Standard Code Library

Shanghai Jiao Tong University
Caladbolg

October 18, 2017

Contents

Chapter 1

沐阳

1.1 AC 自动机

```
//ac 自动机
inline int newnode()
{
  memset(nxt[L],-1,sizeof(nxt[L]));
  return ++L-1;
}
inline void init() { L = 0; root = newnode(); }
inline void insert()
{
  int len = strlen(buf),now = root;
  for (int i = 0;i < len;++i)
  {
    if (nxt[now][buf[i]-'0'] == -1)
      nxt[now][buf[i]-'0'] = newnode();
    now = nxt[now][buf[i]-'0'];
  }
  end[now] = true;
}
inline void build()
{
  int now = root; queue <int> team;
  fail[root] = root;
  for (int i = 0;i < 10;++i)
  {
    if (nxt[now][i] == -1) nxt[now][i] = root;
    else fail[nxt[now][i]] = root,team.push(nxt[now][i]);
  }
  while (!team.empty())
  {
    now = team.front(); team.pop();
    for (int i = 0;i < 10;++i)
    {
      if (nxt[now][i] == -1)
        nxt[now][i] = nxt[fail[now]][i];
      else
      {
        fail[nxt[now][i]] = nxt[fail[now]][i];
        team.push(nxt[now][i]);
      }
    }
  }
}
```

1.2 Dinic Algorithm

```
// dinic
int source,sink,cnt = 1;
int d[maxv],side[maxv],cur[maxv],side[maxe],nxt[maxe],toit[maxe],cap[maxe]; bool in[maxv];

```

```
inline void add(int a,int b,int c) { nxt[++cnt] = side[a]; side[a] = cnt; toit[cnt] = b; cap[cnt] = c; }
inline void ins(int a,int b,int c) { add(a,b,c); add(b,a,0); }

inline bool bfs()
{
  queue <int> team; team.push(source); d[source] = 0;
  memset(in,false,tot+10); in[source] = true; team.push(source);
  while (!team.empty())
  {
    int now = team.front(); team.pop(); cur[now] = side[now];
    for (int i = side[now];i;i = nxt[i])
    {
      if (!cap[i]) continue;
      if (!in[toit[i]])
        in[toit[i]] = true,d[toit[i]] = d[now]+1,team.push(toit[i]);
    }
  }
  return in[sink];
}

inline int dfs(int now,int f)
{
  if (now == sink||!f) return f;
  int used = 0,w;
  for (int &i = cur[now];i;i = nxt[i])
    if (cap[i]&&d[toit[i]] == d[now]+1)
    {
      w = dfs(toit[i],min(cap[i],f-used));
      used += w; cap[i] -= w; cap[i^1] += w;
      if (used == f) break;
    }
  return used;
}

inline int dinic(int S,int T)
{
  source = S; sink = T; int ret = 0;
  while (bfs()) ret += dfs(source,inf);
  return ret;
}
```

1.3 K-Dimension Tree

```
struct Point
{
  double x,y; int id;
  inline Point() = default;
  inline Point(double _x,double _y,int _id):x(_x),y(_y),id(_id) {}
  inline void read(int i = 0) { scanf("%lf %lf",&x,&y); id = i; }
  inline double norm() { return sqrt(x*x+y*y); }
  friend inline Point operator+(const Point &a,const Point &b) { return Point(a.x+b.x,a.y+b.y); }
  friend inline Point operator-(const Point &a,const Point &b) { return Point(a.x-b.x,a.y-b.y); }
  friend inline double operator*(const Point &a,const Point &b) { return a.x*b.x+a.y*b.y; }
  friend inline double operator/(const Point &a,const Point &b) { return a.x*b.y-a.y*b.x; }
}P[maxn];

struct Rectangle
{
  double lx,rx,ly,ry;
  inline Rectangle() = default;
  inline Rectangle(double _lx,double _rx,double _ly,double _ry):lx(_lx),rx(_rx),ly(_ly),ry(_ry) {}
  inline void set(const Point &p) { lx = rx = p.x; ly = ry = p.y; }
  inline void merge(const Point &p)
  {
    lx = min(lx,p.x); rx = max(rx,p.x);
    ly = min(ly,p.y); ry = max(ry,p.y);
  }
```

```
25    inline void merge(const Rectangle &r)
26    {
27      lx = min(lx,r.lx); rx = max(rx,r.rx);
28      ly = min(ly,r.ly); ry = max(ry,r.ry);
29    }
30    // 最小距离，到 4 个角和 4 条边距离
31    inline double dist(const Point &p)
32    {
33      if (p.x <= lx&&p.y <= ly) return (p-Point(lx,ly)).norm();
34      else if (p.x <= rx&&p.y <= ly) return p.y-ly;
35      else if (p.x >= rx&&p.y <= ly) return (p-Point(rx,ly)).norm();
36      else if (p.x >= rx&&p.y <= ry) return p.x-rx;
37      else if (p.x >= rx&&p.y >= ry) return (p-Point(rx,ry)).norm();
38      else if (p.x >= lx&&p.y >= ry) return p.y-ry;
39      else if (p.x <= lx&&p.y >= ry) return (p-Point(lx,ry)).norm();
40      else if (p.x <= lx&&p.y >= ly) return p.x-lx;
41      return 0;
42    }
43    // 最大距离，到 4 个角的距离
44    inline double dist(const Point &p)
45    {
46      double ret = 0;
47      ret += max((rx-p.x)*(rx-p.x),(lx-p.x)*(lx-p.x));
48      ret += max((ry-p.y)*(ry-p.y),(ly-p.y)*(ly-p.y));
49      return ret;
50    }
51  };
52
53  struct Node
54  {
55    int child[2]; Point p; Rectangle r;
56    inline Node() = default;
57    inline Node(const Point &_p,const Rectangle &_r):p(_p),r(_r) { r.set(p); memset(child,0,8); }
58    inline void set(const Point &_p) { p = _p; r.set(p); memset(child,0,8); }
59  }tree[maxn];
60
61  inline bool cmpx(const Point &a,const Point &b)
62  {
63    if (a.x != b.x) return a.x < b.x;
64    else return a.y < b.y;
65  }
66  inline bool cmpy(const Point &a,const Point &b)
67  {
68    if (a.y != b.y) return a.y < b.y;
69    else return a.x < b.x;
70  }
71
72  inline bool cmp(pair <double,int> a,pair <double,int> b)
73  {
74    int sgn = dcmp(a.first-b.first);
75    if (sgn) return sgn < 0;
76    else return a.second < b.second;
77  }
78
79  // 查询 k 大/小
80  inline void query(int now,const Point &p,int k,pair <double,int> ret[],bool dim = false)
81  {
82    if (dcmp(tree[now].r.dist(p)-ret[k].first) > 0) return;
83    pair <double,int> val = make_pair((p-tree[now].p).norm(),tree[now].p.id);
84    for (int i = 1;i <= k;++i)
85      if (cmp(val,ret[i]))
86      {
87        for (int j = k+1;j > i;--j) ret[j] = ret[j-1];
88        ret[i] = val; break;
89      }
90    if ((dim&&cmpx(p,tree[now].p))||(!dim&&cmpy(p,tree[now].p)))
91    {
92      if (tree[now].child[0]) query(tree[now].child[0],p,k,ret,dim^1);
93      if (tree[now].child[1]) query(tree[now].child[1],p,k,ret,dim^1);
94    }
```

```
  else
  {
    if (tree[now].child[1]) query(tree[now].child[1],p,k,ret,dim^1);
    if (tree[now].child[0]) query(tree[now].child[0],p,k,ret,dim^1);
  }
}

// 查询最小/大
inline void query(int x,const Point &p,pair <double,int> ret,bool dim = false)
{
  if (dcmp(tree[now].r.disp(p)-ret.first) > 0) return;
  pair <double,int> val = make_pair((p-tree[now].p).norm(),tree[now].p.id);
  if (cmp(val,ret)) ret = val;
  if ((dim&&cmpx(p,tree[now].p))||(!dim&&cmpy(p,tree[now].p)))
  {
    if (tree[now].child[0]) query(tree[now].child[0],p,ret,dim^1);
    if (tree[now].child[1]) query(tree[now].child[1],p,ret,dim^1);
  }
  else
  {
    if (tree[now].child[1]) query(tree[now].child[1],p,ret,dim^1);
    if (tree[now].child[0]) query(tree[now].child[0],p,ret,dim^1);
  }
}

inline int build(int l,int r,bool dim)
{
  int now = ++size,mid = (l+r)>>1;
  nth_element(vec.begin()+l-1,vec.begin()+mid-1,vec.begin()+r,dim?cmpx:cmpy);
  tree[now].set(vec[mid-1]);
  if (l < mid)
  {
    tree[now].child[0] = build(l,mid-1,dim^1);
    tree[now].r.merge(tree[tree[now].child[0]].r);
  }
  if (r > mid)
  {
    tree[now].child[1] = build(mid+1,r,dim^1);
    tree[now].r.merge(tree[tree[now].child[1]].r);
  }
  return now;
}
```

1.4 KM

```
// Truly O(n^3), 最大权匹配
// 邻接矩阵, 不能连的边设为-INF, 求最小权匹配时边权取负, 但不能连的还是 -INF, 使用时先对 1 -> n 调用 hungary() , 再
struct KM
{
  int w[maxn][maxn],lx[maxn],ly[maxn],match[maxn],way[maxn],slack[maxn];
  bool used[maxn];

  inline void init()
  {
    for (int i = 1;i <= N;++i)
      match[i] = lx[i] = ly[i] = way[i] = 0;
  }

  inline void hungary(int x)
  {
    match[0] = x; int j0 = 0;
    for (int j = 0;j <= N;++j)
      slack[j] = inf,used[j] = false;
    do
    {
      used[j0] = true;
      int i0 = match[j0],delta = inf,j1 = 0;
      for (int j = 1;j <= N;++j)
        if (!used[j])
```

```
25          {
26            int cur = -w[i0][j]-lx[i0]-ly[j];
27            if (cur < slack[j])
28              slack[j] = cur,way[j] = j0;
29            if (slack[j] < delta)
30              delta = slack[j],j1 = j;
31          }
32        for (int j = 0;j <= N;++j)
33        {
34          if (used[j]) lx[match[j]] += delta,ly[j] -= delta;
35          else slack[j] -= delta;
36        }
37        j0 = j1;
38      }
39      while (match[j0]);
40      do
41      {
42        int j1 = way[j0];
43        match[j0] = match[j1];
44        j0 = j1;
45      }
46      while (j0);
47    }
48
49    inline void work() { for (int i = 1;i <= N;++i) hungary(i); }
50
51    inline int get_ans()
52    {
53      int sum = 0;
54      for (int i = 1;i <= N;++i)
55      {
56        // if (w[match[i]][i] == -inf) ; //无解
57        if (match[i] > 0) sum += w[match[i]][i];
58      }
59      return sum;
60    }
61  }km;
62  //最小权匹配
63  struct KM
64  {
65    int w[maxn][maxn],lx[maxn],ly[maxn],match[maxn],way[maxn],slack[maxn]; bool used[maxn];
66
67    inline void init()
68    {
69      for (int i = 1;i <= N;++i)
70        match[i] = lx[i] = ly[i] = way[i] = 0;
71    }
72
73    inline void hungary(int x)
74    {
75      match[0] = x; int j0 = 0;
76      for (int j = 0;j <= N;++j)
77        slack[j] = -inf,used[j] = false;
78      do
79      {
80        used[j0] = true;
81        int i0 = match[j0],delta = -inf,j1 = 0;
82        for (int j = 1;j <= N;++j)
83          if (!used[j])
84          {
85            int cur = -w[i0][j]-lx[i0]-ly[j];
86            if (cur > slack[j]) slack[j] = cur,way[j] = j0;
87            if (slack[j] > delta) delta = slack[j],j1 = j;
88          }
89        for (int j = 0;j <= N;++j)
90        {
91          if (used[j]) lx[match[j]] += delta,ly[j] -= delta;
92          else slack[j] -= delta;
93        }
94        j0 = j1;
```

```
95      }
96      while (match[j0]);
97      do
98      {
99        int j1 = way[j0];
100       match[j0] = match[j1];
101       j0 = j1;
102     }
103     while (j0);
104   }
105
106   inline void work() { for (int i = 1;i <= N;++i) hungary(i); }
107
108   inline int get_ans()
109   {
110     int sum = 0;
111     for (int i = 1;i <= N;++i)
112     {
113       // if (w[match[i]][i] == inf) ; // 无解
114       if (match[i] > 0) sum += w[match[i]][i];
115     }
116     return sum;
117   }
118 }km;
```

1.5 Link Cut Tree

```
1  inline bool isroot(int a) { return ch[fa[a]][0] != a&&ch[fa[a]][1] != a; }
2
3  inline void update(int x) { val[x] = (val[ch[x][0]]+val[ch[x][1]]).merge(x); }
4  inline void pushdown(int x)
5  {
6    if (rev[x])
7    {
8      int &lc = ch[x][0],&rc = ch[x][1];
9      swap(lc,rc);
10     if (lc) rev[lc] ^= 1;
11     if (rc) rev[rc] ^= 1;
12     rev[x] = false;
13   }
14 }
15
16 inline void rotate(int x)
17 {
18   int y = fa[x],z = fa[y],l = ch[y][1] == x,r = l^1;
19   if (!isroot(y)) ch[z][ch[z][1] == y] = x; fa[x] = z;
20   if (ch[x][r]) fa[ch[x][r]] = y; ch[y][l] = ch[x][r];
21   fa[y] = x; ch[x][r] = y; update(y); update(x);
22 }
23 inline void splay(int x)
24 {
25   int top = 0,i;
26   for (i = x;!isroot(i);i = fa[i]) stk[++top] = i; stk[++top] = i;
27   while (top) pushdown(stk[top--]);
28   while (!isroot(x))
29   {
30     int y = fa[x],z = fa[y];
31     if (!isroot(y))
32     {
33       if ((ch[y][0] == x)^(ch[z][0] == y)) rotate(x);
34       else rotate(y);
35     }
36     rotate(x);
37   }
38 }
39
40 inline int access(int x)
41 {
42   int t = 0;
```

```
  for (t = 0;x;t = x,x = fa[x])
    splay(x),ch[x][1] = t,update(x);
  return t;
}
inline int evert(int x) { int t; rev[t = access(x)] ^= 1; return t; }
inline int find(int x)
{
  x = access(x);
  while (pushdown(x),ch[x][0]) x = ch[x][0];
  return x;
}
inline void cut(int x,int y)
{
  evert(x); access(y); splay(y);
  if (ch[y][0] != x||ch[x][1] != 0) return;
  ch[y][0] = fa[x] = 0; update(x); update(y);
}
inline void link(int x,int y) { fa[evert(x)] = y; }


// Magic Forest
#include<algorithm>
#include<cstring>
#include<iostream>
#include<cstdio>
#include<cstdlib>
using namespace std;

const int maxn = 200010,inf = 1<<29;
int N,M,A[maxn],B[maxn],fa[maxn],ch[maxn][2];
int stk[maxn],ans = inf; bool rev[maxn];

struct Value
{
  int ma,mb,id;
  inline Value(int _ma = 0,int _mb = 0,int _id = 0):ma(_ma),mb(_mb),id(_id) {}
  friend inline Value operator +(const Value &a,const Value &b)
  {
    Value ret = Value(max(a.ma,b.ma),max(a.mb,b.mb),a.id);
    if (B[a.id] < B[b.id]) ret.id = b.id;
    return ret;
  }
  inline Value merge(int i)
  {
    Value ret = Value(max(ma,A[i]),max(mb,B[i]),id);
    if (B[i] > B[id]) ret.id = i;
    return ret;
  }
}val[maxn];

inline int gi()
{
  char ch; int ret = 0,f = 1;
  do ch = getchar(); while (!(ch >= '0'&&ch <= '9')&&ch != '-');
  if (ch == '-') f = -1,ch = getchar();
  do ret = ret*10+ch-'0',ch = getchar(); while (ch >= '0'&&ch <= '9');
  return ret*f;
}

struct Edge
{
  int x,y,a,b;
  inline Edge(int _x = 0,int _y = 0,int _a = 0,int _b = 0):x(_x),y(_y),a(_a),b(_b) {}
  inline void read() { x = gi(),y = gi(),a = gi(),b = gi(); }
  friend inline bool operator <(const Edge &s,const Edge &t) { return s.a < t.a; }
}edge[maxn];

inline bool isroot(int a) { return ch[fa[a]][0] != a&&ch[fa[a]][1] != a; }

inline void update(int x) { val[x] = (val[ch[x][0]]+val[ch[x][1]]).merge(x); }
```

```
113 inline void pushdown(int x)
114 {
115   if (rev[x])
116   {
117     int &lc = ch[x][0],&rc = ch[x][1];
118     swap(lc,rc);
119     if (lc) rev[lc] ^= 1;
120     if (rc) rev[rc] ^= 1;
121     rev[x] = false;
122   }
123 }
124
125 inline void rotate(int x)
126 {
127   int y = fa[x],z = fa[y],l = ch[y][1] == x,r = l^1;
128   if (!isroot(y)) ch[z][ch[z][1] == y] = x; fa[x] = z;
129   if (ch[x][r]) fa[ch[x][r]] = y; ch[y][l] = ch[x][r];
130   fa[y] = x; ch[x][r] = y; update(y); update(x);
131 }
132 inline void splay(int x)
133 {
134   int top = 0,i;
135   for (i = x;!isroot(i);i = fa[i]) stk[++top] = i; stk[++top] = i;
136   while (top) pushdown(stk[top--]);
137   while (!isroot(x))
138   {
139     int y = fa[x],z = fa[y];
140     if (!isroot(y))
141     {
142       if ((ch[y][0] == x)^(ch[z][0] == y)) rotate(x);
143       else rotate(y);
144     }
145     rotate(x);
146   }
147 }
148
149 inline int access(int x)
150 {
151   int t = 0;
152   for (t = 0;x;t = x,x = fa[x])
153     splay(x),ch[x][1] = t,update(x);
154   return t;
155 }
156 inline int evert(int x) { int t; rev[t = access(x)] ^= 1; return t; }
157 inline int find(int x)
158 {
159   x = access(x);
160   while (pushdown(x),ch[x][0]) x = ch[x][0];
161   return x;
162 }
163 inline void cut(int x,int y)
164 {
165   evert(x); access(y); splay(y);
166   if (ch[y][0] != x||ch[x][1] != 0) return;
167   ch[y][0] = fa[x] = 0; update(x); update(y);
168 }
169 inline void link(int x,int y) { fa[evert(x)] = y; }
170
171 inline Value query(int x,int y) { evert(x); return val[access(y)]; }
172
173 int main()
174 {
175   // freopen("D.in","r",stdin);
176   N = gi(),M = gi();
177   for (int i = 1;i <= M;++i) edge[i].read();
178   sort(edge+1,edge+M+1);
179   for (int i = 0;i <= N;++i)
180     A[i] = B[i] = -inf,val[i] = Value(A[i],B[i],i);
181   for (int i = 1;i <= M;++i)
182     A[i+N] = edge[i].a,B[i+N] = edge[i].b,val[i+N] = Value(A[i+N],B[i+N],i+N);
```

```
  for (int i = 1;i <= M;++i)
  {
    if (edge[i].x == edge[i].y) continue;
    if (find(edge[i].x) == find(edge[i].y))
    {
      Value res = query(edge[i].x,edge[i].y); int id = res.id-N;
      if (edge[i].b < edge[id].b)
      {
        cut(edge[id].x,id+N),cut(edge[id].y,id+N);
        link(edge[i].x,i+N),link(edge[i].y,i+N);
      }
    }
    else link(edge[i].x,i+N),link(i+N,edge[i].y);
    if (find(1) == find(N))
    {
      Value res = query(1,N);
      ans = min(ans,res.ma+res.mb);
    }
  }
  if (ans == inf) ans = -1;
  printf("%d\n",ans);
  return 0;
}
```

1.6 Numerical Integration

```
//self-adapt simpson
inline long double simpson(long double l,long double r,long double mid,long double Cl,long double
↪  Cr,long double Cm)
{
    long double tCl = calc((l+mid)/2),tCr = calc((mid+r)/2);
  long double
↪  ans=(r-l)*(Cl+Cr+4*Cm)/6,lans=(mid-l)*(Cl+Cm+4*tCl)/6,rans=(r-mid)*(Cr+Cm+4*tCr)/6;
    if (r-l <= 1e-3&&fabs(lans+rans-ans)<eps) return ans;
  // if (dep > lim&&fabs(lans+rans-ans)<eps) return ans;
    else return simpson(l,mid,(l+mid)/2,Cl,Cm,tCl)+simpson(mid,r,(mid+r)/2,Cm,Cr,tCr);
}
```

