Qwertier's Templates
OF
Algorithm And DataStrucures

QWERTIER
gmayukun@gmail.com

Generated on October 20, 2016

目录

1 数学

1.1 素数表（NTT专用）

$r \cdot 2^k$	**r**	**k**	**g**
3	1	1	2
5	1	2	2
17	1	4	3
97	3	5	5
193	3	6	5
257	1	8	3
7681	15	9	17
12289	3	12	11
40961	5	13	3
65537	1	16	3
786433	3	18	10
5767169	11	19	3
7340033	7	20	3
23068673	11	21	3
104857601	25	22	3
167772161	5	25	3
469762049	7	26	3
1004535809	479	21	3
2013265921	15	27	31
2281701377	17	27	3
3221225473	3	30	5
75161927681	35	31	3
77309411329	9	33	7

$r \cdot 2^k$	r	k	g
206158430209	3	36	22
2061584302081	15	37	7
2748779069441	5	39	3
1231453023109121	35	45	3
6597069766657	3	41	5
39582418599937	9	42	5
79164837199873	9	43	5
263882790666241	15	44	7
1337006139375617	19	46	3
3799912185593857	27	47	5
4222124650659841	15	48	19
7881299347898369	7	50	6
31525197391593473	7	52	3
180143985094819841	5	55	6
1945555039024054273	27	56	5
4179340454199820289	29	57	3

1.2 NTT

```
#define MOD 6597069768543436911
namespace NTT {
int m;
#define eps (1e-15)
LL fMul(LL t, LL p) {
  LL ret = t * p - (LL)((long double)t/MOD*p+eps)*MOD;
  if (ret < 0)
    return (ret % MOD + MOD) % MOD;
  else
    return ret;
}
LL fPow(LL t, LL p) {
  LL ret = 1;
  while (p) {
    if (p & 1)
      ret = fMul(ret, t);
```

```
    t = fMul(t, t);
    p >>= 1;
  }
  return ret;
}
int rev[N];
LL g = 19;
void ntt(LL *in, LL *out, int n, int flag) {   // flag = 0 if ntt, 1 if intt
  memset(rev, 0, sizeof(rev));
  REP (i, (1<<m)) {
    REP (j, m) {
      rev[i] |= ((i>>j)&1) << (m-j-1);
    }
  }
  for (int i = 0; i < n; i++)
    out[rev[i]] = in[i];
  for (int s = 1; (1 << s) <= n; s++) {
    int m = 1 << s, m_2 = m /2;
    LL wm;
    if (!flag)
      wm = fPow(g, (MOD - 1) / m);
    else
      wm = fPow(fPow(g, (MOD - 1) / m), MOD - 2);
    for (int k = 0; k < n; k+=m) {
      LL w = 1;
      for (int j = 0; j < m_2; j++) {
        LL t = fMul(w, out[k + j + m_2]),
            u = out[k + j];
        out[k + j] = (u + t) % MOD;
        out[k + j + m_2] = (u - t + MOD) % MOD;
        w = fMul(w, wm);
      }
    }
  }
  if (flag) {
    LL inv = fPow(n, MOD - 2);
    REP (i, n) {
      out[i] = fMul(out[i], inv);
    }
  }
}
}
```

1.3 FFT

```
namespace FFT {
int L;
struct cp {
  double r,i;
  cp(){r=i=0;}
  cp(double _r,double _i):r(_r),i(_i){}
  cp operator*(cp &t){return cp(r*t.r-i*t.i,i*t.r+t.i*r);}
  cp operator+(cp &t){return cp(r+t.r,i+t.i);}
  cp operator-(cp &t){return cp(r-t.r,i-t.i);}
};
const double PI=3.14159265358979323846;
cp tmp[N];
int rev[N];
void fft(cp *a,int flag) {
  REP(i,(1<<L))tmp[i]=a[rev[i]];
  REP(i,(1<<L))a[i]=tmp[i];
  for(int k=2; k<=(1<<L); k<<=1){
    cp wn(cos(2*PI/k),flag*sin(2*PI/k));
    for(int i=0; i<(1<<L); i+=k){
      cp w(1,0),x,y;
      for(int j=i; j<i+k/2; j++){
        x=a[j];
        y=a[j+k/2]*w;
        a[j]=x+y;
        a[j+k/2]=x-y;
        w=w*wn;
      }
    }
  }
  if(flag==-1)REP(i,(1<<L))a[i].r/=(1<<L);
}
void calc_rev() {
  REP(i,(1<<L)){
    rev[i] = 0;
    REP(j,L) {
      if((i>>j)&1)
        rev[i]|=(1<<(L-1-j));
    }
  }
}
}
```

