
Qwertier's Templates OF Algorithms And DataStructures

QWERTIER
gmayukun@gmail.com

GENERATED ON OCTOBER 20, 2016

目录

1	数学	3
1.1	素数表 (NTT专用)	3
1.2	NTT	4
1.3	FFT	6
1.4	FWT	7
1.5	拓展欧几里得	8
1.6	米勒罗宾素数测试	8
1.7	质因数分解	10
1.8	高斯消元	11
2	字符串	12
2.1	KMP算法	12
2.2	AC自动机	12
2.3	后缀数组	13
3	数据结构	15
3.1	树链剖分	15
3.2	Link-Cut Tree	19
4	图论	24
4.1	网络流	24
5	小技巧	26
5.1	Emacs配置文件	26
5.2	两个long long相乘, 对long long取模	26
5.3	读入优化	26

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

```

1  #define MOD 6597069768543436911
2  namespace NTT {
3  int m;
4  #define eps (1e-15)
5  LL fMul(LL t, LL p) {
6      LL ret = t * p - (LL)((long double)t/MOD*p+eps)*MOD;
7      if (ret < 0)
8          return (ret % MOD + MOD) % MOD;
9      else
10         return ret;
11 }
12 LL fPow(LL t, LL p) {
13     LL ret = 1;
14     while (p) {
15         if (p & 1)
16             ret = fMul(ret, t);

```

```

17     t = fMul(t, t);
18     p >>= 1;
19 }
20 return ret;
21 }
22 int rev[N];
23 LL g = 19;
24 void ntt(LL *in, LL *out, int n, int flag) { // flag = 0 if ntt, 1 if intt
25     memset(rev, 0, sizeof(rev));
26     REP (i, (1<<m)) {
27         REP (j, m) {
28             rev[i] |= ((i>>j)&1) << (m-j-1);
29         }
30     }
31     for (int i = 0; i < n; i++)
32         out[rev[i]] = in[i];
33     for (int s = 1; (1 << s) <= n; s++) {
34         int m = 1 << s, m_2 = m / 2;
35         LL wm;
36         if (!flag)
37             wm = fPow(g, (MOD - 1) / m);
38         else
39             wm = fPow(fPow(g, (MOD - 1) / m), MOD - 2);
40         for (int k = 0; k < n; k+=m) {
41             LL w = 1;
42             for (int j = 0; j < m_2; j++) {
43                 LL t = fMul(w, out[k + j + m_2]),
44                     u = out[k + j];
45                 out[k + j] = (u + t) % MOD;
46                 out[k + j + m_2] = (u - t + MOD) % MOD;
47                 w = fMul(w, wm);
48             }
49         }
50     }
51     if (flag) {
52         LL inv = fPow(n, MOD - 2);
53         REP (i, n) {
54             out[i] = fMul(out[i], inv);
55         }
56     }
57 }
58 }

```

1.3 FFT

```
1 namespace FFT {
2 int L;
3 struct cp {
4     double r,i;
5     cp(){r=i=0;}
6     cp(double _r,double _i):r(_r),i(_i){}
7     cp operator*(cp &t){return cp(r*t.r-i*t.i,i*t.r+t.i*r);}
8     cp operator+(cp &t){return cp(r+t.r,i+t.i);}
9     cp operator-(cp &t){return cp(r-t.r,i-t.i);}
10 };
11 const double PI=3.1415926535897932384626;
12 cp tmp[N];
13 int rev[N];
14 void fft(cp *a,int flag) {
15     REP(i,(1<L))tmp[i]=a[rev[i]];
16     REP(i,(1<L))a[i]=tmp[i];
17     for(int k=2; k<=(1<L); k<=1){
18         cp wn(cos(2*PI/k),flag*sin(2*PI/k));
19         for(int i=0; i<(1<L); i+=k){
20             cp w(1,0),x,y;
21             for(int j=i; j<i+k/2; j++){
22                 x=a[j];
23                 y=a[j+k/2]*w;
24                 a[j]=x+y;
25                 a[j+k/2]=x-y;
26                 w=w*wn;
27             }
28         }
29     }
30     if(flag== -1)REP(i,(1<L))a[i].r/=(1<L);
31 }
32 void calc_rev() {
33     REP(i,(1<L)){
34         rev[i] = 0;
35         REP(j,L) {
36             if((i>>j)&1)
37                 rev[i]|=(1<(L-1-j));
38         }
39     }
40 }
41 }
```

