1 Convlution

1.1 FFT

```
typedef complex<double> cd;
const double PI = acos(-1);

void FFT(vector<cd> &a, bool inv){

    int n = a.size();

    for (int i=1, j=0 ; i<n ; i++){
        int bit = (n>>1);
        for ( ; j&bit ; bit>>=1){
            j ^= bit;
        }
        j ^= bit;
        if (i<j){
            swap(a[i], a[j]);
        }
    }

    for (int len=2 ; len<=n ; len<<=1){
        cd wlen = polar(1.0, (inv ? 2 : -2)*PI/len);

        for (int i=0 ; i<n ; i+=len){
            cd w(1);
            for (int j=0 ; j<len/2 ; j++){
                cd u = a[i+j];
                cd v = a[i+j+len/2]*w;
                a[i+j] = u+v;
                a[i+j+len/2] = u-v;
                w *= wlen;
            }
        }
    }

    if (inv){
        for (auto &x : a){
            x /= n;
        }
    }

    return;
}

vector<cd> polyMul(vector<cd> a, vector<cd> b){
    int sa = a.size(), sb = b.size(), n = 1;

    while (n<sa+sb-1) n *= 2;
    a.resize(n);
    b.resize(n);
    vector<cd> c(n);

    FFT(a, 0);
    FFT(b, 0);
    for (int i=0 ; i<n ; i++) c[i] = a[i]*b[i];
    FFT(c, 1);

    c.resize(sa+sb-1);

    return c;
}
```

1.2 NTT-998244353

```
const int MOD = (119 << 23) + 1, ROOT = 62; // = 998244353
// For p < 2^30 there is also e.g. 5 << 25, 7 << 26, 479 << 21
// and 483 << 21 (same root). The last two are > 10^9.

void NTT(vector<int> &a) {
    int n = a.size();
    int L = 31-__builtin_clz(n);
    vector<int> rt(2, 1);
    for (int k=2, s=2 ; k<n ; k*=2, s++){
        rt.resize(n);
    int z[] = {1, qp(ROOT, MOD>>s)};
        for (int i=k ; i<2*k ; i++){
            rt[i] = rt[i/2]*z[i&1]%MOD;
        }
    }

  vector<int> rev(n);
    for (int i=0 ; i<n ; i++){
        rev[i] = (rev[i/2]|(i&1)<<L)/2;
    }
    for (int i=0 ; i<n ; i++){
        if (i<rev[i]){
            swap(a[i], a[rev[i]]);
        }
    }

    for (int k=1 ; k<n ; k*=2){
        for (int i=0 ; i<n ; i+=2*k){
            for (int j=0 ; j<k ; j++){
                int z = rt[j+k]*a[i+j+k]%MOD, &ai = a[i+j];
                a[i+j+k] = ai-z+(z>ai ? MOD : 0);
                ai += (ai+z>=MOD ? z-MOD : z);
            }
        }
    }
}

vector<int> polyMul(vector<int> &a, vector<int> &b){
    if (a.empty() || b.empty()) return {};
    int s = a.size()+b.size()-1, B = 32-__builtin_clz(s), n = 1<<B;
    int inv = qp(n, MOD-2);

    vector<int> L(a), R(b), out(n);
    L.resize(n), R.resize(n);
    NTT(L), NTT(R);
    for (int i=0 ; i<n ; i++){
        out[-i&(n-1)] = L[i]*R[i]%MOD*inv%MOD;
    }
    NTT(out);

    out.resize(s);
    return out;
}
```

1.3 FFT-mod

```
/*
修改 const int MOD = 998244353 更改要取餘的數字
PolyMul(a, b) 回傳多項式乘法的結果（c_k = \sum_{i+j} a_i+b_j mod MOD）

大約可以支援 5e5，a_i, b_i 皆在 MOD 以下的非負整數
*/
const int MOD = 998244353;
typedef complex<double> cd;

void FFT(vector<cd> &a) {
    int n = a.size(), L = 31-__builtin_clz(n);
    vector<complex<long double>> R(2, 1);
    vector<cd> rt(2, 1);
    for (int k=2 ; k<n ; k*=2){
        R.resize(n);
        rt.resize(n);
        auto x = polar(1.0L, acos(-1.0L) / k);
        for (int i=k ; i<2*k ; i++){
            rt[i] = R[i] = (i&1 ? R[i/2]*x : R[i/2]);
        }
    }

    vector<int> rev(n);
    for (int i=0 ; i<n ; i++){
        rev[i] = (rev[i/2] | (i&1)<<L)/2;
    }
    for (int i=0 ; i<n ; i++){
        if (i<rev[i]) swap(a[i], a[rev[i]]);
    }
    for (int k=1 ; k<n ; k*=2){
        for (int i=0 ; i<n ; i+=2*k){
            for (int j=0 ; j<k ; j++){
                auto x = (double *)&rt[j+k];
                auto y = (double *)&a[i+j+k];
                cd z(x[0]*y[0] - x[1]*y[1], x[0]*y[1] + x[1]*y[0]);
                a[i+j+k] = a[i+j]-z;
                a[i+j] += z;
            }
        }
    }
    return;
}

vector<int> PolyMul(vector<int> a, vector<int> b){
    if (a.empty() || b.empty()) return {};

    vector<int> res(a.size()+b.size()-1);
    int B = 32-__builtin_clz(res.size()), n = (1<<B), cut = int(sqrt(MOD));
    vector<cd> L(n), R(n), outs(n), outl(n);

    for (int i=0 ; i<a.size() ; i++){
        L[i] = cd((int) a[i]/cut, (int)a[i]%cut);
    }
    for (int i=0 ; i<b.size() ; i++){
        R[i] = cd((int) b[i]/cut, (int)b[i]%cut);
    }
    FFT(L);
    FFT(R);
    for (int i=0 ; i<n ; i++){
        int j = -i&(n-1);
        outl[j] = (L[i]+conj(L[j])) * R[i]/(2.0*n);
        outs[j] = (L[i]-conj(L[j])) * R[i]/(2.0*n)/1i;
    }
    FFT(outl);
```

```
  FFT(outs);
  for (int i=0 ; i<res.size() ; i++){
    int av = (int)(real(outl[i])+0.5), cv = (int)(imag(
      outs[i])+0.5);
    int bv = (int)(imag(outl[i])+0.5) + (int)(real(outs[i
      ])+0.5);
    res[i] = ((av%MOD*cut+bv) % MOD*cut+cv) % MOD;
  }

  return res;
}
```

1.4 FFT-2

```
typedef complex<double> cd;

void FFT(vector<cd> &a) {
  int n = a.size(), L = 31-__builtin_clz(n);
  vector<complex<long double>> R(2, 1);
  vector<cd> rt(2, 1);
  for (int k=2 ; k<n ; k*=2){
    R.resize(n);
    rt.resize(n);
    auto x = polar(1.0L, acos(-1.0L) / k);
    for (int i=k ; i<2*k ; i++){
      rt[i] = R[i] = (i&1 ? R[i/2]*x : R[i/2]);
    }
  }

  vector<int> rev(n);
  for (int i=0 ; i<n ; i++){
    rev[i] = (rev[i/2] | (i&1)<<L)/2;
  }
  for (int i=0 ; i<n ; i++){
    if (i<rev[i]) swap(a[i], a[rev[i]]);
  }
  for (int k=1 ; k<n ; k*=2){
    for (int i=0 ; i<n ; i+=2*k){
      for (int j=0 ; j<k ; j++){
        auto x = (double *)&rt[j+k];
        auto y = (double *)&a[i+j+k];
        cd z(x[0]*y[0] - x[1]*y[1], x[0]*y[1] + x[1]*
          y[0]);
        a[i+j+k] = a[i+j]-z;
        a[i+j] += z;
      }
    }
  }
  return;
}

vector<double> PolyMul(const vector<double> a, const vector<
  double> b){
 if (a.empty() || b.empty()) return {};
 vector<double> res(a.size()+b.size()-1);
 int L = 32 - __builtin_clz(res.size()), n = 1 << L;
 vector<cd> in(n), out(n);

 copy(a.begin(), a.end(), begin(in));
  for (int i=0 ; i<b.size() ; i++){
    in[i].imag(b[i]);
  }
 FFT(in);
 for (cd& x : in) x *= x;
  for (int i=0 ; i<n ; i++){
    out[i] = in[-i & (n - 1)] - conj(in[i]);
  }
 FFT(out);

  for (int i=0 ; i<res.size() ; i++){
    res[i] = imag(out[i]) / (4 * n);
  }

 return res;
}
```