1.4 FWT

```
namespace FWT {
template <typename T>
void or_fwt(T X[], int l, int r) {
  if (l == r)
    return;
  int m = (l + r) >> 1;
  or_fwt(X, l, m); or_fwt(X, m + 1, r);
  for (int i = 0; i <=  m - l; i++) {
    X[m + 1 + i] += X[l + i];
  }
}
template <typename T>
void or_ifwt(T X[], int l, int r) {
  if (l == r)
    return;
  int m = (l + r) >> 1;
  or_ifwt(X, l, m); or_ifwt(X, m + 1, r);
  for (int i = 0; i <=  m - l; i++) {
    X[m + 1 + i] -= X[l + i];
  }
}
template <typename T>
void and_fwt(T X[], int l, int r) {
  if (l == r)
    return;
  int m = (l + r) >> 1;
  and_fwt(X, l, m); and_fwt(X, m + 1, r);
  for (int i = 0; i <=  m - l; i++) {
    X[l + i] += X[m + 1 + i];
  }
}
template <typename T>
void and_ifwt(T X[], int l, int r) {
  if (l == r)
    return;
  int m = (l + r) >> 1;
  and_ifwt(X, l, m); and_ifwt(X, m + 1, r);
  for (int i = 0; i <=  m - l; i++) {
    X[l + i] -= X[m + 1 + i];
  }
}
template <typename T>
```

```
void xor_fwt(T X[], int l, int r) {
  if (l == r)
    return;
  int m = (l + r) >> 1;
  xor_fwt(X, l, m); xor_fwt(X, m + 1, r);
  for (int i = 0; i <=  m - l; i++) {
    X[l + i] += X[m + 1 + i];
    X[m + 1 + i] = X[l + i] - 2 * X[m + 1 + i];
  }
}
template <typename T>
void xor_ifwt(T X[], int l, int r) {
  if (l == r)
    return;
  int m = (l + r) >> 1;
  for (int i = 0; i <=  m - l; i++) {
    X[l + i] += X[m + 1 + i];
    X[m + 1 + i] = X[l + i] - 2 * X[m + 1 + i];
    X[l+i] /= 2;
    X[m + 1 + i] /= 2;
  }
  xor_ifwt(X, l, m); xor_ifwt(X, m + 1, r);
}
}
```

1.5 拓展欧几里得

```
LL exgcd(LL a, LL b, LL &x, LL &y) {
  if (a==0&&b==0)return -1;
  if (b==0) {x=1;y=0;return a;}
  LL d=extend_gcd(b,a%b,y,x);
  y-=a/b*x;
  return d;
}
```

1.6 米勒罗宾素数测试

```
const int S = 8;
long long mult_mod(long long a,long long b,long long c) {
  a %= c;
  b %= c;
  long long ret = 0;
```

```
  long long tmp = a;
  while(b) {
    if(b & 1) {
      ret += tmp;
      if(ret > c)ret -= c;
    }
    tmp <<= 1;
    if(tmp > c)tmp -= c;
    b >>= 1;
  }
  return ret;
}
long long pow_mod(long long a,long long n,long long mod) {
  long long ret = 1;
  long long temp = a%mod;
  while(n) {
    if(n & 1)ret = mult_mod(ret,temp,mod);
    temp = mult_mod(temp,temp,mod);
    n >>= 1;
  }
  return ret;
}
bool check(long long a,long long n,long long x,long long t) {
  long long ret = pow_mod(a,x,n);
  long long last = ret;
  for(int i = 1;i <= t;i++) {
    ret = mult_mod(ret,ret,n);
    if(ret == 1 && last != 1 && last != n-1)return true;//合数
    last = ret;
  }
  if(ret != 1)return true;
  else return false;
}
bool Miller_Rabin(long long n) {
  if( n < 2)return false;
  if( n == 2)return true;
  if( (n&1) == 0)return false;//偶数
  long long x = n - 1;
  long long t = 0;
  while( (x&1)==0 ){x >>= 1; t++;}
  srand(time(NULL));
  for(int i = 0;i < S;i++) {
    long long a = rand()%(n-1) + 1;
    if( check(a,n,x,t) )
```

```
50        return false;
51    }
52    return true;
53 }
```