1.4 FWT

```
1 namespace FWT {
2 template <typename T>
3 void or_fwt(T X[], int l, int r) {
4     if (l == r)
5         return;
6     int m = (l + r) >> 1;
7     or_fwt(X, l, m); or_fwt(X, m + 1, r);
8     for (int i = 0; i <= m - 1; i++) {
9         X[m + 1 + i] += X[l + i];
10    }
11 }
12 template <typename T>
13 void or_ifwt(T X[], int l, int r) {
14     if (l == r)
15         return;
16     int m = (l + r) >> 1;
17     or_ifwt(X, l, m); or_ifwt(X, m + 1, r);
18     for (int i = 0; i <= m - 1; i++) {
19         X[m + 1 + i] -= X[l + i];
20    }
21 }
22 template <typename T>
23 void and_fwt(T X[], int l, int r) {
24     if (l == r)
25         return;
26     int m = (l + r) >> 1;
27     and_fwt(X, l, m); and_fwt(X, m + 1, r);
28     for (int i = 0; i <= m - 1; i++) {
29         X[l + i] += X[m + 1 + i];
30    }
31 }
32 template <typename T>
33 void and_ifwt(T X[], int l, int r) {
34     if (l == r)
35         return;
36     int m = (l + r) >> 1;
37     and_ifwt(X, l, m); and_ifwt(X, m + 1, r);
38     for (int i = 0; i <= m - 1; i++) {
39         X[l + i] -= X[m + 1 + i];
40    }
41 }
42 template <typename T>
```

```

43 void xor_fwt(T X[], int l, int r) {
44     if (l == r)
45         return;
46     int m = (l + r) >> 1;
47     xor_fwt(X, l, m); xor_fwt(X, m + 1, r);
48     for (int i = 0; i <= m - 1; i++) {
49         X[l + i] += X[m + 1 + i];
50         X[m + 1 + i] = X[l + i] - 2 * X[m + 1 + i];
51     }
52 }
53 template <typename T>
54 void xor_ifwt(T X[], int l, int r) {
55     if (l == r)
56         return;
57     int m = (l + r) >> 1;
58     for (int i = 0; i <= m - 1; i++) {
59         X[l + i] += X[m + 1 + i];
60         X[m + 1 + i] = X[l + i] - 2 * X[m + 1 + i];
61         X[l+i] /= 2;
62         X[m + 1 + i] /= 2;
63     }
64     xor_ifwt(X, l, m); xor_ifwt(X, m + 1, r);
65 }
66 }

```

1.5 拓展欧几里得

```

1 LL exgcd(LL a, LL b, LL &x, LL &y) {
2     if (a==0&&b==0) return -1;
3     if (b==0) {x=1;y=0;return a;}
4     LL d=extend_gcd(b,a%b,y,x);
5     y-=a/b*x;
6     return d;
7 }

```

1.6 米勒罗宾素数测试

```

1 const int S = 8;
2 long long mult_mod(long long a,long long b,long long c) {
3     a %= c;
4     b %= c;
5     long long ret = 0;

