
  build_top(1,1,pa,son,link_top);
}
inline int find_lca(int x, int y, int *depth, int *pa, int *
     link_top)
{
  int tx=link_top[x], ty=link_top[y];
  while(tx!=ty)
  {
    if(depth[tx]<depth[ty])
    {
      swap(tx,ty);
      swap(x,y);
    }
    tx=link_top[x=pa[x]];
  }
  return depth[x]<depth[y]?x:y;
}
//usage:
//build HeavyLightDecomposition: HLD
//find LCA(x,y): find_lca
```

4.14 MaxWeightPerfectBiMatch

```
const int maxn = 500 + 3, INF = 0x3f3f3f3f;
int n, W[maxn][maxn];
int mat[maxn];
int Lx[maxn], Ly[maxn], slack[maxn];
bool S[maxn], T[maxn];

inline void tension(int &a, const int b) {
    if(b < a) a = b;
}

inline bool match(int u) {
    S[u] = true;
    for(int v = 0; v < n; ++v) {
        if(T[v]) continue;
        int t = Lx[u] + Ly[v] - W[u][v];
        if(!t) {
            T[v] = true;
            if(mat[v] == -1 || match(mat[v])) {
                mat[v] = u;
                return true;
            }
        }else tension(slack[v], t);
```

```
    }
    return false;
}

inline void update() {
    int d = INF;
    for(int i = 0; i < n; ++i)
        if(!T[i]) tension(d, slack[i]);
    for(int i = 0; i < n; ++i) {
        if(S[i]) Lx[i] -= d;
        if(T[i]) Ly[i] += d;
    }
}

inline void KM() {
    for(int i = 0; i < n; ++i) {
        Lx[i] = Ly[i] = 0; mat[i] = -1;
        for(int j = 0; j < n; ++j) Lx[i] = max(Lx[i], W[i][j
            ]);
    }
    for(int i = 0; i < n; ++i) {
        fill(slack, slack + n, INF);
        while(true) {
            for(int j = 0; j < n; ++j) S[j] = T[j] = false;
            if(match(i)) break;
            else update();
        }
    }
}
```

4.15 tarjan

```
// copied from sylveon
// ap
void DFS(int v, int fa) //call DFS(v,v) at first
{
    D[v] = L[v] = timestamp++; //timestamp > 0
    int childCount = 0; //定理1 for root
    bool isAP = false;

    for (int w:adj[v])
    {
        if( w==fa ) continue;
        if ( !D[w] ) // D[w] = 0 if not visited
        {
            DFS(w,v);
            childCount++;
            if (D[v]<=L[w]) isAP = true; //定理2
            if (D[v]< L[w]) edgeBridge.emplace_back(v, w);//
                定理3
            L[v] = min(L[v], L[w]);
        }
        L[v] = min(L[v], D[w]);
    }
    if ( v == fa && childCount < 2 ) isAP = false; //定理1, v
        ==fa只是確認root
    if ( isAP ) nodeAP.push_back(v);
    return ;
}
// SCC
void DFS(int v, int fa) { //call DFS(v,v) at first
    D[v] = L[v] = timestamp++; //timestamp > 0
    st.push(v);

    for (int w:adj[v]) {
        if( w==fa ) continue;
        if ( !D[w] ) { // D[w] = 0 if not visited
            DFS(w,v);
            L[v] = min(L[v], L[w]);
        }
        L[v] = min(L[v], D[w]);
    }
    if( D[v] == L[v] ) {
        int x;
        do{
            x = st.top();
            st.pop();
            scc[x] = SCCID;
        }while( x != v );
        SCCID++;
    }
}
```

4.16 dijkstra

```

struct edge{int to, cost;};
typedef pair<int, int> P; //first = min distance, second = v
    id
#define f first
#define s second
#define INF 2147483647

int V, E, S, F;
vector<edge> G[100];
int d[100];

void dijkstra()
{
    priority_queue<P, vector<P>, greater<P>> q;
    fill(d, d + V, INF);
    d[S] = 0;
    q.push(P(0, S));

    while(!q.empty())
    {
        P p = q.top(); q.pop();
        int v = p.s;
        if(d[v] < p.f) continue;
        for(int i = 0; i < G[v].size(); i++)
        {
            edge e = G[v][i];
            if(d[e.to] > d[v] + e.cost)
            {
                d[e.to] = d[v] + e.cost;
                q.push(P(d[e.to], e.to));
            }
        }
    }
}
```

4.17 spfa

