CODE STANDARD LIBRARY

Last build at October 26, 2019

Contents

1 string

1.1 Manacher

```
vector<int> d1(n);
for (int i = 0, l = 0, r = -1; i < n; i++) {
    int k = (i > r) ? 1 : min(d1[l + r - i], r - i);
    while (0 ⩽ i - k && i + k < n && s[i - k] == s[i + k]) {
    k++;
}
    d1[i] = k--;
    if (i + k > r) {
        l = i - k;
        r = i + k;
    }
}
```

1.2 Minimum representation

```
int k = 0, i = 0, j = 1;
while (k < n && i < n && j < n) {
    if (sec[(i + k) % n] == sec[(j + k) % n]) {
        k++;
    } else {
        sec[(i + k) % n] > sec[(j + k) % n] ? i = i + k + 1 : j = j
            + k + 1;
        if (i == j) i++;
        k = 0;
    }
}
i = min(i, j);
```

1.3 AC automation

```
const int N = 156, L = 1e6 + 6;
namespace AC {
const int SZ = N * 80;
int tot, tr[SZ][26];
int fail[SZ], idx[SZ], val[SZ];
int cnt[N];  // 记录第 i 个字符串的出现次数
void init() {
    memset(fail, 0, sizeof(fail));
    memset(tr, 0, sizeof(tr));
    memset(val, 0, sizeof(val));
    memset(cnt, 0, sizeof(cnt));
    memset(idx, 0, sizeof(idx));
    tot = 0;
}
```

```
void insert(char *s, int id) {  // id 表示原始字符串的编号
    int u = 0;
    for (int i = 1; s[i]; i++) {
        if (!tr[u][s[i] - 'a']) tr[u][s[i] - 'a'] = ++tot;
        u = tr[u][s[i] - 'a'];
    }
    idx[u] = id;
}
queue<int> q;
void build() {
    for (int i = 0; i < 26; i++)
        if (tr[0][i]) q.push(tr[0][i]);
    while (q.size()) {
        int u = q.front();
        q.pop();
        for (int i = 0; i < 26; i++) {
        if (tr[u][i])
            fail[tr[u][i]] = tr[fail[u]][i], q.push(tr[u][i]);
        else
            tr[u][i] = tr[fail[u]][i];
        }
    }
}
int query(char *t) {  // 返回最大的出现次数
    int u = 0, res = 0;
    for (int i = 1; t[i]; i++) {
        u = tr[u][t[i] - 'a'];
        for (int j = u; j; j = fail[j]) val[j]++;
    }
    for (int i = 0; i ⩽ tot; i++)
        if (idx[i]) res = max(res, val[i]), cnt[idx[i]] = val[i];
    return res;
    }
}  // namespace AC
```

1.4 suffix automation

```
const int N = 2000010;
const int CHAR_SET_SIZE=26;
int c[N],dfn[N<<1];
struct SAM{
    int ch[N<<1][CHAR_SET_SIZE],fa[N<<1],len[N<<1],siz[N<<1];
    int tot,last;
    int newnode(){
        ++tot;
        memset(ch[tot],0,sizeof ch[tot]);
        fa[tot]=0;
        return tot;
```

```
12      }
13      void init(){
14          tot=1;
15          last=1;
16          len[1]=0;
17          memset(ch[1],0,sizeof ch[1]);
18          memset(siz,0,sizeof siz);
19      }
20      void add(int x){
21          int pos = last,newpos=newnode();
22          last = newpos;
23          siz[newpos]=1;
24          len[newpos]=len[pos]+1;
25          while(pos&&!ch[pos][x]){ch[pos][x]=newpos;pos=fa[pos];}
26          if(!pos)fa[newpos]=1;
27          else{
28              int oldpos=ch[pos][x];
29              if(len[oldpos]==len[pos]+1)fa[newpos]=oldpos;
30              else{
31                  int anp=newnode();
32                  memcpy(ch[anp],ch[oldpos],sizeof ch[anp]);
33                  fa[anp]=fa[oldpos];
34                  len[anp]=len[pos]+1;
35                  fa[oldpos]=fa[newpos]=anp;
36                  while(pos&&ch[pos][x]==oldpos){
37                      ch[pos][x]=anp;
38                      pos=fa[pos];
39                  }
40              }
41          }
42      }
43      long long solve(){
44          long long ans=0;
45          memset(c,0,sizeof c);
46          for(int i=1;i<=tot;i++){c[len[i]]++;}
47          for(int i=1;i<=tot;i++)c[i]+=c[i-1];
48          for(int i=1;i<=tot;i++)dfn[c[len[i]]--]=i;
49          for(int i=tot;i>=1;--i){
50              int now = dfn[i];
51              siz[fa[now]]+=siz[now];
52              if(siz[now]>1)ans=max(ans,1ll*len[now]*siz[now]);
53          }
54          return ans;
55      }
56  }sam;
```