1.7 Splay

```
//splay

inline int find(int rk)
{
  for (int now = root;;)
  {
    if (rk == size[ch[now][0]]+1) return now;
    else if (rk > size[ch[now][0]]+1)
      rk -= size[ch[now][1]]+1,now = ch[now][1];
    else now = ch[now][0];
  }
  return 0;
}

inline int upperbound(int x)
{
  int ret = 0;
  for (int now = root;now;)
  {
    if (key[now] > x) ret = now,now = ch[now][0];
    else now = ch[now][1];
  }
  return ret;
}
inline int lowerbound(int x)
{
  int ret = 0;
  for (int now = root;now;)
```

```
  {
    if (key[now] >= x) ret = now,now = ch[now][0];
    else now = ch[now][1];
  }
  return ret;
}

inline void rotate(int x)
{
  int y = fa[x],z = fa[y],l = ch[y][0] != x,r = l^1;
  if (z) ch[z][ch[z][0] != y] = x; fa[x] = z;
  if (ch[x][r]) fa[ch[x][r]] = y;
  ch[y][l] = ch[x][r]; fa[y] = x; ch[x][r] = y;
  update(y); update(x);
}
inline void splay(int x,int aim) //aim is x's father.
{
  int top = 0;
  for (int i = x;i;i = fa[i]) stack[++top] = i;
  while (top) pushdown(stack[top--]);
  while (fa[x] != aim)
  {
    int y = fa[x],z = fa[y];
    if (z != aim)
    {
      if ((ch[y][0] == x)^(ch[z][0] == y)) rotate(x);
      else rotate(y);
    }
    rotate(x);
  }
  if (!aim) root = x;
}

// 维修数列
#include<cassert>
#include<queue>
#include<algorithm>
#include<cstring>
#include<iostream>
#include<cstdio>
#include<cstdlib>
using namespace std;

const int maxn = 500010,inf = 1<<29;
int N,M,root,cnt,arr[maxn],tag[maxn],key[maxn],fa[maxn],ch[maxn][2],lb[maxn],rb[maxn];
int wb[maxn],sum[maxn],size[maxn],stk[maxn]; bool rev[maxn]; char cmd[20]; queue <int> team;

inline int gi()
{
  char ch; int ret = 0,f = 1;
  do ch = getchar(); while (!(ch >= '0'&&ch <= '9')&&ch != '-');
  if (ch == '-') f = -1,ch = getchar();
  do ret = ret*10+ch-'0',ch = getchar(); while (ch >= '0'&&ch <= '9');
  return ret*f;
}

inline int newnode(int x = 0)
{
  int ret;
  if (!team.empty())
    ret = team.front(),team.pop();
  else ret = ++cnt;
  key[ret] = sum[ret] = lb[ret] = rb[ret] = wb[ret] = x;
  rev[ret] = false; tag[ret] = inf; size[ret] = 1;
  return ret;
}

inline void pushdown(int now)
{
  int lc = ch[now][0],rc = ch[now][1];
```

```
99    if (rev[now])
100   {
101     if (lc)
102     {
103       swap(ch[lc][0],ch[lc][1]);
104       swap(lb[lc],rb[lc]); rev[lc] ^= 1;
105     }
106     if (rc)
107     {
108       swap(ch[rc][0],ch[rc][1]);
109       swap(lb[rc],rb[rc]); rev[rc] ^= 1;
110     }
111     rev[now] = false;
112   }
113   if (tag[now] != inf)
114   {
115     if (lc)
116     {
117       key[lc] = tag[lc] = tag[now]; sum[lc] = tag[lc]*size[lc];
118       if (tag[lc] > 0) lb[lc] = rb[lc] = wb[lc] = sum[lc];
119       else lb[lc] = rb[lc] = wb[lc] = tag[lc];
120     }
121     if (rc)
122     {
123       key[rc] = tag[rc] = tag[now]; sum[rc] = tag[rc]*size[rc];
124       if (tag[rc] > 0) lb[rc] = rb[rc] = wb[rc] = sum[rc];
125       else lb[rc] = rb[rc] = wb[rc] = tag[rc];
126     }
127     tag[now] = inf;
128   }
129 }
130
131 inline void update(int now)
132 {
133   // pushdown(now);
134   int lc = ch[now][0],rc = ch[now][1];
135   size[now] = size[lc]+size[rc]+1;
136   sum[now] = sum[lc]+sum[rc]+key[now];
137   if (lc&&rc)
138   {
139     lb[now] = max(lb[lc],max(sum[lc]+key[now],sum[lc]+key[now]+lb[rc]));
140     rb[now] = max(rb[rc],max(sum[rc]+key[now],sum[rc]+key[now]+rb[lc]));
141     wb[now] = max(wb[lc],wb[rc]); wb[now] = max(wb[now],key[now]);
142     wb[now] = max(wb[now],rb[lc]+key[now]); wb[now] = max(wb[now],lb[rc]+key[now]);
143     wb[now] = max(wb[now],rb[lc]+key[now]+lb[rc]);
144   }
145   else if (lc)
146   {
147     lb[now] = max(lb[lc],sum[lc]+key[now]);
148     rb[now] = max(key[now],key[now]+rb[lc]);
149     wb[now] = max(wb[lc],key[now]);
150     wb[now] = max(wb[now],rb[lc]+key[now]);
151   }
152   else if (rc)
153   {
154     rb[now] = max(rb[rc],sum[rc]+key[now]);
155     lb[now] = max(key[now],key[now]+lb[rc]);
156     wb[now] = max(wb[rc],key[now]);
157     wb[now] = max(wb[now],lb[rc]+key[now]);
158   }
159   else sum[now] = lb[now] = rb[now] = wb[now] = key[now];
160 }
161
162 inline int build(int l,int r)
163 {
164   int mid = (l+r) >> 1,ret = newnode(arr[mid]);
165   if (l < mid) ch[ret][0] = build(l,mid-1),fa[ch[ret][0]] = ret;
166   if (r > mid) ch[ret][1] = build(mid+1,r),fa[ch[ret][1]] = ret;
167   update(ret); return ret;
168 }
```

```
169
170 inline void init()
171 {
172   root = newnode(); ch[root][1] = newnode(); fa[2] = 1;
173   for (int i = 1;i <= N;++i) arr[i] = gi();
174   ch[2][0] = build(1,N); fa[ch[2][0]] = 2;
175   update(2); update(1);
176 }
177
178 inline int find(int rk)
179 {
180   for (int now = root;;)
181   {
182     pushdown(now);
183     if (rk == size[ch[now][0]]+1) return now;
184     else if (rk > size[ch[now][0]]+1)
185       rk -= size[ch[now][0]]+1,now = ch[now][1];
186     else now = ch[now][0];
187   }
188   return 0;
189 }
190
191 inline void rotate(int x)
192 {
193   int y = fa[x],z = fa[y],l = ch[y][0] != x,r = l^1;
194   if (z) ch[z][ch[z][0] != y] = x;
195   fa[x] = z; fa[y] = x; fa[ch[x][r]] = y;
196   ch[y][l] = ch[x][r]; ch[x][r] = y;
197   update(y); update(x);
198 }
199 inline void splay(int x,int aim)
200 {
201   int top = 0;
202   for (int i = x;i;i = fa[i]) stk[++top] = i;
203   while (top) pushdown(stk[top--]);
204   while (fa[x] != aim)
205   {
206     int y = fa[x],z = fa[y];
207     if (z != aim)
208     {
209       if ((ch[y][0] == x)^(ch[z][0] == y)) rotate(x);
210       else rotate(y);
211     }
212     rotate(x);
213   }
214   if (!aim) root = x;
215 }
216
217 inline void Delete(int &now)
218 {
219   if (!now) return;
220   Delete(ch[now][0]);
221   Delete(ch[now][1]);
222   team.push(now); now = 0;
223 }
224
225 inline void print()
226 {
227   for (int i = 1;i <= cnt;++i)
228     printf("%d:%d %d\n",i,ch[i][0],ch[i][1]);
229   for (int i = 1;i <= cnt;++i)
230     printf("%d:%d\n",i,fa[i]);
231
232 }
233
234 inline void laydown(int now)
235 {
236   if (!now) return;
237   pushdown(now);
238   laydown(ch[now][0]);
```

```
239   printf("%d ",key[now]);
240   laydown(ch[now][1]);
241   update(now);
242 }
243
244 int main()
245 {
246   //freopen("C.in","r",stdin);
247   N = gi(); M = gi(); init();
248   while (M--)
249   {
250     scanf("%s",cmd);
251     if (cmd[0] == 'I')
252     {
253       int pos = gi(),a = find(pos+1),b = find(pos+2); N = gi();
254       for (int i = 1;i <= N;++i) arr[i] = gi();
255       splay(a,0); splay(b,a);
256       ch[b][0] = build(1,N); fa[ch[b][0]] = b;
257       update(b); update(a);
258     }
259     else if (cmd[0] == 'D')
260     {
261       int pos = gi(); N = gi();
262       int a = find(pos),b = find(pos+N+1);
263       splay(a,0); splay(b,a);
264       Delete(ch[b][0]); update(b); update(a);
265     }
266     else if (cmd[0] == 'M'&&cmd[2] == 'K')
267     {
268       int pos = gi(); N = gi();
269       int a = find(pos),b = find(pos+N+1);
270       splay(a,0); splay(b,a);
271       key[ch[b][0]] = tag[ch[b][0]] = gi(); sum[ch[b][0]] = tag[ch[b][0]]*size[ch[b][0]];
272       if (tag[ch[b][0]] > 0) lb[ch[b][0]] = rb[ch[b][0]] = wb[ch[b][0]] = sum[ch[b][0]];
273       else lb[ch[b][0]] = rb[ch[b][0]] = wb[ch[b][0]] = tag[ch[b][0]];
274       update(b); update(a);
275     }
276     else if (cmd[0] == 'R')
277     {
278       int pos = gi(); N = gi();
279       int a = find(pos),b = find(pos+N+1);
280       splay(a,0); splay(b,a);
281       rev[ch[b][0]] ^= 1;
282       swap(ch[ch[b][0]][0],ch[ch[b][0]][1]);
283       swap(lb[ch[b][0]],rb[ch[b][0]]);
284       update(b); update(a);
285     }
286     else if (cmd[0] == 'G')
287     {
288       int pos = gi(); N = gi();
289       int a = find(pos),b = find(pos+N+1);
290       splay(a,0); splay(b,a);
291       printf("%d\n",sum[ch[b][0]]);
292     }
293     else
294     {
295       splay(1,0); splay(2,1);
296       printf("%d\n",wb[ch[2][0]]);
297     }
298   }
299   return 0;
300 }
```

1.8 几何基础

```
1 //计算几何常用公式
2 inline int dcmp(double a)
3 {
4   if (fabs(a) <= eps) return 0;
```

```
  else if (a > 0) return 1;
  else return -1;
}
struct Point
{
    double x,y;
  inline Point() = default;
  inline Point(double _x,double _y):x(_x),y(_y) {}
  inline Point unit() const
  {
    double len = norm();
    if (!dcmp(len)) return Point(1,0);
    else return *this/len;
  }
  inline double norm() const { return sqrt(x*x+y*y); }
  inline Point reflect(const Point &p) const
  {
    Point v = *this-p; double len = v.norm();
    v = v/len; return p+v*(1/len);
  }
  inline void read() { scanf("%lf %lf",&x,&y); }
  inline Point vertical() const { return Point(y,-x); }
  inline double angle() const
  {
    double ret = atan2(y,x);
    if (ret < 0) ret += 2*pi;
    return ret;
  }
  friend inline bool operator ==(const Point &a,const Point &b) { return !dcmp(a.x-b.x)&&!dcmp(a.y-b.y); }
    friend inline Point operator -(const Point &a,const Point &b) { return Point(a.x-b.x,a.y-b.y); }
  friend inline Point operator +(const Point &a,const Point &b) { return Point(a.x+b.x,a.y+b.y); }
  friend inline Point operator /(const Point &a,double b) { return Point(a.x/b,a.y/b); }
  friend inline Point operator *(const Point &a,double b) { return Point(a.x*b,a.y*b); }
  friend inline Point operator *(double b,const Point &a) { return Point(a.x*b,a.y*b); }
  friend inline double operator /(const Point &a,const Point &b) { return a.x*b.y-a.y*b.x; }
};
struct Line
{
    Point p,v; double slop;
  inline Line() = default;
  inline Line(const Point &_p,const Point &_v):p(_p),v(_v) {}
    inline void update() { slop = v.alpha(); }
    friend inline bool operator <(const Line &l1,const Line &l2)
  { return l1.slop < l2.slop; }
  inline double dis(const Point &a) { fabs((a-p)/v)/(v.len()); } //点到直线距离
};

inline bool OnLine(const Line &l,const Point &p) { return !dcmp(l.v/(p-l.p)); } //点在直线上

inline Point CrossPoint(const Line &a,const Line &b)  //直线交点
{
    Point u = a.p - b.p;
    double t = (b.v/u)/(a.v/b.v);
    return a.p+a.v*t;
}

inline bool parallel(const Line &a,const Line &b) { return !dcmp(a.v/b.v); } //直线平行
```

1.9 凸包

```
struct Point
{
    inline Point() = default;
  inline Point(double _x,double _y):x(_x),y(_y) {}
  inline Point unit() const
  {
```

```
7       double len = norm();
8       if (!dcmp(len)) return Point(1,0);
9       else return *this/len;
10    }
11    inline double norm() const { return sqrt(x*x+y*y); }
12    inline Point reflect(const Point &p) const
13    {
14      Point v = *this-p; double len = v.norm();
15      v = v/len; return p+v*(1/len);
16    }
17    inline void read() { scanf("%lf %lf",&x,&y); }
18    inline Point vertical() const { return Point(y,-x); }
19    inline double angle() const
20    {
21      double ret = atan2(y,x);
22      if (ret < 0) ret += 2*pi;
23      return ret;
24    }
25    friend inline bool operator ==(const Point &a,const Point &b) { return
      ↪  !dcmp(a.x-b.x)&&!dcmp(a.y-b.y); }
26      friend inline Point operator -(const Point &a,const Point &b) { return
      ↪  Point(a.x-b.x,a.y-b.y); }
27    friend inline Point operator +(const Point &a,const Point &b) { return Point(a.x+b.x,a.y+b.y);
      ↪  }
28    friend inline Point operator /(const Point &a,double b) { return Point(a.x/b,a.y/b); }
29    friend inline Point operator *(const Point &a,double b) { return Point(a.x*b,a.y*b); }
30    friend inline Point operator *(double b,const Point &a) { return Point(a.x*b,a.y*b); }
31    friend inline double operator /(const Point &a,const Point &b) { return a.x*b.y-a.y*b.x; }
32    friend inline bool operator <(const Point &a,const Point &b)
33    {
34      if (a.x != b.x) return a.x < b.x;
35      else return a.y < b.y;
36    }
37  }P[maxn],convex[maxn];
38
39  inline void ConvexHull()
40  {
41      sort(P+1,P+N+1);  //x 第一关键字，y 第二关键字从小到大排序
42      for (int i = 1;i <= N;++i)
43      {
44          while (m > 1&&(convex[m]-convex[m-1])/(P[i]-convex[m-1]) <= 0) --m;
45          convex[++m] = P[i];
46      }
47      int k = m;
48      for (int i = N-1;i;--i)
49      {
50          while (m > k&&(convex[m]-convex[m-1])/(P[i]-convex[m - 1]) <= 0) --m;
51          convex[++m] = P[i];
52      }
53      if (N > 1) m--;
54  }
```

1.10 匈牙利

```
1   //匈牙利算法
2   //Version1
3   inline bool find(int x)
4   {
5     if (cor[x]) return false;
6     for (int i = side[x];i;i = next[i]) if (!used[toit[i]])
7     {
8       used[toit[i]] = true;
9       if (!cho[toit[i]]||find(cho[toit[i]]))
10      {
11        cho[toit[i]] = x; map[x] = toit[i];
12         return true;
13      }
14    }
15    return false;
```

```
}

inline void hungry()
{
  for (int i = 1;i <= p;++i)
    memset(used,false,sizeof(used)),find(i);
  for (int i = 1;i <= m;++i)
  {
    memset(used,false,sizeof(used)),cho[map[i]] = 0;
    find(i),cor[i] = true;
  }
}
//Version2
inline int find(int x)
{
    for (int i = 1;i <= n;++i)
        if (f[x][i]&&!used[i])
        {
            used[i] = true;
            if (!cho[i]||find(cho[i])) { cho[i] = x; return true; }
        }
    return false;
}

inline int hungry()
{
    int ret = 0;
    for (int i = 1;i <= n;++i)
    {
        memset(used,false,sizeof(used));
        if (find(i)) ret++;
    }
    return ret;
}
```

1.11 半平面交

```
//半平面交，直线左侧半平面，注意最后是 tail-head <= 0 还是 tail-head <= 1
inline int dcmp(double a)
{
  if (-eps <= a&&a <= eps) return 0;
  else if (a > 0) return 1; else return -1;
}

struct Point
{
  double x,y;
  inline Point() = default;
  inline Point(double _x,double _y):x(_x),y(_y) {}
  inline void read() { x = gi(),y = gi(); }
  inline Point vertical() const { return Point(-y,x); }
  inline Point unit() const
  {
    double len = norm();
    if (!dcmp(len)) return Point(1,0);
    else return *this/len;
  }
  inline double norm() const { return sqrt(x*x+y*y); }
  inline double angle() const { return atan2(y,x); }
  friend inline Point operator+(const Point &a,const Point &b) { return Point(a.x+b.x,a.y+b.y); }
  friend inline Point operator-(const Point &a,const Point &b) { return Point(a.x-b.x,a.y-b.y); }
  friend inline Point operator*(const Point &a,double b) { return Point(a.x*b,a.y*b); }
  friend inline Point operator*(double b,const Point &a) { return Point(a.x*b,a.y*b); }
  friend inline double operator/(const Point &a,const Point &b) { return a.x*b.y-a.y*b.x; }
}P[maxn],pp[maxn],pol[maxn];

struct Line
{
  Point p,v;
```

```
  inline Line(const Point _p = Point(),const Point _v = Point()):p(_p),v(_v) {}
  inline double slop() const { return v.angle(); }
  friend inline bool operator<(const Line &a,const Line &b) { return a.slop() < b.slop(); }
}line[maxn],qq[maxn];

inline bool onleft(const Line &L,const Point &p)
{
  return dcmp(L.v/(p-L.p)) > 0;
}
inline bool parallel(const Line &a,const Line &b) { return !dcmp(a.v/b.v); }
inline Point crosspoint(const Line &a,const Line &b)
{
  Point u = a.p-b.p;
  double t = (b.v/u)/(a.v/b.v);
  return a.p+(a.v*t);
}

inline int half_plane_intersection()
{
    sort(lines+1,lines+tot+1);  //直线按斜率排序
    int head,tail;
    qq[head = tail = 1] = lines[1];
    for (int i = 2;i <= tot;++i)
    {
        while (head < tail&&!onleft(lines[i],pp[tail-1])) --tail;
        while (head < tail&&!onleft(lines[i],pp[head])) ++head;
        qq[++tail] = lines[i];
        if (parallel(qq[tail],qq[tail-1]))
        {
            tail--;
            if (onleft(qq[tail],lines[i].p)) qq[tail] = lines[i];
        }
        if (head < tail) pp[tail-1] = crosspoint(qq[tail],qq[tail-1]);
    }
    while (head < tail && !onleft(qq[head],pp[tail-1])) --tail;
    if (tail-head <= 0) return 0;
    pp[tail] = crosspoint(qq[tail],qq[head]);
    for (int i = head;i <= tail;++i) pol[++m] = pp[i];  //半平面交点
    pol[0] = pol[m];
    return m;
}
```

1.12 后缀数组

```
inline void build(char *buf,int *Sa,int *Rank,int *Height,int n,int now,int m)
{
    int i,j,k,*x = t1,*y = t2;
    memset(c,0,4*m);
    for (i = 0;i < n;++i) c[x[i] = buf[i]-'A']++;
    for (i = 1;i < m;++i) c[i] += c[i-1];
    for (i = n-1;i >= 0;--i) Sa[--c[x[i]]] = i;
    for (k = 1;k < n;k <<= 1)
    {
        int p = 0;
        for (i = n-k;i < n;++i) y[p++] = i;
        for (i = 0;i < n;++i) if (Sa[i] >= k) y[p++] = Sa[i] - k;
        memset(c,0,4*m);
        for (i = 0;i < n;++i) c[x[y[i]]]++;
        for (i = 1;i < m;++i) c[i] += c[i-1];
        for (i = n-1;i >= 0;--i) Sa[--c[x[y[i]]]] = y[i];
        swap(x,y); p = 1; x[Sa[0]] = 0;
        for (i = 1;i < n;++i)
            x[Sa[i]] = y[Sa[i-1]] == y[Sa[i]]&&y[Sa[i-1]+k] == y[Sa[i]+k]?p-1:p++;
        if (p >= n) break; m = p;
    }
    for (i = 0;i < n;++i) Rank[Sa[i]] = i;
    for (i = k = 0;i < n;++i)
    {
        if (k) --k; if (!Rank[i]) continue;
```

```
        j = Sa[Rank[i]-1];
        while (i+k<n&&j+k<n&&buf[i+k]==buf[j+k]) ++k;
        Height[Rank[i]] = k;
    }
}
```