```



```

6   long long tmp = a;
7   while(b) {
8       if(b & 1) {
9           ret += tmp;
10          if(ret > c) ret -= c;
11      }
12      tmp <<= 1;
13      if(tmp > c) tmp -= c;
14      b >>= 1;
15  }
16  return ret;
17 }
18 long long pow_mod(long long a, long long n, long long mod) {
19     long long ret = 1;
20     long long temp = a%mod;
21     while(n) {
22         if(n & 1) ret = mult_mod(ret, temp, mod);
23         temp = mult_mod(temp, temp, mod);
24         n >>= 1;
25     }
26     return ret;
27 }
28 bool check(long long a, long long n, long long x, long long t) {
29     long long ret = pow_mod(a, x, n);
30     long long last = ret;
31     for(int i = 1; i <= t; i++) {
32         ret = mult_mod(ret, ret, n);
33         if(ret == 1 && last != 1 && last != n-1) return true; //合数
34         last = ret;
35     }
36     if(ret != 1) return true;
37     else return false;
38 }
39 bool Miller_Rabin(long long n) {
40     if( n < 2) return false;
41     if( n == 2) return true;
42     if( (n&1) == 0) return false; //偶数
43     long long x = n - 1;
44     long long t = 0;
45     while( (x&1) == 0 ) { x >>= 1; t++; }
46     srand(time(NULL));
47     for(int i = 0; i < S; i++) {
48         long long a = rand()%(n-1) + 1;
49         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;

```

```

37 while( p >= n)
38     p = pollard_rho(p,c--); //值变化, 防止死循环k
39 findfac(p,k);
40 findfac(n/p,k);
41 }

```

1.8 高斯消元

```

1 bool gauss(double *m[N], int n) {
2     for(int i=0; i<n; i++){
3         int c=i;
4         for(int j=i; j<n; j++) if(fabs(m[j][i])>fabs(m[c][i])) c=j;
5         if (fabs(m[c][i]) < eps)
6             return false;
7         for(int j=0; j<=n; j++) swap(m[i][j],m[c][j]);
8         for(int j=i+1; j<n; j++){
9             double f=m[j][i]/m[i][i];
10            for(int k=n; k>=i; k--)
11                m[j][k]-=m[j][i]*m[i][k]/m[i][i];
12        }
13    }
14    for(int i=n-1; i>=0; i--){
15        for(int j=i+1; j<n; j++)
16            m[i][n]-=m[j][n]*m[i][j];
17        m[i][n]/=m[i][i];
18    }
19    return true;
20 }

```

2 字符串

2.1 KMP算法

```
1 namespace KMP {
2 int fa[L];
3 void get_fa(char *s, int n) {
4     int j = fa[0] = -1;
5     FOR (i, n - 1) {
6         while (j != -1 && s[j + 1] != s[i])
7             j = fa[j];
8         if (s[j + 1] == s[i])
9             j++;
10        fa[i] = j;
11    }
12 }
13 int find(char *s, int n, char *t, int m) {
14     int j = -1, ret = 0;
15     REP (i, m) {
16         while (j != -1 && s[j + 1] != t[i])
17             j = fa[j];
18         if (s[j + 1] == t[i])
19             j++;
20         if (j == n - 1) {
21             ret++;
22             j = fa[j];
23         }
24     }
25     return ret;
26 }
27 }
```

2.2 AC自动机

```
1 namespace AC {
2 int sz, ch[SZ][26], val[SZ];
3 void insert(char *str, int v) {
4     int u = 0, len = 0;
5     while (*str) {
6         int c = (*str++) - 'a';
7         len++;
8         if (!ch[u][c]) {
9             ch[u][c] = ++sz;
```

```

10     memset(ch[sz], 0, sizeof(ch[sz]));
11     val[sz] = 0;
12 }
13 u = ch[u][c];
14 }
15 val[u] += v;
16 }
17 int fr, rr, que[SZ], fa[SZ], lst[SZ];
18 void calc_fa() {
19     fr=0,rr=-1;
20     REP (i, 26) {
21         if(ch[0][i]) {
22             que[++rr]=ch[0][i];
23             fa[ch[0][i]] = 0;
24             lst[ch[0][i]] = 0;
25         }
26     }
27     while (fr <= rr) {
28         int r = que[fr++];
29         REP (i,26) {
30             int u=ch[r][i];
31             if (!u) {
32                 ch[r][i] = ch[fa[r]][i];
33                 continue;
34             }
35             fa[u] = ch[fa[r]][i];
36             lst[u] = val[fa[u]] ? fa[u] : lst[fa[u]];
37             val[u] += val[lst[u]];
38             que[++rr]=u;
39         }
40     }
41 }
42 void init() {
43     sz = 0;
44     memset(ch[0], 0, sizeof(ch[0]));
45     val[0] = 0;
46     lst[0] = 0;
47 }
48 }