1.5 PAM

```
class PA {
    private:
        static const int N = 100010;
        struct Node {
            int len;
            int ptr[26], fail;
            Node(int len = 0) : len(len), fail(0) { memset(ptr, 0,
               sizeof(ptr)); }
        } nd[N];

        int size, cnt;  // size为字符串长度，cnt为节点个数
        int cur;  //当前指针停留的位置，即最后插入字符所对应的节点
        char s[N];

        int getfail(int x)  //沿着fail指针找到第一个回文后缀
        {
        while (s[size - nd[x].len - 1] != s[size]) {
            x = nd[x].fail;
        }
        return x;
        }

    public:
        PA() : size(0), cnt(0), cur(0) {
            nd[cnt] = Node(0);
            nd[cnt].fail = 1;
            nd[++cnt] = Node(-1);
            nd[cnt].fail = 0;
            s[0] = '$';
        }

        void extend(char c) {
            s[++size] = c;
            int now = getfail(cur);  //找到插入的位置
            if (!nd[now].ptr[c - 'a'])  //若没有这个节点，则新建并
               求出它的fail指针
            {
                int tmp = ++cnt;
                nd[tmp] = Node(nd[now].len + 2);
                nd[tmp].fail = nd[getfail(nd[now].fail)].ptr[c - 'a
                   '];
                nd[now].ptr[c - 'a'] = tmp;
            }
            cur = nd[now].ptr[c - 'a'];
        }

        int qlen() { return nd[cur].len; }
} A, B;
```

2 math

2.1 prime/phi/mu

```
void pj() {
    int n = 10000000;
    pr = 0;
    mu[1] = 1;
    for (int i = 2; i ⩽ n; i++) {
        if (v[i] == 0) {
            v[i] = i;
            prime[pr++] = i;
            phi[i] = i - 1;
            mu[i] = -1;
        }
        for (int j = 0; j < pr && n / i ⩾ prime[j]; j++) {
            v[i * prime[j]] = prime[j];
            if (v[i] == prime[j])
                mu[i * prime[j]] = 0;
            else
                mu[i * prime[j]] = -mu[i];
            phi[i*prime[j]] = phi[i] * (i%prime[j]?prime[j]-1:prime
               [j]);
            if (v[i] ⩽ prime[j]) break;
        }
    }
}
```

2.2 Congruence Equation

```
int ex_gcd(int a, int b, int& x, int& y) {
    if (b == 0) {
        x = 1;
        y = 0;
        return a;
    }
    int d = ex_gcd(b, a % b, x, y);
    int temp = x;
    x = y;
    y = temp - a / b * y;
    return d;
}
bool liEu(int a, int b, int c, int& x, int& y) {
    int d = ex_gcd(a, b, x, y);
    if (c % d ≠ 0) return 0;
    int k = c / d;
    x *= k;
    y *= k;
    return 1;
```

```
}
```

2.3 CRT

```
lt ai[maxn],bi[maxn];

lt mul(lt a,lt b,lt mod)
{
    lt res=0;
    while(b>0)
    {
        if(b&1) res=(res+a)%mod;
        a=(a+a)%mod;
        b>>=1;
    }
    return res;
}

lt exgcd(lt a,lt b,lt &x,lt &y)
{
    if(b==0){x=1;y=0;return a;}
    lt gcd=exgcd(b,a%b,x,y);
    lt tp=x;
    x=y; y=tp-a/b*y;
    return gcd;
}

lt excrt()
{
    lt x,y,k;
    lt M=bi[1],ans=ai[1];//第一个方程的解特判
    for(int i=2;i<=n;i++)
    {
        lt a=M,b=bi[i],c=(ai[i]-ans%b+b)%b;//ax≡c(mod b)
        lt gcd=exgcd(a,b,x,y),bg=b/gcd;
        if(c%gcd!=0) return -1; //判断是否无解，然而这题其实不用

        x=mul(x,c/gcd,bg);
        ans+=x*M;//更新前k个方程组的答案
        M*=bg;//M为前k个m的lcm
        ans=(ans%M+M)%M;
    }
    return (ans%M+M)%M;
}
```