1.13 圆与多边形面积交

```
const int maxn = 510;
const double eps = 1e-9;

inline int dcmp(double a)
{
  if (a > eps) return 1;
  else if (a < -eps) return -1;
  else return 0;
}

struct Point
{
  double x,y;
  Point() = default;
  Point(double _x,double _y):x(_x),y(_y) {}
  inline double norm() const { return sqrt(x*x+y*y); }
  inline Point unit() const { double len = norm(); return Point(x/len,y/len); }
  friend Point operator +(const Point &a,const Point &b) { return Point(a.x+b.x,a.y+b.y); }
  friend Point operator -(const Point &a,const Point &b) { return Point(a.x-b.x,a.y-b.y); }
  friend Point operator *(const Point &a,double b) { return Point(a.x*b,a.y*b); }
  friend Point operator *(double b,const Point &a) { return Point(a.x*b,a.y*b); }
  friend Point operator /(const Point &a,double b) { return Point(a.x/b,a.y/b); }
  friend double operator /(const Point &a,const Point &b) { return a.x*b.y-b.x*a.y; }
  friend double operator *(const Point &a,const Point &b) { return a.x*b.x+a.y*b.y; }
  inline void read() { scanf("%lf %lf",&x,&y); }
}P[maxn],A,B;
int N; double K;

inline double getSectorArea(const Point &a,const Point &b,double r)
{
  double c = (2*r*r-((a-b)*(a-b)))/(2*r*r);
  double alpha = acos(c);
  return r*r*alpha/2.0;
}

inline pair <double,double> getSolution(double a,double b,double c)
{
  double delta = b*b-4*a*c;
  if (dcmp(delta) < 0) return make_pair(0,0);
  else return make_pair((-b-sqrt(delta))/(2*a),(-b+sqrt(delta))/(2*a));
}

inline pair <Point,Point> getIntersection(const Point &a,const Point &b,double r)
{
  Point d = b-a;
  double A = d*d,B = 2*(d*a),C = (a*a)-r*r;
  pair <double,double> s = getSolution(A,B,C);
  return make_pair(a+(d*s.first),a+(d*s.second));
}

inline double getPointDist(const Point &a,const Point &b)
{
  Point d = b-a;
  int sA = dcmp(a*d),sB = dcmp(b*d);
  if (sA*sB <= 0) return (a/b)/((a-b).norm());
  else return min(a.norm(),b.norm());
}

double getArea(const Point &a,const Point &b,double r)
{
  double dA = a*a,dB = b*b,dC = getPointDist(a,b),ans = 0;
```

```
  if (dcmp(dA-r*r) <= 0&&dcmp(dB-r*r) <= 0) return (a/b)/2;
  Point tA = a.unit()*r,tB = b.unit()*r;
  if (dcmp(dC-r) > 0) return getSectorArea(tA,tB,r);
  pair <Point,Point> ret = getIntersection(a,b,r);
  if (dcmp(dA-r*r) > 0&&dcmp(dB-r*r) > 0)
  {
    ans += getSectorArea(tA,ret.first,r);
    ans += (ret.first/ret.second)/2;
    ans += getSectorArea(ret.second,tB,r);
    return ans;
  }
  if (dcmp(dA-r*r) > 0) return (ret.first/b)/2+getSectorArea(tA,ret.first,r);
  else return (a/ret.second)/2.0+getSectorArea(ret.second,tB,r);
}

double getArea(int n,Point *p,const Point &c,double r)
{
  double ret = 0;
  for (int i = 0;i < n;++i)
  {
    int sgn = dcmp((p[i]-c)/(p[(i+1)%n]-c));
    if (sgn > 0) ret += getArea(p[i]-c,p[(i+1)%n]-c,r);
    else ret -= getArea(p[(i+1)%n]-c,p[i]-c,r);
  }
  return fabs(ret);
}
```

1.14 圆的 K 次交

```
//modified
const double eps = 1e-7,pi = acos(-1.0);
int N,M; double area[maxn]; // area[k] -> area of intersections >= k.

inline int dcmp(double a)
{
  if (-eps <= a&&a <= eps) return 0;
  else if (a > 0) return 1; else return -1;
}

struct Point
{
  double x,y;
  inline Point() = default;
  inline Point(double _x,double _y):x(_x),y(_y) {}
  inline void read() { x = gi(),y = gi(); }
  inline double norm() const { return sqrt(x*x+y*y); }
  inline double angle() const { return atan2(y,x); }
  inline Point unit() const { double len = norm(); return Point(x/len,y/len); }
  friend inline Point operator-(const Point &a,const Point &b) { return Point(a.x-b.x,a.y-b.y); }
  friend inline Point operator+(const Point &a,const Point &b) { return Point(a.x+b.x,a.y+b.y); }
  friend inline Point operator*(const Point &a,double b) { return Point(a.x*b,a.y*b); }
  friend inline Point operator*(double b,const Point &a) { return Point(a.x*b,a.y*b); }
  friend inline Point operator/(const Point &a,double b) { return Point(a.x/b,a.y/b); }
  friend inline double operator/(const Point &a,const Point &b) { return a.x*b.y-a.y*b.x; }
};
struct Circle
{
  Point C; double r; int sgn;
  inline Circle() = default;
  inline Circle(const Point &_C,double _r,int _sgn):C(_C),r(_r),sgn(_sgn) {}
  // sgn 代表该圆的权值，默认 1
  friend inline bool operator==(const Circle &a,const Circle &b)
  {
    if (dcmp(a.r-b.r)) return false;
    if (dcmp(a.C.x-b.C.x)) return false;
    if (dcmp(a.C.y-b.C.y)) return false;
    if (a.sgn != b.sgn) return false;
    return true;
  }
```

```
40    friend inline bool operator!=(const Circle &a,const Circle &b) { return !(a == b); }
41  }cir[maxn];
42
43  inline Point rotate(const Point &p,double cost,double sint)
44  {
45    double x = p.x,y = p.y;
46    return Point(x*cost-y*sint,x*sint+y*cost);
47  }
48  inline pair <Point,Point> crosspoint(const Point &ap,double ar,const Point &bp,double br)
49  {
50    double d = (ap-bp).norm(),cost = (ar*ar+d*d-br*br)/(2*ar*d),sint = sqrt(1-cost*cost);
51    Point v = ((bp-ap).unit())*ar;
52    return make_pair(ap+rotate(v,cost,-sint),ap+rotate(v,cost,sint));
53  }
54  inline pair <Point,Point> crosspoint(const Circle &a,const Circle &b) { return
      ↪  crosspoint(a.C,a.r,b.C,b.r); }
55
56  inline bool overlap(const Circle &a,const Circle &b) { return dcmp(a.r-b.r-(a.C-b.C).norm()) >=
      ↪  0; } // b 是不是在 a 里面
57  inline bool intersect(const Circle &a,const Circle &b)
58  {
59    if (overlap(a,b)) return false;
60    if (overlap(b,a)) return false;
61    return dcmp((a.C-b.C).norm()-a.r-b.r) < 0;
62  }
63
64  struct Event
65  {
66    Point p; double a; int d;
67    inline Event() = default;
68    inline Event(const Point &_p,double _a,double _d):p(_p),a(_a),d(_d) {}
69    friend inline bool operator <(const Event &a,const Event &b) { return a.a < b.a; }
70  };
71
72  inline void solve()
73  {
74    for (int i = 1;i <= M;++i) area[i] = 0;
75    for (int i = 1;i <= M;++i)
76    {
77      int cnt = cir[i].sgn; if (cnt<0) cnt = 0; vector <Event> event;
78      for (int j = 1;j < i;++j) if (cir[i] == cir[j]) cnt += cir[j].sgn;
79      for (int j = 1;j <= M;++j)
80        if (j != i&&cir[i] != cir[j]&&overlap(cir[j],cir[i])) cnt += cir[j].sgn;
81      for (int j = 1;j <= M;++j)
82        if (j != i&&intersect(cir[i],cir[j]))
83        {
84          pair <Point,Point> res = crosspoint(cir[i],cir[j]); swap(res.first,res.second);
85          double alpha1 = (res.first-cir[i].C).angle(),alpha2 = (res.second-cir[i].C).angle();
86          event.push_back(Event(res.second,alpha2,cir[j].sgn));
87          event.push_back(Event(res.first,alpha1,-cir[j].sgn));
88          cnt += (alpha2 > alpha1)*cir[j].sgn;
89        }
90      if (!event.size()) area[cnt] += pi*cir[i].r*cir[i].r*cir[i].sgn;
91      else
92      {
93        sort(event.begin(),event.end());
94        event.push_back(event.front());
95        for (int j = 0;j+1 < (int)event.size();++j)
96        {
97          cnt += event[j].d;
98          area[cnt] += event[j].p/event[j+1].p/2*cir[i].sgn;
99          double alpha = event[j+1].a-event[j].a;
100         if (alpha < 0) alpha += 2*pi;
101         if (!dcmp(alpha)) continue;
102         area[cnt] += alpha*cir[i].r*cir[i].r/2*cir[i].sgn;
103         area[cnt] += -sin(alpha)*cir[i].r*cir[i].r/2*cir[i].sgn;
104       }
105     }
106   }
107 }
```

```
108
109 // origin
110 struct Event {
111   Point p;
112   double ang;
113   int delta;
114   Event (Point p = Point(0, 0), double ang = 0, double delta = 0) : p(p), ang(ang), delta(delta)
    ↪  {}
115 };
116 bool operator < (const Event &a, const Event &b) {
117   return a.ang < b.ang;
118 }
119 void addEvent(const Circle &a, const Circle &b, vector<Event> &evt, int &cnt) {
120   double d2 = (a.o - b.o).len2(),
121        dRatio = ((a.r - b.r) * (a.r + b.r) / d2 + 1) / 2,
122        pRatio = sqrt(-(d2 - sqr(a.r - b.r)) * (d2 - sqr(a.r + b.r)) / (d2 * d2 * 4));
123   Point d = b.o - a.o, p = d.rotate(PI / 2),
124       q0 = a.o + d * dRatio + p * pRatio,
125       q1 = a.o + d * dRatio - p * pRatio;
126   double ang0 = (q0 - a.o).ang(),
127        ang1 = (q1 - a.o).ang();
128   evt.push_back(Event(q1, ang1, 1));
129   evt.push_back(Event(q0, ang0, -1));
130   cnt += ang1 > ang0;
131 }
132 bool issame(const Circle &a, const Circle &b) { return sign((a.o - b.o).len()) == 0 && sign(a.r -
    ↪  b.r) == 0; }
133 bool overlap(const Circle &a, const Circle &b) { return sign(a.r - b.r - (a.o - b.o).len()) >= 0;
    ↪  }
134 bool intersect(const Circle &a, const Circle &b) { return sign((a.o - b.o).len() - a.r - b.r) <
    ↪  0; }
135 Circle c[N];
136 double area[N];  // area[k] -> area of intersections >= k.
137 Point centroid[N];  //k 次圆的质心
138 bool keep[N];
139 void add(int cnt, DB a, Point c) {
140   area[cnt] += a;
141   centroid[cnt] = centroid[cnt] + c * a;
142 }
143 void solve(int C) {
144   for (int i = 1; i <= C; ++ i) {
145         area[i] = 0;
146         centroid[i] = Point(0, 0);
147     }
148   for (int i = 0; i < C; ++i) {
149     int cnt = 1;
150     vector<Event> evt;
151     for (int j = 0; j < i; ++j) if (issame(c[i], c[j])) ++cnt;
152     for (int j = 0; j < C; ++j) {
153       if (j != i && !issame(c[i], c[j]) && overlap(c[j], c[i])) {
154         ++cnt;
155       }
156     }
157     for (int j = 0; j < C; ++j) {
158       if (j != i && !overlap(c[j], c[i]) && !overlap(c[i], c[j]) && intersect(c[i], c[j])) {
159         addEvent(c[i], c[j], evt, cnt);
160       }
161     }
162     if (evt.size() == 0u) {
163       add(cnt, PI * c[i].r * c[i].r, c[i].o);
164     } else {
165       sort(evt.begin(), evt.end());
166       evt.push_back(evt.front());
167       for (int j = 0; j + 1 < (int)evt.size(); ++j) {
168         cnt += evt[j].delta;
169         add(cnt, det(evt[j].p, evt[j + 1].p) / 2, (evt[j].p + evt[j + 1].p) / 3);
170         double ang = evt[j + 1].ang - evt[j].ang;
171         if (ang < 0) {
172           ang += PI * 2;
173         }
```

```
174                 if (sign(ang) == 0) continue;
175           double ang0 = evt[j].a,ang1 = evt[j+1].a;
176                 add(cnt, ang * c[i].r * c[i].r / 2, c[i].o +
177                     Point(sin(ang1) - sin(ang0), -cos(ang1) + cos(ang0)) * (2 / (3 * ang) *
    ↪  c[i].r));
178           add(cnt, -sin(ang) * c[i].r * c[i].r / 2, (c[i].o + evt[j].p + evt[j + 1].p) / 3);
179         }
180       }
181     }
182       for (int i = 1; i <= C; ++ i)
183       if (sign(area[i])) {
184         centroid[i] = centroid[i] / area[i];
185       }
186   }
```

1.15 平面图

```
1   // 包括平面图转对偶图
2   inline int dcmp(double a)
3   {
4     if (fabs(a) <= eps) return 0;
5     else if (a > 0) return 1;
6     else return -1;
7   }
8   struct Point
9   {
10    double x,y;
11    inline Point(double _x = 0,double _y = 0):x(_x),y(_y) {}
12    inline void read() { x = gi(),y = gi(); }
13    friend inline Point operator-(const Point &a,const Point &b) { return Point(a.x-b.x,a.y-b.y); }
14    friend inline double operator/(const Point &a,const Point &b) { return a.x*b.y-a.y*b.x; }
15    inline double angle() { return atan2(y,x); }
16  }pp[maxn];
17  struct Segment
18  {
19    int from,to,h,id,sur; // from 号点到 to 号点，h 为边权,suf 为这条有向边维出来的平面编号。
20    inline Segment(int _from = 0,int _to = 0,int _h = 0,int _id = 0,int _sur =
    ↪  0):from(_from),to(_to),h(_h),id(_id),sur(_sur) {}
21    friend inline bool operator<(const Segment &a,const Segment &b) { return
    ↪  (pp[a.to]-pp[a.from]).angle() < (pp[b.to]-pp[b.from]).angle(); }
22  }edge[maxm*2];
23  vector <int> G[maxn];
24
25  inline void nadd(int u,int v,int h) { ++ncnt; G[u].push_back(ncnt); edge[ncnt] = Segment(u,v,h);
    ↪  }
26  inline void nins(int u,int v,int h) { nadd(u,v,h); nadd(v,u,h); }
27
28  inline bool cmp(int a,int b) { return edge[a] < edge[b]; }
29
30  inline void find_surface()
31  {
32    for (int i = 1;i <= N;++i) sort(G[i].begin(),G[i].end(),cmp);
33    for (int i = 1;i <= N;++i)
34    {
35      int nn = G[i].size();
36      for (int j = 0;j < nn;++j)
37        edge[G[i][j]].id = j;
38    }
39    for (int i = 2;i <= ncnt;++i)
40      if (!edge[i].sur)
41      {
42        ++tot; int j = i,p,nn; vector <Point> vec;
43        while (!edge[j].sur)
44        {
45          edge[j].sur = tot; vec.push_back(pp[edge[j].from]);
46          p = edge[j].to; nn = G[p].size();
47          j ^= 1; j = G[p][(edge[j].id+1)%nn];
48        }
49        double res = 0; nn = vec.size();
```

```
50        for (j = 0;j < nn;++j)
51          res += (vec[j]-vec[0])/(vec[(j+1)%nn]-vec[0]);
52        res /= 2; space[tot] = res;
      ↪  // 第 tot 个平面的有向面积，外面的大平面面积为正，其余为负，大平面可能有多个（平面图不连通）
53      }
54    // 开始建边，以 mst 为例
55    // for (int i = 2;i <= cnt;i += 2)
56    // {
57    //   if (space[edge[i].sur]<0&&space[edge[i^1].sur]<0)
58    //     arr[++all] = (ARR) { edge[i].sur,edge[i^1].sur,edge[i].h };
59    //   else arr[++all] = (ARR) { edge[i].sur,edge[i^1].sur,inf};
60    // }
61  }
62
63  // 点定位
64  struct Scan
65  {
66    double x,y; int bel,sign;
67    inline Scan(double _x = 0,double _y = 0,int _bel = 0,int _sign =
      ↪  0):x(_x),y(_y),bel(_bel),sign(_sign) {}
68    friend inline bool operator < (const Scan &a,const Scan &b)
69    {
70      if (a.x != b.x) return a.x < b.x;
71      else return a.sign > b.sign;
72    }
73  }bac[maxn*4];
74
75  struct Splay
76  {
77    int num,root,ch[maxn][2],fa[maxn],key[maxn]; queue <int> team;
78
79    inline int newnode()
80    {
81      int ret;
82      if (team.empty()) ret = ++num;
83      else ret = team.front(),team.pop();
84      fa[ret] = ch[ret][0] = ch[ret][1] = 0;
85      return ret;
86    }
87
88    inline void init() { num = 0; root = newnode(); key[root] = cnt; }
89
90    inline void rotate(int x)
91    {
92      int y = fa[x],z = fa[y],l = ch[y][1] == x,r = l^1;
93      if (z != 0) ch[z][ch[z][1] == y] = x;
94      fa[x] = z; fa[y] = x; fa[ch[x][r]] = y;
95      ch[y][l] = ch[x][r]; ch[x][r] = y;
96    }
97
98    inline void splay(int x)
99    {
100     while (fa[x] != 0)
101     {
102       int y = fa[x],z = fa[y];
103       if (fa[y] != 0)
104       {
105         if ((ch[y][0] == x)^(ch[z][0] == y)) rotate(x);
106         else rotate(y);
107       }
108       rotate(x);
109     }
110     root = x;
111   }
112
113   inline int lower_bound(const Point &p)
114   {
115     int now = root,ret = 0;
116     while (now)
117     {
```

```
118         int k = key[now];
119         if ((p-pp[edge[k].from])/(pp[edge[k].to]-pp[edge[k].from]) >= 0)
120           ret = k,now = ch[now][0];
121         else now = ch[now][1];
122       }
123       return ret;
124     }
125
126     inline int find(int w)
127     {
128       int now = root;
129       double x = pp[edge[w].to].x,y = pp[edge[w].to].y;
130       double ang = (pp[edge[w].to] - pp[edge[w].from]).angle();
131       while (now)
132       {
133         int k = key[now];
134         if (k == w) return now;
135         NODE p = pp[edge[k].to] - pp[edge[k].from],q = pp[edge[k].from];
136         double xx = x - q.x,yy = q.y+xx/p.x*p.y;
137         if (equal(yy,y))
138         {
139           double t = p.angle();
140           now = ch[now][ang < t];
141         }
142         else now = ch[now][y > yy];
143       }
144     }
145
146     inline void erase(int w)
147     {
148       int p = find(w);
149       while (ch[p][0] || ch[p][1])
150       {
151         if (ch[p][0])
152         {
153           rotate(ch[p][0]);
154           if (p == root) root = fa[p];
155         }
156         else
157         {
158           rotate(ch[p][1]);
159           if (p == root) root = fa[p];
160         }
161       }
162       team.push(p);
163       ch[fa[p]][ch[fa[p]][1] == p] = 0;
164       fa[p] = 0;
165     }
166
167     inline void insert(int w)
168     {
169       int now = root,pre;
170       double x = pp[edge[w].from].x,y = pp[edge[w].from].y;
171       double ang = (pp[edge[w].to] - pp[edge[w].from]).angle();
172       double xx,yy;
173       while (true)
174       {
175         int k = key[now];
176         NODE p = pp[edge[k].to] - pp[edge[k].from],q = pp[edge[k].from];
177         xx = x - q.x,yy = q.y+xx/p.x*p.y;
178         if (equal(yy,y))
179         {
180           double t = p.angle();
181           pre = now,now = ch[now][ang > t];
182           if (!now)
183           {
184             now = newnode();
185             fa[now] = pre; ch[pre][ang > t] = now; key[now] = w;
186             break;
187           }
```

```
188       }
189       else
190       {
191         pre = now,now = ch[now][y > yy];
192         if (!now)
193         {
194           now = newnode();
195           fa[now] = pre; ch[pre][y>yy] = now; key[now] = w;
196           break;
197         }
198       }
199     }
200     splay(now);
201   }
202 }S;
203
204 inline void locate()
205 {
206   int nn = 0;
207   for (int i = 2;i <= cnt;i += 2)
208   {
209     if (!dcmp(pp[edge[i].from].x-pp[edge[i].to].x)) continue;
210     bac[++nn] = Scan(pp[edge[i].from].x,pp[edge[i].from].y,i,2);
211     bac[++nn] = Scan(pp[edge[i].to].x,pp[edge[i].to].y,i,3);
212   }
213   scanf("%d",&T); double x,y;
214   // 查询 (x,y) 所在平面
215   for (int i = 1;i <= T;++i)
216   {
217     scanf("%lf %lf",&x,&y);
218     bac[++nn] = Scan(x,y,i,0);
219     scanf("%lf %lf",&x,&y);
220     bac[++nn] = Scan(x,y,i,1);
221   }
222   sort(bac+1,bac+nn+1);
223   pp[++n] = Point(-oo,-oo); pp[++n] = (oo,-oo);
224   edge[++cnt] = Edge(n-1,n);
225   S.init(); int p;
226   for (int i = 1;i <= nn;++i)
227   {
228     if (bac[i].sign == 2||bac[i].sign == 3)
229     {
230       if (bac[i].sign == 2) S.insert(bac[i].bel);
231       else S.erase(bac[i].bel);
232     }
233     else
234     {
235       p = S.lower_bound(Point(bac[i].x,bac[i].y));
236       query[bac[i].bel][bac[i].sign] = edge[p].sur;
237     }
238   }
239 }
```

1.16 弦图染色最大势

```
1  #include<algorithm>
2  #include<queue>
3  #include<cstdio>
4  #include<cstdlib>
5  #include<set>
6  using namespace std;
7
8  #define maxn 10010
9  #define maxc 510
10 #define maxm 1000010
11 int tot,n,m,cnt,color[maxn][maxc],label[maxn],all;
12 int side[maxn],next[maxm*2],toit[maxm*2],per[maxn];
13 bool in[maxn];
14 struct node
```

```
{
  int key,ord;
  friend bool operator < (node a,node b) {return a.key > b.key; }
};
multiset <node> S;

inline void add(int a,int b)
{
  next[++cnt] = side[a]; side[a] = cnt; toit[cnt] = b;
}

inline void ins(int a,int b){add(a,b); add(b,a);}

inline void mcs()
{
  int i,u;
  for (i = 1;i <= n;++i) S.insert((node){0,i});
  while (all < n)
  {
    u = (*S.begin()).ord; S.erase(S.begin()); if (in[u]) continue;
    in[u] = true; per[++all] = u;
    for (i = side[u];i;i = next[i])
      if (!in[toit[i]])
      {
        label[toit[i]]++;
        S.insert((node){label[toit[i]],toit[i]});
      }
  }
}

inline void paint()
{
  int p,i,j,t;
  for (p = 1;p <= n;++p)
  {
    i = per[p];
    for (j = 1;j <= tot;++j)
      if (!color[i][j]) {t = j; break; }
    if (j == tot + 1) t = ++tot;
    for (j = side[i];j;j = next[j])
      color[toit[j]][t] = true;
  }
}

int main()
{
  freopen("1006.in","r",stdin);
  freopen("1006.out","w",stdout);
  scanf("%d %d",&n,&m);
  for (int i = 1;i <= m;++i)
  { int a,b; scanf("%d %d",&a,&b); ins(a,b); }
  mcs();
  paint();
  printf("%d",tot);
  fclose(stdin); fclose(stdout);
  return 0;
}
```