2 Data-Structure

2.1 GP-Hash-Table

```
#include <ext/pb_ds/assoc_container.hpp>
using namespace __gnu_pbds;
typedef tree<int,null_type,less<int>,rb_tree_tag,
  tree_order_statistics_node_update> order_set;
struct custom_hash {
  static uint64_t splitmix64(uint64_t x) {
    // http://xorshift.di.unimi.it/splitmix64.c
    x += 0x9e3779b97f4a7c15;
    x = (x ^ (x >> 30)) * 0xbf58476d1ce4e5b9;
    x = (x ^ (x >> 27)) * 0x94d049bb133111eb;
    return x ^ (x >> 31);
  }

  size_t operator()(uint64_t x) const {
    static const uint64_t FIXED_RANDOM = chrono::
      steady_clock::now().time_since_epoch().count();
    return splitmix64(x + FIXED_RANDOM);
  }
};

gp_hash_table<int, int, custom_hash> ss;
```

2.2 Sparse-Table

```
vector<vector<int>> st;
void build(vector<int> v){
 int h = __lg(v.size());
 st.resize(h+1);
 st[0] = v;

 for (int i=1 ; i<=h ; i++){
  int gap = (1<<(i-1));
  for (int j=0 ; j+gap<st[i-1].size() ; j++){
   st[i].push_back(min(st[i-1][j], st[i-1][j+gap]));
  }
 }
}

// 回傳 [ll, rr) 的最小值
int RMQ(int ll, int rr){
 int h = __lg(rr-ll);
 return min(st[h][ll], st[h][rr-(1<<h)]);
}
```

2.3 Order-Set

```
// .find_by_order(k) 回傳第 k 小的值（based-0）
// .order_of_key(k) 回傳有多少元素比 k 小

#include <ext/pb_ds/tree_policy.hpp>
using namespace __gnu_pbds;
typedef tree<int,null_type,less<int>,rb_tree_tag,
  tree_order_statistics_node_update> order_set;
```

2.4 BIT

```
vector<int> BIT(MAX_SIZE);
void update(int pos, int val){
 for (int i=pos ; i<MAX_SIZE ; i+=i&-i){
  BIT[i]+=val;
 }
}

int query(int pos){
 int ret=0;
 for (int i=pos ; i>0 ; i-=i&-i){
  ret+=BIT[i];
 }
 return ret;
}

// const int MAX_N = (1<<20)
// const int LOG_N = 20;
int k_th(int k){ // 回傳 BIT 中第 k 小的元素（based-1）
 int target = k-1, now = 0;
  for (int i=LOG_N-1 ; i>=0 ; i--){
    if (BIT[now+(1<<i)]<=k){
      k -= BIT[now+(1<<i)];
      now += 1<<i;
    }
  }
  return now+1;
}
```

2.5 Add-Set-Segment-Tree

```
// [ll, rr), based-0
// 使用前記得 init(陣列大小), build(陣列名稱)
// add(ll, rr): 區間修改
// set(ll, rr): 區間賦值
// query(ll, rr): 區間求和 / 求最大值
struct segment_tree{
  struct node{
    int add_tag=0;
    int set_tag=0;
    int sum=0;
```

```cpp
        int ma=0;
    };

    vector<node> arr;

    void init(int n){
        arr.resize(n<<2);
    }

    node pull(node A, node B){
        node C;
        C.sum=A.sum+B.sum;
        C.ma=max(A.ma, B.ma);
        return C;
    }

    void push(int idx, int ll, int rr){
        if (arr[idx].set_tag>0){
            // set 優先實作
            arr[idx].sum=(rr-ll)*arr[idx].set_tag;
            arr[idx].ma=arr[idx].set_tag;

            if (rr-ll>1){
                arr[idx*2+1].add_tag=0;
                arr[idx*2+1].set_tag=arr[idx].set_tag;
                arr[idx*2+2].add_tag=0;
                arr[idx*2+2].set_tag=arr[idx].set_tag;
            }

            arr[idx].set_tag=0;
        }
        if (arr[idx].add_tag>0){
            // add 次要實作
            arr[idx].sum+=(rr-ll)*arr[idx].add_tag;
            arr[idx].ma+=arr[idx].add_tag;

            if (rr-ll>1){
                arr[idx*2+1].add_tag+=arr[idx].add_tag;
                arr[idx*2+2].add_tag+=arr[idx].add_tag;
            }

            arr[idx].add_tag=0;
        }
    }

    void build(vector<int> &v, int idx=0, int ll=0, int rr=n){
        if (rr-ll==1){
            arr[idx].sum=v[ll];
            arr[idx].ma=v[ll];
        }else{
            int mid=(ll+rr)/2;
            build(v, idx*2+1, ll, mid);
            build(v, idx*2+2, mid, rr);
            arr[idx]=pull(arr[idx*2+1], arr[idx*2+2]);
        }
    }

    void add(int ql, int qr, int val, int idx=0, int ll=0, int rr=n){
        push(idx, ll, rr);
        if (rr<=ql || qr<=ll) return;
        if (ql<=ll && rr<=qr){
            arr[idx].add_tag+=val;
            push(idx, ll, rr);
            return;
        }
        int mid=(ll+rr)/2;
        add(ql, qr, val, idx*2+1, ll, mid);
        add(ql, qr, val, idx*2+2, mid, rr);
        arr[idx]=pull(arr[idx*2+1], arr[idx*2+2]);
    }

    void set(int ql, int qr, int val, int idx=0, int ll=0, int rr=n){
        push(idx, ll, rr);
        if (rr<=ql || qr<=ll) return;
        if (ql<=ll && rr<=qr){
            arr[idx].add_tag=0;
            arr[idx].set_tag=val;
            push(idx, ll, rr);
            return;
        }
        int mid=(ll+rr)/2;
        set(ql, qr, val, idx*2+1, ll, mid);
        set(ql, qr, val, idx*2+2, mid, rr);
        arr[idx]=pull(arr[idx*2+1], arr[idx*2+2]);
    }

    node query(int ql, int qr, int idx=0, int ll=0, int rr=n){
        push(idx, ll, rr);
        if (rr<=ql || qr<=ll) return node();
        if (ql<=ll && rr<=qr) return arr[idx];

        int mid=(ll+rr)/2;
        return pull(query(ql, qr, idx*2+1, ll, mid), query(ql, qr, idx*2+2, mid, rr));
    }
} ST;
```

2.6 Treap

```cpp
struct Treap{
    Treap *l, *r;
    int pri, val, sz;

    Treap(int _val){
        l = nullptr;
        r = nullptr;
        pri = rand();
        val = _val;
        sz = 1;
    }
} *root;

int size(Treap *a){
    return a ? a->sz : 0;
}

void pull(Treap *t){
    t->sz = size(t->l)+size(t->r)+1;
}

Treap *merge(Treap *a, Treap *b){
    // 如果一個為空，就回傳另一個
    if (!a || !b) return a ? a : b;

    if (a->pri>b->pri){
        a->r = merge(a->r, b);
        pull(a);
        return a;
    }else{
        b->l = merge(a, b->l);
        pull(b);
        return b;
    }

}

void split(Treap *&t, int k, Treap *&a, Treap *&b){
    // 如果樹為空就直接返回
    if (!t) a = b = nullptr;

    else if (size(t->l)+1<=k){ // 用 k 分割 treap
        // 如果以左子節點為根 + 目前節點合法：
        a = t;
        split(t->r, k-size(t->l)-1, a->r, b);
        pull(a);
    }else{
        b = t;
        split(t->l, k, a, b->l);
        pull(b);
    }
}

ostream & operator << (ostream &os, Treap *t){
    if (t==0) return os;
    os << t->l;
    os << (char)t->val;
    os << t->r;
    return os;
}

void print(Treap *t){
    if (t->l!=0) print(t->l);
    cout << (char)t->val;
    if (t->r!=0) print(t->r);
}
```