```
typedef pair<int, ll> P;
#define idx first
#define w second
int vn, en;
vector<P> graph[N];
ll dist[N];

bool spfa()  // return true if neg cycle
{
    for(int i = 0; i < vn; i++) dist[i] = INF; dist[0] = 0;
    int cnt[N] = {0};
    bool inq[N] = {false};
    queue<int> q; q.push(0); inq[0] = true;
    while(!q.empty())
    {
        int s = q.front(); q.pop();
        inq[s] = false;
        for(auto e:graph[s])
        {
            if(dist[e.idx] > dist[s]+e.w)
            {
                dist[e.idx] = dist[s]+e.w;
                if(++cnt[e.idx] >= vn) return true;
                if(!inq[e.idx])
                {
                    inq[e.idx] = true;
                    q.push(e.idx);
                }
            }
        }
    }
    return false;
}
```

4.18 MaxBiMatching

```
//注意: 變數V
#define MAXV 505
int V;//# of vertex
vector<int> G[MAXV];
int match[MAXV];
int used[MAXV];

void add_edge(int u, int v)
{
    G[u].pb(v);
    G[v].pb(u);
}

bool dfs(int u)
{
    used[u]=true;
    for(int i = 0; i < G[u].size(); i++)
    {
        int v = G[u][i], w = match[v];
        if(w<0 || !used[w]&&dfs(w) )
        {
            match[u]=v;
            match[v]=u;
            return true;
        }
    }
    return false;
```

```
}

int bip_match()
{
    int res=0;
    memset(match,-1,sizeof(match));
    for(int v=0; v<V; v++)
    {
        if(match[v]<0)
        {
            memset(used,0,sizeof(used));
            if(dfs(v))res++;
        }
    }
    return res;
}
```

5 Math

5.1 modeq

```
ll solve(ll *a, ll *b, ll *m, int N)
{
  ll k = 0, h = 1, gcd, n, t, ar;
  for (ll i = 0; i < N; i++)
  {
    gcd = extgcd(a[i] * h, m[i], ar, t);
    t = (b[i] - a[i] * k) / gcd, n = abs(m[i] / gcd);
    if (t * gcd != b[i] - a[i] * k) return -1;
    t = ((ar * t) % n + n) % n;
    k += h * t, h *= n, k %= h;
  }
  return (k % h + h) % h;
}
// 解n組a*x=b%m, 回傳x.
// 回傳的是最小非負整數解. 無解回傳-1.
```

5.2 FFT

```
const double PI = acos(-1.0);
struct Complex
{
  double x,y;
  Complex(){}
  Complex(double a):x(a),y(0){}
  Complex(double a, double b):x(a),y(b){}
  Complex operator+ (const Complex &a){ return Complex(x+a.x,
       y+a.y); }
  Complex operator- (const Complex &a){ return Complex(x-a.x,
       y-a.y); }
  Complex operator* (const Complex &a){ return Complex(x*a.x-
       y*a.y,x*a.y+y*a.x); }
};
inline vector<Complex> fft(vector<Complex> rtn, int Rev = 1)
{
  int fft_n = rtn.size();
  for(int i=0,j=0;i<fft_n;i++)
  {
    if(i>j) swap(rtn[i],rtn[j]);
    for(int k=(fft_n>>1);(j^=k)<k;k>>=1);
  }
  for(int i=2,m;i<=fft_n;i<<=1)
  {
    m = i>>1;
    for(int j=0;j<fft_n;j+=i)
    {
      for(int k=0;k<m;k++)
      {
        Complex y = rtn[j+k+m]*Complex(cos(2*PI/i*k), Rev*sin
             (2*PI/i*k));
        rtn[j+k+m] = rtn[j+k]-y;
        rtn[j+k] = rtn[j+k]+y;
      }
    }
  }
  for(int i=0;!~Rev&&i<fft_n;i++)
    rtn[i].x = rtn[i].x/fft_n;
  return rtn;
}
// Complex的x為實部, y為虛部.
// 把原多項式包成Complex的vector(poly), 並把項次拓展到2^i, 用
     fft(poly)即可得到轉換後的結果.
// Rev為1時為FFT, 為-1時為InvFFT.
```

5.3 NTT