```

2.3 后缀数组

```

1 namespace SA{

```

```

2  int sa[N], cnt[N], *x = new int[N], *y = new int[N];
3  inline bool eq(int i, int j) {
4      return (i >= n && j >= n) || (i < n && j < n && y[i] == y[j]);
5  }
6  void calc_sa() {
7      memset(cnt, 0, sizeof(int)*m);
8      REP(i, n) cnt[x[i] = s[i] - 'a']++;
9      FOR(i, m-1) cnt[i] += cnt[i-1];
10     for(int i = n-1; i >= 0; i--) sa[--cnt[x[i]]] = i;
11     for(int k = 1; k <= n; k <= 1) {
12         int p = 0;
13         for(int i = n-k; i < n; i++) y[p++] = i;
14         REP(i, n) if(sa[i] >= k) y[p++] = sa[i] - k;
15
16         REP(i, m) cnt[i] = 0;
17         REP(i, n) cnt[x[y[i]]]++;
18         FOR(i, m-1) cnt[i] += cnt[i-1];
19         for(int i = n-1; i >= 0; i--) sa[--cnt[x[y[i]]]] = y[i];
20
21         swap(x, y);
22         x[sa[0]] = 0;
23         p = 1;
24         FOR(i, n-1)
25             x[sa[i]] = eq(sa[i], sa[i-1]) && eq(sa[i] + k, sa[i-1] + k) ? p-1 : p++;
26         if(p >= n) break;
27         m = p;
28     }
29 }
30 int h[N];
31 void calc_hi() {
32     int j, k = 0;
33     REP(i, n) {
34         if(x[i] == 0) {k = 0; continue;}
35         if(k) k--;
36         j = sa[x[i]-1];
37         while(s[i+k] == s[j+k]) k++;
38         h[x[i]] = k;
39     }
40 }
41 }

```

3 数据结构

3.1 树链剖分

```
1  /*
2   SDOI 2012 染色: 给定一棵树, 节点有色。现有两种查询:
3   1. 修改某点的颜色
4   2. 查询一条链上不同的颜色段数
5  */
6  #include <algorithm>
7  #include <stdio.h>
8  #include <string.h>
9  #include <map>
10 #define PROB
11 #define N 100010
12 #define For(i,n) for(int i=1; i<=n; i++)
13 using namespace std;
14
15 int n;
16
17 int le[N], pe[N<<1], ev[N<<1], ecnt;
18 void addEdge(int u, int v) {
19     ecnt++;
20     pe[ecnt]=le[u];
21     le[u]=ecnt;
22     ev[ecnt]=v;
23 }
24
25 int ori[N], a[N];
26
27 #define lc (o<<1)
28 #define rc (o<<1|1)
29 #define mid ((l+r)>>1)
30 int cl[N<<2], cr[N<<2], sumv[N<<2], setv[N<<2], L, R, setc;
31 inline void maintain(int o) {
32     cl[o]=cl[lc];
33     cr[o]=cr[rc];
34     sumv[o]=sumv[lc]+sumv[rc]-(cr[lc]==cl[rc]);
35 }
36 inline void pushdown(int o) {
37     if(setv[o]!=-1) {
38         cl[lc]=cr[lc]=setv[lc]=setv[o];
39         sumv[lc]=1;
40         cl[rc]=cr[rc]=setv[rc]=setv[o];
```

```

41     sumv[rc]=1;
42     setv[o]=-1;
43 }
44 }
45 void build(int o=1,int l=1,int r=n){
46     if(l==r){
47         cl[o]=cr[o]=a[l];
48         sumv[o]=1;
49         return;
50     }
51     build(lc,l,mid);
52     build(rc,mid+1,r);
53     maintain(o);
54 }
55 void update(int o=1,int l=1,int r=n){
56     if(L<=l&&R<=r){
57         cl[o]=cr[o]=setv[o]=setc;
58         sumv[o]=1;
59         return;
60     }
61     pushdown(o);
62     if(L<=mid) update(lc,l,mid);
63     if(R>mid) update(rc,mid+1,r);
64     maintain(o);
65 }
66 struct ANS{
67     int cl,cr,sumv;
68     ANS(int a,int b,int c):cl(a),cr(b),sumv(c){}
69     ANS operator+(const ANS& r){
70         return ANS(cl,r.cr,sumv+r.sumv-(cr==r.cl));
71     }
72 };
73 ANS query(int o=1,int l=1,int r=n){
74     if(L<=l&&R<=r)
75         return ANS(cl[o],cr[o],sumv[o]);
76     pushdown(o);
77     if(R<=mid) return query(lc,l,mid);
78     if(L>mid) return query(rc,mid+1,r);
79     return query(lc,l,mid)+query(rc,mid+1,r);
80 }
81
82 int fa[N],sz[N],hson[N],dep[N];
83 void dfs1(int u){
84     sz[u]=1;