1.7 质因数分解

```
1  long long factor[100];//质因素分解结果（刚返回时时无序的）
2  int tol;//质因数的个数，编号0 tol-1
3  long long gcd(long long a,long long b) {
4    long long t;
5    while(b) {
6      t = a;
7      a = b;
8      b = t%b;
9    }
10   if(a >= 0)return a;
11   else return -a;
12 }
13 // 找出一个因子
14 long long pollard_rho(long long x,long long c) {
15   long long i = 1, k = 2;
16   srand(time(NULL));
17   long long x0 = rand()%(x-1) + 1;
18   long long y = x0;
19   while(1) {
20     i ++;
21     x0 = (mult_mod(x0,x0,x) + c)%x;
22     long long d = gcd(y - x0,x);
23     if( d != 1 && d != x)return d;
24     if(y == x0)return x;
25     if(i == k){y = x0; k += k;}
26   }
27 }
28 //对n进行素因子分解，存入factor. k设置为107左右即可
29 void findfac(long long n,int k) {
30   if(n == 1)return;
31   if(Miller_Rabin(n)) {
32     factor[tol++] = n;
33     return;
34   }
35   long long p = n;
36   int c = k;
```

```
  while( p >= n)
    p = pollard_rho(p,c--);//值变化，防止死循环k
  findfac(p,k);
  findfac(n/p,k);
}
```

1.8 高斯消元

```
bool gauss(double *m[N], int n) {
  for(int i=0; i<n; i++){
    int c=i;
    for(int j=i; j<n; j++)if(fabs(m[j][i])>fabs(m[c][i]))c=j;
    if (fabs(m[c][i]) < eps)
      return false;
    for(int j=0; j<=n; j++)swap(m[i][j],m[c][j]);
    for(int j=i+1; j<n; j++){
      double f=m[j][i]/m[i][i];
      for(int k=n; k>=i; k--)
        m[j][k]-=m[j][i]*m[i][k]/m[i][i];
    }
  }
  for(int i=n-1; i>=0; i--){
    for(int j=i+1; j<n; j++)
      m[i][n]-=m[j][n]*m[i][j];
    m[i][n]/=m[i][i];
  }
  return true;
}
```

2 字符串

2.1 KMP算法

```
namespace KMP {
int fa[L];
void get_fa(char *s, int n) {
  int j = fa[0] = -1;
  FOR (i, n - 1) {
    while (j != -1 && s[j + 1] != s[i])
      j = fa[j];
    if (s[j + 1] == s[i])
      j++;
    fa[i] = j;
  }
}
int find(char *s, int n, char *t, int m) {
  int j = -1, ret = 0;
  REP (i, m) {
    while (j != -1 && s[j + 1] != t[i])
      j = fa[j];
    if (s[j + 1] == t[i])
      j++;
    if (j == n - 1) {
      ret++;
      j = fa[j];
    }
  }
  return ret;
}
}
```

2.2 AC自动机

```
namespace AC {
int sz, ch[SZ][26], val[SZ];
void insert(char *str, int v) {
  int u = 0, len = 0;
  while (*str) {
    int c=(*str++)-'a';
    len++;
    if (!ch[u][c]) {
      ch[u][c]=++sz;
```

```
      memset(ch[sz], 0, sizeof(ch[sz]));
      val[sz] = 0;
    }
    u = ch[u][c];
  }
  val[u] += v;
}
int fr, rr, que[SZ], fa[SZ], lst[SZ];
void calc_fa() {
  fr=0,rr=-1;
  REP (i, 26) {
    if(ch[0][i]) {
      que[++rr]=ch[0][i];
      fa[ch[0][i]] = 0;
      lst[ch[0][i]] = 0;
    }
  }
  while (fr <= rr) {
    int r = que[fr++];
    REP (i,26) {
      int u=ch[r][i];
      if (!u) {
        ch[r][i] = ch[fa[r]][i];
        continue;
      }
      fa[u] = ch[fa[r]][i];
      lst[u] = val[fa[u]] ? fa[u] : lst[fa[u]];
      val[u] += val[lst[u]];
      que[++rr]=u;
    }
  }
}
void init() {
  sz = 0;
  memset(ch[0], 0, sizeof(ch[0]));
  val[0] = 0;
  lst[0] = 0;
}
}
```