2.7 Persistent-Segment-Tree

```cpp
/*
全部都是 0-based

宣告
Persistent_Segment_Tree st(n+q);
st.build(v, 0);

函式：
update_version(pos, val, ver)：對版本 ver 的 pos 位置改成 val
query_version(ql, qr, ver)：對版本 ver 查詢 [ql, qr) 的區間和
clone_version(ver)：複製版本 ver 到最新的版本
*/
struct Persistent_Segment_Tree{
    int node_cnt = 0;
    struct Node{
        int lc = -1;
        int rc = -1;
        int val = 0;
    };
    vector<Node> arr;
    vector<int> version;

    Persistent_Segment_Tree(int sz){
        arr.resize(32*sz);
        version.push_back(node_cnt++);
        return;
    }

    void pull(Node &c, Node a, Node b){
        c.val = a.val+b.val;
        return;
    }

    void build(vector<int> &v, int idx, int ll = 0, int rr = n){
        auto &now = arr[idx];

        if (rr-ll==1){
            now.val = v[ll];
            return;
        }

        int mid = (ll+rr)/2;
        now.lc = node_cnt++;
        now.rc = node_cnt++;
        build(v, now.lc, ll, mid);
        build(v, now.rc, mid, rr);
        pull(now, arr[now.lc], arr[now.rc]);
        return;
    }

    void update(int pos, int val, int idx, int ll = 0, int rr = n){
        auto &now = arr[idx];

        if (rr-ll==1){
            now.val = val;
            return;
        }

        int mid = (ll+rr)/2;
        if (pos<mid){
            arr[node_cnt] = arr[now.lc];
            now.lc = node_cnt;
            node_cnt++;
            update(pos, val, now.lc, ll, mid);
        }else{
            arr[node_cnt] = arr[now.rc];
            now.rc = node_cnt;
            node_cnt++;
            update(pos, val, now.rc, mid, rr);
        }
        pull(now, arr[now.lc], arr[now.rc]);
        return;
    }

    void update_version(int pos, int val, int ver){
        update(pos, val, version[ver]);
    }

    Node query(int ql, int qr, int idx, int ll = 0, int rr = n){
        auto &now = arr[idx];

        if (ql<=ll && rr<=qr) return now;
        if (rr<=ql || qr<=ll) return Node();

        int mid = (ll+rr)/2;

        Node ret;
        pull(ret, query(ql, qr, now.lc, ll, mid), query(ql, qr, now.rc, mid, rr));
        return ret;
    }

    Node query_version(int ql, int qr, int ver){
        return query(ql, qr, version[ver]);
    }

    void clone_version(int ver){
        version.push_back(node_cnt);
        arr[node_cnt] = arr[version[ver]];
        node_cnt++;
    }
};
```

2.8 Trie

```cpp
struct Trie{
    struct Data{
        int nxt[2]={0, 0};
    };

    int sz=0;
    vector<Data> arr;

    void init(int n){
        arr.resize(n);
    }

    void insert(int n){
        int now=0;
        for (int i=N ; i>=0 ; i--){
            int v=(n>>i)&1;
            if (!arr[now].nxt[v]){
                arr[now].nxt[v]=++sz;
            }
            now=arr[now].nxt[v];
        }
    }

    int query(int n){
        int now=0, ret=0;
        for (int i=N ; i>=0 ; i--){
            int v=(n>>i)&1;
            if (arr[now].nxt[1-v]){
                ret+=(1<<i);
                now=arr[now].nxt[1-v];
            }else if (arr[now].nxt[v]){
                now=arr[now].nxt[v];
            }else{
                return ret;
            }
        }
        return ret;
    }

} tr;
```

2.9 LC-Segment-Tree

```cpp
/*
全部都是 0-based

宣告
LC_Segment_Tree st(n);

函式：
update(val)：將一個 pair <a, b> 代表插入一條 y=ax+b 的直線
query(x)：查詢所有直線在位置 x 的最小值
*/
const int MAX_V = 1e6+10; // 值域最大值

struct LC_Segment_Tree{
    struct Node{ // y = ax+b
        int a = 0;
        int b = INF;

        int y(int x){
            return a*x+b;
        }
    };
    vector<Node> arr;

    LC_Segment_Tree(int n = 0){
        arr.resize(4*n);
    }

    void update(Node val, int idx = 0, int ll = 0, int rr = MAX_V){
        if (rr-ll==1){
            if (val.y(ll)<arr[idx].y(ll)){
                arr[idx] = val;
            }
            return;
        }

        int mid = (ll+rr)/2;
        if (arr[idx].a > val.a) swap(arr[idx], val); // 原本的線斜率要比較小
        if (arr[idx].y(mid) < val.y(mid)){ // 交點在左邊
            update(val, idx*2+1, ll, mid);
        }else{ // 交點在右邊
            swap(arr[idx], val); // 在左子樹中，新線比舊線還要好
            update(val, idx*2+2, mid, rr);
        }
        return;
    }

    int query(int x, int idx = 0, int ll = 0, int rr = MAX_V){
        if (rr-ll==1){
            return arr[idx].y(ll);
        }

        int mid = (ll+rr)/2;
        if (x<mid){
            return min(arr[idx].y(x), query(x, idx*2+1, ll, mid));
        }else{
            return min(arr[idx].y(x), query(x, idx*2+2, mid, rr));
        }
    }
};
```

2.10 Persistent-Disjoint-Set

```cpp
struct Persistent_Disjoint_Set{
    Persistent_Segment_Tree arr, sz;

    void init(int n){
        arr.init(n);
        vector<int> v1;
        for (int i=0 ; i<n ; i++){
            v1.push_back(i);
        }
        arr.build(v1, 0);

        sz.init(n);
        vector<int> v2;
        for (int i=0 ; i<n ; i++){
            v2.push_back(1);
        }
        sz.build(v2, 0);
    }

    int find(int a){
        int res = arr.query_version(a, a+1, arr.version.size()-1).val;
        if (res==a) return a;
        return find(res);
    }

    bool unite(int a, int b){
        a = find(a);
        b = find(b);

        if (a!=b){

            int sz1 = sz.query_version(a, a+1, arr.version.size()-1).val;
            int sz2 = sz.query_version(b, b+1, arr.version.size()-1).val;

            if (sz1<sz2){
                arr.update_version(a, b, arr.version.size()-1);
                sz.update_version(b, sz1+sz2, arr.version.size()-1);
            }else{
                arr.update_version(b, a, arr.version.size()-1);
                sz.update_version(a, sz1+sz2, arr.version.size()-1);
            }
            return true;
        }
        return false;
    }
};
```

3 Dynamic-Programming

3.1 SOS-DP

```cpp
// 總時間複雜度為 O(n 2^n)
// 計算 dp[i] = i 所有 bit mask 子集的和
for (int i=0 ; i<n ; i++){
  for (int mask=0 ; mask<(1<<n) ; mask++){
    if ((mask>>i)&1){
      dp[mask] += dp[mask^(1<<i)];
    }
  }
}
```

3.2 Digit-DP

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

long long l, r;
long long dp[20][10][2][2]; // dp[pos][pre][limit] = 後 pos 位，pos 前一位是 pre，（是/否）有上界，（是/否）有前綴零的答案數量

long long memorize_search(string &s, int pos, int pre, bool limit, bool lead){

    // 已經被找過了，直接回傳值
    if (dp[pos][pre][limit][lead]!=-1) return dp[pos][pre][limit][lead];

    // 已經搜尋完畢，紀錄答案並回傳
    if (pos==(int)s.size()){
        return dp[pos][pre][limit][lead] = 1;
    }

    // 枚舉目前的位數數字是多少
    long long ans = 0;
    for (int now=0 ; now<=(limit ? s[pos]-'0' : 9) ; now++){
        if (now==pre){

            // 1~9 絕對不能連續出現
            if (pre!=0) continue;

            // 如果已經不在前綴零的範圍內，0 不能連續出現
            if (lead==false) continue;
        }

        ans += memorize_search(s, pos+1, now, limit&(now==(s[pos]-'0')), lead&(now==0));
    }

    // 已經搜尋完畢，紀錄答案並回傳
    return dp[pos][pre][limit][lead] = ans;
}

// 回傳 [0, n] 有多少數字符合條件
long long find_answer(long long n){
    memset(dp, -1, sizeof(dp));
    string tmp = to_string(n);

    return memorize_search(tmp, 0, 0, true, true);
}

int main(){

    // input
    cin >> l >> r;