```
typedef long long ll;

const ll P = (479<<21)+1;
const ll G = 3;
inline ll fpw(ll x, ll y, ll m)
{
  ll rtn = 1;
  for(x=(x>=m?x%m:x);y;y>>=1)
  {
    if(y&1) rtn = rtn*x%m;
    x = x*x%m;
  }
  return rtn;
}
inline vector<ll> ntt(vector<ll> rtn, int Rev = 1)
{
  int ntt_n = rtn.size();
  for(int i=0,j=0;i<ntt_n;i++)
  {
    if(i>j) swap(rtn[i],rtn[j]);
    for(int k=(ntt_n>>1);(j^=k)<k;k>>=1);
  }
  for(int i=2,m=1;i<=ntt_n;i<<=1,m++)
  {
    ll w = 1, wn = fpw(G,(P-1)>>m,P), u, t;
    int mh = i>>1;
    for(int j=0;j<mh;j++)
    {
      for(int k=j;k<ntt_n;k+=i)
      {
        u = rtn[k], t = w*rtn[k+mh]%P;
        rtn[k] = (u+t)%P;
        rtn[k+mh] = (u-t+P)%P;
        w = w*wn%P;
      }
    }
  }
  if(!~Rev)
  {
    for(int i=1;i<ntt_n/2;i++) swap(rtn[i],rtn[ntt_n-i]);
    ll Revn = fpw(ntt_n,P-2,P);
    for(int i=0;i<ntt_n;i++) rtn[i] = rtn[i]*Revn%P;
  }
  return rtn;
}
// 把原多項式包成long long的vector(poly), 並把項次拓展到2^i.
// 用ntt(poly)即可得到轉換後的結果.
// Rev為1時為NTT, 為-1時為InvNTT.
```

5.4 EulerPhi

```
//find in O(sqrt(N))

int euler_phi(int N)
{
    int res=N;
    for(int i=2;i*i<=N;i++)
    {
        if(N%i==0)
        {
            res=res/i*(i-1);
            for(;N%i==0;N/=i);
        }
    }
    if(N!=1)res=res/N*(N-1);//self=prime
    return res;
}

//tabulate in O(MAXN)

int euler[MAXN];

void euler_phi2()
{
    for(int i=0;i<MAXN;i++)euler[i]=i;
    for(int i=2;i<MAXN;i++)
    {
        if(euler[i]==i)
        {
            for(int j=i;j<MAXN;j+=i)
            {
                euler[j]=euler[j]/i*(i-1);
            }
        }
    }
}
```

5.5 BigInt