```



```

85     for(int i=le[u]; i; i=pe[i]){
86         int &v=ev[i];
87         if(v==fa[u]) continue;
88         fa[v]=u;
89         dep[v]=dep[u]+1;
90         dfs1(v);
91         sz[u]+=sz[v];
92         if(sz[v]>sz[hson[u]]) hson[u]=v;
93     }
94 }
95 int dfs_clock,top[N],id[N];
96 void dfs2(int u){
97     id[u]=++dfs_clock;
98     if(hson[u]){
99         top[hson[u]]=top[u];
100        dfs2(hson[u]);
101    }
102    for(int i=le[u]; i; i=pe[i]){
103        int &v=ev[i];
104        if(v==fa[u] || v==hson[u]) continue;
105        top[v]=v;
106        dfs2(v);
107    }
108 }
109
110 int Query(int u,int v){
111     ANS ans1(-1,-1,0),ans2(-2,-2,0);
112     while(top[u]!=top[v]){
113         if(dep[top[u]]>dep[top[v]]){
114             L=id[top[u]],R=id[u];
115             ANS q=query();
116             ans1=q+ans1;
117             u=fa[top[u]];
118         }else{
119             L=id[top[v]],R=id[v];
120             ANS q=query();
121             ans2=q+ans2;
122             v=fa[top[v]];
123         }
124     }
125     if(dep[u]>dep[v]){
126         L=id[v],R=id[u];
127         ANS q=query();
128         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);

```

```

173     printf("%d\n",Query(u,v));
174 }else{
175     scanf("%d%d%d",&u,&v,&setc);
176     Modify(u,v,c);
177 }
178 }
179 return 0;
180 }

```

3.2 Link-Cut Tree

```

1  /*水管局局长加强版：加边，维护最小生成树（森林）
2
3  */
4  #include<stdio.h>
5  #include<algorithm>
6  #include<cstring>
7  #include<vector>
8  #include<map>
9  #include<set>
10 #define REP(i,b,e) for(int i=b; i<=e; i++)
11 #define RREP(i,b,e) for(int i=b; i>=e; i--)
12
13 int getint(){
14     char ch = getchar();
15     for ( ; ch > '9' || ch < '0'; ch = getchar());
16     int tmp = 0;
17     for ( ; '0' <= ch && ch <= '9'; ch = getchar())
18         tmp = tmp * 10 + int(ch) - 48;
19     return tmp;
20 }
21
22 using namespace std;
23 #define N 100010
24 #define M 1000010
25
26 int pa[N];
27 int findset(int u){
28     if(pa[u]!=u)pa[u]=findset(pa[u]);
29     return pa[u];
30 }
31
32 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     #ifndef 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);