    // output - 計算 [l, r] 有多少數字任意兩個位數都不相同
    cout << find_answer(r)-find_answer(l-1) << "\n";

    return 0;
}
```

3.3 整數拆分

$dp[i][x] =$ 要將整數 x 拆成 i 堆的「組合數」

$dp[i+1][x+1]+=dp[i][x]$ (創造新的一堆)
$dp[i][x+i]+=dp[i][x]$ (把每一堆都增加 1)

4 Geometry

4.1 Point-Struct

```cpp
// 記得確定 point 是要整數點還是浮點數
const double EPS = 1e-6;

struct point{
    double x, y;

    // 純量乘、除法
    point operator * (int a){return {a*x, a*y};};
    point operator / (int a){return {a/x, a/y};};

    // 向量加、減法
    point operator + (point a){return {x+a.x, y*a.y};};
    point operator - (point a){return {x-a.x, y-a.y};};

    // 內積、外積
    double operator * (point a){return x*a.x+y*a.y;};
    double operator ^ (point a){return x*a.y-y*a.x;};

    // 極角排序（順時鐘）
    bool operator < (const point &a) const {return (x*a.y<a.x*y);}

    // 長度
    double len(){return sqrt(x*x+y*y);};
};
```

```
// 判斷向量正負：1=正數, 0=0, -1=負數
int sign(double a){
    if (abs(a)<EPS) return 0;
    else return (a>0 ? 1 : -1);
}

// 判斷 ab 到 ac 的方向： 1=逆時鐘, 0=重疊, -1=順時鐘
int ori(point a, point b, point c){
    return sign((b-a)^(c-a));
}
```

4.2 Line-Intersection

```
bool same_seg(point a, point b, point c){
    return sign((b-a)^(c-a))==0;
}

// c 是否在 ab 裡面
bool banana(point a, point b, point c){
    if (!same_seg(a, b, c)) return 0;
    return sign((a-c)*(b-c))<=0;
}

// 判斷 ab 是否跟 cd 相交
bool seg_cross(point a, point b, point c, point d){
    int s1=ori(a, b, c);
    int s2=ori(a, b, d);
    int s3=ori(c, d, a);
    int s4=ori(c, d, b);
    if (banana(a, b, c) || banana(a, b, d) || banana(c, d, a) || banana(c, d, b)) return 1;
    return (s1*s2<0) && (s3*s4<0);
}
```

4.3 Pick's-Theorem

給定頂點坐標均是整點的簡單多邊形，面積 = 內部格點數 + 邊上格點數/2 - 1

4.4 Convex-Hull

```
vector<point> convex_hull(vector<point> points){

    sort(points.begin(), points.end());
    vector<point> hull;

    for (int _=0 ; _<2 ; _++){
        int sz=hull.size();

        for (int i=0 ; i<points.size() ; i++){
            while (hull.size()>=sz+2 && ori(hull[hull.size()-2], hull[hull.size()-1], points[i])<0){
                hull.pop_back();
            }
            hull.push_back(points[i]);
        }

        hull.pop_back();
        reverse(points.begin(), points.end());
    }

    return hull;
}
```

5 Graph

5.1 Find-Bridge

```
vector<int> dep(MAX_N), low(MAX_N);
vector<pair<int, int>> bridge;
bitset<MAX_N> vis;

void dfs(int now, int pre){
    vis[now] = 1;
    low[now] = dep[now] = (now==1 ? 0 : dep[pre]+1);

    for (auto x : G[now]){
        if (x==pre){
            continue;
        }else if (vis[x]==0){
            // 沒有走過的節點
            dfs(x, now);
            low[now] = min(low[now], low[x]);
        }else if (vis[x]==1){
            low[now] = min(low[now], dep[x]);
        }
    }

    if (now!=1 && low[now]==dep[now]){
        bridge.push_back({now, pre});
    }
    return;
}
```

5.2 Find-AP

```
vector<int> dep(MAX_N), low(MAX_N), AP;
bitset<MAX_N> vis;

void dfs(int now, int pre){
    int cnt = 0;
    bool ap = 0;
    vis[now] = 1;
    low[now] = dep[now] = (now==1 ? 0 : dep[pre]+1);

    for (auto x : G[now]){
        if (x==pre){
            continue;
        }else if (vis[x]==0){
            cnt++;
            dfs(x, now);
            low[now] = min(low[now], low[x]);
            if (low[x]>=dep[now]) ap=1;
        }else{
            low[now] = min(low[now], dep[x]);
        }
    }

    if ((now==pre && cnt>=2) || (now!=pre && ap)){
        AP.push_back(now);
    }
}
```

5.3 SCC 與縮點

```
/*
給定一個有向圖，迴回傳縮點後的圖、SCC 的資訊
所有點都以 based-0 編號

函式：
SCC_compress G(n): 宣告一個有 n 個點的圖
.add_edge(u, v): 加上一條邊 u -> v
.compress: O(n log n) 計算 G3、SCC、SCC_id 的資訊
*/
struct SCC_compress{
    int n = 0, m = 0;
    vector<vector<int>> G, inv_G, result;
    vector<pair<int, int>> edges;
    vector<bool> vis;
    vector<int> order;

    vector<vector<int>> SCC; // SCC[i] = 某個 SCC 中的所有點
    vector<int> SCC_id;      // SCC_id[i] = 第 i 個點在第幾個 SCC

    SCC_compress(int _n){ // 點的數量
        n = _n;
        G.resize(n);
        inv_G.resize(n);
        result.resize(n);
        vis.resize(n);
        SCC_id.resize(n);
    }

    void add_edge(int u, int v){
        G[u].push_back(v);
        inv_G[v].push_back(u);
        edges.push_back({u, v});
        m++;
    }

    void dfs1(vector<vector<int>> &G, int now){
        vis[now] = 1;
        for (auto x : G[now]){
            if (vis[x]==0){
                dfs1(G, x);
            }
        }
        order.push_back(now);
        return;
    }

    void dfs2(vector<vector<int>> &G, int now){
        SCC_id[now] = SCC.size()-1;
        SCC.back().push_back(now);
        vis[now] = 1;
        for (auto x : G[now]){
            if (vis[x]==0){
                dfs2(G, x);
            }
        }
        return;
    }

    void compress(){
        // 找反圖順序
        fill(vis.begin(), vis.end(), 0);
        for (int i=0 ; i<n ; i++){
            if (vis[i]==0){
                dfs1(G, i);
            }
        }

        // 找到 SCC
        fill(vis.begin(), vis.end(), 0);
        reverse(order.begin(), order.end());
        for (int i=0 ; i<n ; i++){
            if (vis[order[i]]==0){
                SCC.push_back(vector<int>());
                dfs2(inv_G, order[i]);
            }
        }

        // 縮點做 DAG
        for (int i=0 ; i<m ; i++){
            if (SCC_id[edges[i].first]!=SCC_id[edges[i].
                 second]){
                result[SCC_id[edges[i].first]].push_back(
                    SCC_id[edges[i].second]);
            }
        }
        for (int i=0 ; i<SCC.size() ; i++){
            sort(result[i].begin(), result[i].end());
            result[i].resize(unique(result[i].begin(), result
                [i].end())-result[i].begin());
        }
    }
};
```

5.4 MCMF

```
// frostray 會用，所以他超強
struct Flow {
    struct Edge {
        int u, rc, k, rv;
    };
    vector<vector<Edge>> G;
    vector<int> par, par_eid;
    Flow(int n) : G(n + 1), par(n + 1), par_eid(n + 1) {}

    // v->u | 流量：c, cost: k
    void add(int v, int u, int c, int k) {
        G[v].push_back({u, c, k, (int)G[u].size()});
        G[u].push_back({v, 0, -k, (int)G[v].size() - 1});
    }
    int spfa(int s, int t) {
        fill(par.begin(), par.end(), -1);
        vector<int> dis(par.size(), LONG_LONG_MAX);
        vector<bool> in_q(par.size(), false);
        queue<int> Q;
        dis[s] = 0; in_q[s] = true;
        Q.push(s);
        while (! Q.empty()) {
            int v = Q.front(); Q.pop();
            in_q[v] = false;
            for (int i = 0; i < (int)G[v].size(); i++) {
                auto [u, rc, k, rv] = G[v][i];
                if (rc > 0 && dis[v] + k < dis[u]) {
                    dis[u] = dis[v] + k;
                    par[u] = v;
                    par_eid[u] = i;
                    if (! in_q[u]) Q.push(u);
                    in_q[u] = true;
                }
            }
        }
        return dis[t];
    }