```
#define MAX_N 1000
#define MAX 100000
#define MAX_LOG 5
```

```
class BigInt
{
  public:
    int sign;
    long long m[MAX_N];
    int l;
    long long operator [](int i) const { return m[i]; }
    long long &operator [](int i) { return m[i]; }
    BigInt(){ l=1, m[0]=0; sign=1; }
    BigInt(int x){ (*this)=x; }
    BigInt(const char *s){ (*this)=s; }
    BigInt operator =(int x)
    {
      if(x<0) x=-x, sign=-1;
      else sign=1;
      for(l=1, m[l-1]=x%MAX, x/=MAX; x; m[l++]=x%MAX, x/=MAX)
          ;
      if(sign==-1&&l==1&&m[0]==0) sign=1;
      return *this;
    }
    BigInt operator =(const char *t)
    {
      int i, j, len;
      const char *s;
      if(t[0]=='-') sign=-1, s=t+1;
      else sign=1, s=t;
      for(len=0; s[len]>='0' && s[len]<='9'; len++);
      for(l=(len+MAX_LOG-1)/MAX_LOG, i=0; i<l; i++)
        for(m[i]=0, j=0; j<MAX_LOG; j++)
          if(len-i*MAX_LOG-MAX_LOG+j>=0)
            m[i]=m[i]*10+s[len-i*MAX_LOG-MAX_LOG+j]-'0';
      if(sign==-1&&l==1&&m[0]==0) sign=1;
      return *this;
    }
    bool scan()
    {
      char s[MAX_N*MAX_LOG+10];
      if(scanf("%s", s)==EOF) return 0;
      else { *this=s; return 1; }
    }
    void print()
    {
      int i;
      char s[8];
      if(sign==-1) printf("-");
      for(sprintf(s, "%%0%dlld", MAX_LOG), printf("%lld", m[l
          -1]), i=l-2; i>=0; printf(s, m[i]), i--);
    }
};
bool operator <(const BigInt &x, const BigInt &y)
{
  if(x.sign!=y.sign) return x.sign<y.sign;
  int i;
  if(x.l!=y.l) return (x.l<y.l) ^ (x.sign==-1);
  for(i=x.l-1; i>=0 && x[i]==y[i]; i--);
  return (i>=0 && x[i]<y[i]) ^ (x.sign==-1);
}
bool operator ==(const BigInt &x, const BigInt &y)
{
  if(x.sign!=y.sign) return false;
  int i;
  if(x.l!=y.l) return 0;
  for(i=x.l-1; i>=0 && x[i]==y[i]; i--);
  return i<0;
}
BigInt operator +(BigInt x, const BigInt &y)
{
  int i;
  long long h;
  for(h=0, i=0; i<x.l || i<y.l || h; i++)
  {
    h+=(i<x.l)*x[i]*x.sign+(i<y.l)*y[i]*y.sign;
    x[i]=h%MAX;
    h/=MAX;
  }
  x.l=i;
  for(; x.l>1 && !x[x.l-1]; x.l--);
  x.sign=(x[x.l-1]>0?1:-1);
  if(x[x.l-1]>0){ for(i=0; i<x.l; i++) if(x[i]<0) x[i+1]--, x
      [i]+=MAX; }
  else for(i=0; i<x.l; i++) if(x[i]>0) x[i+1]++, x[i]-=MAX;
  for(i=0; i<x.l; i++) x[i]*=x.sign;
  if(x.sign==-1&&x.l==1&&x[0]==0) x.sign=1;
  return x;
}
BigInt operator -(BigInt x, const BigInt &y)
{
  int i;
  long long h;
  for(h=0, i=0; i<x.l || i<y.l || h; i++)
  {
    h+=(i<x.l)*x[i]*x.sign-(i<y.l)*y[i]*y.sign;
    x[i]=h%MAX;
    h/=MAX;
  }
  x.l=i;
  for(; x.l>1 && !x[x.l-1]; x.l--);
  x.sign=(x[x.l-1]>0?1:-1);
  if(x[x.l-1]>0){ for(i=0; i<x.l; i++) if(x[i]<0) x[i+1]--, x
      [i]+=MAX; }
  else for(i=0; i<x.l; i++) if(x[i]>0) x[i+1]++, x[i]-=MAX;
  for(i=0; i<x.l; i++) x[i]*=x.sign;
  if(x.sign==-1&&x.l==1&&x[0]==0) x.sign=1;
  return x;
}
BigInt operator *(BigInt x, int y)
{
  int i, sign=1;
  long long h;
  if(y<0) y=-y, sign=-1;
  for(h=0, i=0; i<x.l || h; i++)
  {
    h+=(i<x.l)*x[i]*y;
    x[i]=h%MAX;
    h/=MAX;
  }
  for(x.l=i; x.l>1 && !x[x.l-1]; x.l--);
  x.sign=(x.sign==sign?1:-1);
  if(x.sign==-1&&x.l==1&&x[0]==0) x.sign=1;
  return x;
}
BigInt operator /(BigInt x, int y)
{
  int i, sign=1;
  long long h;
  if(y<0) y=-y, sign=-1;
  for(h=0, i=x.l-1; i>=0; i--)
  {
    h=h*MAX+x[i];
    x[i]=h/y;
    h%=y;
  }
  for(; x.l>1 && !x[x.l-1]; x.l--);
  x.sign=(x.sign==sign?1:-1);
  if(x.sign==-1&&x.l==1&&x[0]==0) x.sign=1;
  return x;
}
int operator %(BigInt x, int y)
{
  int i;
  long long h;
  for(h=0, i=x.l-1; i>=0; i--)
  {
    h=h*MAX+x[i];
    h%=y;
  }
  if(x.sign==-1) h=-h;
  return h;
}
long long fl(double x) { return x<0?x-0.5:x+0.5; }
BigInt operator *(BigInt x, const BigInt &y)
{
  if(y.l==1) return x*(y[0]*y.sign);
  int i, N;
  long long t;
  vector<Complex> a, b;
  for(i=0; i<x.l; i++) a.emplace_back(x[i]);
  for(i=0; i<y.l; i++) b.emplace_back(y[i]);
  for(N=1; N<x.l+y.l; N<<=1);
  while(N!=(int)(a.size())) a.emplace_back(0);
  while(N!=(int)(b.size())) b.emplace_back(0);
  a=fft(a), b=fft(b);
  for(i=0; i<N; i++) a[i]=a[i]*b[i];
  a=fft(a,-1);
  for(i=0, t=0, x.l=0; i<N; i++)
  {
    t+=fl(a[i].x);
    x[x.l++]=t%MAX;
    t/=MAX;
  } x[x.l++]=t;
  for(; x.l>1 && !x[x.l-1]; x.l--);
  x.sign=(x.sign==y.sign?1:-1);
  if(x.sign==-1&&x.l==1&&x[0]==0) x.sign=1;
  return x;
}
BigInt operator /(BigInt x, const BigInt &y)
{
  if(y.l==1) return x/(y[0]*y.sign);
  int i;
  BigInt h;
  for(h=0, i=x.l-1; i>=0; i--)
  {
    h=h*MAX+x[i];
    if(h.l>y.l) x[i]=(h[h.l-1]*MAX*MAX+h[h.l-2]*MAX+h[h.l-3])
        ;
    if(h.l==y.l) x[i]=(h[h.l-1]*MAX+h[h.l-2]);
    x[i]/=(y[y.l-1]*MAX+y[y.l-2]);
    for(; x[i] && h<y*(x[i]*y.sign); x[i]--);
    h=h-(y*(x[i]*y.sign));
  }
  for(; x.l>1 && !x[x.l-1]; x.l--);
  x.sign=(x.sign==y.sign?1:-1);
  if(x.sign==-1&&x.l==1&&x[0]==0) x.sign=1;
  return x;
}
BigInt operator %(BigInt x, BigInt y)
{
```