1.17 强连通分量

```
int dfn[maxn],low[maxn],timestamp;
stack <int> stk; vector <int> scc[maxn];
void tarjan(int now)
{
  dfn[now] = low[now] = ++timestamp;
  stk.push(now);
  for (int i = side[now];i;i = nxt[i])
  {
    if (!dfn[toit[i]])
```

```
      tarjan(toit[i]),low[now] = min(low[now],low[toit[i]]);
    else if (!bel[toit[i]]) low[now] = min(low[now],dfn[toit[i]]);
  }
  if (dfn[now] == low[now])
  {
    ++tot;
    while (stk.top() != now)
    {
      scc[tot].push_back(stk.top());
      bel[stk.top()] = tot; stk.pop();
    }
    scc[tot].push_back(stk.top());
    bel[stk.top()] = tot; stk.pop();
  }
}
```

1.18 支配树

```
//建出来的树点的编号 i 在原图中是 redfn[i]
int N,M,Ts,cnt,side[maxn],nxt[maxn],toit[maxn],dfn[maxn],redfn[maxn],idom[maxn],best[maxn],semi[maxn];
int ans[maxn],anc[maxn],fa[maxn],child[maxn],size[maxn]; vector <int> prod[maxn],bucket[maxn],son[maxn];

inline void init()
{
  cnt = 1; memset(side,0,sizeof side); memset(ans,0,sizeof ans);
  for (int i = 0;i <= N;++i) prod[i].clear(),bucket[i].clear(),son[i].clear();
}

inline void add(int a,int b) { nxt[++cnt] = side[a]; side[a] = cnt; toit[cnt] = b; }

inline int gi()
{
  char ch; int ret = 0,f = 1;
  do ch = getchar(); while (!(ch >= '0'&&ch <= '9')&&ch != '-');
  if (ch == '-') f = -1,ch = getchar();
  do ret = ret*10+ch-'0',ch = getchar(); while (ch >= '0'&&ch <= '9');
  return ret*f;
}

inline void dfs(int now)
{
  dfn[now] = ++Ts; redfn[Ts] = now;
  anc[Ts] = idom[Ts] = child[Ts] = size[Ts] = 0;
  semi[Ts] = best[Ts] = Ts;
  for (int i = side[now];i;i = nxt[i])
  {
    if (!dfn[toit[i]])
      dfs(toit[i]),fa[dfn[toit[i]]] = dfn[now];
    prod[dfn[toit[i]]].push_back(dfn[now]);
  }
}

inline void compress(int now)
{
  if (anc[anc[now]] != 0)
  {
    compress(anc[now]);
    if (semi[best[now]] > semi[best[anc[now]]])
      best[now] = best[anc[now]];
    anc[now] = anc[anc[now]];
  }
}

inline int eval(int now)
{
  if (!anc[now]) return now;
  else
```

```
  {
    compress(now);
    return semi[best[anc[now]]] >= semi[best[now]]?best[now]:best[anc[now]];
  }
}

inline void link(int v,int w)
{
  int s = w;
  while (semi[best[w]] < semi[best[child[w]]])
  {
    if (size[s]+size[child[child[s]]] >= 2*size[child[s]])
      anc[child[s]] = s,child[s] = child[child[s]];
    else size[child[s]] = size[s],s = anc[s] = child[s];
  }
  best[s] = best[w]; size[v] += size[w];
  if (size[v] < 2*size[w]) swap(s,child[v]);
  while (s) anc[s] = v,s = child[s];
}

inline void lengauer_tarjan()
{
  memset(dfn,0,sizeof dfn); memset(fa,-1,sizeof fa); Ts = 0;
  dfs(N); fa[1] = 0;
  for (int w = Ts;w > 1;--w)
  {
    for (auto x:prod[w])
    {
      int u = eval(x);
      if (semi[w] > semi[u]) semi[w] = semi[u];
    }
    bucket[semi[w]].push_back(w);
    link(fa[w],w); if (!fa[w]) continue;
    for (auto x:bucket[fa[w]])
    {
      int u = eval(x);
      if (semi[u] < fa[w]) idom[x] = u;
      else idom[x] = fa[w];
    }
    bucket[fa[w]].clear();
  }
  for (int w = 2;w <= Ts;++w)
    if (idom[w] != semi[w])
      idom[w] = idom[idom[w]];
  idom[1] = 0;
  for (int i = Ts;i > 1;--i)
  {
    if (fa[i] == -1) continue;
    son[idom[i]].push_back(i);
  }
}

// 例题：询问 i 号点到 N 号点所有必经点编号和
#include<algorithm>
#include<cstring>
#include<iostream>
#include<cstdio>
#include<cstdlib>
using namespace std;

const int maxn = 100010;
int
 ↪  N,M,Ts,cnt,side[maxn],nxt[maxn],toit[maxn],dfn[maxn],redfn[maxn],idom[maxn],best[maxn],semi[maxn];
int ans[maxn],anc[maxn],fa[maxn],child[maxn],size[maxn]; vector <int>
 ↪  prod[maxn],bucket[maxn],son[maxn];

inline void init()
{
  cnt = 1; memset(side,0,sizeof side); memset(ans,0,sizeof ans);
  for (int i = 0;i <= N;++i) prod[i].clear(),bucket[i].clear(),son[i].clear();
```

```
118 }
119
120 inline void add(int a,int b) { nxt[++cnt] = side[a]; side[a] = cnt; toit[cnt] = b; }
121
122 inline int gi()
123 {
124   char ch; int ret = 0,f = 1;
125   do ch = getchar(); while (!(ch >= '0'&&ch <= '9')&&ch != '-');
126   if (ch == '-') f = -1,ch = getchar();
127   do ret = ret*10+ch-'0',ch = getchar(); while (ch >= '0'&&ch <= '9');
128   return ret*f;
129 }
130
131 inline void dfs(int now)
132 {
133   dfn[now] = ++Ts; redfn[Ts] = now;
134   anc[Ts] = idom[Ts] = child[Ts] = size[Ts] = 0;
135   semi[Ts] = best[Ts] = Ts;
136   for (int i = side[now];i;i = nxt[i])
137   {
138     if (!dfn[toit[i]])
139       dfs(toit[i]),fa[dfn[toit[i]]] = dfn[now];
140     prod[dfn[toit[i]]].push_back(dfn[now]);
141   }
142 }
143
144 inline void compress(int now)
145 {
146   if (anc[anc[now]] != 0)
147   {
148     compress(anc[now]);
149     if (semi[best[now]] > semi[best[anc[now]]])
150       best[now] = best[anc[now]];
151     anc[now] = anc[anc[now]];
152   }
153 }
154
155 inline int eval(int now)
156 {
157   if (!anc[now]) return now;
158   else
159   {
160     compress(now);
161     return semi[best[anc[now]]] >= semi[best[now]]?best[now]:best[anc[now]];
162   }
163 }
164
165 inline void link(int v,int w)
166 {
167   int s = w;
168   while (semi[best[w]] < semi[best[child[w]]])
169   {
170     if (size[s]+size[child[child[s]]] >= 2*size[child[s]])
171       anc[child[s]] = s,child[s] = child[child[s]];
172     else size[child[s]] = size[s],s = anc[s] = child[s];
173   }
174   best[s] = best[w]; size[v] += size[w];
175   if (size[v] < 2*size[w]) swap(s,child[v]);
176   while (s) anc[s] = v,s = child[s];
177 }
178
179 inline void lengauer_tarjan()
180 {
181   memset(dfn,0,sizeof dfn); memset(fa,-1,sizeof fa); Ts = 0;
182   dfs(N); fa[1] = 0;
183   for (int w = Ts;w > 1;--w)
184   {
185     for (auto x:prod[w])
186     {
187       int u = eval(x);
```

```
188         if (semi[w] > semi[u]) semi[w] = semi[u];
189       }
190       bucket[semi[w]].push_back(w);
191       link(fa[w],w); if (!fa[w]) continue;
192       for (auto x:bucket[fa[w]])
193       {
194         int u = eval(x);
195         if (semi[u] < fa[w]) idom[x] = u;
196         else idom[x] = fa[w];
197       }
198       bucket[fa[w]].clear();
199     }
200     for (int w = 2;w <= Ts;++w)
201       if (idom[w] != semi[w])
202         idom[w] = idom[idom[w]];
203     idom[1] = 0;
204     for (int i = Ts;i > 1;--i)
205     {
206       if (fa[i] == -1) continue;
207       son[idom[i]].push_back(i);
208     }
209   }
210
211   inline void get_ans(int now)
212   {
213     ans[redfn[now]] += redfn[now];
214     for (auto x:son[now])
215       ans[redfn[x]] += ans[redfn[now]],get_ans(x);
216   }
217
218   int main()
219   {
220     //freopen("I.in","r",stdin);
221     while (scanf("%d %d",&N,&M) != EOF)
222     {
223       init();
224       for (int i = 1,a,b;i <= M;++i)
225         a = gi(),b = gi(),add(a,b);
226       lengauer_tarjan(); get_ans(1);
227       for (int i = 1;i <= N;++i)
228         printf("%d%c",ans[i]," \n"[i == N]);
229     }
230     return 0;
231   }
```

1.19 点双联通分量

```
1   #include<algorithm>
2   #include<vector>
3   #include<stack>
4   #include<iostream>
5   #include<cstdio>
6   #include<cstdlib>
7   using namespace std;
8
9   const int maxn = 400010;
10  int N,M,Q,cnt = 1,side[maxn],toit[maxn],nxt[maxn],f[maxn][25],father[maxn],low[maxn];
11  int tot,dep[maxn],dfn[maxn],nside[maxn],ntoit[maxn],nnxt[maxn]; bool cut[maxn];
12  stack <int> S; vector <int> bel[maxn],bcc[maxn]; bool vis[maxn];
13
14  inline int find(int a) { if (father[a] != a) father[a] = find(father[a]); return father[a]; }
15
16  inline void add(int a,int b) { nxt[++cnt] = side[a]; side[a] = cnt; toit[cnt] = b; }
17  inline void ins(int a,int b) { add(a,b); add(b,a); }
18  inline void nadd(int a,int b) { nnxt[++cnt] = nside[a]; nside[a] = cnt; ntoit[cnt] = b; }
19  inline void nins(int a,int b) { nadd(a,b); nadd(b,a); }
20
21  inline int gi()
22  {
```

```
23    char ch; int ret = 0,f = 1;
24    do ch = getchar(); while (!(ch >= '0'&&ch <= '9')&&ch != '-');
25    if (ch == '-') f = -1,ch = getchar();
26    do ret = ret*10+ch-'0',ch = getchar(); while (ch >= '0'&&ch <= '9');
27    return ret*f;
28  }
29
30  inline void tj(int now,int fa)
31  {
32    dfn[now] = low[now] = ++cnt; int child = 0;
33    for (int i = side[now];i;i = nxt[i])
34    {
35      if (toit[i] == fa) continue;
36      if (!dfn[toit[i]])
37      {
38        S.push(i>>1); tj(toit[i],now); ++child;
39        low[now] = min(low[now],low[toit[i]]);
40        if (low[toit[i]] >= dfn[now])
41        {
42          cut[now] = true; ++tot;
43          while (true)
44          {
45            int t = S.top(); S.pop();
46            bel[toit[t<<1]].push_back(tot);  bel[toit[t<<1|1]].push_back(tot);
47            bcc[tot].push_back(toit[t<<1]); bcc[tot].push_back(toit[t<<1|1]);
48            if (t == (i>>1)) break;
49          }
50        }
51      }
52      else low[now] = min(low[now],dfn[toit[i]]);
53    }
54    if (!fa&&child == 1) cut[now] = false;
55  }
56
57  inline void build()
58  {
59    vector <int> cuts; cnt = 1;
60    for (int i = 1;i <= tot;++i)
61    {
62      sort(bcc[i].begin(),bcc[i].end());
63      bcc[i].erase(unique(bcc[i].begin(),bcc[i].end()),bcc[i].end());
64    }
65    for (int i = 1;i <= N;++i) if (cut[i]) cuts.push_back(i);
66    for (auto x:cuts)
67    {
68      sort(bel[x].begin(),bel[x].end());
69      bel[x].erase(unique(bel[x].begin(),bel[x].end()),bel[x].end());
70      ++tot; for (auto y:bel[x]) nins(tot,y);
71      bel[x].clear(); bel[x].push_back(tot); bcc[tot].push_back(x);
72    }
73  }
74
75  inline void dfs(int now)
76  {
77    vis[now] = true;
78    for (int i = 1;(1<<i) <= dep[now];++i) f[now][i] = f[f[now][i-1]][i-1];
79    for (int i = nside[now];i;i = nnxt[i])
80    {
81      if (vis[ntoit[i]]) continue; f[ntoit[i]][0] = now;
82      dep[ntoit[i]] = dep[now]+1; dfs(ntoit[i]);
83    }
84  }
85
86  inline int jump(int a,int b) { for (int i = 0;b;++i,b >>= 1) if (b&1) a = f[a][i]; return a; }
87  inline int lca(int a,int b)
88  {
89    if (dep[a] < dep[b]) swap(a,b);
90    a = jump(a,dep[a]-dep[b]); if (a == b) return a;
91    for (int i = 0;i >= 0;)
92    {
```

```
93      if (f[a][i] != f[b][i]) a = f[a][i],b = f[b][i],++i;
94      else --i;
95    }
96    return f[a][0];
97  }
98
99  inline bool check(int u,int v,int w)
100 {
101   if (find(u) != find(v)||find(v) != find(w)) return false;
102   if (u == w||v == w) return true; if (u == v) return false;
103   int uu = bel[u][0],vv = bel[v][0],ww = bel[w][0],su,sv;
104   if (uu == ww||vv == ww) return true;
105   if (lca(uu,ww) == ww) su = jump(uu,dep[uu]-dep[ww]-1); else su = f[ww][0];
106   if (lca(vv,ww) == ww) sv = jump(vv,dep[vv]-dep[ww]-1); else sv = f[ww][0];
107   if (su == sv)
108   {
109     if (!cut[w]) return false;
110     else
111     {
112       if (su == uu||sv == vv) return true; int ssu,ssv;
113       if (lca(su,uu) == su) ssu = jump(uu,dep[uu]-dep[su]-1); else ssu = f[su][0];
114       if (lca(sv,vv) == sv) ssv = jump(vv,dep[vv]-dep[sv]-1); else ssv = f[sv][0];
115       if (ssu == ssv) return false; else return true;
116     }
117   }
118   else return true;
119 }
120
121 int main()
122 {
123   freopen("J.in","r",stdin);
124   freopen("J.out","w",stdout);
125   N = gi(); M = gi(); Q = gi();
126   for (int i = 1;i <= N;++i) father[i] = i;
127   for (int i = 1,a,b;i <= M;++i)
128   {
129     ins(a = gi(),b = gi());
130     a = find(a),b = find(b);
131     if (a != b) father[a] = b;
132   }
133   cnt = 0; for (int i = 1;i <= N;++i) if (!dfn[i]) tj(i,0);
134   build(); for (int i = 1;i <= N;++i) if (!vis[i]) dfs(i);
135   while (Q--)
136   {
137     int u = gi(),v = gi(),w = gi();
138     if (check(u,v,w)) puts("YES"); else puts("NO");
139   }
140   return 0;
141 }
```

1.20 线性规划

```
1  #include<iostream>
2  #include<cstdio>
3  #include<cstdlib>
4  using namespace std;
5
6  #define maxn (30)
7  #define eps (1e-8)
8
9  int N,M,op,tot,q[maxn],idx[maxn],idy[maxn]; double a[maxn][maxn],A[maxn];
10
11 inline void pivot(int x,int y)
12 {
13   swap(idy[x],idx[y]);
14   double tmp = a[x][y]; a[x][y] = 1/a[x][y];
15   for (int i = 0;i <= N;++i) if (y != i) a[x][i] /= tmp;
16   tot = 0; for (int i = 0;i <= N;++i) if (i != y&&(a[x][i] > eps||a[x][i] < -eps)) q[++tot] = i;
17   for (int i = 0;i <= M;++i)
```

```
  {
    if ((x == i)||(a[i][y] < eps&&a[i][y] > -eps)) continue;
    for (int j = 1;j <= tot;++j) a[i][q[j]] -= a[x][q[j]]*a[i][y];
    a[i][y] = -a[i][y]/tmp;
  }
}

int main()
{
  freopen("179.in","r",stdin);
  freopen("179.out","w",stdout);
  scanf("%d %d %d",&N,&M,&op); srand(233);
  for (int i = 1;i <= N;++i) scanf("%lf",a[0]+i);
  for (int i = 1;i <= M;++i)
  {
    for (int j = 1;j <= N;++j) scanf("%lf",a[i]+j);
    scanf("%lf",a[i]);
  }
  for (int i = 1;i <= N;++i) idx[i] = i;
  for (int i = 1;i <= M;++i) idy[i] = i+N;
  while (true)
  {
    int x = 0,y = 0;
    for (int i = 1;i <= M;++i) if (a[i][0] < -eps&&((!x)||(rand()&1))) x = i; if (!x) break;
    for (int i = 1;i <= N;++i) if (a[x][i] < -eps&&((!y)||(rand()&1))) y = i; if (!y) return
 ↪  puts("Infeasible"),0;
    pivot(x,y);
  }
  while (true)
  {
    int x = 0,y = 0; double mn = 1e15;
    for (int i = 1;i <= N;++i) if (a[0][i] > eps) { y = i; break; } if (!y) break;
    for (int i = 1;i <= M;++i) if (a[i][y] > eps && a[i][0]/a[i][y] < mn) mn = a[i][0]/a[i][y],x
 ↪  = i; if (!x) return puts("Unbounded"),0;
    pivot(x,y);
  }
  printf("%.8lf\n",-a[0][0]); if (!op) return 0;
  for (int i = 1;i <= M;++i) if (idy[i] <= N) A[idy[i]] = a[i][0];
  for (int i = 1;i <= N;++i) printf("%.8lf ",A[i]);
  fclose(stdin); fclose(stdout);
  return 0;
}
```

1.21 费用流

```
int side[maxv],nxt[maxe],toit[maxe],cost[maxe],pre[maxv];
int cap[maxv],arr[maxv],dis[maxv]; bool in[maxv];
int source,sink;

inline void add(int a,int b,int c,int d) { nxt[++cnt] = side[a]; side[a] = cnt; toit[cnt] = b;
 ↪  cap[cnt] = c; cost[cnt] = d; }
inline void ins(int a,int b,int c,int d) { add(a,b,c,d); add(b,a,0,-d); }

inline bool spfa(int &Flow,int &Cost)
{
    queue <int> team; team.push(source);
    memset(dis,0x7,4*(sink+5));
    dis[source] = 0; in[source] = true;
  arr[source] = inf; arr[sink] = 0;
    while (!team.empty())
    {
        int now = team.front(); team.pop();
        for (int i = side[now];i;i = nxt[i])
        {
            if (!cap[i]) continue;
            if (dis[toit[i]] > dis[now]+cost[i])
            {
                arr[toit[i]] = min(cap[i],arr[now]); pre[toit[i]] = i;
                dis[toit[i]] = dis[now]+cost[i];
```

```
24                  if (!in[toit[i]]) in[toit[i]] = true,team.push(toit[i]);
25              }
26          }
27          in[now] = false;
28      }
29      if (!arr[sink]) return false;
30      Flow += arr[sink];
31      for (int now = sink,i;now != source;now = toit[i^1])
32      {
33          i = pre[now]; Cost += cost[pre[now]]*arr[sink];
34          cap[i] -= arr[sink]; cap[i^1] += arr[sink];
35      }
36      return true;
37  }
```

Chapter 2

司宇

2.1 FFT

```
#include<iostream>
#include<cstdio>
#include<cmath>
using namespace std;
const double eps=1e-8;
const double PI=acos(-1.0);
struct Complex
{
    double real,image;
    Complex(double _real,double _image)
    {
        real=_real;
        image=_image;
    }
    Complex(){real=0;image=0;}
};

Complex operator + (const Complex &c1, const Complex &c2)
{
    return Complex(c1.real + c2.real, c1.image + c2.image);
}

Complex operator - (const Complex &c1, const Complex &c2)
{
    return Complex(c1.real - c2.real, c1.image - c2.image);
}

Complex operator * (const Complex &c1, const Complex &c2)
{
    return Complex(c1.real*c2.real - c1.image*c2.image, c1.real*c2.image + c1.image*c2.real);
}

int rev(int id,int len)
{
    int ret=0;
    for(int i=0;(1<<i)<len;i++)
    {
        ret<<=1;
        if(id&(1<<i))
      ret|=1;
    }
    return ret;
}
Complex* IterativeFFT(Complex* a,int len,int DFT)
{
    Complex* A=new Complex[len];
    for(int i=0;i<len;i++)
        A[rev(i,len)]=a[i];
    for(int s=1;(1<<s)<=len;s++)
    {
        int m=(1<<s);
```

```
52          Complex wm=Complex(cos(DFT*2*PI/m),sin(DFT*2*PI/m));
53          for(int k=0;k<len;k+=m)
54          {
55              Complex w=Complex(1,0);
56              for(int j=0;j<(m>>1);j++)
57              {
58                  Complex t=w*A[k+j+(m>>1)];
59                  Complex u=A[k+j];
60                  A[k+j]=u+t;
61                  A[k+j+(m>>1)]=u-t;
62                  w=w*wm;
63              }
64          }
65      }
66      if(DFT==-1)
67    for(int i=0;i<len;i++)
68    {
69      A[i].real/=len;
70      A[i].image/=len;
71    }
72      return A;
73  }
74  char s[101010],t[101010];
75  Complex a[202020],b[202020],c[202020];
76  int pr[202020];
77  int main()
78  {
79    int len;
80    scanf("%d",&len);
81    scanf("%s",s);
82    scanf("%s",t);
83    for(int i=0;i<len;i++)
84      a[i]=Complex(s[len-i-1]-'0',0);
85    for(int i=0;i<len;i++)
86      b[i]=Complex(t[len-i-1]-'0',0);
87    int tmp=1;
88    while(tmp<=len)
89      tmp*=2;
90    len=tmp*2;
91    Complex* aa=IterativeFFT(a,len,1);
92    Complex* bb=IterativeFFT(b,len,1);
93    for(int i=0;i<len;i++)
94      c[i]=aa[i]*bb[i];
95    Complex* ans=IterativeFFT(c,len,-1);
96    for(int i=0;i<len;i++)
97      pr[i]=round(ans[i].real);
98    for(int i=0;i<=len;i++)
99    {
100     pr[i+1]+=pr[i]/10;
101     pr[i]%=10;
102   }
103   bool flag=0;
104   for(int i=len-1;i>=0;i--)
105   {
106     if(pr[i]>0)
107       flag=1;
108     if(flag)
109       printf("%d",pr[i]);
110   }
111   printf("\n");
112     return 0;
113 }
```