    // <最大流，最小費用>
    pair<int, int> flow(int s, int t) {
        int fl = 0, cost = 0, d;
        while ((d = spfa(s, t)) < LONG_LONG_MAX) {
            int cur = LONG_LONG_MAX;
            for (int v = t; v != s; v = par[v])
                cur = min(cur, G[par[v]][par_eid[v]].rc);
            fl += cur;
            cost += d * cur;
            for (int v = t; v != s; v = par[v]) {
                G[par[v]][par_eid[v]].rc -= cur;
                G[v][G[par[v]][par_eid[v]].rv].rc += cur;
            }
        }
        return {fl, cost};
    }
};
```

5.5 HLD

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

const int N = 100005;
vector <int> G[N];
struct HLD {
    vector<int> pa, sz, depth, mxson, topf, id;
    int n, idcnt = 0;
    HLD(int _n) : n(_n), pa(_n + 1), sz(_n + 1), depth(_n +
        1), mxson(_n + 1), topf(_n + 1), id(_n + 1) {}
    void dfs1(int v = 1, int p = -1) {
        pa[v] = p; sz[v] = 1; mxson[v] = 0;
        depth[v] = (p == -1 ? 0 : depth[p] + 1);
        for (int u : G[v]) {
            if (u == p) continue;
            dfs1(u, v);
            sz[v] += sz[u];
            if (sz[u] > sz[mxson[v]]) mxson[v] = u;
        }
    }
    void dfs2(int v = 1, int top = 1) {
        id[v] = ++idcnt;
        topf[v] = top;
        if (mxson[v]) dfs2(mxson[v], top);
        for (int u : G[v]) {
            if (u == mxson[v] || u == pa[v]) continue;
            dfs2(u, u);
        }
    }
    // query 為區間資料結構
    int path_query(int a, int b) {
        int res = 0;
        while (topf[a] != topf[b]) { /// 若不在同一條鍊上
            if (depth[topf[a]] < depth[topf[b]]) swap(a, b);
            res = max(res, 0ll); // query : l = id[topf[a]],
                r = id[a]
            a = pa[topf[a]];
        }
        /// 此時已在同一條鍊上
        if (depth[a] < depth[b]) swap(a, b);
        res = max(res, 0ll); // query : l = id[b], r = id[a]
        return res;
    }
};
```

5.6 Tree-Isomorphism

```
#include <bits/stdc++.h>
#pragma GCC optimize("O3,unroll-loops")
#define fastio ios::sync_with_stdio(0), cin.tie(0), cout.tie
    (0)
#define dbg(x) cerr << #x << " = " << x << endl
#define int long long
using namespace std;

// declare
const int MAX_SIZE = 2e5+5;
const int INF = 9e18;
const int MOD = 1e9+7;
const double EPS = 1e-6;
typedef vector<vector<int>> Graph;
typedef map<vector<int>, int> Hash;

int n, a, b;
int id1, id2;
pair<int, int> c1, c2;
vector<int> sz1(MAX_SIZE), sz2(MAX_SIZE);
vector<int> we1(MAX_SIZE), we2(MAX_SIZE);
Graph g1(MAX_SIZE), g2(MAX_SIZE);
Hash m1, m2;
int testcase=0;

void centroid(Graph &g, vector<int> &s, vector<int> &w, pair<
     int, int> &rec, int now, int pre){
    s[now]=1;
    w[now]=0;
    for (auto x : g[now]){
        if (x!=pre){
            centroid(g, s, w, rec, x, now);
            s[now]+=s[x];
            w[now]=max(w[now], s[x]);
        }
```

```cpp
    }

    w[now]=max(w[now], n-s[now]);
    if (w[now]<=n/2){
        if (rec.first==0) rec.first=now;
        else rec.second=now;
    }
}

int dfs(Graph &g, Hash &m, int &id, int now, int pre){
    vector<int> v;
    for (auto x : g[now]){
        if (x!=pre){
            int add=dfs(g, m, id, x, now);
            v.push_back(add);
        }
    }
    sort(v.begin(), v.end());

    if (m.find(v)!=m.end()){
        return m[v];
    }else{
        m[v]=++id;
        return id;
    }
}


void solve1(){

    // init
    id1=0;
    id2=0;
    c1={0, 0};
    c2={0, 0};
    fill(sz1.begin(), sz1.begin()+n+1, 0);
    fill(sz2.begin(), sz2.begin()+n+1, 0);
    fill(we1.begin(), we1.begin()+n+1, 0);
    fill(we2.begin(), we2.begin()+n+1, 0);
    for (int i=1 ; i<=n ; i++){
        g1[i].clear();
        g2[i].clear();
    }
    m1.clear();
    m2.clear();

    // input
    cin >> n;
    for (int i=0 ; i<n-1 ; i++){
        cin >> a >> b;
        g1[a].push_back(b);
        g1[b].push_back(a);
    }
    for (int i=0 ; i<n-1 ; i++){
        cin >> a >> b;
        g2[a].push_back(b);
        g2[b].push_back(a);
    }

    // get tree centroid
    centroid(g1, sz1, we1, c1, 1, 0);
    centroid(g2, sz2, we2, c2, 1, 0);

    // process
    int res1=0, res2=0, res3=0;
    if (c2.second!=0){
        res1=dfs(g1, m1, id1, c1.first, 0);
        m2=m1;
        id2=id1;
        res2=dfs(g2, m1, id1, c2.first, 0);
        res3=dfs(g2, m2, id2, c2.second, 0);
    }else if (c1.second!=0){
        res1=dfs(g2, m1, id1, c2.first, 0);
        m2=m1;
        id2=id1;
        res2=dfs(g1, m1, id1, c1.first, 0);
        res3=dfs(g1, m2, id2, c1.second, 0);
    }else{
        res1=dfs(g1, m1, id1, c1.first, 0);
        res2=dfs(g2, m1, id1, c2.first, 0);
    }

    // output
    cout << (res1==res2 || res1==res3 ? "YES" : "NO") << endl;

    return;
}

signed main(void){
    fastio;

    int t=1;
    cin >> t;
    while (t--){
        solve1();
    }
    return 0;
}
```

5.7 Bridge BCC

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

const int N = 200005;
vector <int> G[N];
int low[N], depth[N];
bool vis[N];
vector <vector <int>> bcc;
stack <int> stk;

void dfs(int v, int p) {
    stk.push(v);
    vis[v] = true;
    low[v] = depth[v] = (p == -1 ? 1 : depth[p] + 1);
    for (int u : G[v]) {
        if (u == p) continue;
        if (!vis[u]) {
            /// (v, u) 是樹邊
            dfs(u, v);
            low[v] = min(low[v], low[u]);
        } else {
            /// (v, u) 是回邊
            low[v] = min(low[v], depth[u]);
        }
    }
    /// v 在不依靠父邊的情況下永遠沒辦法走到它的祖先
    if (low[v] == depth[v]) {
        bcc.emplace_back();
        while (stk.top() != v) {
            bcc.back().push_back(stk.top());
            stk.pop();
        }
        bcc.back().push_back(stk.top());
        stk.pop();
    }
}
```

5.8 Cut BCC

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

const int N = 200005;
vector <int> G[N];
int low[N], depth[N];
bool vis[N];
vector <vector <int>> bcc;
stack <int> stk;

void dfs(int v, int p) {
    stk.push(v);
    vis[v] = true;
    low[v] = depth[v] = (p == -1 ? 1 : depth[p] + 1);
    for (int u : G[v]) {
        if (u == p) continue;
        if (!vis[u]) {
            /// (v, u) 是樹邊
            dfs(u, v);
            low[v] = min(low[v], low[u]);
            /// u 無法在不經過父邊的情況走到 v 的祖先
            if (low[u] >= depth[v]) {
                bcc.emplace_back();
                while (stk.top() != u) {
                    bcc.back().push_back(stk.top());
                    stk.pop();
                }
                bcc.back().push_back(stk.top());
                stk.pop();
                bcc.back().push_back(v);
            }
        } else {
            /// (v, u) 是回邊
            low[v] = min(low[v], depth[u]);
        }
    }
}
```