```
  if(y.l==1) return x%(y[0]*y.sign);
  return x-(x/y)*y;
}
```

5.6 GaussianJordan

```
const double EPS = 1e-8;
typedef vector<double> vec;
typedef vector<vec> mat;

//solve Ax=b
//if no sol/inf sol, return vec of size 0
vec gauss_jordan(const mat& A, const vec& b)
{
    int n = A.size();
    mat B(n, vec(n+1));
    for(int i=0;i<n;i++)for(int j=0;j<n;j++)B[i][j]=A[i][j];

    for(int i=0;i<n;i++)B[i][n]=b[i];

    for(int i=0;i<n;i++)
    {
        int pivot=i;
        for(int j=i;j<n;j++)
        {
            if(abs(B[j][i])>abs(B[pivot][i]))pivot=j;
        }
        swap(B[i],B[pivot]);
        if(abs(B[i][i])<EPS)return vec();//no/inf sol

        for(int j=i+1;j<=n;j++)B[i][j]/=B[i][i];
        for(int j=0;j<n;j++)
        {
            if(i!=j)
            {
                for(int k=i+1;k<=n;k++)
                    B[j][k]-=B[j][i]*B[i][k];
            }
        }
    }
    vec x(n);
    for(int i=0;i<n;i++)x[i]=B[i][n];
    return x;
}
```

5.7 mobius

```
/* Mobius Function
 * m(x) = 0, x has repeated factors
 * m(x) = 1, x = 1
 * m(x) = (-1)^k, x is the product of k distinct primes
*/
const int M = 100005;
ll sp[M], mobius[M];
void sieve()
{
    for(ll i = 0; i < M; i++) sp[i] = i;
    for(ll i = 2; i*i < M; i++) if(sp[i] == i)
    {
        for(ll j = i*i; j < M; j += i)
            if(sp[j] == j) sp[j] = i;
    }
}
void makeMobius()
{
    for(ll i = 0; i < M; i++) mobius[i] = 1;
    mobius[0] = 0;
    for(ll i = 2; i < M; i++) if(sp[i] == i)
    {
        for(ll j = i; j < M; j += i) mobius[j] = -mobius[j];
        for(ll j = i*i; j < M; j += i*i) mobius[j] = 0;
    }
}
```

5.8 extgcd