2.3 后缀数组

```
namespace SA{
```

```
int sa[N], cnt[N], *x = new int[N], *y = new int[N];
inline bool eq(int i,int j){
  return (i>=n && j>=n)||(i<n && j<n && y[i]==y[j]);
}
void calc_sa(){
  memset(cnt, 0, sizeof(int)*m);
  REP(i,n)cnt[x[i]=s[i]-'a']++;
  FOR(i,m-1)cnt[i]+=cnt[i-1];
  for(int i=n-1; i>=0; i--)sa[--cnt[x[i]]]=i;
  for(int k=1; k<=n; k<<=1){
    int p=0;
    for(int i=n-k; i<n; i++)y[p++]=i;
    REP(i,n)if(sa[i]>=k)y[p++]=sa[i]-k;

    REP(i,m)cnt[i]=0;
    REP(i,n)cnt[x[y[i]]]++;
    FOR(i,m-1)cnt[i]+=cnt[i-1];
    for(int i=n-1; i>=0; i--)sa[--cnt[x[y[i]]]]=y[i];

    swap(x,y);
    x[sa[0]]=0;
    p=1;
    FOR(i,n-1)
      x[sa[i]]=eq(sa[i],sa[i-1])&&eq(sa[i]+k,sa[i-1]+k)?p-1:p++;
    if(p>=n)break;
    m=p;
  }
}
int h[N];
void calc_hi(){
  int j,k=0;
  REP(i,n){
    if(x[i]==0){k=0;continue;}
    if(k)k--;
    j=sa[x[i]-1];
    while(s[i+k]==s[j+k])k++;
    h[x[i]]=k;
  }
}
}
```