2.2 NTT

```
1  #include <iostream>
2  #include <cstdio>
3  #include <cstring>
4  #include <algorithm>
```

```cpp
#include <cmath>
using namespace std;
const int N=(1<<18)+5, INF=1e9;
const double PI=acos(-1);
long long P=1004535809;
long long Pow(long long a, long long b,long long P)
{
    long long ans=1;
    for(; b; b>>=1, a=a*a%P)
        if(b&1) ans=ans*a%P;
    return ans;
}
struct NumberTheoreticTransform {
    int n, rev[N];
    long long g;
    void ini(int lim) {
        g=3;
        n=1; int k=0;
        while(n<lim) n<<=1, k++;
        for(int i=0; i<n; i++) rev[i] = (rev[i>>1]>>1) | ((i&1)<<(k-1));
    }
    void dft(long long *a, int flag) {
        for(int i=0; i<n; i++) if(i<rev[i]) swap(a[i], a[rev[i]]);
        for(int l=2; l<=n; l<<=1) {
            int m=l>>1;
            long long wn = Pow(g, flag==1 ? (P-1)/l : P-1-(P-1)/l, P);
            for(long long *p=a; p!=a+n; p+=l) {
                long long w=1;
                for(int k=0; k<m; k++) {
                    long long t = w * p[k+m]%P;
                    p[k+m]=(p[k]-t+P)%P;
                    p[k]=(p[k]+t)%P;
                    w=w*wn%P;
                }
            }
        }
        if(flag==-1) {
            long long inv=Pow(n, P-2, P);
            for(int i=0; i<n; i++) a[i]=a[i]*inv%P;
        }
    }
    void mul(long long *a, long long *b, int m) {
        ini(m);
        dft(a, 1); dft(b, 1);
        for(int i=0; i<n; i++) a[i]=a[i]*b[i];
        dft(a, -1);
    }
}f;

int n1, n2, m, c[N];
long long a[N], b[N];
char s1[N], s2[N];
int main()
{
  int n;
  scanf("%d",&n);
    scanf("%s%s",s1,s2);
    n1=strlen(s1); n2=strlen(s2);
    for(int i=0;i<n1;i++)
    a[i]=s1[n1-i-1]-'0';
    for(int i=0;i<n2;i++)
    b[i]=s2[n2-i-1]-'0';
    m=n1+n2-1;
    f.mul(a,b,m);
    for(int i=0;i<m;i++) c[i]=a[i];
    for(int i=0;i<m;i++) c[i+1]+=c[i]/10, c[i]%=10;
    if(c[m])
    m++;
    for(int i=m-1; i>=0; i--)
    printf("%d",c[i]);
```

```
  return 0;
}
```

2.3 SAM

```
#include<iostream>
#include<cstring>
using namespace std;
const int MaxPoint=1010101;
struct Suffix_AutoMachine{
  int son[MaxPoint][27],pre[MaxPoint],step[MaxPoint],right[MaxPoint],last,root,num;
  int NewNode(int stp)
  {
    num++;
    memset(son[num],0,sizeof(son[num]));
    pre[num]=0;
    step[num]=stp;
    return num;
  }
  Suffix_AutoMachine()
  {
    num=0;
    root=last=NewNode(0);
  }
  void push_back(int ch)
  {
    int np=NewNode(step[last]+1);
    right[np]=1;
    step[np]=step[last]+1;
    int p=last;
    while(p&&!son[p][ch])
    {
      son[p][ch]=np;
      p=pre[p];
    }
    if(!p)
      pre[np]=root;
    else
    {
      int q=son[p][ch];
      if(step[q]==step[p]+1)
        pre[np]=q;
      else
      {
        int nq=NewNode(step[p]+1);
        memcpy(son[nq],son[q],sizeof(son[q]));
        step[nq]=step[p]+1;
        pre[nq]=pre[q];
        pre[q]=pre[np]=nq;
        while(p&&son[p][ch]==q)
        {
          son[p][ch]=nq;
          p=pre[p];
        }
      }
    }
    last=np;
  }
};
/*

int arr[1010101];
bool Step_Cmp(int x,int y)
{
  return S.step[x]<S.step[y];
}
void Get_Right()
{
  for(int i=1;i<=S.num;i++)
```

```
    arr[i]=i;
  sort(arr+1,arr+S.num+1,Step_Cmp);
  for(int i=S.num;i>=2;i--)
    S.right[S.pre[arr[i]]]+=S.right[arr[i]];
}
*/
int main()
{

  return 0;
}
```

2.4 manacher

```
#include<iostream>
#include<cstring>
using namespace std;
char Mana[202020];
int cher[202020];
int Manacher(char *S)
{
  int len=strlen(S),id=0,mx=0,ret=0;
  Mana[0]='$';
  Mana[1]='#';
  for(int i=0;i<len;i++)
  {
    Mana[2*i+2]=S[i];
    Mana[2*i+3]='#';
  }
  Mana[2*len+2]=0;
  for(int i=1;i<=2*len+1;i++)
  {
    if(i<mx)
      cher[i]=min(cher[2*id-i],mx-i);
    else
      cher[i]=0;
    while(Mana[i+cher[i]+1]==Mana[i-cher[i]-1])
      cher[i]++;
    if(cher[i]+i>mx)
    {
      mx=cher[i]+i;
       id=i;
    }
    ret=max(ret,cher[i]);
  }
  return ret;
}
char S[101010];
int main()
{
  ios::sync_with_stdio(false);
  cin.tie(0);
  cout.tie(0);
  cin>>S;
  cout<<Manacher(S)<<endl;
  return 0;
}
```

2.5 中国剩余定理

```
// 51nod 1079
#include<iostream>
using namespace std;
int gcd(int x,int y)
{
  if(x==0)
    return y;
  if(y==0)
```

```
    return x;
  return gcd(y,x%y);
}
long long exgcd(long long a,long long b,long long &x,long long &y)
{
    if(b==0)
    {
        x=1;
        y=0;
        return a;
    }
    long long ans=exgcd(b,a%b,x,y);
    long long temp=x;
    x=y;
    y=temp-a/b*y;
    return ans;
}
void fix(long long &x,long long &y)
{
  x%=y;
  if(x<0)
    x+=y;
}
bool solve(int n, std::pair<long long, long long> input[],std::pair<long long, long long>
 ↪  &output)
{
  output = std::make_pair(1, 1);
  for(int i = 0; i < n; ++i)
  {
    long long number, useless;
    exgcd(output.second, input[i].second, number, useless);
    long long divisor = gcd(output.second, input[i].second);
    if((input[i].first - output.first) % divisor)
    {
      return false;
    }
    number *= (input[i].first - output.first) / divisor;
    fix(number,input[i].second);
    output.first += output.second * number;
    output.second *= input[i].second / divisor;
    fix(output.first, output.second);
  }
  return true;
}
pair<long long,long long> input[101010],output;
int main()
{
  int n;
  cin>>n;
  for(int i=0;i<n;i++)
    cin>>input[i].second>>input[i].first;
  solve(n,input,output);
  cout<<output.first<<endl;
  return 0;
}
```

2.6 回文自动机

```
//Tsinsen A1280 最长双回文串
#include<iostream>
#include<cstring>
using namespace std;

const int maxn = 100005;// n(空间复杂度 o(n*ALP)), 实际开 n 即可
const int ALP = 26;

struct PAM{ // 每个节点代表一个回文串
    int next[maxn][ALP]; // next 指针, 参照 Trie 树
    int fail[maxn]; // fail 失配后缀链接
```

```
12      int cnt[maxn]; // 此回文串出现个数
13      int num[maxn];
14      int len[maxn]; // 回文串长度
15      int s[maxn]; // 存放添加的字符
16      int last; //指向上一个字符所在的节点，方便下一次 add
17      int n; // 已添加字符个数
18      int p; // 节点个数
19
20      int newnode(int w)
21    {// 初始化节点，w= 长度
22          for(int i=0;i<ALP;i++)
23              next[p][i] = 0;
24          cnt[p] = 0;
25          num[p] = 0;
26          len[p] = w;
27          return p++;
28      }
29      void init()
30    {
31          p = 0;
32          newnode(0);
33          newnode(-1);
34          last = 0;
35          n = 0;
36          s[n] = -1; // 开头放一个字符集中没有的字符，减少特判
37          fail[0] = 1;
38      }
39      int get_fail(int x)
40    { // 和 KMP 一样，失配后找一个尽量最长的
41          while(s[n-len[x]-1] != s[n]) x = fail[x];
42          return x;
43      }
44      int add(int c)
45    {
46          c -= 'a';
47          s[++n] = c;
48          int cur = get_fail(last);
49          if(!next[cur][c])
50      {
51              int now = newnode(len[cur]+2);
52              fail[now] = next[get_fail(fail[cur])][c];
53              next[cur][c] = now;
54              num[now] = num[fail[now]] + 1;
55          }
56          last = next[cur][c];
57          cnt[last]++;
58          return len[last];
59      }
60      void count()
61    {
62          // 最后统计一遍每个节点出现个数
63          // 父亲累加儿子的 cnt，类似 SAM 中 parent 树
64          // 满足 parent 拓扑关系
65          for(int i=p-1;i>=0;i--)
66              cnt[fail[i]] += cnt[i];
67      }
68  }pam;
69  char S[101010];
70  int l[101010],r[101010];
71  int main()
72  {
73    cin>>S;
74    int len=strlen(S);
75    pam.init();
76    for(int i=0;i<len;i++)
77      l[i]=pam.add(S[i]);
78    pam.init();
79    for(int i=len-1;i>=0;i--)
80      r[i]=pam.add(S[i]);
81    pam.init();
```

```
  int ans=0;
  for(int i=0;i<len-1;i++)
    ans=max(ans,l[i]+r[i+1]);
  cout<<ans<<endl;
  return 0;
}
```

2.7 多项式开方

```
//    ²
//Nlog^2N
#include <cstdio>
#include <algorithm>
#define FOR(i,j,k) for(i=j;i<=k;++i)
#define rep(i,j,k) for(i=j;i<k;++i)
#define gmod(i) (((i)%mod+mod)%mod)
const int N = 262144, mod = 998244353, inv2 = 499122177;
using namespace std;
typedef long long ll;
ll qpow(ll x, int y) {
    ll z = 1;
    for (; y; x = x * x % mod, y /= 2)
        if (y & 1) z = z * x % mod;
    return z;
}
namespace NTT {
    int n, rev[N], inv_n, m = -1;
    void init(int c) {
        int k = -1, i;
        if (m == c) return; else m = c;
        for (n = 1; n <= m; n <<= 1) ++k;
        inv_n = qpow(n, mod - 2);
        rep(i,0,n) rev[i] = (rev[i >> 1] >> 1) | ((i & 1) << k);
    }
    void ntt(int *a, int f) {
        int h, i, j;
        rep(i,0,n) if (i < rev[i]) swap(a[i], a[rev[i]]);
        for (h = 2; h <= n; h *= 2) {
            int wn = qpow(3, (mod - 1) / h);
            for (i = 0; i < n; i += h) {
                int w = 1;
                rep(j,0,h/2) {
                    int u = a[i + j], t = 1ll * a[i + j + h / 2] * w % mod;
                    a[i + j + h / 2] = (u - t + mod) % mod;
                    a[i + j] = (u + t) % mod;
                    w = 1ll * w * wn % mod;
                }
            }
        }
        if (f) {
            rep(i,1,n/2) swap(a[i], a[n - i]);
            rep(i,0,n) a[i] = 1ll * a[i] * inv_n % mod;
        }
    }
}
void inv(int *a, int *b, int n) {
    static int t[N];
    int i;
    if (n == 1) { b[0] = qpow(a[0], mod - 2); return; }
    inv(a, b, n / 2);
    rep(i,0,n) t[i] = a[i]; rep(i,n,2*n) t[i] = 0;
    NTT::init(n);
    NTT::ntt(t, 0); NTT::ntt(b, 0);
    rep(i,0,NTT::n) t[i] = (ll) b[i] * gmod(2ll - (ll) t[i] * b[i] % mod) % mod;
    NTT::ntt(t, 1);
    rep(i,0,n) b[i] = t[i]; rep(i,n,2*n) b[i] = 0;
}
void sqrt(int *a, int *b, int n) {
    static int t[N], b1[N];
```

```
    if (n == 1) { b[0] = 1; return; }
    int i;
    sqrt(a, b, n / 2);
    rep(i,0,n) b1[i] = 0;
    inv(b, b1, n);
    rep(i,0,n) t[i] = a[i]; rep(i,n,2*n) t[i] = 0;
    NTT::init(n);
    NTT::ntt(t, 0), NTT::ntt(b, 0), NTT::ntt(b1, 0);
    rep(i,0,NTT::n) t[i] = inv2 * ((b[i] + (ll) b1[i] * t[i] % mod) % mod) % mod;
    NTT::ntt(t, 1);
    rep(i,0,n) b[i] = t[i]; rep(i,n,2*n) b[i] = 0;
}
int main() {
    static int c[N], sc[N], ic[N];
    int i, x, n, m, l;
    scanf("%d%d", &n, &m);
    FOR(i,1,n) scanf("%d", &x), ++c[x];
    c[0] = gmod(1 - c[0]);
    FOR(i,1,m) c[i] = gmod(-4 * c[i]);
    for (l = 1; l <= m; l <<= 1);
    sqrt(c, sc, l);
    (++sc[0]) %= mod;
    inv(sc, ic, l);
    FOR(i,0,m) ic[i] = 2ll * ic[i] % mod;
    FOR(i,1,m) printf("%d\n", ic[i]);
    return 0;
}
```

2.8 多项式求逆

```
//3  F bzoj3456
#include<iostream>
#include<cstdio>
#include<algorithm>
#include<cstring>
#include<cmath>
#define N 5000003
#define LL long long
#define p 1004535809
using namespace std;
int n,m;
int a[N],b[N],c[N],jc[N],inv_j[N],wn[N];
LL quickpow(LL num,LL x)
{
    LL base=num%p; LL ans=1;
    while (x) {
        if (x&1) ans=ans*base%p;
        x>>=1;
        base=base*base%p;
    }
    return ans;
}
void init()
{
    jc[0]=1; inv_j[0]=quickpow(jc[0],p-2);
    for (int i=1;i<=n;i++)
     jc[i]=(LL)jc[i-1]*i%p,inv_j[i]=quickpow(jc[i],p-2);
    for (int i=1;i<=n*8;i<<=1)
     wn[i]=quickpow(3,(p-1)/(i<<1));
}
void NTT(int n,int *a,int opt)
{
    for (int i=0,j=0;i<n;i++) {
        if (i>j) swap(a[i],a[j]);
        for (int l=n>>1;(j^=l)<l;l>>=1);
    }
    for (int i=1;i<n;i<<=1) {
        LL wn1=wn[i];
        for (int p1=i<<1,j=0;j<n;j+=p1) {
```

```
            LL w=1;
            for (int k=0;k<i;k++,w=(LL)w*wn1%p) {
                int x=a[j+k]; int y=(LL)a[j+k+i]*w%p;
                a[j+k]=(x+y)%p; a[j+k+i]=(x-y+p)%p;
            }
        }
    }
    if (opt==-1) reverse(a+1,a+n);
}
void inverse(int n,int *a,int *b,int *c)
{
    if (n==1) b[0]=quickpow(a[0],p-2);
    else {
        inverse((n+1)>>1,a,b,c);
        int k=0;
        for (k=1;k<=(n<<1);k<<=1);
        for (int i=0;i<n;i++) c[i]=a[i];
        for (int i=n;i<k;i++) c[i]=0;
        NTT(k,c,1);
        NTT(k,b,1);
        for (int i=0;i<k;i++) {
            b[i]=(LL)(2-(LL)c[i]*b[i]%p)*b[i]%p;
            if (b[i]<0) b[i]+=p;
        }
        NTT(k,b,-1);
        int inv=quickpow(k,p-2);
        for (int i=0;i<k;i++) b[i]=(LL)b[i]*inv%p;
        for (int i=n;i<k;i++) b[i]=0;
    }
}
int main()
{
    scanf("%d",&n); init();
    int n1=0;
    for (n1=1;n1<=n*2;n1<<=1);
    a[0]=1;
    for (int i=1;i<=n;i++) a[i]=(LL)quickpow(2,(LL)i*(i-1)/2)*inv_j[i]%p;
    inverse(n1,a,b,c);
    memset(c,0,sizeof(c));
    for (int i=1;i<=n;i++) c[i]=(LL)quickpow(2,(LL)i*(i-1)/2)*inv_j[i-1]%p;
    NTT(n1,b,1); NTT(n1,c,1);
    for (int i=0;i<=n1;i++) b[i]=(LL)b[i]*c[i]%p;
    NTT(n1,b,-1);
    LL inv=quickpow(n1,p-2);
    for (int i=0;i<=n1;i++) b[i]=(LL)b[i]*inv%p;
    printf("%d\n",(LL)b[n]*jc[n-1]%p);
    return 0;
}
```

2.9 广义 SAM

```
#include<iostream>
#include<cstring>
using namespace std;
const int MaxPoint=1010101;
struct Suffix_AutoMachine{
  int son[MaxPoint][27],pre[MaxPoint],step[MaxPoint],right[MaxPoint],root,num;
  int NewNode(int stp)
  {
    num++;
    memset(son[num],0,sizeof(son[num]));
    pre[num]=0;
    step[num]=stp;
    return num;
  }
  Suffix_AutoMachine()
  {
    num=0;
    root=NewNode(0);
```

```
  }
  int push_back(int ch,int p)
  {
    int np=NewNode(step[p]+1);
    right[np]=1;
    step[np]=step[p]+1;
    while(p&&!son[p][ch])
    {
      son[p][ch]=np;
      p=pre[p];
    }
    if(!p)
      pre[np]=root;
    else
    {
      int q=son[p][ch];
      if(step[q]==step[p]+1)
        pre[np]=q;
      else
      {
        int nq=NewNode(step[p]+1);
        memcpy(son[nq],son[q],sizeof(son[q]));
        step[nq]=step[p]+1;
        pre[nq]=pre[q];
        pre[q]=pre[np]=nq;
        while(p&&son[p][ch]==q)
        {
          son[p][ch]=nq;
          p=pre[p];
        }
      }
    }
    return np;
  }
};
int main()
{

  return 0;
}
```

2.10 循环串最小表示

```
int getmin( char s[] )
{
    int i , j , k , m , t ;
    m = strlen( s ) ;
    i = 0 ; j = 1 ; k = 0 ;
    while( i < m && j < m && k < m )
    {
        t = s[( i + k ) % m] - s[( j + k ) % m] ;
        if( !t )
            ++ k ;
        else
        {
            if( t > 0 )
                i += k + 1 ;
            else
                j += k + 1 ;
            if( i == j )
                j ++ ;
            k = 0 ;
        }

    }
    return min(i,j);
}
```

2.11 扩展欧几里得

```
long long exgcd(long long a,long long b,long long &x,long long &y)
{
    if(b==0)
    {
        x=1;
        y=0;
        return a;
    }
    long long ans=exgcd(b,a%b,x,y);
    long long temp=x;
    x=y;
    y=temp-a/b*y;
    return ans;
}
```

2.12 最大团搜索

```
#include<iostream>
using namespace std;
int ans;
int num[1010];
int path[1010];
int a[1010][1010],n;
bool dfs(int *adj,int total,int cnt)
{
    int i,j,k;
    int t[1010];
    if(total==0)
  {
        if(ans<cnt)
    {
            ans=cnt;
      return 1;
        }
        return 0;
    }
    for(i=0;i<total;i++)
  {
        if(cnt+(total-i)<=ans)
      return 0;
        if(cnt+num[adj[i]]<=ans)
      return 0;
        for(k=0,j=i+1;j<total;j++)
    if(a[adj[i]][adj[j]])
            t[k++]=adj[j];
        if(dfs(t,k,cnt+1))
      return 1;
    }
  return 0;
}
int MaxClique()
{
    int i,j,k;
    int adj[1010];
    if(n<=0)
    return 0;
    ans=1;
    for(i=n-1;i>=0;i--)
  {
        for(k=0,j=i+1;j<n;j++)
        if(a[i][j])
      adj[k++]=j;
        dfs(adj,k,1);
        num[i]=ans;
    }
    return ans;
}
```

```
int main()
{
  ios::sync_with_stdio(0);
  cin.tie(0);
  cout.tie(0);
  while(cin>>n)
  {
    if(n==0)
      break;
    for(int i=0;i<n;i++)
    for(int j=0;j<n;j++)
      cin>>a[i][j];
    cout<<MaxClique()<<endl;
  }
  return 0;
}
```