2.4 dyhshai

```
ll T, n, pri[maxn], cur, mu[maxn], sum_mu[maxn];
```

```
bool vis[maxn];
map<ll, ll> mp_mu;
ll S_mu(ll x) {
    if (x < maxn) return sum_mu[x];
    if (mp_mu[x]) return mp_mu[x];
    ll ret = 1ll;
    for (ll i = 2, j; i ⩽ x; i = j + 1) {
        j = x / (x / i);
        ret -= S_mu(x / i) * (j - i + 1);
    }
    return mp_mu[x] = ret;
}
ll S_phi(ll x) {
    ll ret = 0ll;
    for (ll i = 1, j; i ⩽ x; i = j + 1) {
        j = x / (x / i);
        ret += (S_mu(j) - S_mu(i - 1)) * (x / i) * (x / i);
    }
    return ((ret - 1) >> 1) + 1;
}
int main() {
    scanf("%lld", &T);
    mu[1] = 1;
    for (int i = 2; i < maxn; i++) {
    if (!vis[i]) {
        pri[++cur] = i;
        mu[i] = -1;
    }
    for (int j = 1; j ⩽ cur && i * pri[j] < maxn; j++) {
        vis[i * pri[j]] = true;
        if (i % pri[j])
            mu[i * pri[j]] = -mu[i];
        else {
            mu[i * pri[j]] = 0;
            break;
        }
    }
    }
    for (int i = 1; i < maxn; i++) sum_mu[i] = sum_mu[i - 1] + mu[i
       ];
    while (T--) {
        scanf("%lld", &n);
        printf("%lld %lld\n", S_phi(n), S_mu(n));
    }
    return 0;
}
```

2.5 FFT

```
const int N = 4000005;
const double PI = acos ( -1. );
int r[N];
struct Complex {
    double r, i;
    Complex ( double _r = 0.0, double _i = 0.0 ) : r ( _r ), i ( _i ) {};
    inline void real ( const double &x ) {
        r = x;
    }
    inline double real() {
        return r;
    }
    inline Complex operator+ ( const Complex x ) const {
        return Complex ( r + x.r, i + x.i );
    }
    inline Complex operator- ( const Complex x ) const {
        return Complex ( r - x.r, i - x.i );
    }
    inline Complex operator* ( const Complex x ) const {
        return Complex ( r * x.r - i * x.i, x.r * i + r * x.i );
    }
    inline void operator/= ( const double x ) {
        r /= x;
    }
};
Complex a[N], b[N];
void change ( Complex y[], int len ) {
    for ( int i = 0; i < len; i++ ) {
        if ( i < r[i] ) {
            swap ( y[i], y[r[i]] );
        }
    }
}
void fft ( Complex y[], int len, int on ) {
    change ( y, len );
    for ( int m = 1; m < len; m <<= 1 ) {
        Complex wn ( cos ( PI / m ), sin ( on * PI / m ) );
        for ( int r = m << 1, j = 0; j < len; j += r ) {
            Complex w ( 1.0, 0 );
            for ( int k = 0; k < m; k++, w = w * wn ) {
                Complex u = y[j + k];
                Complex t = w * y[j + k + m];
                y[j + k] = u + t;
                y[j + k + m] = u - t;
            }
```

```
        }
    }
    if ( on == -1 )
        for ( int i = 0; i < len; i++ ) {
            y[i] /= len;
        }
}
int main() {
    int n, m;
    while ( ~scanf ( "%d %d", &n, &m ) ) {
        int l = 0, len = 1;
        while ( len <= n+m ) {
            len <<= 1;
            l++;
        }
        for ( int i = 0; i < len; i++ ) {
            r[i] = ( r[i >> 1] >> 1 ) | ( ( i & 1 ) << ( l - 1 ) );
        }
        int tmp;
        for ( int i = 0; i <= n; i++ ) {
            scanf ( "%d", &tmp );
            a[i] = Complex ( tmp * 1.0, 0 );
        }
        for ( int i = n + 1; i < len; i++ ) {
            a[i] = Complex ( 0, 0 );
        }
        for ( int i = 0; i <= m; i++ ) {
            scanf ( "%d", &tmp );
            b[i] = Complex ( tmp * 1.0, 0 );
        }
        for ( int i = m + 1; i < len; i++ ) {
            b[i] = Complex ( 0, 0 );
        }
        fft ( a, len, 1 );
        fft ( b, len, 1 );
        for ( int i = 0; i < len; i++ ) {
            a[i] = a[i] * b[i];
        }
        fft ( a, len, -1 );
        for ( int i = 0; i <= n + m; i++ ) {
            printf ( "%lld%c", ( long long ) ( a[i].r + 0.5 ), i ==
                n + m ? '\n' : ' ' );
        }
    }
}
```