3 数据结构

3.1 树链剖分

```
/*
  SDOI 2012 染色：给定一棵树，节点有色。现有两种查询：
  1. 修改某点的颜色
  2. 查询一条链上不同的颜色段数
*/
#include <algorithm>
#include <stdio.h>
#include <string.h>
#include <map>
#define PROB
#define N 100010
#define For(i,n) for(int i=1; i<=n; i++)
using namespace std;

int n;

int le[N],pe[N<<1],ev[N<<1],ecnt;
void addEdge(int u,int v){
  ecnt++;
  pe[ecnt]=le[u];
  le[u]=ecnt;
  ev[ecnt]=v;
}

int ori[N],a[N];

#define lc (o<<1)
#define rc (o<<1|1)
#define mid ((l+r)>>1)
int cl[N<<2],cr[N<<2],sumv[N<<2],setv[N<<2],L,R,setc;
inline void maintain(int o){
  cl[o]=cl[lc];
  cr[o]=cr[rc];
  sumv[o]=sumv[lc]+sumv[rc]-(cr[lc]==cl[rc]);
}
inline void pushdown(int o){
  if(setv[o]!=-1){
    cl[lc]=cr[lc]=setv[lc]=setv[o];
    sumv[lc]=1;
    cl[rc]=cr[rc]=setv[rc]=setv[o];
```

```
    sumv[rc]=1;
    setv[o]=-1;
  }
}
void build(int o=1,int l=1,int r=n){
  if(l==r){
    cl[o]=cr[o]=a[l];
    sumv[o]=1;
    return;
  }
  build(lc,l,mid);
  build(rc,mid+1,r);
  maintain(o);
}
void update(int o=1,int l=1,int r=n){
  if(L<=l&&r<=R){
    cl[o]=cr[o]=setv[o]=setc;
    sumv[o]=1;
    return;
  }
  pushdown(o);
  if(L<=mid)update(lc,l,mid);
  if(R>mid)update(rc,mid+1,r);
  maintain(o);
}
struct ANS{
  int cl,cr,sumv;
  ANS(int a,int b,int c):cl(a),cr(b),sumv(c){}
  ANS operator+(const ANS& r){
    return ANS(cl,r.cr,sumv+r.sumv-(cr==r.cl));
  }
};
ANS query(int o=1,int l=1,int r=n){
  if(L<=l&&r<=R)
    return ANS(cl[o],cr[o],sumv[o]);
  pushdown(o);
  if(R<=mid)return query(lc,l,mid);
  if(L>mid)return query(rc,mid+1,r);
  return query(lc,l,mid)+query(rc,mid+1,r);
}

int fa[N],sz[N],hson[N],dep[N];
void dfs1(int u){
  sz[u]=1;
```

```
  for(int i=le[u]; i; i=pe[i]){
    int &v=ev[i];
    if(v==fa[u])continue;
    fa[v]=u;
    dep[v]=dep[u]+1;
    dfs1(v);
    sz[u]+=sz[v];
    if(sz[v]>sz[hson[u]])hson[u]=v;
  }
}
int dfs_clock,top[N],id[N];
void dfs2(int u){
  id[u]=++dfs_clock;
  if(hson[u]){
    top[hson[u]]=top[u];
    dfs2(hson[u]);
  }
  for(int i=le[u]; i; i=pe[i]){
    int &v=ev[i];
    if(v==fa[u] || v==hson[u])continue;
    top[v]=v;
    dfs2(v);
  }
}

int Query(int u,int v){
  ANS ans1(-1,-1,0),ans2(-2,-2,0);
  while(top[u]!=top[v]){
    if(dep[top[u]]>dep[top[v]]){
      L=id[top[u]],R=id[u];
      ANS q=query();
      ans1=q+ans1;
      u=fa[top[u]];
    }else{
      L=id[top[v]],R=id[v];
      ANS q=query();
      ans2=q+ans2;
      v=fa[top[v]];
    }
  }
  if(dep[u]>dep[v]){
    L=id[v],R=id[u];
    ANS q=query();
    swap(ans2.cr,ans2.cl);
```

```
129     return (ans2+q+ans1).sumv;
130   }else{
131     L=id[u],R=id[v];
132     ANS q=query();
133     swap(ans1.cl,ans1.cr);
134     return (ans1+q+ans2).sumv;
135   }
136 }
137 void Modify(int u,int v,int c){
138   while(top[u]!=top[v]){
139     if(dep[top[u]]<dep[top[v]])swap(u,v);
140     L=id[top[u]],R=id[u];
141     update();
142     u=fa[top[u]];
143   }
144   if(dep[u]<dep[v])swap(u,v);
145   L=id[v],R=id[u];
146   update();
147 }
148 int m;
149 char op[10];
150 int main(){
151 #ifndef ONLINE_JUDGE
152   freopen("in.txt","r",stdin);
153 #endif
154   scanf("%d%d",&n,&m);
155   For(i,n)scanf("%d",&ori[i]);
156   For(i,n-1){
157     int u,v;
158     scanf("%d%d",&u,&v);
159     addEdge(u,v);
160     addEdge(v,u);
161   }
162   dfs1(1);
163   top[1]=1;
164   dfs2(1);
165   For(i,n)a[id[i]]=ori[i];
166   memset(setv,-1,sizeof(setv));
167   build();
168   while(m--){
169     int u,v,c;
170     scanf("%s",op);
171     if(op[0]=='Q'){
172       scanf("%d%d",&u,&v);
```

```
      printf("%d\n",Query(u,v));
    }else{
      scanf("%d%d%d",&u,&v,&setc);
      Modify(u,v,c);
    }
  }
  return 0;
}
```