2.13 极大团计数

```
#include<cstdio>
#include<cstring>
using namespace std;
const int N=130;
int ans,a[N][N],R[N][N],P[N][N],X[N][N];
bool Bron_Kerbosch(int d,int nr,int np,int nx)
{
    int i,j;
    if(np==0&&nx==0)
    {
        ans++;
        if(ans>1000)//
            return 1;
        return 0;
    }
    int u,max=0;
    u=P[d][1];
    for(i=1;i<=np;i++)
    {
        int cnt=0;
        for(j=1;j<=np;j++)
        {
            if(a[P[d][i]][P[d][j]])
                cnt++;
        }
        if(cnt>max)
        {
            max=cnt;
            u=P[d][i];
        }
    }
    for(i=1;i<=np;i++)
    {
        int v=P[d][i];
        if(a[v][u]) continue;
        for(j=1;j<=nr;j++)
            R[d+1][j]=R[d][j];
        R[d+1][nr+1]=v;
        int cnt1=0;
        for(j=1;j<=np;j++)
            if(P[d][j]&&a[P[d][j]][v])
                P[d+1][++cnt1]=P[d][j];
        int cnt2=0;
        for(j=1;j<=nx;j++)
            if(a[X[d][j]][v])
                X[d+1][++cnt2]=X[d][j];
        if(Bron_Kerbosch(d+1,nr+1,cnt1,cnt2))
            return 1;
        P[d][i]=0;
        X[d][++nx]=v;
```

```
    }
    return 0;
}
int main()
{
    int n,i,m,x,y;
    while(scanf("%d%d",&n,&m)!=EOF)
    {
        memset(a,0,sizeof(a));
        while(m--)
        {
            scanf("%d%d",&x,&y);
            a[x][y]=a[y][x]=1;
        }
        ans=0;
        for(i=1;i<=n;i++)
            P[1][i]=i;
        Bron_Kerbosch(1,0,n,0);
        if(ans>1000)
            printf("Too many maximal sets of friends.\n");
        else
            printf("%d\n",ans);
    }
    return 0;
}
```

2.14 求原根

```
//51Nod - 1135
#include <iostream>
#include <string.h>
#include <algorithm>
#include <stdio.h>
#include <math.h>
#include <bitset>

using namespace std;
typedef long long LL;

const int N = 1000010;

bitset<N> prime;
int p[N],pri[N];
int k,cnt;

void isprime()
{
    prime.set();
    for(int i=2; i<N; i++)
    {
        if(prime[i])
        {
            p[k++] = i;
            for(int j=i+i; j<N; j+=i)
                prime[j] = false;
        }
    }
}

void Divide(int n)
{
    cnt = 0;
    int t = (int)sqrt(1.0*n);
    for(int i=0; p[i]<=t; i++)
    {
        if(n%p[i]==0)
        {
            pri[cnt++] = p[i];
            while(n%p[i]==0) n /= p[i];
```

```
        }
    }
    if(n > 1)
        pri[cnt++] = n;
}

LL quick_mod(LL a,LL b,LL m)
{
    LL ans = 1;
    a %= m;
    while(b)
    {
        if(b&1)
        {
            ans = ans * a % m;
            b--;
        }
        b >>= 1;
        a = a * a % m;
    }
    return ans;
}

int main()
{
    int P;
    isprime();
    while(cin>>P)
    {
        Divide(P-1);
        for(int g=2; g<P; g++)
        {
            bool flag = true;
            for(int i=0; i<cnt; i++)
            {
                int t = (P - 1) / pri[i];
                if(quick_mod(g,t,P) == 1)
                {
                    flag = false;
                    break;
                }
            }
            if(flag)
            {
                int root = g;
                cout<<root<<endl;
        break;
            }
        }
    }
    return 0;
}
```

2.15 线性递推多项式

```
//³μ¸
void linear_recurrence(long long n, int m, int a[], int c[], int p)
{
  long long v[M] = {1 % p},u[M << 1], msk = !!n;
  for(long long i(n); i > 1; i >>= 1)
  {
    msk <<= 1;
  }
  for(long long x(0); msk; msk >>= 1, x <<= 1)
  {
    fill_n(u, m << 1, 0);
    int b(!!(n & msk));
    x |= b;
    if(x < m)
```

```
15      {
16        u[x] = 1 % p;
17      }
18      else
19      {
20        for(int i(0); i < m; i++)
21        {
22          for(int j(0), t(i + b); j < m; j++, t++)
23          {
24            u[t] = (u[t] + v[i] * v[j]) % p;
25          }
26        }
27        for(int i((m << 1) - 1); i >= m; i--)
28        {
29          for(int j(0), t(i - m); j < m; j++, t++)
30          {
31            u[t] = (u[t] + c[j] * u[i]) % p;
32          }
33        }
34      }
35      copy(u, u + m, v);
36    }
37    for(int i(m); i < 2 * m; i++)
38    {
39      a[i] = 0;
40      for(int j(0); j < m; j++)
41      {
42        a[i] = (a[i] + (long long)c[j] * a[i + j - m]) % p;
43      }
44    }
45    for(int j(0); j < m; j++)
46    {
47      b[j] = 0;
48      for(int i(0); i < m; i++)
49      {
50        b[j] = (b[j] + v[i] * a[i + j]) % p;
51      }
52    }
53    for(int j(0); j < m; j++)
54    {
55      a[j] = b[j];
56    }
57  }
```

Chapter 3

尧尧

3.1 bcc

```cpp
#include <cstdio>
#include <vector>
using namespace std;

const int N = 1000+10;
const int M = N*N;

struct Edge {
    int u, v;
    Edge( int u, int v ):u(u),v(v){}
};

int n, m;
int head[N], dest[M], next[M], etot;
int dfn[N], low[N], bccno[N], iscut[N], bcc_cnt, idc;
vector<int> bcc[N];
vector<Edge> stk;

void adde( int u, int v ) {
    etot++;
    dest[etot] = v;
    next[etot] = head[u];
    head[u] = etot;
}
void dfs( int u, int fa ) {
    dfn[u] = low[u] = ++idc;
    int child = 0;
    for( int t=head[u]; t; t=next[t] ) {
        int v=dest[t];
        if( v==fa ) continue;

        if( !dfn[v] ) {
            stk.push_back( Edge(u,v) );
            dfs(v,u);
            low[u] = min( low[u], low[v] );
            child++;
            if( low[v]>=dfn[u] ) {
                iscut[u] = true;
                bcc_cnt++;
                while(1) {
                    Edge e=stk.back();
                    stk.pop_back();
                    if( !bccno[e.u] ) bccno[e.u]=bcc_cnt,bcc[bcc_cnt].push_back(e.u);
                    if( !bccno[e.v] ) bccno[e.v]=bcc_cnt,bcc[bcc_cnt].push_back(e.v);
                    if( e.u==u && e.v==v ) break;
                }
            }
        } else if( dfn[v]<dfn[u] ) {
            low[u] = min( low[u], dfn[v] );
        }
    }
```

```
    if( u==fa && child<=1 ) iscut[u]=false;
}
int main() {
    scanf( "%d%d", &n, &m );
    for( int i=1,u,v; i<=m; i++ ) {
        scanf( "%d%d", &u, &v );
        adde( u, v );
        adde( v, u );
    }
    dfs(1,1);
    for( int u=1; u<=n; u++ )
        if( iscut[u] ) printf( "%d ", u );
    printf( "\n" );
}
```

3.2 fft

```
#include <bits/stdc++.h>
using namespace std;

struct FFT {
  typedef complex<double> Complex;
  Complex w[33];

  FFT() {
    double dpi = 2.0 * acos(-1);
    for(int p = 0; p <= 30; p++)
      w[p] = Complex(cos(dpi/(1<<p)),sin(dpi/(1<<p)));
  }
  int reverse( int pmax, int a ) {
    int b = 0;
    for( int i=0; i<pmax; i++ )
      if( a&(1<<i) ) b |= 1<<(pmax-1-i);
    return b;
  }
  void fft( vector<Complex> &a, int pmax, bool r ) {
    int n = (int)a.size();;
    for( int i=0; i<n; i++ ) {
      int j=reverse(pmax,i);
      if( i<j ) swap(a[i],a[j]);
    }
    for( int p=1; p<=pmax; p++ ) {
      for( int i=0; i<n; i+=(1<<p) ) {
        Complex wo, wk;
        int l = 1<<(p-1);
        if( !r ) wo = w[p];
        else wo = conj(w[p]);
        wk = w[0];
        for( int j=0; j<l; j++,wk=wk*wo ) {
          Complex lf = a[i+j], rg = a[i+j+l];
          a[i+j] = lf + wk*rg;
          a[i+j+l] = lf - wk*rg;
        }
      }
    }
    if( r ) for( int i=0; i<n; i++ )
      a[i].real(a[i].real()/n);
  }
  vector<double> multiply( vector<double> a, vector<double> b ) {
    int pmax;
    for( pmax = 0; a.size() > (1u<<pmax) || b.size() > (1u<<pmax); pmax++ );
    pmax++;
    vector<Complex> ca, cb, cc;
    for( int t = 0; t < (int)a.size(); t++ )
      ca.push_back(Complex(a[t],0));
    for( int t = 0; t < (int)b.size(); t++ )
      cb.push_back(Complex(b[t],0));
    ca.resize(1<<pmax);
    cb.resize(1<<pmax);
```

```
53      cc.resize(1<<pmax);
54      fft( ca, pmax, 0 );
55      fft( cb, pmax, 0 );
56      for( int t = 0; t < (int)cc.size(); t++ )
57        cc[t] = ca[t] * cb[t];
58      fft( cc, pmax, 1 );
59      vector<double> c;
60      for( int t = 0; t < (int)cc.size(); t++ )
61        c.push_back(cc[t].real());
62      return c;
63    }
64  }fft;
65
66  const int N = 500000 + 10;
67
68  int n;
69  char ss[N];
70  vector<double> va, vb, vc;
71  bool ans[N];
72
73  int main() {
74    int T;
75    scanf("%d", &T);
76    for( int cas = 1; cas <= T; cas++ ) {
77      scanf("%d", &n);
78      scanf("%s", ss);
79      va.resize(n), vb.resize(n);
80      for(int i = 0; i < n; i++) {
81        if(ss[i] == 'V') {
82          va[i] = 1.0;
83          vb[i] = 0.0;
84        } else if(ss[i] == 'K') {
85          va[i] = 0.0;
86          vb[i] = 1.0;
87        } else {
88          va[i] = vb[i] = 0.0;
89        }
90      }
91      reverse(va.begin(), va.end());
92      vc = fft.multiply(va, vb);
93
94      for(int i = 0; i <= n; i++ )
95        ans[i] = false;
96      for(int i = 0; i < n + n - 1; i++) {
97        bool exist = fabs(vc[i]) > 0.4;
98        int dif = abs(n - 1 - i);
99        if(exist) ans[dif] = true;
100     }
101     for(int i = 1; i < n; i++) {
102       for(int j = i + i; j < n; j += i) {
103         if(ans[j]) {
104           ans[i] = true;
105           break;
106         }
107       }
108     }
109     int tot = 0;
110     for(int i = 1; i < n; i++)
111       tot += (ans[i] == false);
112     printf("%d\n", tot + 1);
113     for(int i = 1; i < n; i++)
114       if(ans[i] == false) printf("%d ", i);
115     printf("%d\n", n);
116   }
117 }
```

3.3 hungary

```
const int N = 1000 + 10;

int n, m;
int match[N], visit[N], stamp;
vector<int> edge[N];

bool dfs( int u ) {
  for( int t = 0; t < (int)edge[u].size(); t++ ) {
    int v = edge[u][t];
    if( visit[v] == stamp ) continue;
    visit[v] = stamp;
    if( match[v] == 0 || dfs(match[v]) ) {
      match[v] = u;
      return true;
    }
  }
  return false;
}
int hungary() {
  int ans = 0;
  memset( visit, 0, sizeof(visit) );
  memset( match, 0, sizeof(match) );
  stamp = 0;
  for( int u = 1; u <= n; u++ ) {
    ++stamp;
    ans += dfs( u );
  }
  return ans;
}
```

3.4 kth.shortest.path

```
#include <cstdio>
#include <cstring>
#include <queue>
using namespace std;

const int N = 1010;
const int M = 100010;
const int oo = 0x3f3f3f3f;

struct Elist {
  int _head[N], _dest[M], _dist[M], _last[M], etot;
  inline void adde( int u, int v, int d ) {
    etot++;
    _dest[etot] = v;
    _dist[etot] = d;
    _last[etot] = _head[u];
    _head[u] = etot;
  }
  inline int head( int u ) { return _head[u]; }
  inline int dest( int t ) { return _dest[t]; }
  inline int dist( int t ) { return _dist[t]; }
  inline int last( int t ) { return _last[t]; }
};
struct Stat {
  int u, d;
  Stat(){}
  Stat( int u, int d ):u(u),d(d){}
};

int n, m, K;
Elist e, re;
int src, dst;
int rdis[N];
bool done[N];

```

```
36  bool operator<( const Stat &r, const Stat &s ) {
37    return r.d + rdis[r.u] > s.d + rdis[s.u];
38  }
39  void dijkstra() {
40    memset( done, false, sizeof(done) );
41    memset( rdis, 0x3f, sizeof(rdis) );
42    priority_queue<pair<int,int> > Q;
43    Q.push( make_pair(0,dst) );
44    rdis[dst] = 0;
45    while( !Q.empty() ) {
46      int u = Q.top().second;
47      Q.pop();
48      if( done[u] ) continue;
49      done[u] = true;
50      for( int t = re.head(u); t; t = re.last(t) ) {
51        int v = re.dest(t), d = re.dist(t);
52        if( done[v] ) continue;
53        if( rdis[v] > rdis[u] + d ) {
54          rdis[v] = rdis[u] + d;
55          Q.push( make_pair( -rdis[v], v ) );
56        }
57      }
58    }
59  }
60  int astar() {
61    int pcnt = 0;
62    priority_queue<Stat> Q;
63
64    if( rdis[src] == oo ) return -1;
65    if( src == dst ) K++;
66    Q.push( Stat( src, 0 ) );
67    while( !Q.empty() ) {
68      Stat s = Q.top();
69      Q.pop();
70      if( s.u == dst ) {
71        pcnt++;
72        if( pcnt == K )
73          return s.d;
74      }
75      for( int t = e.head(s.u); t; t = e.last(t) ) {
76        int v = e.dest(t), d = e.dist(t);
77        if( rdis[v] == oo ) continue;
78        Q.push( Stat( v, s.d + d ) );
79      }
80    }
81    return -1;
82  }
83  int main() {
84    scanf( "%d%d", &n, &m );
85    for( int i = 1; i <= m; i++ ) {
86      int u, v, d;
87      scanf( "%d%d%d", &u, &v, &d );
88      e.adde( u, v, d );
89      re.adde( v, u, d );
90    }
91    scanf( "%d%d%d", &src, &dst, &K );
92    dijkstra();
93    printf( "%d\n", astar() );
94  }
```

3.5 mobius

```
1  /*************************************************************
2      Problem: 3529
3      User: idy002
4      Language: C++
5      Result: Accepted
6      Time:3568 ms
7      Memory:3192 kb
```

```
8   ************************************************************/
9
10  #include <cstdio>
11  #include <algorithm>
12  using namespace std;
13
14  struct Query {
15      int n, m, a, id;
16      bool operator<( const Query & b ) const {
17          return a<b.a;
18      }
19  };
20
21  int prm[10000], isnot[100010], mu[100010], f[100010], h[100010], ptot;
22  int order[100010], cur;
23  int q, ans[20010];
24  Query qry[20010];
25
26  bool cmp( int a, int b ) { return f[a]<f[b]; }
27  void modify( int pos, int delta ) {
28      for( int i=pos; i<=100000; i+=i&-i )
29          h[i] += delta;
30  }
31  int query( int pos ) {
32      int rt = 0;
33      for( int i=pos; i; i-=i&-i )
34          rt += h[i];
35      return rt;
36  }
37  void init( int n ) {
38      mu[1] = 1;
39      for( int i=2; i<=n; i++ ) {
40          if( !isnot[i] ) {
41              prm[++ptot] = i;
42              mu[i] = -1;
43          }
44          for( int j=1; j<=ptot && i*prm[j]<=n; j++ ) {
45              isnot[i*prm[j]] = true;
46              if( i%prm[j]==0 ) {
47                  mu[i*prm[j]] = 0;
48                  break;
49              }
50              mu[i*prm[j]] = -mu[i];
51          }
52      }
53      for( int i=1; i<=n; i++ )
54          for( int j=i; j<=n; j+=i )
55              f[j] += i;
56      for( int i=1; i<=n; i++ )
57          order[i] = i;
58      sort( order+1, order+1+n, cmp );
59  }
60
61  int main() {
62      init(100000);
63      scanf( "%d", &q );
64      for( int i=1; i<=q; i++ ) {
65          scanf( "%d%d%d", &qry[i].n, &qry[i].m, &qry[i].a );
66          if( qry[i].n>qry[i].m ) swap( qry[i].n, qry[i].m );
67          qry[i].id = i;
68      }
69      sort( qry+1, qry+1+q );
70      for( int i=1; i<=q; i++ ) {
71          while( cur+1<=100000 && f[order[cur+1]]<=qry[i].a ) {
72              cur++;
73              int j=order[cur];
74              for( int k=j; k<=100000; k+=j )
75                  modify( k, f[j]*mu[k/j] );
76          }
77          int &tans = ans[qry[i].id];
```

```
        int n = qry[i].n, m = qry[i].m;
        for( int j=1; j<=n; j++ ) {
            int jj=min( n/(n/j), m/(m/j) );
            tans += (query(jj)-query(j-1))*(n/j)*(m/j);
            j = jj;
        }
    }
    for( int i=1; i<=q; i++ )
        printf( "%d\n", ans[i]&((1U<<31)-1) );
}
```

3.6 ntt

```
#include <bits/stdc++.h>
using namespace std;

struct NTT {
  int mod, g, maxp;
  int w[33], rw[33], rb[33];

  vector<int> pfactor(int a) {
    if(a <= 1) return vector<int>();
    vector<int> pfac;
    for(int d = 2; d * d <= a; d++) {
      if(a % d == 0) {
        pfac.push_back(d);
        do a /= d;
        while(a % d == 0);
      }
    }
    if(a != 1) pfac.push_back(a);
    return pfac;
  }
  int findroot(int mod) {
    if(mod == 2) return 1;
    vector<int> pfac = pfactor(mod - 1);
    for(int g = 2; ; g++) {
      bool ok = true;
      for(int p : pfac) {
        if(mpow(g, (mod - 1) / p, mod) == 1) {
          ok = false;
          break;
        }
      }
      if(ok) return g;
    }
  }
  int mpow(int a, int b, int mod) {
    int rt;
    for(rt = 1; b; b>>=1,a=(1LL*a*a)%mod)
      if(b & 1) rt=(1LL*rt*a)%mod;
    return rt;
  }
  void setmod(int m) {
    mod = m;
    g = findroot(mod);
    maxp = 0;
    for(int t = mod - 1; (t & 1) == 0; t >>= 1)
      maxp++;
    w[maxp] = mpow(g, (mod - 1) / (1 << maxp), mod);
    for(int p = maxp - 1; p >= 0; p--)
      w[p] = 1LL * w[p + 1] * w[p + 1] % mod;
    for(int p = 0; p <= maxp; p++) {
      rw[p] = mpow(w[p], mod - 2, mod);
      rb[p] = mpow(1<<p, mod - 2, mod);
    }
  }
  NTT() {}
  int reverse(int pmax, int a) {
```

```
57      int b = 0;
58      for( int i=0; i<pmax; i++ )
59        if( a&(1<<i) ) b |= 1<<(pmax-1-i);
60      return b;
61    }
62    void ntt(vector<int> &a, int pmax, bool r) {
63      int n = (int)a.size();
64      for(int i=0,j=0; i<n; i++){
65        if(i>j) swap(a[i],a[j]);
66        for(int l=n>>1;(j^=l)<l;l>>=1);
67      }
68      /*
69      for(int i = 0; i < n; i++) {
70        int j = reverse(pmax, i);
71        if(i < j) swap(a[i], a[j]);
72      }
73      */
74      for(int p = 1; p <= pmax; p++) {
75        for(int i = 0; i < n; i += (1<<p)) {
76          int wo, wk;
77          int l = 1<<(p-1);
78          if(!r) wo = w[p];
79          else wo = rw[p];
80          wk = w[0];
81          for(int j = 0; j < l; j++,wk=1LL*wk*wo%mod) {
82            int lf = a[i+j], rg = a[i+j+l];
83            int val = 1LL * wk * rg % mod;
84            a[i+j] = (lf + val) % mod;
85            a[i+j+l] = (lf + mod - val) % mod;
86          }
87        }
88      }
89      if(r) for(int i = 0; i < n; i++)
90        a[i] = 1LL * a[i] * rb[pmax] % mod;
91    }
92    vector<int> multiply(vector<int> ca, vector<int> cb) {
93      int pmax;
94      for( pmax = 0; ca.size() > (1u<<pmax) || cb.size() > (1u<<pmax); pmax++ );
95      pmax++;
96      if(pmax > maxp) { assert("check this" == 0); }
97      vector<int> cc;
98      ca.resize(1<<pmax);
99      cb.resize(1<<pmax);
100     cc.resize(1<<pmax);
101     ntt( ca, pmax, 0 );
102     ntt( cb, pmax, 0 );
103     for( int t = 0; t < (int)cc.size(); t++ )
104       cc[t] = 1LL * ca[t] * cb[t] % mod;
105     ntt( cc, pmax, 1 );
106     return cc;
107   }
108 }ntt;
109
110 int main() {
111   int n, m;
112   vector<int> a, b, c;
113   scanf("%d%d", &n, &m);
114   a.resize(n + 1);
115   b.resize(m + 1);
116   for(int i = 0; i <= n; i++)
117     scanf("%d", &a[i]);
118   for(int i = 0; i <= m; i++)
119     scanf("%d", &b[i]);
120   ntt.setmod((479<<21) + 1);
121   c = ntt.multiply(a,b);
122   for(int i = 0; i <= n + m; i++)
123     printf("%d%c", c[i], " \n"[i == n + m]);
124 }
```