2.6 NTT

```
const int N = 4000005;
const double PI = acos ( -1. );
const int mod=998244353, G = 3, Gi = 332748118;//gmin=3;
long long quickpow(long long a,long long n){
    if(a==0)return 0;
    long long ans=1;
    while(n){
        if(n&1)ans=ans*a%mod;
        n>>=1;
        a=a*a%mod;
    }
    return ans;
}
int r[N];
long long a[N],b[N];
long long inv;
void exgcd(long long a,long long b,long long &x,long long &y){
    if(b){
        exgcd(b,a%b,y,x);
        y-=x*(a/b);
    }else{
        x=1;
        y=0;
        return ;
    }
}
long long gao(long long a){
    long long x,y;
    exgcd(a,mod,x,y);
    if(x<0)x+=mod;
    return x;
}
inline void change ( long long y[], int len ) {
    for ( int i = 0; i < len; i++ ) {
        if ( i < r[i] ) {
            swap ( y[i], y[r[i]] );
        }
    }
}
void fft ( long long y[], int len, int on ) {
    change ( y, len );
    for ( int m = 1; m < len; m <<= 1 ) {
        long long wn =quickpow(on==1?G:Gi,(mod-1)/(m<<1));
        for ( int r = m << 1, j = 0; j < len; j += r ) {
            long long  w=1;
            for ( int k = 0; k < m; k++, w = w * wn%mod ) {
                long long u = y[j + k];
                long long t = w * y[j + k + m]%mod;
```

```
                y[j + k] =(u + t)%mod;
                y[j + k + m] = (u - t+mod)%mod;
            }
        }
    }
    if ( on == -1 )
        for ( int i = 0; i < len; i++ ) {
            y[i]=y[i]*inv%mod;
        }
}
int main() {
    int n, m;
    while ( ~scanf ( "%d %d", &n, &m ) ) {
        int l = 0, len = 1;
        while ( len <= n+m ) {
            len <<= 1;
            l++;
        }
        inv=gao(len);
        for ( int i = 0; i < len; i++ ) {
            r[i] = ( r[i >> 1] >> 1 ) | ( ( i & 1 ) << ( l - 1 ) );
        }
        for ( int i = 0; i <= n; i++ ) {
            scanf ( "%lld", &a[i] );
        }
        for ( int i = n + 1; i < len; i++ ) {
            a[i] =0;
        }
        for ( int i = 0; i <= m; i++ ) {
            scanf ( "%lld", &b[i] );
        }
        for ( int i = m + 1; i < len; i++ ) {
            b[i] = 0;
        }
        fft ( a, len, 1 );
        fft ( b, len, 1 );
        for ( int i = 0; i < len; i++ ) {
            a[i] = a[i] * b[i]%mod;
        }
        fft ( a, len, -1 );
        for ( int i = 0; i <= n + m; i++ ) {
            printf ( "%lld%c", a[i], i == n + m ? '\n' : ' ' );
        }
    }
}
```

2.7 FWT

```
long long inv;
long long a1[150000], b1[150000];
long long a2[150000], b2[150000];
long long a3[150000], b3[150000];
void FWT1 ( long long n ) {
    for ( int d = 1; d < n; d <<= 1 )
        for ( int m = d << 1, i = 0; i < n; i += m )
            for ( int j = 0; j < d; j++ ) {
                long long  x1 = a1[i + j], y1 = a1[i + j + d];
                long long  x2 = a2[i + j], y2 = a2[i + j + d];
                long long  x3 = a3[i + j], y3 = a3[i + j + d];
                a1[i + j + d] =  ( x1 + y1 ) % mod ;
                a2[i + j]     =  ( x2 + y2 ) % mod ;
                a3[i + j]     =  ( x3 + y3 ) % mod ;
                a3[i + j + d] =  ( x3 - y3 + mod ) % mod ;
                //xor:a[i+j]=x+y,a[i+j+d]=(x-y+mod)%mod;
                //and:a[i+j]=x+y;
                //or:a[i+j+d]=x+y;
            }
}
void FWT2 ( long long n ) {
    for ( int d = 1; d < n; d <<= 1 )
        for ( int m = d << 1, i = 0; i < n; i += m )
            for ( int j = 0; j < d; j++ ) {
                long long  x1 = b1[i + j], y1 = b1[i + j + d];
                long long  x2 = b2[i + j], y2 = b2[i + j + d];
                long long  x3 = b3[i + j], y3 = b3[i + j + d];
                b1[i + j + d] =  ( x1 + y1 ) % mod ;
                b2[i + j]     =  ( x2 + y2 ) % mod ;
                b3[i + j]     =  ( x3 + y3 ) % mod ;
                b3[i + j + d] =  ( x3 - y3 + mod ) % mod ;
                //xor:a[i+j]=x+y,a[i+j+d]=(x-y+mod)%mod;
                //and:a[i+j]=x+y;
                //or:a[i+j+d]=x+y;
            }
}
void UFWT ( long long n ) {
    for ( long long d = 1; d < n; d <<= 1 )
        for ( long long  m = d << 1, i = 0; i < n; i += m )
            for ( long long  j = 0; j < d; j++ ) {
                long long  x1 = a1[i + j], y1 = a1[i + j + d];
                long long  x2 = a2[i + j], y2 = a2[i + j + d];
                long long  x3 = a3[i + j], y3 = a3[i + j + d];
                a1[i + j + d] =  ( y1 - x1 + mod) % mod ;
                a2[i + j]     =  ( x2 - y2 + mod ) % mod ;
                a3[i + j]     =  ( x3 + y3 ) * inv % mod ;
                a3[i + j + d] =  ( x3 - y3 + mod ) * inv % mod ;
                //xor:a[i+j]=(x+y)/2,a[i+j+d]=(x-y)/2;
                //and:a[i+j]=x-y;
```

```
                //or:a[i+j+d]=y-x;
            }
}
void solve ( long long n ) {
    FWT1 ( n );
    FWT2 ( n );
    for ( long long i = 0; i < n; i++ ) {
        a1[i] = a1[i] * b1[i] % mod;
        a2[i] = a2[i] * b2[i] % mod;
        a3[i] = a3[i] * b3[i] % mod;
    }
    UFWT ( n );
}
int main() {
    long long n;
    while ( ~scanf ( "%lld", &n ) ) {
        long long res = 1 << n;
        for ( long long i = 0; i < res; i++ ) {
            scanf ( "%lld", &a1[i] );
            a3[i] = a2[i] = a1[i];
        }
        for ( long long i = 0; i < res; i++ ) {
            scanf ( "%lld", &b1[i] );
            b3[i] = b2[i] = b1[i];
        }
        inv = mod - ( mod >> 1 );
        solve ( res );
        for ( long long i = 0; i < res; i++ ) {
            printf ( "%lld%c", a1[i], i == res - 1 ? '\n' : ' ' );
        }
        for ( long long i = 0; i < res; i++ ) {
            printf ( "%lld%c", a2[i], i == res - 1 ? '\n' : ' ' );
        }
        for ( long long i = 0; i < res; i++ ) {
            printf ( "%lld%c", a3[i], i == res - 1 ? '\n' : ' ' );
        }
    }
    return 0;
}
```