```
int extgcd(int a, int b, int &x, int &y)
{
  int gcd = a;
  if(b != 0)
  {
    gcd = extgcd(b, a%b, y, x);
    y -= (a/b)*x;
  }
  else x = 1, y = 0;
  return gcd;
}
//維護a*x+b*y=gcd(a, b)
```

6 String

6.1 LCP

```
//build query in O(nlogn), query LCP(i,j) in O(1)
int dp_height[MAX_N][20];
void height_build(int *SA, int *Rank, char *S, int *Height)
{
  int len=strlen(S), k=0;
  for(int i=0;i<len;i++)
  {
    if(Rank[i]==0) continue;
    while(S[i+k] == S[SA[Rank[i]-1]+k]) k++;
    Height[Rank[i]]=k;
    if(k) k--;
  } Height[0]=0;
  for(int i=0;i<len;i++) dp_height[i][0]=Height[i];
  for(int i=0;i<len;i++) for(int j=1;i+(1<<j)<len;j++)
    dp_height[i][j]=min(dp_height[i][j-1], dp_height[i+(1<<(j
        -1))][j-1]);
}
int height_query(int x, int y)
{
  int k=0;
  while((1<<(k+1))<=y-x) k++;
  return min(dp_height[x+1][k], dp_height[y-(1<<k)+1][k]);
}
```

6.2 AC-Automation

```
#define SZ 25000
int nx[SZ][26], spt;
int fl[SZ], efl[SZ], ed[SZ];
int newnode()
{
  for(int i=0;i<26;i++) nx[spt][i]=0;
  ed[spt]=0;
  return spt++;
}
int add(char *s, int sptnow)
{
  for(int i=0;s[i];i++)
  {
    int tmp=s[i]-'a';
    if(nx[sptnow][tmp]==0) nx[sptnow][tmp]=newnode();
    sptnow=nx[sptnow][tmp];
  }
  ed[sptnow]=1;
  return sptnow;
}
int bfsq[SZ], qs, qe;
void make_fl(int root)
{
  fl[root]=efl[root]=qs=qe=0;
  bfsq[qe++]=root;
  while(qs!=qe)
  {
    int p=bfsq[qs++];
    for(int i=0;i<26;i++)
    {
      int t=nx[p][i];
      if(t==0) continue;
      int tmp=fl[p];
      for(;tmp&&nx[tmp][i]==0;tmp=fl[tmp]);
      fl[t]=tmp?nx[tmp][i]:root;
      efl[t]=ed[fl[t]]?fl[t]:efl[fl[t]];
      bfsq[qe++]=t;
    }
  }
}
```

6.3 KMP

```
void failure_build(const char *p, int *fail)
{
  for(int i=1, j=fail[0]=-1; p[i]; i++)
  {
    while(j>=0&&p[j+1]!=p[i]) j=fail[j];
    if(p[j+1]==p[i]) j++;
    fail[i]=j;
  }
}
int KMP(const char *T, const char *P, int *fail)
{
  failure_build(P, fail);
  for(int i=0, j=-1; T[i]; i++)
  {
    while(j>=0&&P[j+1]!=T[i]) j=fail[j];
    if(P[j+1]==T[i]) j++;
    if(!P[j+1]) return i-j;
```