3.2 Link-Cut Tree

```
/*水管局局长加强版：加边，维护最小生成树（森林）

*/
#include<stdio.h>
#include<algorithm>
#include<cstring>
#include<vector>
#include<map>
#include<set>
#define REP(i,b,e) for(int i=b; i<=e; i++)
#define RREP(i,b,e) for(int i=b; i>=e; i--)

int getint(){
  char ch = getchar();
  for ( ; ch > '9' || ch < '0'; ch = getchar());
  int tmp = 0;
  for ( ; '0' <= ch && ch <= '9'; ch = getchar())
    tmp = tmp * 10 + int(ch) - 48;
  return tmp;
}

using namespace std;
#define N 100010
#define M 1000010

int pa[N];
int findset(int u){
  if(pa[u]!=u)pa[u]=findset(pa[u]);
  return pa[u];
}

struct Node{
```

```
33    int v,rev,maxv;
34    int ch[2],fa,pfa;
35    inline void pushdown();
36    inline void maintain();
37    inline void change(int,int);
38  }node[N+M];
39  inline void Node::pushdown(){
40    if(rev){
41      node[ch[0]].rev^=1;
42      node[ch[1]].rev^=1;
43      swap(ch[0],ch[1]);
44      rev=0;
45    }
46  }
47  inline void Node::maintain(){
48    maxv=max(v,max(node[ch[0]].maxv,node[ch[1]].maxv));
49  }
50  inline void Node::change(int v,int cur){
51    node[v].fa=node[ch[1]].pfa=cur;
52    node[v].pfa=node[ch[1]].fa=0;
53    ch[1]=v;
54    maintain();
55  }
56  void rotate(int o,int d){
57    int k=node[o].ch[d];
58    node[o].ch[d]=node[k].ch[1^d];
59    node[node[k].ch[1^d]].fa=o;
60
61    int d2=node[node[o].fa].ch[1]==o;
62    node[node[o].fa].ch[d2]=k;
63    node[k].fa=node[o].fa;
64
65    node[k].ch[1^d]=o;
66    node[o].fa=k;
67    node[o].maintain();
68    node[k].maintain();
69  }
70  int top[N+M];
71  void splay(int cur){
72    top[0]=0;
73    while(cur){
74      top[++top[0]]=cur;
75      cur=node[cur].fa;
76    }
```

```
77    cur=top[1];
78    if(top[0]>1){
79      node[cur].pfa=node[top[top[0]]].pfa;
80      node[top[top[0]]].pfa=0;
81    }
82    RREP(i,top[0],1)
83      node[top[i]].pushdown();
84    int fa,gfa,d1,d2;
85    while(node[cur].fa){
86      fa=node[cur].fa,gfa=node[fa].fa;
87      d1=node[gfa].ch[1]==fa,d2=node[fa].ch[1]==cur;
88      if(!gfa){
89        rotate(fa,d2);
90      }else{
91        if(d1==d2){rotate(gfa,d1);rotate(fa,d2);}
92        else{rotate(fa,d2);rotate(gfa,d1);}
93      }
94    }
95    node[cur].pushdown();
96  }
97  void access(int u){
98    for(int v=0; u; u=node[v].pfa){
99      splay(u);
100     node[u].change(v,u);
101     v=u;
102   }
103 }
104 void evert(int u){
105   access(u);
106   splay(u);
107   node[u].rev^=1;
108 }
109 void cut(int u,int v){
110   evert(u);
111   access(v);
112   splay(u);
113   node[u].change(0,u);
114   node[v].pfa=0;
115 }
116 void link(int u,int v){
117   evert(v);
118   splay(v);
119   access(u);
120   splay(u);
```

```
121   node[v].pfa=u;
122 }
123 int getmax(int u,int v){
124   evert(u);
125   access(v);
126   splay(v);
127   return node[v].maxv;
128 }
129 int findmax(int ret){
130   while(node[ret].v!=node[ret].maxv){
131     node[ret].pushdown();
132     if(node[node[ret].ch[0]].maxv==node[ret].maxv)
133       ret=node[ret].ch[0];
134     else
135       ret=node[ret].ch[1];
136   }
137   return ret;
138 }
139
140 int n,m,Q;
141 struct Query{int k,x,y;}q[N];
142 struct E{int u,v,w;bool operator<(const E& rhs)const{return w<rhs.w;}}e[M];
143 map<int,int> id[N];
144 int del[M],ans[N];
145 int main(){
146 #ifdef QWERTIER
147   freopen("in","r",stdin);
148 #endif
149   n=getint();m=getint();Q=getint();
150   REP(i,1,n)pa[i]=i;
151   REP(i,1,m){ e[i].u=getint();e[i].v=getint();e[i].w=getint();}
152   sort(e+1,e+m+1);
153   REP(i,1,m)id[e[i].u][e[i].v]=id[e[i].v][e[i].u]=i;
154   REP(i,1,Q){
155     q[i].k=getint();q[i].x=getint();q[i].y=getint();
156     if(q[i].k==2)
157       del[id[q[i].x][q[i].y]]=1;
158   }
159   REP(i,1,m){
160     node[i+n].v=node[i+n].maxv=e[i].w;
161     if(del[i])continue;
162     if(findset(e[i].u)!=findset(e[i].v)){
163       pa[pa[e[i].u]]=pa[e[i].v];
164       link(e[i].u,i+n);
```

```
      link(e[i].v,i+n);
    }
  }
  RREP(i,Q,1){
    int &k=q[i].k,&x=q[i].x,&y=q[i].y;
    if(k==1){
      ans[i]=getmax(x,y);
    }else{
      int maxw=getmax(x,y);
      if(maxw<=e[id[x][y]].w)continue;
      else{
        splay(y);
        int u=findmax(y);
        splay(u);
        cut(e[u-n].u,u);
        cut(e[u-n].v,u);
        link(x,n+id[x][y]);
        link(y,n+id[x][y]);
      }
    }
  }
  REP(i,1,Q)if(q[i].k==1)printf("%d\n",ans[i]);
  return 0;
}
```