5.9 圓方樹

```cpp
#include <bits/stdc++.h>
#define lp(i,a,b) for(int i=(a);i<(b);i++)
#define pii pair<int,int>
#define pb push_back
#define ins insert
#define ff first
```

```cpp
#define ss second
#define opa(x) cerr << #x << " = " << x << ", ";
#define op(x) cerr << #x << " = " << x << endl;
#define ops(x) cerr << x;
#define etr cerr << endl;
#define spc cerr << ' ';
#define BAE(x) (x).begin(), (x).end()
#define STL(x) cerr << #x << " : "; for(auto &qwe:x) cerr << qwe << ' '; cerr << endl;
#define deb1 cerr << "deb1" << endl;
#define deb2 cerr << "deb2" << endl;
#define deb3 cerr << "deb3" << endl;
#define deb4 cerr << "deb4" << endl;
#define deb5 cerr << "deb5" << endl;
#define bye exit(0);
using namespace std;

const int mxn = (int)(2e5) + 10;
const int mxlg = 17;
int last_special_node = (int)(1e5) + 1;
vector<int> E[mxn], F[mxn];

struct edg{
    int fr, to;
    edg(int _fr, int _to){
        fr = _fr;
        to = _to;
    }
};
ostream& operator<<(ostream& os, edg x){os << x.fr << "--" << x.to;}
vector<edg> EV;

void tarjan(int v, int par, stack<int>& S){
    static vector<int> dfn(mxn), low(mxn);
    static vector<bool> to_add(mxn);
    static int nowT = 0;

    int childs = 0;
    nowT += 1;
    dfn[v] = low[v] = nowT;
    for(auto &ne:E[v]){
        int i = EV[ne].to;
        if(i == par) continue;
        if(!dfn[i]){
            S.push(ne);
            tarjan(i, v, S);
            childs += 1;
            low[v] = min(low[v], low[i]);

            if(par >= 0 && low[i] >= dfn[v]){
                vector<int> bcc;
                int tmp;
                do{
                    tmp = S.top(); S.pop();
                    if(!to_add[EV[tmp].fr]){
                        to_add[EV[tmp].fr] = true;
                        bcc.pb(EV[tmp].fr);
                    }
                    if(!to_add[EV[tmp].to]){
                        to_add[EV[tmp].to] = true;
                        bcc.pb(EV[tmp].to);
                    }
                }while(tmp != ne);
                for(auto &j:bcc){
                    to_add[j] = false;
                    F[last_special_node].pb(j);
                    F[j].pb(last_special_node);
                }
                last_special_node += 1;
            }
        }
        else{
            low[v] = min(low[v], dfn[i]);
            if(dfn[i] < dfn[v]){ // edge i--v will be visited twice at here, but we only need one.
                S.push(ne);
            }
        }
    }
}

int dep[mxn], jmp[mxn][mxlg];
void dfs_lca(int v, int par, int depth){
    dep[v] = depth;
    for(auto &i:F[v]){
        if(i == par) continue;
        jmp[i][0] = v;
        dfs_lca(i, v, depth + 1);
    }
}

inline void build_lca(){
    jmp[1][0] = 1;
    dfs_lca(1, -1, 1);
    lp(j,1,mxlg){
        lp(i,1,mxn){
            jmp[i][j] = jmp[jmp[i][j-1]][j-1];
        }
    }
}

inline int lca(int x, int y){
    if(dep[x] < dep[y]){ swap(x, y); }

    int diff = dep[x] - dep[y];
    lp(j,0,mxlg){
        if((diff >> j) & 1){
            x = jmp[x][j];
        }
    }
    if(x == y) return x;

    for(int j = mxlg - 1; j >= 0; j--){
        if(jmp[x][j] != jmp[y][j]){
            x = jmp[x][j];
            y = jmp[y][j];
        }
    }
    return jmp[x][0];
}

inline bool can_reach(int fr, int to){
    if(dep[to] > dep[fr]) return false;

    int diff = dep[fr] - dep[to];
    lp(j,0,mxlg){
        if((diff >> j) & 1){
            fr = jmp[fr][j];
        }
    }
    return fr == to;
}

int main(){
    ios::sync_with_stdio(false); cin.tie(0);
//    freopen("test_input.txt", "r", stdin);
    int n, m, q; cin >> n >> m >> q;
    lp(i,0,m){
        int u, v; cin >> u >> v;
        E[u].pb(EV.size());
        EV.pb(edg(u, v));
        E[v].pb(EV.size());
        EV.pb(edg(v, u));
    }
    E[0].pb(EV.size());
    EV.pb(edg(0, 1));
    stack<int> S;
    tarjan(0, -1, S);
    build_lca();

    lp(queries,0,q){
        int fr, to, relay; cin >> fr >> to >> relay;
        if(fr == relay || to == relay){
            cout << "NO\n";
            continue;
        }
        if((can_reach(fr, relay) || can_reach(to, relay)) && dep[relay] >= dep[lca(fr, to)]){
            cout << "NO\n";
            continue;
        }
        cout << "YES\n";
    }
}
```

5.10 Dijkstra

```cpp
// 可以在 O(E log E) 的時間複雜度解決在無負權有向圖單點源最短路
const int INF = 2e18; // 要確保 INF 開的足夠大

vector<vector<pair<int, int>>> G(n); // G[i] = <節點, 權重>
vector<int> dis(n, INF);
priority_queue<pair<int, int>, vector<pair<int, int>>, greater<pair<int, int>>> pq;
dis[s] = 0;
pq.push({0, s});

while (pq.size()){
    int now_dis = pq.top().first;
    int now_node = pq.top().second;
    pq.pop();

    if (now_dis>dis[now_node]) continue;

    for (auto x : G[now_node]){
        if (now_dis+x.second<dis[x.first]){
            dis[x.first] = now_dis+x.second;
            pq.push({x.first, dis[x.first]});
        }
    }
}
```

6 Math

6.1 Burnside's-Lemma

$$\sum_{k=1}^{n}\frac{c(k)}{n}$$

- n：有多少種置換方式（例如：旋轉方式）
- $c(k)$：所有可能中，經過 k 次旋轉後，仍不會和別人相同的方式的數量

6.2 線性篩

```cpp
const int MAX_N = 5e5;

// lpf[i] = i 的最小質因數
vector<int> prime, lpf(MAX_N);

void prime_init(){
    for (int i=2 ; i<MAX_N ; i++){
        if (lpf[i]==0){
            lpf[i]=i;
            prime.push_back(i);
        }

        for (int j : prime){
            if (i*j>=MAX_N) break;
            lpf[i*j]=j;
            if (lpf[i]==j) break;
        }
    }
}
```

6.3 Lucas's-Theorem

```cpp
// 對於很大的 C^n_{m} 對質數 p 取模，只要 p 不大就可以用。
int Lucas(int n, int m, int p){
    if (m==0) return 1;
    return (C(n%p, m%p, p)*Lucas(n/p, m/p, p)%p);
}
```

6.4 Miller-Rabin

```cpp
// O(1)
typedef Uint unsigned long long
Uint modmul(Uint a, Uint b, Uint m) {
  int ret = a*b - m*(Uint)(1.L/m*a*b);
  return ret + m*(ret < 0) - m*(ret>=(int)m);
}

int qp(int b, int p, int m){
    int ret = 1;
    for ( ; p ; p>>=1){
        if (p&1){
            ret = modmul(ret, b, m);
        }
        b = modmul(b, b, m);
    }
    return ret;
}

vector<int> llsprp = {2, 325, 9375, 28178, 450775, 9780504, 1795265022};
bool isprime(int n, vector<int> sprp = llsprp){
    if (n==2) return 1;
    if (n<2 || n%2==0) return 0;

    int t = 0;
    int u = n-1;
    for ( ; u%2==0 ; t++) u>>=1;

    for (int i=0 ; i<sprp.size() ; i++){
        int a = sprp[i]%n;
        if (a==0 || a==1 || a==n-1) continue;
        int x = qp(a, u, n);
        if (x==1 || x==n-1) continue;
        for (int j=0 ; j<t ; j++){
            x = modmul(x, x, n);
            if (x==1) return 0;
            if (x==n-1) break;
        }

        if (x==n-1) continue;
        return 0;
    }

    return 1;
}
```