2.8 liner basis

```
struct basis {
    static const int MAXL=62;
    long long a[MAXL+1];
    basis() {
        reset();
    }
```

```
    void reset() {
        memset(a,0,sizeof a);
    }
    void insert(long long x) {
        for(int i=MAXL; i⩾0; --i) {
            if(!(x>>i)&1)continue;

            if(a[i])x^=a[i];
            else {
                for(int j=0; j<i; j++)if((x>>j)&1)x^=a[j];
                for(int j=i+1; j⩽MAXL; j++)if((a[j]>>i)&1)a[j]^=x;
                a[i]=x;
                return ;
            }
        }
    }
    long long qmax() {
        long long ans=0;
        for(int i=0; i⩽MAXL; i++)ans^=a[i];
        return ans;
    }
};
inline void insert(long long x) {
  for (int i = 55; i + 1; i--) {
    if (!(x >> i))  // x的第i位是0
      continue;
    if (!p[i]) {
      p[i] = x;
      break;
    }
    x ^= p[i];
  }
}
```

2.9 high precision

```
#define MAXN 9999
// MAXN 是一位中最大的数字
#define MAXSIZE 10024
// MAXSIZE 是位数
#define DLEN 4
// DLEN 记录压几位
struct Big {
    int a[MAXSIZE], len;
    bool flag;  //标记符号'-'
    Big() {
        len = 1;
        memset(a, 0, sizeof a);
        flag = 0;
```

```
14      }
15      Big(const int);
16      Big(const char*);
17      Big(const Big&);
18      Big& operator=(const Big&);  //注意这里operator有&，因为赋值有
            修改……
19      //由于OI中要求效率
20      //此处不使用泛型函数
21      //故不重载
22      // istream& operator>>(istream&,  BigNum&);   //重载输入运算符
23      // ostream& operator<<(ostream&,  BigNum&);   //重载输出运算符
24      Big operator+(const Big&) const;
25      Big operator-(const Big&) const;
26      Big operator*(const Big&)const;
27      Big operator/(const int&) const;
28      // TODO: Big / Big;
29      Big operator^(const int&) const;
30      // TODO: Big ^ Big;
31
32      // TODO: Big 位运算;
33
34      int operator%(const int&) const;
35      // TODO: Big ^ Big;
36      bool operator<(const Big&) const;
37      bool operator<(const int& t) const;
38      inline void print();
39  };
40  // README::不要随随便便把参数都变成引用，那样没办法传值
41  Big::Big(const int b) {
42      int c, d = b;
43      len = 0;
44      // memset(a,0,sizeof a);
45      CLR(a);
46      while (d > MAXN) {
47          c = d - (d / (MAXN + 1) * (MAXN + 1));
48          d = d / (MAXN + 1);
49          a[len++] = c;
50      }
51      a[len++] = d;
52  }
53  Big::Big(const char* s) {
54      int t, k, index, l;
55      CLR(a);
56      l = strlen(s);
57      len = l / DLEN;
58      if (l % DLEN) ++len;
59      index = 0;
60      for (int i = l - 1; i ⩾ 0; i -= DLEN) {
```

```
61          t = 0;
62          k = i - DLEN + 1;
63          if (k < 0) k = 0;
64          g(j, k, i) t = t * 10 + s[j] - '0';
65          a[index++] = t;
66      }
67  }
68  Big::Big(const Big& T) : len(T.len) {
69      CLR(a);
70      f(i, 0, len) a[i] = T.a[i];
71      // TODO:重载此处?
72  }
73  Big& Big::operator=(const Big& T) {
74      CLR(a);
75      len = T.len;
76      f(i, 0, len) a[i] = T.a[i];
77      return *this;
78  }
79  Big Big::operator+(const Big& T) const {
80      Big t(*this);
81      int big = len;
82      if (T.len > len) big = T.len;
83      f(i, 0, big) {
84          t.a[i] += T.a[i];
85          if (t.a[i] > MAXN) {
86              ++t.a[i + 1];
87              t.a[i] -= MAXN + 1;
88          }
89      }
90      if (t.a[big])
91      t.len = big + 1;
92      else
93      t.len = big;
94      return t;