```
  }
  return -1;
}

//使用方法: KMP(主字串, 待匹配字串, failure array)
//回傳: 第一個完全匹配的位置
```

6.4 BWT

```
// use with suffix array
int pivot;
// BWT array size must be double of the data size
inline void BWT(char *tmp, char *in, char *out, int *SA, int
     *Rank)
{
  int len=strlen(in);
  for(int i=0;i<len;i++) tmp[i]=tmp[i+len]=in[i];
  tmp[len*2]='\0';
  SA_build(SA,Rank,tmp);
  for(int i=0, j=0;i<2*len;i++)
  {
    if(SA[i]==len) pivot=j;
    if(SA[i]<len)
      out[j++]=in[(SA[i]+len-1)%len];
  }
  out[len]='\0';
}

inline void IBWT(char *in, char *out, int *tmp)
{
  int len=strlen(in);
  vector<int> idx[256];
  for(int i=0;i<len;i++)
    idx[in[i]].emplace_back(i);
  for(int i=0,k=0;i<256;i++)
    for(int j=0;j<(int)(idx[i].size());j++)
      tmp[k++]=idx[i][j];
  int p=pivot;
  for(int i=0;i<len;i++)
    out[i]=in[p=tmp[p]];
  out[len]='\0';
}
```

6.5 Z-value

```
void Z_build(const char *S, int *Z)
{
  Z[0]=0;
  int b=0;
  for(int i=1;S[i];i++)
  {
    if(Z[b]+b<i) Z[i]=0;
    else Z[i]=min(Z[b]+b-i,Z[i-b]);
    while(S[i+Z[i]]&&S[Z[i]]==S[i+Z[i]]) Z[i]++;
    if(Z[i]+i>Z[b]+b) b=i;
  }
}
```

6.6 SuffixArray

```
void SA_radix_sort(int *s, int *e, int *Rank, int rankcnt)
{
  int box[MAX_N], tmp[MAX_N], len=e-s;
  memset(box,0,sizeof(int)*rankcnt);
  for(int i=0;i<len;i++) box[Rank[i]]++;
  for(int i=1;i<rankcnt;i++) box[i]=box[i]+box[i-1];
  for(int i=len-1;i>=0;i--) tmp[--box[Rank[s[i]]]]=s[i];
  for(int i=0;i<len;i++) s[i]=tmp[i];
}
#define equal(a,b,c) c[a]!=c[b]||a+k>=len||c[a+k]!=c[b+k]
void SA_build(int *SA, int *Rank, char *S)
{
  int ranktmp[MAX_N], len=strlen(S), rankcnt='z'+1;
  for(int i=0;i<len;i++) Rank[i]=S[i];
  for(int k=1;rankcnt!=len;k*=2)
  {
    for(int i=0;i<len;i++) SA[i]=(i+len-k)%len;
    SA_radix_sort(SA+k, SA+len, Rank+k, rankcnt);
    SA_radix_sort(SA, SA+len, Rank, rankcnt);
    ranktmp[SA[0]]=0, rankcnt=0;
    for(int i=1;i<len;i++)
      ranktmp[SA[i]]=rankcnt+=equal(SA[i-1], SA[i], Rank);
    rankcnt++;
    for(int i=0;i<len;i++) Rank[i]=ranktmp[i];
  }
}
#undef equal
```

7 other

7.1 2sat

```
const int N = 10;    // 變數數量
bool adj[20][20];    // adjacency matrix
int visit[20];       // DFS visit record
int sat[20];         // 解

int not(int a) {return a<N ? a+N : a-N;}

// 另外一種方式
/*
int not(int a) {return a&1 ? a : a+1;}
int not(int a) {return a^1;}
*/

bool dfs_try(int i)
{
    if (visit[i] == 1 || sat[i] == 1) return true;
    if (visit[i] == 2 || sat[i] == 2) return false;
    visit[i] = 1;
    visit[not(i)] = 2;
    for (int j=0; j<N+N; ++j)
        if (adj[i][j] && !dfs_try(j))
            return false;
    return true;
}

void dfs_mark(int i)
{
    if (sat[i] == 1) return;
    sat[i] = 1;
    sat[not(i)] = 2;
    for (int j=0; j<N+N; ++j)
        if (adj[i][j])
            dfs_mark(j);
}

void two_satisfiability()
{
    // 一次輸入一個括號
    memset(adj, false, sizeof(adj));
    int a, b;
    while (cin >> a >> b)
    {
        map[not(a)][b] = true;
        map[not(b)][a] = true;
    }

    // 找出一組解
    for (int i=0; i<N; ++i)
    {
        memset(visit, 0, sizeof(visit));
        if (dfs_try(i)) {dfs_mark(i); continue;}

        memset(visit, 0, sizeof(visit));
        if (dfs_try(not(i))) {dfs_mark(not(i)); continue;}