6.5 Matrix

```cpp
struct Matrix{
    int n, m;
    vector<vector<int>> arr;

    Matrix(int _n, int _m){
        n = _n;
        m = _m;
        arr.resize(n, vector<int>(m));
    }

    Matrix operator * (const Matrix B){
        Matrix ret(n, B.m);

        for (int i=0 ; i<n ; i++){
            for (int j=0 ; j<B.m ; j++){
                for (int k=0 ; k<m ; k++){
                    ret.arr[i][j] += arr[i][k]*B.arr[k][j];
                    ret.arr[i][j] %= MOD;
                }
            }
        }

        return ret;
    }
};
```

6.6 Pollard-Rho

```cpp
mt19937 seed(chrono::steady_clock::now().time_since_epoch().count());
int rnd(int l, int r){
    return uniform_int_distribution<int>(l, r)(seed);
}

// O(n^{1/4}) 回傳 1 或自己的因數、記得先判斷 n 是不是質數（用 Miller-Rabin）
int Pollard_Rho(int n){
    int s = 0, t = 0;
    int c = rnd(1, n-1);

    int step = 0, goal = 1;
    int val = 1;

    for (goal=1 ; ; goal<<=1, s=t, val=1){
        for (step=1 ; step<=goal ; step++){

            t = ((__int128)t*t+c)%n;
            val = (__int128)val*abs(t-s)%n;

            if ((step % 127) == 0){
                int d = __gcd(val, n);
                if (d>1) return d;
            }
        }

        int d = __gcd(val, n);
        if (d>1) return d;
    }
}
```

6.7 最大質因數

```cpp
void max_fac(int n, int &ret){
    if (n<=ret || n<2) return;
    if (isprime(n)){
        ret = max(ret, n);
        return;
    }

    int p = Pollard_Rho(n);
    max_fac(p, ret), max_fac(n/p, ret);
}
```

6.8 中國剩餘定理（m 互質）

```cpp
vector<int> a, m;

int extgcd(int a, int b, int &x, int &y){
    if (b==0){
        x=1, y=0;
        return a;
    }

    int ret=extgcd(b, a%b, y, x);
    y-=a/b*x;
    return ret;
}

// n = 有幾個式子，求解 x \equiv a_i \bmod m_i
int CRT(int n, vector<int> &a, vector<int> &m){
    int p=1, ans=0;

    vector<int> M(n), inv_M(n);

    for (int i=0 ; i<n ; i++) p*=m[i];
    for (int i=0 ; i<n ; i++){
        M[i]=p/m[i];
        extgcd(M[i], m[i], inv_M[i], tmp);
        ans+=a[i]*inv_M[i]*M[i];
        ans%=p;
    }

    return (ans%p+p)%p;
}
```

6.9 中國剩餘定理（m 不互質）

```cpp
int extgcd(int a, int b, int &x, int &y){
    if (b==0){
        x=1, y=0;
        return a;
    }

    int ret=extgcd(b, a%b, y, x);
    y-=a/b*x;
    return ret;
}

// 對於方程組的式子兩兩求解
// {是否有解, {a, m}}
pair<bool, pair<int, int>> CRT(int a1, int m1, int a2, int m2){
    int g=__gcd(m1, m2);
    if ((a2-a1)%g!=0) return {0, {-1, -1}};

    int x, y;
    extgcd(m1, m2, x, y);

    x=(a2-a1)*x/g; // 兩者不能相反
    a1=x*m1+a1;
    m1=m1*m2/g;
    a1=(a1%m1+m1)%m1;
    return {1, {a1, m1}};
}
```

6.10 歐拉公式

```cpp
// phi(n) = 小於 n 並與 n 互質的正整數數量。
// O(sqrt(n))，回傳 phi(n)
int phi(int n){
    int ret = n;

    for (int i=2 ; i*i<=n ; i++){
        if (n%i==0){
            while (n%i==0) n /= i;
            ret = ret*(i-1)/i;
        }
    }
    if (n>1) ret = ret*(n-1)/n;

    return ret;
}

// O(n log n)，回傳 1~n 的 phi 值
vector<int> phi_1_to_n(int n){
    vector<int> phi(n+1);
    phi[0]=0;
    phi[1]=1;

    for (int i=2 ; i<=n ; i++){
        phi[i]=i-1;
    }

    for (int i=2 ; i<=n ; i++){
        for (int j=2*i ; j<=n ; j+=i){ // 枚舉所有倍數
            phi[j]-=phi[i];
        }
    }

    return phi;
}
```

6.11 卡特蘭數

任意括號序列：$C_n = \frac{1}{n+1}\binom{2n}{n}$

6.12 歐拉定理

若 a, m 互質，則：

$$a^n \bmod m = a^{n \bmod \varphi(m)} \bmod m$$

若 a, m 可能是任何數，則：

$$a^{\varphi(m)+[n \bmod \varphi(m)]} \bmod m$$

6.13 Fraction

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

/// Fraction template starts ///
#define fraction_template_bonus_check
const long long ll_overflow_warning_value = (long long)(3e9);

long long gcd(long long a, long long b){
    if(a == 0) return 0;
    if(b == 0) return a;
    if(a < b) return gcd(b,a);
    return gcd(b, a%b);
}
struct frac{
    long long a, b;
    frac(long long _a = 0, long long _b = 1){
        a = _a; b = _b;
        if(b == 0){
            cerr << "Error: division by zero\n";
            cerr << "Called : Constructor(" << _a << ", " << _b << ")\n";
            return;
        }
        if(a == 0){b = 1; return;}
        if(b < 0){a = -a; b = -b;}
        long long gcd_ab = gcd(std::abs(a), b);
        if(gcd_ab != 1){a /= gcd_ab; b /= gcd_ab;}

        #ifdef fraction_template_bonus_check
        if(std::abs(a) > ll_overflow_warning_value || b > ll_overflow_warning_value){
            cerr << "Overflow warning : " << a << "/" << b << "\n";}
        #endif // fraction_template_bonus_check
    }
    frac operator+(frac const &B){
        return frac(a*(B.b)+(B.a)*b, b*(B.b));}
    frac operator-(frac const &B){
        return frac(a*(B.b)-(B.a)*b, b*(B.b));}
    frac operator*(frac const &B){
        return frac(a*(B.a), b*(B.b));}
    frac operator/(frac const &B){
        return frac(a*(B.b), b*(B.a));}

    frac operator+=(frac const &B){
        *this = frac(a*(B.b)+(B.a)*b, b*(B.b));}
    frac operator-=(frac const &B){
        *this = frac(a*(B.b)-(B.a)*b, b*(B.b));}
    frac operator*=(frac const &B){
        *this = frac(a*(B.a), b*(B.b));}
    frac operator/=(frac const &B){
        *this = frac(a*(B.b), b*(B.a));}

    frac abs(){
        a = std::abs(a);
        return *this;
    }

    bool operator<(frac const &B){
        return a*B.b < B.a*b;}
    bool operator<=(frac const &B){
        return a*B.b <= B.a*b;}
    bool operator>(frac const &B){
        return a*B.b > B.a*b;}
    bool operator>=(frac const &B){
        return a*B.b >= B.a*b;}
    bool operator==(frac const &B){
        return a * B.b == B.a * b;}
    bool operator!=(frac const &B){
        return a * B.b != B.a * b;}
};
ostream& operator<<(ostream &os, const frac& A){
    os << A.a << "/" << A.b;
    return os;
}
/// Fraction template ends ///

void test(frac A, frac B){
    cout << "A = " << A << endl;
    cout << "B = " << B << endl;
    cout << endl;
    cout << "A + B = " << A + B << endl;
    cout << "A - B = " << A - B << endl;
    cout << "A * B = " << A * B << endl;
    cout << "A / B = " << A / B << endl;
    cout << endl;
    cout << "(A < B) = " << (A < B) << endl;
    cout << "(A <= B) = " << (A <= B) << endl;
    cout << "(A > B) = " << (A > B) << endl;
    cout << "(A >= B) = " << (A >= B) << endl;
    cout << "(A == B) = " << (A == B) << endl;
    cout << "(A != B) = " << (A != B) << endl;
    cout << "---------------\n";
    return;
}

int main(){
    frac tmp1(-7, 2);
    frac tmp2(5, 3);
    test(tmp1, tmp2);

    frac tmp3(-7);
    frac tmp4(0);
    test(tmp3, tmp4);
    return 0;
}
```

6.14 錯排公式

錯排公式：(n 個人中，每個人皆不再原來位置的組合數)

$$dp_i = \begin{cases} 1 & i = 0 \\ 0 & i = 1 \\ (i-1)(dp_{i-1} + dp_{i-2}) & \text{otherwise} \end{cases}$$

6.15 Quick-Pow