95  }
96  Big Big::operator-(const Big& T) const {
97      int big;
98      bool ctf;
99      Big t1, t2;
100     if (*this < T) {
101     t1 = T;
102     t2 = *this;
103     ctf = 1;
104     } else {
105     t1 = *this;
106     t2 = T;
107     ctf = 0;
108     }
109     big = t1.len;
```

```
110     int j = 0;
111     f(i, 0, big) {
112     if (t1.a[i] < t2.a[i]) {
113         j = i + 1;
114         while (t1.a[j] == 0) ++j;
115         --t1.a[j--];
116         // WTF?
117         while (j > i) t1.a[j--] += MAXN;
118         t1.a[i] += MAXN + 1 - t2.a[i];
119     } else
120         t1.a[i] -= t2.a[i];
121     }
122     t1.len = big;
123     while (t1.len > 1 && t1.a[t1.len - 1] == 0) {
124     --t1.len;
125     --big;
126     }
127     if (ctf) t1.a[big - 1] = -t1.a[big - 1];
128     return t1;
129 }
130 Big Big::operator*(const Big& T) const {
131     Big res;
132     int up;
133     int te, tee;
134     f(i, 0, len) {
135         up = 0;
136         f(j, 0, T.len) {
137             te = a[i] * T.a[j] + res.a[i + j] + up;
138             if (te > MAXN) {
139                 tee = te - te / (MAXN + 1) * (MAXN + 1);
140                 up = te / (MAXN + 1);
141                 res.a[i + j] = tee;
142             } else {
143                 up = 0;
144                 res.a[i + j] = te;
145             }
146         }
147         if (up) res.a[i + T.len] = up;
148     }
149     res.len = len + T.len;
150     while (res.len > 1 && res.a[res.len - 1] == 0) --res.len;
151     return res;
152 }
153 Big Big::operator/(const int& b) const {
154     Big res;
155     int down = 0;
156     gd(i, len - 1, 0) {
157         res.a[i] = (a[i] + down * (MAXN + 1) / b);
158         down = a[i] + down * (MAXN + 1) - res.a[i] * b;
```

```
159     }
160     res.len = len;
161     while (res.len > 1 && res.a[res.len - 1] == 0) --res.len;
162     return res;
163 }
164 int Big::operator%(const int& b) const {
165     int d = 0;
166     gd(i, len - 1, 0) d = (d * (MAXN + 1) % b + a[i]) % b;
167     return d;
168 }
169 Big Big::operator^(const int& n) const {
170     Big t(n), res(1);
171     // TODO::快速幂这样写好丑= =//DONE:)
172     int y = n;
173     while (y) {
174         if (y & 1) res = res * t;
175         t = t * t;
176         y >>= 1;
177     }
178     return res;
179 }
180 bool Big::operator<(const Big& T) const {
181     int ln;
182     if (len < T.len) return 233;
183     if (len == T.len) {
184         ln = len - 1;
185         while (ln >= 0 && a[ln] == T.a[ln]) --ln;
186         if (ln >= 0 && a[ln] < T.a[ln]) return 233;
187         return 0;
188     }
189     return 0;
190 }
191 inline bool Big::operator<(const int& t) const {
192     Big tee(t);
193     return *this < tee;
194 }
195 inline void Big::print() {
196     printf("%d", a[len - 1]);
197     gd(i, len - 2, 0) { printf("%04d", a[i]); }
198 }
199
200 inline void print(Big s) {
201     // s不要是引用，要不然你怎么print(a * b);
202     int len = s.len;
203     printf("%d", s.a[len - 1]);
204     gd(i, len - 2, 0) { printf("%04d", s.a[i]); }
205 }
206 char s[100024];
```

2.10 Fibonacci

```
pair<int, int> fib(int n) {
    if (n == 0) return {0, 1};
    auto p = fib(n >> 1);
    int c = p.first * (2 * p.second - p.first);
    int d = p.first * p.first + p.second * p.second;
    if (n & 1)
        return {d, c + d};
    else
        return {c, d};
}
```