3.7 sam

```
/*
后缀自动机是一种确定性有限自动机 (DFA), 它可以且仅可以匹配一个给定串的任意后缀。

构造一个可以接受一个给定串的所有后缀的不确定性有限自动机 (NFA) 是很容易的, 我们发现我们用通用的将 NFA 转换成对应 DFA

后缀自动机的增量算法:
*/
struct Sam {
    int son[S][26], val[S], pnt[S], ntot, last;
    Sam() { pnt[0] = -1; }
    void append( int c ) {
        int p = last;
        int np = ++ntot;
        val[np] = val[p]+1;
        while( p!=-1 && !son[p][c] )
            son[p][c]=np, p=pnt[p];
        if( p==-1 ) {
            pnt[np] = 0;
        } else {
            int q=son[p][c];
            if( val[q]==val[p]+1 ) {
                pnt[np] = q;
            } else {
                int nq = ++ntot;
                memcpy( son[nq], son[q], sizeof(son[nq]) );
                val[nq] = val[p]+1;
                pnt[nq] = pnt[q];
                pnt[q] = pnt[np] = nq;
                while( p!=-1 && son[p][c]==q )
                    son[p][c]=nq, p=pnt[p];
            }
        }
        last = np;
    }
};
/*
后缀自动机构造完成之后, 我们得到了 4 个东西: 转移 DAG, Parent 树, 每个点的 right 集合,
每个点的字符串集合的长度区间。(其中最后一个可以由地一个 DP 出来)

转移 DAG 最直接的用处是子串判定问题, 因为它将原串的所有子串唯一的映射到了该 DAG 上的某个节点,
并且将该子串放到 DAG 上跑, 会跑到该节点。每个节点可能代表多个串。

可以在 DAG 上进行 DP, 得出从某点开始最多匹配多少个本质不同的子串, 如果再加上 right 集合,
就可以算出普通字串个数 (位置不同本质相同也算)。

转移 DAG 加上 Parent 树可以提供给我们访问原串某个子串的所有子串的能力。

对于处理多个串的问题, 我们可以先用分割符连接各个串, 然后构造后缀自动机, 并且重新定义一个点代表的字串:
原本的串去除带有分割符的串, 我们可以 DP 算出每个节点代表的串的数量以及长度区间。
这样可以通过刚才访问子串的子串的方法访问一个串的所有子串。
*/
```

3.8 scc

```
#include <cstdio>
#include <cstring>
#include <vector>
#define maxn 10010
using namespace std;

int n, m;
vector<int> g[maxn], gg[maxn], scc[maxn], stk;
int dfn[maxn], low[maxn], sccno[maxn], scc_cnt, dfs_clock;
int indgr[maxn];


void dfs( int u ) {
```

```
  dfn[u] = low[u] = ++dfs_clock;
  stk.push_back( u );
  for( int t=0; t<g[u].size(); t++ ) {
    int v=g[u][t];
    if( !dfn[v] ) {
      dfs(v);
      low[u] = min( low[u], low[v] );
    } else if( !sccno[v] )
      low[u] = min( low[u], dfn[v] );
  }
  if( dfn[u]==low[u] ) {
    scc_cnt++;
    while(1) {
      int v = stk.back();
      sccno[v] = scc_cnt;
      scc[scc_cnt].push_back(v);
      stk.pop_back();
      if( v==u ) break;
    }
  }
}
int main() {
  scanf( "%d%d", &n, &m );
  for( int i=1,u,v; i<=m; i++ ) {
    scanf( "%d%d", &u, &v );
    g[v].push_back(u);
  }
  for( int i=1; i<=n; i++ )
    if( !dfn[i] ) dfs(i);
  for( int u=1; u<=n; u++ ) {
    for( int t=0; t<g[u].size(); t++ ) {
      int v = g[u][t];
      if( sccno[u]!=sccno[v] )
        indgr[sccno[v]]++;
    }
  }
  int ans, cnt=0;
  for( int i=1; i<=scc_cnt; i++ )
    if( indgr[i]==0 ) {
      cnt++;
      if( cnt>1 ) {
        printf( "0\n" );
        return 0;
      }
      ans = (int)scc[i].size();
    }
  printf( "%d\n", ans );
}
```

3.9 伪虚树

```
/*
第一道"虚树"题目（好吧，我也不知道这是不是虚树，但和虚树的思想肯定是一样的，都是简化树结构）

这一类算法核心思想都是简化树结构，只取我们必须的节点和一些信息，然后在简化后的树结构上工作。

首先，如果这道题只有一次询问，那么很容易想到树形 DP 的解法，但这道题又多组询问，并且限制了所有询问的关键点个数，
这意味着我们必须设计出一种算法，她回答一组询问的复杂度只和关键点个数相关（O(k) 或 O(klogk) 都是可接受的），而和原图无

然后就有了虚树，我们可以构建一个新的树，这棵树上有所有关键点，以及相邻 dfs 序上相邻的两个关键点的 lca，我们发现，
这样的图包括了所有关键点的 lca 以及所有关键点，然后改一下 DP 就可以在这棵树上快速的搞了（因为节点个数是 O(2*k)，
所以这样 DP 的复杂度就从 O(n) 变成了 O(k)）。
DP:

dp[i] 表示将 i 及其子树中所有关键点与跟节点断开所需的最小代价（可以砍他们上面的边）

构简化图：

对于一个询问，我们先将其关键点按照 DFS 序排序。然后维护一个栈保存当前走了的关键点或关键点之间的 LCA，
```

```
19  当我们要插入一个新的关键节点时，我们根据当前节点与当前栈顶节点 LCA 的深度与栈顶元素的深度来判断是否需要弹出节点，一直
20  直到深度大于等于栈顶元素（注意，这个 LCA 是最初的栈顶与新的关键节点的 LCA）
21  */
22  /**************************************************************
23      Problem: 2286
24      User: idy002
25      Language: C++
26      Result: Accepted
27      Time:6700 ms
28      Memory:32060 kb
29  ****************************************************************/
30
31  #include <cstdio>
32  #include <vector>
33  #include <algorithm>
34  #define min(a,b) ((a)<(b)?(a):(b))
35  #define oo 0x3f3f3f3f
36  #define N 250010
37  #define P 17
38  using namespace std;
39
40  typedef long long dnt;
41
42  int n, m;
43  int head[N], dest[N+N], wght[N+N], next[N+N], ntot;
44  int dfn[N], dep[N], bst[N], anc[N][P+1], idc;
45  int qcnt, aa[N], stk[N], top;
46  dnt dp[N];
47
48  void adde( int u, int v, int w ) {
49      ntot++;
50      wght[ntot] = w;
51      dest[ntot] = v;
52      next[ntot] = head[u];
53      head[u] = ntot;
54  }
55  void dfs( int u ) {
56      dfn[u] = ++idc;
57      for( int p=1; p<=P; p++ )
58          anc[u][p] = anc[anc[u][p-1]][p-1];
59      for( int t=head[u]; t; t=next[t] ) {
60          int v=dest[t], w=wght[t];
61          if( v==anc[u][0] ) continue;
62          anc[v][0] = u;
63          bst[v] = min( bst[u], w );
64          dep[v] = dep[u]+1;
65          dfs(v);
66      }
67  }
68  bool cmp( int u, int v ) {
69      return dfn[u]<dfn[v];
70  }
71  int lca( int u, int v ) {
72      if( dep[u]<dep[v] ) swap(u,v);
73      int t=dep[u]-dep[v];
74      for( int p=0; t; t>>=1,p++ )
75          if( t&1 ) u=anc[u][p];
76      if( u==v ) return u;
77      for( int p=P; p>=0 && anc[u][0]!=anc[v][0]; p-- )
78          if( anc[u][p]!=anc[v][p] )
79              u=anc[u][p], v=anc[v][p];
80      return anc[u][0];
81  }
82  void sov() {
83      scanf( "%d", &qcnt );
84      for( int i=1; i<=qcnt; i++ )
85          scanf( "%d", aa+i );
86      sort( aa+1, aa+1+qcnt, cmp );
87
88      stk[top=1] = 1;
```

```
    dp[1] = 0;
    for( int i=1; i<=qcnt; i++ ) {
        int ca=lca(aa[i],stk[top]);
        while( dep[stk[top]]>dep[ca] ) {
            int fa, u;
            u = stk[top];
            top--;
            if( dep[stk[top]]<=dep[ca] ) {
                if( dep[stk[top]]<dep[ca] ) {
                    stk[++top] = ca;
                    dp[ca] = 0;
                }
                fa = stk[top];

                dp[u] = min( dp[u], bst[u] );
                dp[fa] += dp[u];
                break;
            }
            fa = stk[top];

            dp[u] = min( dp[u], bst[u] );
            dp[fa] += dp[u];
        }
        int u=aa[i];
        stk[++top] = u;

        dp[u] = bst[u];
    }
    while( top ) {
        if( top-1 ) {
            int fa=stk[top-1], u=stk[top];

            dp[u] = min( dp[u], bst[u] );
            dp[fa] += dp[u];
        }
        top--;
    }
    printf( "%lld\n", dp[1] );
}
int main() {
    scanf( "%d", &n );
    for( int i=1,u,v,w; i<n; i++ ) {
        scanf( "%d%d%d", &u, &v, &w );
        adde( u, v, w );
        adde( v, u, w );
    }
    anc[1][0] = 1;
    dep[1] = 1;
    bst[1] = oo;
    dfs(1);
    scanf( "%d", &m );
    for( int i=1; i<=m; i++ )
        sov();
}
```

3.10 元根

```
/*
学习了元根的一些知识，哈哈。

总结一下：

几个概念：

阶：对于模数 m 和整数 a，并且 gcd(m,a)==1，那么定义 a 在模 m 下的阶 r 为满足 a^r=1 mod m 的最小正整数。

性质 1：r in [1,phi(m)]  （由欧拉定理）

性质 2：r | phi(m)  （ a^r=a^phi(m) mod m，然后用反证法）
```

```
13
14  性质 3: r 是整数 a 模 m 的阶当且仅当满足: 1) a^r=1 mod m   2) a^(r/p(r))   1 mod m (后面推前面也用反正法)。
15
16  元根:
17
18  如果对于一个模数 m, 存在一个数 a, 满足 a 在模 m 下的阶是 phi(m), 那么就称 a 是模数 m 的一个元根。
19
20  性质: 所有质数有元根 (更一般的, 2,4,pe,2pe 有元根, p 是奇质数)
21
22  元根应用
23
24  元根依靠离散对数, 将对数运算引入了模数的缩系下。从而可以解决很多数论中关于指数的问题。
25
26  ind a = ind b mod phi(m) <=> a = b mod m
27
28  ind a^k = k ind a mod phi(m)
29
30  ind ab = ind a + ind b mod phi(m)
31  */
32
33
34  /**************************************************************
35      Problem: 3285
36      User: idy002
37      Language: C++
38      Result: Accepted
39      Time:756 ms
40      Memory:32072 kb
41  ****************************************************************/
42
43  #include <cstdio>
44  #include <cmath>
45  #include <cstring>
46  #include <cctype>
47  #define N 1000010
48
49  typedef long long dnt;
50
51  const int Hmod = 60793;
52  struct Hash {
53      int head[N], key[N], val[N], next[N], etot;
54      void init() {
55          etot = 0;
56          memset( head, 0, sizeof(head) );
57      }
58      void insert( int k, int v ) {
59          int kk = k%Hmod;
60          etot++;
61          key[etot] = k;
62          val[etot] = v;
63          next[etot] = head[kk];
64          head[kk] = etot;
65      }
66      int query( int k ) {
67          int kk = k%Hmod;
68          for( int t=head[kk]; t; t=next[t] )
69              if( key[t]==k ) return val[t];
70          return -1;
71      }
72  }hash;
73
74  dnt mpow( dnt a, int b, int c ) {
75      dnt rt;
76      for( rt=1; b; b>>=1,a=(a*a)%c )
77          if( b&1 ) rt=(rt*a)%c;
78      return rt;
79  }
80  int findroot( int p ) {
81      int phi = p-1;
82      int tmp = phi;
```

```
    int stk[50], top;
    top = 0;
    for( int i=2; i<=(1<<16); i++ ) {
        if( tmp%i==0 ) {
            stk[++top] = i;
            do {
                tmp/=i;
            }while( tmp%i==0 );
        }
    }
    if( tmp!=1 )
        stk[++top] = tmp;
    for( int r=1; ; r++ ) {
        bool ok = true;
        for( int i=1; i<=top; i++ ) {
            if( mpow(r,phi/stk[i],p)==1 ) {
                ok=false;
                break;
            }
        }
        if( ok ) return r;
    }
}
dnt ind( dnt r, int a, int p ) {    //  ind_r(a) mod p-1
    int m = ceil(sqrt(p-1));
    hash.init();
    dnt cur = 1;
    for( int i=0; i<m; i++ ) {
        if( cur==a ) return i;
        hash.insert( cur, i );
        cur = (cur*r) % p;
    }
    dnt base;
    base = cur = mpow(cur,p-2,p);
    for( int i=m; i<p; i+=m,cur=(cur*base)%p ) {
        int j = hash.query( a*cur%p );
        if( j!=-1 ) return i+j;
    }
    return -1;  //  impossible
}
dnt gcd( dnt a, dnt b ) {
    return b ? gcd(b,a%b) : a;
}
void exgcd( dnt a, dnt b, dnt &d, dnt &x, dnt &y ) {
    if( b==0 ) {
        d=a, x=1, y=0;
    } else {
        exgcd(b,a%b,d,y,x);
        y-=a/b*x;
    }
}
dnt meq( dnt a, dnt b, dnt c ) {    //  ax=b mod c
    dnt d, dd, x, y;
    a = (a%c+c)%c;
    b = (b%c+c)%c;
    d = gcd(a,c);
    if( b%d!=0 ) return -1;
    exgcd(a/d,c/d,dd,x,y);
    x = x*(b/d);
    x = (x%(c/d)+(c/d))%(c/d);
    if( x==0 ) x+=c/d;
    return x;
}

dnt a, b, c, g, p, r;
int aa[N], bb[N], cc[N], gg[N];

void read( int a[] ) {
    int i;
    char ch;
```

```
153     for( i=0; isdigit(ch=getchar()); i++ )
154         a[i] = ch-'0';
155     a[i] = -1;
156 }
157 dnt modulo( int a[], dnt mod ) {
158     dnt rt = 0;
159     for( int i=0; a[i]!=-1; i++ )
160         rt = (rt*10 + a[i]) % mod;
161     return rt;
162 }
163 int main() {
164     read(aa);
165     read(bb);
166     read(cc);
167     read(gg);
168     scanf( "%lld", &p );
169     a = modulo(aa,p-1);
170     b = modulo(bb,p-1),
171     c = modulo(cc,p);
172     g = modulo(gg,p);
173 
174     if( g%p==0 || c%p==0 ) {
175         if( g%p==0 && c%p==0 )
176             printf( "1\n" );
177         else
178             printf( "no solution\n" );
179         return 0;
180     }
181     r = findroot(p);
182 //  fprintf( stderr, "%d\n", (int)r );
183     dnt ans = meq( a*ind(r,g,p), ind(r,c,p)-b*ind(r,g,p), p-1 );
184     if( ans<0 )
185         printf( "no solution\n" );
186     else
187         printf( "%lld\n", ans );
188 }
189 #include <cstdio>
190 #include <cstring>
191 #include <cmath>
192 #include <vector>
193 #include <algorithm>
194 using namespace std;
195 
196 typedef long long dnt;
197 
198 const int mod = 8543;
199 const int elen = 100010;
200 struct Hash {
201     int head[mod], val[elen], rat[elen], next[elen], etot;
202     void init() {
203         memset( head, 0, sizeof(head) );
204         etot = 0;
205     }
206     void insert( int v, int r ) {
207         int k = v % mod;
208         etot++;
209         next[etot] = head[k];
210         rat[etot] = r;
211         val[etot] = v;
212         head[k] = etot;
213     }
214     int query( int v ) {
215         int k = v % mod;
216         for( int t=head[k]; t; t=next[t] )
217             if( val[t]==v ) return rat[t];
218         return -1;
219     }
220 }hash;
221 
222 int p, a, k;
```

```
223 int g, b;
224 vector<int> stk;
225
226 dnt mpow( dnt a, int b, int m ) {
227     dnt rt;
228     for( rt=1; b; b>>=1,a=(a*a)%m )
229         if( b&1 ) rt=(rt*a)%m;
230     return rt;
231 }
232 void exgcd( int a, int b, int &d, dnt &x, dnt &y ) {
233     if( b==0 ) d=a, x=1, y=0;
234     else {
235         exgcd( b, a%b, d, y, x );
236         y-=a/b*x;
237     }
238 }
239 int gcd( int a, int b ) {
240     return b ? gcd(b,a%b) : a;
241 }
242 int findroot( int n ) {         //    n is prime
243     if( n==2 ) return 1;
244
245     vector<int> pfac;
246     int maxi = (int)ceil(sqrt(n-1));
247     int remain=n-1;
248     for( int i=2; i<=maxi; i++ ) {
249         if( remain%i==0 ) {
250             pfac.push_back(i);
251             while( remain%i==0 )
252                 remain/=i;
253         }
254     }
255     if( remain!=1 ) pfac.push_back( remain );
256     for( int i=1; ; i++ ) {
257         bool ok = true;
258         for( int t=0; t<pfac.size(); t++ )
259             if( mpow(i,(n-1)/pfac[t],n)==1 ) {
260                 ok = false;
261                    break;
262             }
263         if( ok ) return i;
264     }
265 }
266 dnt inv( int a, int n ) {
267     return mpow(a,n-2,n);
268 }
269 dnt ind( int g, int b, int n ) {    //    n is prime, g is root, return v in [0,n-1]
270     hash.init();
271     int m = (int)ceil(sqrt(n-1));
272     dnt s = 1;
273     for( int i=0; i<m; i++ ) {
274         if( s==b ) return i;
275         hash.insert( s, i );
276         s = (s*g) % n;
277     }
278     int am = s;
279     s = b;
280     for( int i=m,j; i<n; i+=m ) {
281         s = (s*inv(am,n)) % n;
282         if( (j=hash.query(s))!=-1 )
283             return i+j;
284     }
285     return -1;    //    impossible
286 }
287 void meq( int a, int b, int m ) {
288     stk.clear();
289     int d = gcd(a,m);
290     if( b%d ) return;
291     int aa=a/d, bb=b/d, mm=m/d, dd;
292     dnt x0, y0;
```

```
293     exgcd( aa, mm, dd, x0, y0 );
294     x0 = (x0%mm+mm)%mm;
295     for( dnt k=0; k<d; k++ )
296         stk.push_back( (x0*bb+k*mm)%m );
297 }
298
299 int main() {
300     scanf( "%d%d%d", &p, &k, &a );     //    x^k = a mod p
301     if( a==0 ) {
302         printf( "1\n0 \n" );
303         return 0;
304     }
305     //    find the root of p: g
306     g = findroot(p);
307     //    ind a: b
308     b = ind( g, a, p );
309     //    kx=b mod phi(p)
310     meq( k, b, p-1 );
311     //    decode
312     for( int t=0; t<stk.size(); t++ )
313         stk[t] = mpow( g, stk[t], p );
314     sort( stk.begin(), stk.end() );
315     //    output
316     printf( "%d\n", stk.size() );
317     for( int t=0; t<stk.size(); t++ )
318         printf( "%d ", stk[t] );
319     printf( "\n" );
320 }
```

3.11 决策单调性

```
1  /*
2  决策单调性，对于一个 1D/1D（状态是一维，转移也是一维）的 DP，如果 DP 的决策具有单调性，那么就可以做到 O(nlogn) 的复杂
3  */
4
5  /**************************************************************
6      Problem: 2216
7      User: idy002
8      Language: C++
9      Result: Accepted
10     Time:4916 ms
11     Memory:14476 kb
12 ****************************************************************/
13
14 #include <cstdio>
15 #include <algorithm>
16 #include <cmath>
17 #define N 500010
18 using namespace std;
19
20 struct Trid {
21     int p, l, r;
22     Trid(){}
23     Trid( int p, int l, int r ):p(p),l(l),r(r){}
24 };
25
26 int n;
27 int aa[N];
28 int f[N], g[N], h[N];
29 Trid stk[N]; int top;
30
31 double calc( int j, int i ) {
32     return aa[j]-aa[i]+sqrt(abs(i-j));
33 }
34 void dodp( int dp[N] ) {
35     stk[top=1] = Trid(1,1,n);
36     for( int i=2; i<n; i++ ) {
37         if( calc(stk[top].p,n)>calc(i,n) ) continue;
38
```

```
        while( stk[top].l>=i &&
                calc(stk[top].p,stk[top].l)<calc(i,stk[top].l) )
            top--;
        if( stk[top].r==i-1 ) {
            stk[++top] = Trid( i, i, n );
        } else {
            int lf = max( stk[top].l+1, i );
            int rg = min( stk[top].r+1, n );
            int p = stk[top].p;
            while( lf<rg ) {
                int mid=(lf+rg)>>1;
                if( calc(p,mid) > calc(i,mid) ) lf=mid+1;
                else rg=mid;
            }
            stk[top].r = lf-1;
            stk[++top] = Trid( i, lf, n );
        }
    }
    for( int i=1; i<=top; i++ )
        for( int j=stk[i].l; j<=stk[i].r; j++ )
            dp[j] = stk[i].p;
}
int main() {
    scanf( "%d", &n );
    for( int i=1; i<=n; i++ )
        scanf( "%d", aa+i );
    dodp(f);
    reverse( aa+1, aa+1+n );
    dodp(g);
    reverse( aa+1, aa+1+n );

    for( int i=1; i<=n; i++ )
        g[i] = n+1-g[i];
    reverse( g+1, g+1+n );

    for( int i=1; i<=n; i++ )
        if( calc(f[i],i)>calc(g[i],i) ) h[i]=f[i];
        else h[i]=g[i];
    for( int i=1; i<=n; i++ )
        printf( "%lld\n", (long long)ceil(calc(h[i],i)) );
}
```