```

```

165     link(e[i].v,i+n);
166 }
167 }
168 RREP(i,Q,1){
169     int &k=q[i].k,&x=q[i].x,&y=q[i].y;
170     if(k==1){
171         ans[i]=getmax(x,y);
172     }else{
173         int maxw=getmax(x,y);
174         if(maxw<=e[id[x][y]].w) continue;
175         else{
176             splay(y);
177             int u=findmax(y);
178             splay(u);
179             cut(e[u-n].u,u);
180             cut(e[u-n].v,u);
181             link(x,n+id[x][y]);
182             link(y,n+id[x][y]);
183         }
184     }
185 }
186 REP(i,1,Q) if(q[i].k==1) printf("%d\n",ans[i]);
187 return 0;
188 }

```

4 图论

4.1 网络流

```
1 namespace nf{
2 int le[N],pe[M],ecnt,data[M],ev[M];
3 void addEdge(int u,int v,int cap){
4     pe[ecnt]=le[u];
5     ev[ecnt]=v;
6     data[ecnt]=cap;
7     le[u]=ecnt++;
8
9     pe[ecnt]=le[v];
10    ev[ecnt]=u;
11    data[ecnt]=0;
12    le[v]=ecnt++;
13 }
14 int S,T;
15 void init(int s,int t){
16     memset(le,-1,sizeof(le));
17     ecnt=0;
18     S=s,T=t;
19 }
20 void setST(int s, int t) {
21     S = s, T = t;
22 }
23 int dist[N],q[N],front,rear;
24 int bfs(){
25     memset(dist,-1,sizeof(dist));
26     front=rear=0;
27     q[front]=S;
28     dist[S]=0;
29     while(front<=rear){
30         int u=q[front++];
31         for(int i=le[u]; i!=-1; i=pe[i]){
32             if(data[i] > 0 && dist[ev[i]]==-1){
33                 dist[ev[i]]=dist[u]+1;
34                 q[++rear]=ev[i];
35                 if (ev[i] == T)
36                     return 1;
37             }
38         }
39     }
40     return 0;
```



```

41 }
42 int cur[N];
43 int dfs(int u, int a) {
44     if (u == T || a == 0) return a;
45     int ret = 0, f;
46     for (int &i = cur[u]; i != -1; i = pe[i]) {
47         int &v = ev[i];
48         if (dist[v] != dist[u] + 1) continue;
49         f = dfs(v, min(a, data[i]));
50         data[i] -= f;
51         data[i ^ 1] += f;
52         a -= f;
53         ret += f;
54         if (a == 0)
55             break;
56     }
57     return ret;
58 }
59 int maxFlow() {
60     int ret = 0;
61     while (bfs()) {
62         memcpy(cur, le, sizeof(le));
63         ret += dfs(S, INF);
64     }
65     return ret;
66 }
67 }

```

5 小技巧

5.1 Emacs配置文件

```
1 ;;compile and run
2 (defun comp_run ()
3   (interactive)
4   (save-some-buffers t)
5   (setq file_name (buffer-file-name(current-buffer)))
6   (setq short_file_name (substring (buffer-file-name(current-buffer)) (
7     string-match "[^/]*$" file_name) (string-match "[A-Za-z]*$" file_name)))
8   (compile (concat "g++ \"" file_name "\" -o \"" short_file_name ".out\" -lm -
9     DQWERTIER -g --std=gnu++11 -Wall && ./" short_file_name ".out"))
10 )
11 (global-set-key (kbd "M-#") 'comp_run)
12 (global-set-key [(f9)] 'comp_run)
```

5.2 两个long long相乘，对long long取模

```
1 LL fMul(LL t, LL p) {
2   static long double eps = 1e-15;
3   LL ret = t * p - (LL)((long double)t/MOD*p+eps)*MOD;
4   if (ret < 0)
5     return (ret % MOD + MOD) % MOD;
6   else
7     return ret;
8 }
```

5.3 读入优化

```
1 namespace IStream{
2   const int L=1<<20;
3   char buffer[L], *S, *T;
4   inline char get_char() {
5     if(S==T) {
6       T = (S = buffer) + fread(buffer, 1, L, stdin);
7       if (S == T) return EOF;
8     }
9     return *S++;
10  }
11  inline int get_int() {
12    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  }

```