3 math

3.1 matrix tree

```
const int MOD = 31011;
int n, m;
int bel[110];
struct edge {
    int u, v;
    ll w;
    bool operator<(const edge &tmp) const { return w < tmp.w; }
} e[1010], tr[110];
int fa[110], use[110], is[110];
inline int get(int x) {
    while (x ≠ fa[x]) x = fa[x] = fa[fa[x]];
    return x;
}
inline void uni(int x, int y) { fa[get(x)] = get(y); }
int a[110][110];
int tot, val[110], cnt;
void addTreeEdge(int v) {
    for (int i = 1; i ⩽ cnt; i++) {
        if (tr[i].w ≠ v) {
            uni(tr[i].u, tr[i].v);
        }
    }
}
void add(int u, int v) { --a[u][v], --a[v][u], ++a[u][u], ++a[v][v
   ]; }
int getblock() {
    int blo = 0;
    for (int i = 1; i ⩽ n; i++) {
        if (!bel[get(i)]) {
            bel[get(i)] = ++blo;
        }
        bel[i] = bel[get(i)];
    }
    return blo;
}
int Gauss(int n) {
    int ans = 1;
    for (int i = 1; i ⩽ n; i++)
        for (int j = 1; j ⩽ n; j++)
            if (a[i][j] < 0) a[i][j] += MOD;
    for (int i = 1; i ⩽ n; ++i) {
        for (int k = i + 1; k ⩽ n; ++k) {
            while (a[k][i]) {
                int d = a[i][i] / a[k][i];
                for (int j = i; j ⩽ n; ++j)
```

```
45                      a[i][j] = (a[i][j] - 1LL * d * a[k][j] % MOD +
                            MOD) % MOD;
46                  std::swap(a[i], a[k]), ans = -ans;
47              }
48          }
49          ans = 1LL * ans * a[i][i] % MOD, ans = (ans + MOD) % MOD;
50      }
51      return ans;
52  }
53  void rebuild(int v) {
54      memset(a, 0, sizeof(a));
55      for (int i = 1; i ⩽ m; ++i)
56          if (e[i].w == v) add(bel[e[i].u], bel[e[i].v]);
57  }
58  int main() {
59      // freopen("in.txt","r",stdin);
60      // freopen("out.txt","w",stdout);
61      scanf("%d%d", &n, &m);
62      for (int i = 1; i ⩽ m; i++) {
63          scanf("%d%d%lld", &e[i].u, &e[i].v, &e[i].w);
64      }
65      sort(e + 1, e + m + 1);
66      int g = n;
67      cnt = 0, tot = 0;
68      for (int i = 1; i ⩽ n; i++) fa[i] = i;
69      for (int i = 1, fx, fy; i ⩽ m && g > 1; i++) {
70          fx = get(e[i].u), fy = get(e[i].v);
71          if (fx ≠ fy) {
72              fa[fx] = fy;
73              tr[++cnt] = e[i];
74              if (e[i].w ≠ val[tot]) val[++tot] = e[i].w;
75              --g;
76          }
77      }
78      if (g > 1) {
79          puts("0");
80          return 0;
81      }
82      ll ans = 1;
83      for (int i = 1; i ⩽ tot; i++) {
84          for (int j = 1; j ⩽ n; j++) fa[j] = j, bel[j] = 0;
85          addTreeEdge(val[i]);
86          int blo = getblock();
87          rebuild(val[i]);
88          ans = 1LL * ans * Gauss(blo - 1) % MOD;
89      }
90      printf("%lld\n", ans);
91      return 0;
```

```
92 }
```