3.12 弦图

```
/*
一些定义：

弦图是一种特殊图：它的所有极小环都只有 3 个顶点。

单纯点：该顶点与其邻接点在原图中的导出子图是一个完全图。

图 G 的完美消去序列：一个顶点序列 a1a2a3...an，使得对于每个元素 ai，ai 在 ai、ai+1、ai+2...an 的导出子图中是一个单纯

弦图有一个性质：任何一个弦图都至少存在一个单纯点（该点和其邻接点组成一个完全图）

弦图另一个性质：一个图是弦图当且仅当其存在完美消去序列。(归纳证明)

最大势算法（msc)：若原图是弦图，则该算法计算出的序列是完美消去序列。

算法大致思想：从后往前计算序列，每次选择点 v 作为序列中的元素，v 是还未选的点中与已经选了的点连边最多的点。

然后检查该序列是否是完美消去序列。
*/
/*************************************************************
    Problem: 1242
```

```
    User: idy002
    Language: C++
    Result: Accepted
    Time:544 ms
    Memory:1816 kb
**************************************************************/

#include <cstdio>
#include <cstring>
#define N 1010
#define M N*N*2

int n, m;
bool c[N][N];
int qu[N], inq[N], dgr[N];
int stk[N], top;

void msc() {
    dgr[0] = -1;
    for( int i=n; i>=1; i-- ) {
        int s = 0;
        for( int u=1; u<=n; u++ )
            if( !inq[u] && dgr[u]>dgr[s] ) s=u;
        qu[i] = s;
        inq[s] = true;
        for( int u=1; u<=n; u++ )
            if( !inq[u] && c[s][u] ) dgr[u]++;
    }
}
bool check() {
    for( int i=n; i>=1; i-- ) {
        int s=qu[i];
        top = 0;
        for( int j=i+1; j<=n; j++ )
            if( c[s][qu[j]] ) stk[++top] = qu[j];
        if( top==0 ) continue;
        for( int j=2; j<=top; j++ )
            if( !c[stk[1]][stk[j]] ) return false;
    }
    return true;
}
int main() {
    scanf( "%d%d", &n, &m );
    for( int i=1,u,v; i<=m; i++ ) {
        scanf( "%d%d", &u, &v );
        c[u][v] = c[v][u] = 1;
    }
    msc();
    printf( "%s\n", check() ? "Perfect" : "Imperfect" );
}

/*
给定一个弦图，问最少染色数。

对于弦图的一个完美消去序列，从后往前染色，每次染可以染的最小编号的颜色，由完美消去序列的定义，
序列任一后缀的点的导出子图中，由该后缀第一个元素及其邻接点导出的子图一定是完全图，所以，序列中
某一元素染的颜色编号是该完全图的大小。所以最小染色数小于等于最大团的点数，而显然前者又大于等于后者，
故弦图的最小染色数等于最大团的大小。
*/

/**************************************************************
    Problem: 1006
    User: idy002
    Language: C++
    Result: Accepted
    Time:1672 ms
    Memory:11968 kb
**************************************************************/

#include <cstdio>
```

```cpp
#include <vector>
#define maxn 10010
using namespace std;

int n, m;
vector<int> g[maxn];
bool done[maxn];
int label[maxn], pos[maxn];

int msc() {
    int rt = 0;
    for( int i=n; i>=1; i-- ) {
        int mu = 0;
        for( int u=1; u<=n; u++ ) {
            if( !done[u] ) {
                if( !mu || label[u]>label[mu] )
                    mu = u;
            }
        }
        done[mu] = true;
        pos[mu] = i;
        int cnt = 0;
        for( int t=0; t<g[mu].size(); t++ ) {
            int v = g[mu][t];
            if( done[v] ) {
                cnt++;
            } else {
                label[v]++;
            }
        }
        rt = max( rt, cnt+1 );
    }
    return rt;
}
int main() {
    scanf( "%d%d", &n, &m );
    for( int i=1,u,v; i<=m; i++ ) {
        scanf( "%d%d", &u, &v );
        g[u].push_back(v);
        g[v].push_back(u);
    }
    printf( "%d\n", msc() );
}
```

3.13 树上莫队带修改

```cpp
/*
   1、带修改，其实就是暴力，只是将同一块的查询再按照时间顺序排，这样就能减少在修改操作上“爬”的时间，
  其实就是利用了数据随机这个特点，可以构造数据来卡。

    2、以前排序的方法是 u 按照块，v 按照 dfs 序，这次两个都是按照块，其实差不多。
*/
/**************************************************************
    Problem: 3052
    User: idy002
    Language: C++
    Result: Accepted
    Time:101223 ms
    Memory:21792 kb
****************************************************************/

#include <cstdio>
#include <cmath>
#include <vector>
#include <algorithm>
#define P(p) ((1)<<(p))
#define maxn 100010
#define maxp 16
using namespace std;
```

```
24
25  typedef long long dint;
26
27  int n, m, q, qq, qm;
28  int cc[maxn], ww[maxn], vv[maxn];
29  vector<int> g[maxn], stk;
30  int mno[maxn], mcc_siz, mcc_cnt;
31  int anc[maxn][maxp+1], depth[maxn], dfn[maxn], dfs_clock;
32  bool stat[maxn];
33  dint cnt[maxn], cur_ans;
34  dint ans[maxn];
35  int mdu[maxn], mdc[maxn], mdo[maxn], mdcc[maxn];
36
37  struct Qu {
38      int u, v, t, id;
39      bool operator<( const Qu & b ) const {
40          if( mno[u]^mno[b.u] ) return mno[u]<mno[b.u];
41          if( mno[v]^mno[b.v] ) return mno[v]<mno[b.v];
42          return t<b.t;
43      }
44  };
45  Qu qu[maxn];
46
47  int dfs( int u ) {
48      dfn[u] = ++dfs_clock;
49      depth[u] = depth[anc[u][0]]+1;
50      for( int p=1; p<=maxp; p++ ) {
51          anc[u][p] = anc[anc[u][p-1]][p-1];
52          if( !anc[u][p] ) break;
53      }
54
55      int sz = 0;
56      for( int t=0; t<g[u].size(); t++ ) {
57          int v = g[u][t];
58          if( v==anc[u][0] ) continue;
59          anc[v][0] = u;
60          sz += dfs(v);
61          if( sz > mcc_siz ) {
62              mcc_cnt++;
63              for( int i=1; i<=sz; i++ ) {
64                  mno[stk.back()] = mcc_cnt;
65                  stk.pop_back();
66              }
67              sz = 0;
68          }
69      }
70      stk.push_back( u );
71      return sz+1;
72  }
73
74  int lca( int u, int v ) {
75      if( depth[u]<depth[v] ) swap(u,v);
76      int t = depth[u]-depth[v];
77      for( int p=0; t; t>>=1, p++ )
78          if( t&1 ) u=anc[u][p];
79      if( u==v ) return u;
80      for( int p=maxp; p>=0 && anc[u][0]!=anc[v][0]; p-- )
81          if( anc[u][p]!=anc[v][p] )
82              u = anc[u][p], v = anc[v][p];
83      return anc[u][0];
84  }
85
86  void inv_sig( int u ) {
87      if( stat[u] ) {
88          cur_ans -= (dint)ww[cnt[cc[u]]]*vv[cc[u]];
89          cnt[cc[u]]--;
90      } else {
91          cnt[cc[u]]++;
92          cur_ans += (dint)ww[cnt[cc[u]]]*vv[cc[u]];
93      }
```

```
94      stat[u] ^= 1;
95  }
96
97  void chg_sig( int u, int type ) {
98      if( stat[u] ) {
99          inv_sig(u);
100         cc[u] = type;
101         inv_sig(u);
102     } else cc[u] = type;
103 }
104
105 void inv_chain( int u, int v ) {
106     int ca = lca(u,v);
107     for( ; u!=ca; u=anc[u][0] ) inv_sig(u);
108     for( ; v!=ca; v=anc[v][0] ) inv_sig(v);
109 }
110 void app_time( int fm, int to ) {
111     while( fm<to ) {
112         fm++;
113         chg_sig(mdu[fm],mdc[fm]);
114     }
115     while( to<fm ) {
116         chg_sig(mdu[fm],mdo[fm]);
117         fm--;
118     }
119 }
120
121 void work() {
122     sort( qu+1, qu+1+qq );
123     int ou=qu[1].u;
124     int ov=qu[1].u;
125     int ot=0;
126     for( int i=1; i<=qq; i++ ) {
127         int u = qu[i].u, v = qu[i].v;
128         inv_chain( u, ou );
129         inv_chain( v, ov );
130         app_time( ot, qu[i].t );
131         ot = qu[i].t;
132         ou = u;
133         ov = v;
134         int ca = lca(u,v);
135         inv_sig( ca );
136         ans[qu[i].id] = cur_ans;
137         inv_sig( ca );
138     }
139 }
140
141 int main() {
142     scanf( "%d%d%d", &n, &m, &q );
143     for( int i=1; i<=m; i++ ) scanf( "%d", vv+i );
144     for( int i=1; i<=n; i++ ) scanf( "%d", ww+i );
145     for( int i=1,u,v; i<n; i++ ) {
146         scanf( "%d%d", &u, &v );
147         g[u].push_back(v);
148         g[v].push_back(u);
149     }
150     mcc_siz = (int)(pow(n,2.0/3.0))+1;
151     dfs(1);
152     while( !stk.empty() ) {
153         mno[stk.back()] = mcc_cnt;
154         stk.pop_back();
155     }
156     for( int i=1; i<=n; i++ ) {
157         scanf( "%d", cc+i );
158         mdcc[i] = cc[i];
159     }
160     for( int i=1,type,x,y; i<=q; i++ ) {
161         scanf( "%d%d%d", &type, &x, &y );
162         if( !type ) {
163             qm++;
```

```
164             mdu[qm]=x, mdc[qm]=y, mdo[qm]=mdcc[x];
165             mdcc[x] = y;
166         } else {
167             qq++;
168             qu[qq].u=x, qu[qq].v=y, qu[qq].t=qm, qu[qq].id=qq;
169         }
170     }
171     work();
172     for( int i=1; i<=qq; i++ )
173         printf( "%lld\n", ans[i] );
174 }
```

3.14 点分

```
1   /*    1. 关于重心，对于一个无向图，我们这样给每条边重新确定方向：u<->v 这条边将原图分成两个部分 Su,Sv,w(S) 表示 S
2
3             u->v 当 w(Su)<w(Sv)
4
5             u<-v 当 w(Su)>w(Sv)
6
7             u<->v 当 w(Su)=w(Sv)
8
9           那么从一个点沿着边的方向走，直到走到一个区域，使得走不出这个区域，那么这个区域中的任何一个点就是重心.
10
11      2. 有了第一条，就可以运用点分来搞一些询问某个点的的东西了 (这里就是询问以这个点为重心的权和).
12
13        再用点分最多 log 层和类似 1 的性质，就可以做了.
14
15      3. 点分不擅长求全局最... 的东西，而是擅长关于某个点.... 的东西.
16
17         这道题可以用是因为有重心位置的单调性.
18
19         还有捉迷藏那道题是用堆来辅助的.
20
21
22  */
23
24  #include <cstdio>
25  #include <vector>
26  #include <map>
27  #define max(a,b) ((a)>(b)?(a):(b))
28  #define oo 0x3f3f3f3f3f3f3f3fLL
29  #define N 100010
30  #define M N<<1
31  #define P 17
32  using namespace std;
33
34  typedef long long dnt;
35  struct Info {
36      int p, s;
37      Info(){}
38      Info( int p, int s ):p(p),s(s){}
39  };
40  struct Pair {
41      int s, c;
42      Pair(){}
43      Pair( int s, int c ):s(s),c(c){}
44  };
45
46  int n, m;
47  int head[N], dest[M], wght[M], next[M], etot;
48  int idv[M], vid[N], dep[N], stu[M][P+1], stp[M][P+1];
49  int bac[N], fat[N], siz[N], vis[N], dis[N];
50  int bin[P+1], log[M], qu[N], bg, ed;
51  //Info info[N][P+1]; int icnt[N];
52  //Pair gc[N][22]; int gcnt[N];
53  vector<Info> info[N];
54  vector<Pair> gc[N];
55  dnt cans[N], fans[N], sumw[N];
```

```
dnt rans[N]; int tag[N], curt;

inline void uMax( int &u, int v ) { if( u<v ) u=v; }
void adde( int u, int v, int w ) {
    etot++;
    dest[etot] = v;
    wght[etot] = w;
    next[etot] = head[u];
    head[u] = etot;
}
void build_lca( int s ) {
    //    qu fat dep dis, siz
    fat[s] = 0;
    dep[s] = 1;
    dis[s] = 0;
    siz[s] = 0;
    qu[bg=ed=1] = s;
    while( bg<=ed ) {
        int u=qu[bg++];
        for( int t=head[u]; t; t=next[t] ) {
            int v=dest[t], w=wght[t];
            if( v==fat[u] ) continue;
            fat[v] = u;
            siz[v] = 0;
            dep[v] = dep[u]+1;
            dis[v] = dis[u]+w;
            qu[++ed] = v;
        }
    }
    //    siz
    for( register int i=ed; i>=2; i-- ) {
        int u=qu[i], p=fat[u];
        siz[u]++;
        siz[p]+=siz[u];
    }
    siz[s]++;
    //    idv vid
    vid[s] = 1;
    idv[1] = s;
    for( register int i=1; i<=ed; i++ ) {
        int u=qu[i];
        int cur=vid[u]+1;
        for( register int t=head[u]; t; t=next[t] ) {
            int v=dest[t];
            if( v==fat[u] ) continue;
            idv[cur] = u;
            cur++;
            vid[v] = cur;
            idv[cur] = v;
            cur += siz[v]+siz[v]-1;
        }
    }
    idv[n+n] = s;
    //    bin log
    int idc = n+n;
    bin[0] = 1;
    for( int i=1; i<=P; i++ ) bin[i] = bin[i-1]<<1;
    log[0] = -1;
    for( int i=1; i<=idc; i++ ) log[i] = log[i>>1]+1;
    //    stu stp
    for( int i=1; i<=idc; i++ ) {
        stu[i][0] = idv[i];
        stp[i][0] = dep[idv[i]];
    }
    for( int p=1; p<=log[idc]; p++ ) {
        for( register int i=1; i<=idc-bin[p]+1; i++ ) {
            if( stp[i][p-1] < stp[i+bin[p-1]][p-1] ) {
                stp[i][p] = stp[i][p-1];
                stu[i][p] = stu[i][p-1];
            } else {
```

```
126                 stp[i][p] = stp[i+bin[p-1]][p-1];
127                 stu[i][p] = stu[i+bin[p-1]][p-1];
128             }
129         }
130     }
131 }
132 inline int lca( int u, int v ) {
133     int lf=vid[u], rg=vid[v];
134     if( lf>rg ) swap(lf,rg);
135     int p = log[rg-lf+1];
136     if( stp[lf][p] < stp[rg-bin[p]+1][p] )
137         return stu[lf][p];
138     else
139         return stu[rg-bin[p]+1][p];
140 }
141 inline int qu_dis( int u, int v ) {
142     int ca = lca(u,v);
143     return dis[u]+dis[v]-(dis[ca]<<1);
144 }
145 int build_vdcp( int s ) {
146     int c;
147     //    qu fat, siz bac
148     fat[s] = 0;
149     siz[s] = bac[s] = 0;
150     qu[bg=ed=1] = s;
151     while( bg<=ed ) {
152         int u=qu[bg++];
153         for( int t=head[u]; t; t=next[t] ) {
154             int v=dest[t];
155             if( v==fat[u] || vis[v] ) continue;
156             fat[v] = u;
157             siz[v] = bac[v] = 0;
158             qu[++ed] = v;
159         }
160     }
161     //    siz bac
162     for( register int i=ed; i>=2; i-- ) {
163         int u=qu[i], p=fat[u];
164         siz[u]++;
165         siz[p]+=siz[u];
166         uMax( bac[p], siz[u] );
167     }
168     siz[s]++;
169     //    bac c
170     c = 0;
171     for( register int i=1; i<=ed; i++ ) {
172         int u=qu[i];
173         uMax( bac[u], siz[s]-siz[u] );
174         if( bac[u]<bac[c] ) c=u;
175     }
176     //    qu info
177     vis[c] = true;
178     vector<int> stk;
179     for( int t=head[c]; t; t=next[t] ) {
180         int s=dest[t], cc;
181         if( vis[s] ) continue;
182 
183         qu[bg=ed=1] = s;
184         fat[s] = c;
185         stk.clear();
186         while( bg<=ed ) {
187             int u=qu[bg++];
188             stk.push_back( u );
189             for( int t=head[u]; t; t=next[t] ) {
190                 int v=dest[t];
191                 if( v==fat[u] || vis[v] ) continue;
192                 qu[++ed] = v;
193                 fat[v] = u;
194             }
195         }
```

```
196         cc = build_vdcp(s);
197         gc[c].push_back(Pair(s,cc));
198 //        gc[c][gcnt[c]] = Pair(s,cc);
199 //        gcnt[c]++;
200         for( register int t=stk.size()-1; t>=0; t-- )  {
201             int u=stk[t];
202             info[u].push_back( Info(c,cc) );
203 //            info[u][icnt[u]] = Info(c,cc);
204 //            icnt[u]++;
205         }
206     }
207     return c;
208 }
209 dnt query( int u ) {
210     if( tag[u]==curt ) return rans[u];
211     tag[u] = curt;
212     dnt rt = cans[u];
213     for( int t=0; t<info[u].size(); t++ ) {
214 //    for( int t=icnt[u]-1; t>=0; t-- ) {
215         int p=info[u][t].p, s=info[u][t].s;
216         rt += cans[p]-fans[s]+(sumw[p]-sumw[s])*qu_dis(u,p);
217     }
218     return rans[u]=rt;
219 }
220 dnt search( int u ) {
221     dnt su = query(u);
222     for( int t=0; t<gc[u].size(); t++ ) {
223 //    for( int t=gcnt[u]-1; t>=0; t-- ) {
224         Pair &p = gc[u][t];
225         dnt a=query(p.s);
226         if( a<su ) return search(p.c);
227     }
228     return su;
229 }
230 void modify( int u, int delta ) {
231     sumw[u] += delta;
232     for( int t=info[u].size()-1; t>=0; t-- ) {
233 //    for( int t=icnt[u]-1; t>=0; t-- ) {
234         int p=info[u][t].p, s=info[u][t].s;
235         dnt d = (dnt)delta*qu_dis(u,p);
236         cans[p] += d;
237         fans[s] += d;
238         sumw[p] += delta;
239     }
240 }
241 int main() {
242     scanf( "%d%d", &n, &m );
243     for( int i=1,u,v,w; i<n; i++ ) {
244         scanf( "%d%d%d", &u, &v, &w );
245         adde( u, v, w );
246         adde( v, u, w );
247     }
248     bac[0] = n;
249     int core = build_vdcp(1);
250     build_lca(1);
251     for( int t=1,u,d; t<=m; t++ ) {
252         scanf( "%d%d", &u, &d );
253         modify( u, d );
254         curt = t;
255         printf( "%lld\n", search(core) );
256     }
257 }
```

3.15 集合幂级数

```
1  #include <cstdio>
2
3  const int N = 10;
4
```

```
5  int n, U;
6  int a[1<<N], b[1<<N], c[1<<N];
7
8  void trans( int a[], int flag ) {
9      for( int b=0; b<n; b++ ) {
10         int u = U ^ (1<<b);
11         for( int s=u,t=1<<(n-1); t; s=(s-1)&u,t-- ) {
12             int l=a[s], r=a[s|(1<<b)];
13             /*
14             NOT AND
15             if( flag==1 ) {
16                 a[s] = l+r;
17                 a[s|(1<<b)] = r;
18             } else {
19                 a[s] = r;
20                 a[s|(1<<b)] = l-r;
21             }
22              */
23             /*
24             NOT XOR
25             if( flag==1 ) {
26                 a[s] = l+r;
27                 a[s|(1<<b)] = l-r;
28             } else {
29                 a[s] = (l-r)/2;
30                 a[s|(1<<b)] = (l+r)/2;
31             }
32              */
33             /*
34             NOT OR
35             if( flag==1 ) {
36                 a[s] = l;
37                 a[s|(1<<b)] = l+r;
38             } else {
39                 a[s] = r-l;
40                 a[s|(1<<b)] = l;
41             }
42             */
43             /*
44             OR
45             if( flag==1 ) {
46                 a[s] = l;
47                 a[s|(1<<b)] = l+r;
48             } else {
49                 a[s] = l;
50                 a[s|(1<<b)] = r-l;
51             }
52             */
53             /*
54             AND
55             if( flag==1 ) {
56                 a[s] = l+r;
57                 a[s|(1<<b)] = r;
58             } else {
59                 a[s] = l-r;
60                 a[s|(1<<b)] = r;
61             }
62             */
63             /*
64             XOR
65             if( flag==1 ) {
66                 a[s] = l+r;
67                 a[s|(1<<b)] = l-r;
68             } else {
69                 a[s] = (l+r)/2;
70                 a[s|(1<<b)] = (l-r)/2;
71             }
72             */
73         }
74     }
```

```
75  }
76  int main() {
77      scanf( "%d", &n );
78      U = (1<<n)-1;
79      for( int i=0; i<=U; i++ )
80          scanf( "%d", a+i );
81      for( int i=0; i<=U; i++ )
82          scanf( "%d", b+i );
83      trans(a,1);
84      trans(b,1);
85      for( int s=0; s<=U; s++ )
86          c[s] = a[s]*b[s];
87      trans(c,-1);
88      for( int s=0; s<=U; s++ )
89          printf( "%d ", c[s] );
90      printf( "\n" );
91  }
```