4 图论

4.1 网络流

```
namespace nf{
int le[N],pe[M],ecnt,data[M],ev[M];
void addEdge(int u,int v,int cap){
  pe[ecnt]=le[u];
  ev[ecnt]=v;
  data[ecnt]=cap;
  le[u]=ecnt++;

  pe[ecnt]=le[v];
  ev[ecnt]=u;
  data[ecnt]=0;
  le[v]=ecnt++;
}
int S,T;
void init(int s,int t){
  memset(le,-1,sizeof(le));
  ecnt=0;
  S=s,T=t;
}
void setST(int s, int t) {
  S = s, T = t;
}
int dist[N],q[N],front,rear;
int bfs(){
  memset(dist,-1,sizeof(dist));
  front=rear=0;
  q[front]=S;
  dist[S]=0;
  while(front<=rear){
    int u=q[front++];
    for(int i=le[u]; i!=-1; i=pe[i]){
      if(data[i] > 0 && dist[ev[i]]==-1){
        dist[ev[i]]=dist[u]+1;
        q[++rear]=ev[i];
        if (ev[i] == T)
          return 1;
      }
    }
  }
  return 0;
```

```
}
int cur[N];
int dfs(int u,int a){
  if(u==T || a==0)return a;
  int ret=0,f;
  for (int &i=cur[u]; i!=-1; i=pe[i]) {
    int &v=ev[i];
    if(dist[v]!=dist[u]+1)continue;
    f=dfs(v,min(a,data[i]));
    data[i]-=f;
    data[i^1]+=f;
    a-=f;
    ret+=f;
    if (a == 0)
      break;
  }
  return ret;
}
int maxFlow(){
  int ret=0;
  while(bfs()){
    memcpy(cur, le, sizeof(le));
    ret+=dfs(S,INF);
  }
  return ret;
}
}
```

5 小技巧

5.1 Emacs配置文件

```
;;compile and run
(defun comp_run ()
  (interactive)
  (save-some-buffers t)
  (setq file_name (buffer-file-name(current-buffer)))
  (setq short_file_name (substring (buffer-file-name(current-buffer)) (
      string-match "[^/]*$" file_name) (string-match "[.][A-Za-z]*$" file_name)))
  (compile (concat "g++ \"" file_name "\" -o \"" short_file_name ".out\" -lm -
      DQWERTIER -g --std=gnu++11 -Wall && ./" short_file_name ".out"))
)
(global-set-key (kbd "M-#") 'comp_run)
(global-set-key [(f9)] 'comp_run)
```

5.2 两个long long相乘，对long long取模

```
LL fMul(LL t, LL p) {
  static long double eps = 1e-15;
  LL ret = t * p - (LL)((long double)t/MOD*p+eps)*MOD;
  if (ret < 0)
    return (ret % MOD + MOD) % MOD;
  else
    return ret;
}
```

5.3 读入优化

```
namespace IStream{
const int L=1<<20;
char buffer[L], *S, *T;
inline char get_char() {
  if(S==T) {
    T = (S = buffer) + fread(buffer, 1, L, stdin);
    if (S == T) return EOF;
  }
  return *S++;
}
inline int get_int() {
  char c;
```

```
13    int re = 0, nega = 0;
14    for (c = get_char(); (c < '0' || c > '9') && c != '-'; c = get_char());
15    if (c == '-') {
16      nega = 1;
17      c = get_char();
18    }
19    while (c >= '0' && c <= '9')
20      re = (re << 1) + (re << 3) + (c - '0'), c = get_char();
21    if (!nega)
22      return re;
23    else
24      return re * -1;
25  }
26  inline char get_alpha() {
27    char c;
28    for (c = get_char(); (c < 'a' || c > 'z') && (c < 'A' || c > 'Z'); c = get_char
          ());
29    return c;
30  }
31  }
```