        // 無解則立即結束。
        return;
    }

    // 印出一組解。
    for (int i=1; i<N; ++i)
        if (sat[i] == 1)
            cout << i;
        else /*if (sat[i] == 2)*/
            cout << "not" << i;
}
```

7.2 PojTree

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

using namespace std;
typedef long long int ll;
typedef pair<int, ll> P;
#define idx first
#define w second

const int N = 10004;
const ll INF = (1ll << 60);

int vn;
ll k;
vector<P> graph[N];
vector<int> dist;
ll subtreeSz[N];
bool isCentroid[N];

void init()
```

```
{
    for(int i = 1; i <= vn; i++)
        graph[i].clear(), isCentroid[i] = false;
}

void buildTree()
{
    for(int i = 1; i < vn; i++)
    {
        int u, v, l; scanf("%d %d %d", &u, &v, &l);
        graph[u].push_back(P(v, l));
        graph[v].push_back(P(u, l));
    }
}

ll calSubsz(int v, int p)
{
    subtreeSz[v] = 1;
    for(auto c:graph[v])
    {
        if(isCentroid[c.idx] || c.idx == p) continue;
        subtreeSz[v] += calSubsz(c.idx, v);
    }
    return subtreeSz[v];
}


P getCentroid(int v, int p, ll subsz)
{
    P cen(-1, INF);
    ll mxsonSz = -1;
    for(auto c:graph[v])
    {
        if(c.idx == p || isCentroid[c.idx]) continue;
        P res = getCentroid(c.idx, v, subsz);
        if(res.w < cen.w) cen = res;
        mxsonSz = max(mxsonSz, subtreeSz[c.idx]);
    }
    mxsonSz = max(mxsonSz, subsz-subtreeSz[v]);
    if(mxsonSz < cen.w) cen = P(v, mxsonSz);
    return cen;
}

void getDist(int v, int p, ll w)
{
    if(w > k) return;
    dist.push_back(w);
    for(auto c:graph[v])
    {
        if(c.idx == p || isCentroid[c.idx]) continue;
        getDist(c.idx, v, w+c.w);
    }
}

ll calValidPair(int idx, ll w)
{
    dist.clear();
    getDist(idx, -1, w);
    sort(dist.begin(), dist.end());
    ll sum = 0;
    for(int l = 0, r = dist.size()-1; l < r; )
    {
        if(dist[r]+dist[l] <= k) sum += r-l, l++;
        else r--;
    }
    return sum;
}

ll treedc(int v)
{
    ll sum = 0;
    // find centroid
    calSubsz(v, v);
    int cen = getCentroid(v, v, subtreeSz[v]).idx;
    isCentroid[cen] = true;

    sum += calValidPair(cen, 0);
    for(auto c:graph[cen])
    {
        if(isCentroid[c.idx]) continue;
        sum -= calValidPair(c.idx, c.w);
        sum += treedc(c.idx);
    }
    return sum;
}

int main()
{
    while(scanf("%d %lld", &vn, &k) && vn && k)
    {
        init();
        buildTree();
        printf("%lld\n", treedc(1));
    }
    return 0;
}
```

7.3 definesss

```
#include <bits/stdc++.h>
using namespace std;
#define pb push_back
#define pii pair<int,int>
#define pll pair<ll, ll>
#define pil pair<int,ll>
#define pli pair<ll,int>
#define ppi pair<pii,int>
#define pip pair<int,pii>
#define pdd pair<double, double>
#define f first
#define s second
#define MOD 1000000007
#define mkp make_pair
#define M_PI 3.14159265358979323846
#define FOR(i,l,r) for (int i=l;i<=r;i++)
#define LOR(i,l,r) for (ll i=l;i<=r;i++)
#define FORD(i,r,l) for (int i=r;i>=l;i--)
#define LORD(i,r,l) for (ll i=r;i>=l;i--)
#define INF 1000000000
#define CL(x) memset(x,0,sizeof(x))
typedef long long ll;

int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0);

    return 0;
}
```

ACM ICPC Team Reference - NTHU_FurryForce

Contents