3.2 isap

```
 1
 2 class ISAP {
 3     private:
 4         static const int N = 10010;//endpoint_num
 5         static const int M = 240010;//edge_num
 6         static const int INF = 0x3f3f3f3f;
 7         int tot,n,m,s,t;
 8         int carc[N],gap[N];//curarc and gap
 9         int pre[N];
10         int Head[N],nxt[M],ver[M],flow[M];//base
11         int d[N];//depth
12         bool visited[N];
13     public:
14         void init(int _n,int _m,int _s,int _t) {
15             tot=1;
16             n=_n,m=_m,s=_s,t=_t;
17             fill(Head,Head+n+1,0);
18         }
19
20         void addedge(int u,int v,int w) {
21             ver[++tot]=v;
22             flow[tot]=w;
23             nxt[tot]=Head[u];
24             Head[u]=tot;
25
26             ver[++tot]=u;
27             flow[tot]=0;
28             nxt[tot]=Head[v];
29             Head[v]=tot;
30         }
31         bool bfs() {// calculate the depth
32             fill(visited,visited+n+1,0);
33             queue<int>q;
34             visited[t]=1;
35             d[t]=0;
36             q.push(t);
37             while(q.size()) {
38                 int u = q.front();
39                 q.pop();
40                 for(int i = Head[u]; i; i=nxt[i]) {
41                     int v = ver[i];
42                     if(i&1&&!visited[v]) {
43                         visited[v]=true;
44                         d[v]=d[u]+1;
45                         q.push(v);
```

```
46                          //printf("!! !! !! !d[%d]=%d  d[%d]=%d\n",u,d[
                               u],v,d[v]);
47                      }
48                  }
49              }
50              return visited[s];
51          }
52          int aug() {
53              int u=t, df=INF;
54              while(u≠s) {// calculate the flow
55                  df=min(df,flow[pre[u]]);
56                  u=ver[pre[u] ^ 1];
57              }
58              u=t;
59              while(u≠s) {
60                  flow[pre[u]]-=df;
61                  flow[pre[u] ^ 1]+=df;
62                  u=ver[pre[u] ^ 1];
63              }
64              return df;
65          }
66          int maxflow();
67  } flow;
68  int ISAP :: maxflow() {
69      int ans=0;
70      fill(gap,gap+n+1,0);
71      for(int i=1; i⩽n; i++) carc[i]=Head[i];//copy the head for
           ignore the useless edge
72      bfs();
73      for(int i=1; i⩽n; i++)gap[d[i]]++;//Using array gap to store
           how many endpoint's depth is k. When we found some gap is 0
           or d[source]>n mean there are no another augmenting path.
74      int u = s;
75      while(d[s]⩽n) {
76          if(u==t) {
77              ans+=aug();
78              u=s;
79          }
80          bool advanced=false;
81          for(int i=carc[u]; i; i=nxt[i]) {
82              if(flow[i] && d[u]==d[ver[i]]+1) {
83                  advanced=true;
84                  pre[ver[i]]=i;
85                  carc[u]=i;//carc
86                  u=ver[i];
87                  break;
88              }
89          }
90          if(!advanced) {
```

```
91              int mindep=n-1;
92              for(int i=Head[u]; i; i=nxt[i]) {
93                  if(flow[i]) {
94                      mindep=min(mindep,d[ver[i]]);
95                  }
96              }
97              //if(d[s]>n)puts("d[s]>n cause exit");
98              if(--gap[d[u]]==0)break;
99              gap[d[u]=mindep+1]++;
100
101             carc[u]=Head[u];
102             if(u!=s)u=ver[pre[u] ^ 1];
103         }
104         // if(d[s]>n)puts("d[s]>n cause exit");
105     }
106     return ans;
107 }
```

3.3 dinic

```
1  class dinic {
2      private:
3          static const int N = 10010;//TODO :endpoint_num
4          static const int M = 200010;//edge_num
5          static const int INF = 0x3f3f3f3f;
6          int tot,n,m,s,t;
7          int carc[N];//curarc and gap
8          int Head[N],nxt[M],ver[M],flow[M];//base
9          int d[N];//depth
10     public:
11         void init(int _n,int _m,int _s,int _t) {
12             tot=1;
13             n=_n,m=_m,s=_s,t=_t;
14             fill(Head,Head+n+1,0);
15         }
16         void addedge(int u,int v,int w) {
17             ver[++tot]=v;
18             flow[tot]=w;
19             nxt[tot]=Head[u];
20             Head[u]=tot;
21
22             ver[++tot]=u;
23             flow[tot]=0;
24             nxt[tot]=Head[v];
25             Head[v]=tot;
26         }
27         bool bfs() {
28             fill(d,d+n+1,0);
29             queue<int>q;
```

```
30              d[s]=1;
31              q.push(s);
32              while(q.size()) {
33                  int u = q.front();
34                  q.pop();
35                  for(int i = Head[u]; i; i=nxt[i]) {
36                      int v = ver[i];
37                      if(d[v]==0&&flow[i]) {
38                          d[v]=d[u]+1;
39                          q.push(v);
40                      }
41                  }
42              }
43              return d[t]!=0;
44          }
45          int dfs(int u,int minn);
46          int maxflow() {
47              int ans=0;
48              while(bfs()) {
49                  copy(Head+1,Head+n+1,carc+1);
50                  ans+=dfs(s,INF);
51              }
52              return ans;
53          }
54  } flow;
55  int dinic::dfs(int u,int minn) {
56      if(u==t)return minn;
57      int ret=0;
58      for(int i = carc[u]; minn&&i; i=nxt[i]) {
59          carc[u]=i;
60          int v = ver[i];
61          if(flow[i]&&d[v]==d[u]+1) {
62              int final=dfs(v,min(flow[i],minn));
63              if(final>0) {
64                  flow[i]-=final;
65                  flow[i ^ 1]+=final;
66                  minn-=final;
67                  ret+=final;
68              } else d[v]=-1;
69          }
70      }
71      return ret;
72  }
```

4 others

4.1 cppHeader

```cpp
#include <algorithm>
#include <bitset>
#include <cassert>
#include <cmath>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <ctime>
#include <iostream>
#include <map>
#include <queue>
#include <random>
#include <set>
#include <stack>
#include <string>
#include <unordered_map>
#include <unordered_set>
#include <vector>
using namespace std;
typedef long long ll;
const double PI = acos(-1.);
mt19937 myrand(time(0));
```