```
int qp(int b, int p, int m = MOD){
    int ret = 1;
    for ( ; p ; p>>=1){
        if (p&1) ret = ret*b%m;
        b = b*b%m;
    }
    return ret;
}
```

6.16 二元一次方程式

$$\begin{cases} ax + by = c \\ dx + ey = f \end{cases} = \begin{cases} x = \frac{ed-bf}{ad-bc} \\ y = \frac{af-ec}{ad-bc} \end{cases}$$

若 $x = \frac{0}{0}$ 且 $y = \frac{0}{0}$，則代表無限多組解。若 $x = \frac{*}{0}$ 且 $y = \frac{*}{0}$，則代表無解。

6.17 Josephus

```
// 有 n 個人，第偶數個報數的人被刪掉，問第 k 個被踢掉的是誰
int solve(int n, int k){
    if (n==1) return 1;
    if (k<=(n+1)/2){
        if (2*k>n) return 2*k%n;
        else return 2*k;
    }else{
        int res=solve(n/2, k-(n+1)/2);
        if (n&1) return 2*res+1;
        else return 2*res-1;
    }
}
```

7 Misc

7.1 Xor-Basis

```
vector<int> basis;
void add_vector(int x){
    for (auto v : basis){
        x=min(x, x^v);
    }
    if (x) basis.push_back(x);
}

// 給一數字集合 S，求能不能 XOR 出 x
bool check(int x){
  for (auto v : basis){
    x=min(x, x^v);
  }
  return x;
}

// 給一數字集合 S，求能 XOR 出多少數字
// 答案等於 2^{basis 的大小}

// 給一數字集合 S，求 XOR 出最大的數字
int get_max(){
  int ans=0;
  for (auto v : basis){
    ans=max(ans, ans^v);
  }
  return ans;
}
```

7.2 Default-Code

```
#include <bits/stdc++.h>
#define int long long
#define fastio ios::sync_with_stdio(0), cin.tie(0);
using namespace std;

#ifdef LOCAL
void debug(){cerr << "\n";}
template<class T, class ... U>
void debug(T a, U ... b){cerr << a << " ", debug(b...);}
template<class T> void pary(T l, T r){
    while (l!=r) cerr << *l << " ", l++;
    cerr << "\n";
}
#else
#pragma GCC optimize("O3,unroll-loops")
#pragma GCC target("avx,avx2,sse,sse2,sse3,sse4,popcnt")
#define debug(...) void()
#define pary(...) void()
#endif

const int MAX_N = 5e5+10;
const int INF = 2e18;

int n, tmp;
vector<int> v;

void solve1(){

    return;
}

signed main(){

    fastio;

    int t = 1;
    while (t--){
        solve1();
    }

    return 0;
}
```

7.3 Fast-Input

```
// fast IO
inline char readchar(){
    static char buffer[BUFSIZ], * now = buffer + BUFSIZ, *
        end = buffer + BUFSIZ;
    if (now == end)
    {
        if (end < buffer + BUFSIZ)
            return EOF;
        end = (buffer + fread(buffer, 1, BUFSIZ, stdin));
        now = buffer;
    }
    return *now++;
}
inline int nextint(){
    int x = 0, c = readchar(), neg = false;
    while(('0' > c || c > '9') && c!='-' && c!=EOF) c =
        readchar();
    if(c == '-') neg = true, c = readchar();
    while('0' <= c && c <= '9') x = (x<<3) + (x<<1) + (c^'0')
        , c = readchar();
    if(neg) x = -x;
    return x; // returns 0 if EOF
}
```

7.4 Radix-Sort

```
// 值域限制：0 ~ 1073741823(2^30-1)
inline void radix_sort(vector<int> &a, int n){
    static int cnt[32768] = {0};
    vector<int> tmpa(n);
    for(int i = 0; i < n; ++i)
        ++cnt[a[i] & 32767];
    for(int i = 1; i < 32768; ++i)
        cnt[i] += cnt[i-1];
    static int temp;
    for(int i = n-1; i >= 0; --i){
        temp = a[i] & 32767;
        --cnt[temp];
        tmpa[cnt[temp]] = a[i];
    }

    static int cnt2[32768] = {0};
    for(int i = 0; i < n; ++i)
        ++cnt2[(tmpa[i]>>15)];
    for(int i = 1; i < 32768; ++i)
        cnt2[i] += cnt2[i-1];

    for(int i = n-1; i >= 0; --i){
        temp = (tmpa[i]>>15);
        --cnt2[temp];
        a[cnt2[temp]] = tmpa[i];
    }
    return;
}
```

7.5 Set-Pq-Sort

```
// priority_queue
struct cmp{
    bool operator () (Data a, Data b){
        return a.x<b.x;
    }
};
priority_queue<Data, vector<Data>, cmp> pq;

// set
struct Data{
    int x;

    bool operator < (const Data &b){
        return x<b.x;
    }
};
```

7.6 2-SAT

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

struct TWO_SAT {
    int n, N;
    vector<vector<int>> G, rev_G;
    deque<bool> used;
    vector<int> order, comp;
    deque<bool> assignment;
    void init(int _n) {
        n = _n;
        N = _n * 2;
        G.resize(N + 5);
        rev_G.resize(N + 5);
    }
    void dfs1(int v) {
        used[v] = true;
        for (int u : G[v]) {
            if (!used[u])
                dfs1(u);
        }
        order.push_back(v);
    }
    void dfs2(int v, int cl) {
        comp[v] = cl;
        for (int u : rev_G[v]) {
            if (comp[u] == -1)
                dfs2(u, cl);
        }
    }
    bool solve() {
        order.clear();
        used.assign(N, false);
        for (int i = 0; i < N; ++i) {
            if (!used[i])
                dfs1(i);
        }
        comp.assign(N, -1);
        for (int i = 0, j = 0; i < N; ++i) {
            int v = order[N - i - 1];
            if (comp[v] == -1)
                dfs2(v, j++);
        }
        assignment.assign(n, false);
        for (int i = 0; i < N; i += 2) {
            if (comp[i] == comp[i + 1])
                return false;
            assignment[i / 2] = (comp[i] > comp[i + 1]);
        }
        return true;
    }
    void add_disjunction(int a, bool na, int b, bool nb) { // A or B
        // na means whether a is negative or not
        // nb means whether b is negative or not
        a = 2 * a ^ na;
        b = 2 * b ^ nb;
        int neg_a = a ^ 1;
        int neg_b = b ^ 1;
        G[neg_a].push_back(b);
        G[neg_b].push_back(a);
        rev_G[b].push_back(neg_a);
        rev_G[a].push_back(neg_b);
        return;
    }
    void get_result(vector<int>& res) {
        res.clear();
        for (int i = 0; i < n; i++)
            res.push_back(assignment[i]);
    }
};
/* CSES Giant Pizza
3 5
+ 1 + 2
- 1 + 3

- + + + -
*/
int main() {
    int n, m;
    cin >> n >> m;
    TWO_SAT E;
    E.init(m);

    char c1, c2;
    int inp1, inp2;
    for (int i = 0; i < n; i++) {
        cin >> c1 >> inp1;
        cin >> c2 >> inp2;
        E.add_disjunction(inp1 - 1, c1 == '-', inp2 - 1, c2 == '-');
    }

    bool able = E.solve();
    if (able) {
        vector <int> ans;
        E.get_result(ans);
        for (int i : ans)
            cout << (i == true ? '+' : '-') << ' ';
        cout << '\n';
    } else {
        cout << "IMPOSSIBLE\n";
    }

    return 0;
}
```

7.7 Enumerate-Subset

```
// 時間複雜度 O(3^n)
// 枚舉每個 mask 的子集
for (int mask=0 ; mask<(1<<n) ; mask++){
  for (int s=mask ; s>=0 ; s=(s-1)&m){
    // s 是 mask 的子集
    if (s==0) break;
  }
}
```

8 String

8.1 Hash

```
mt19937 rnd(chrono::steady_clock::now().time_since_epoch().count());
int A = rnd(), B = 1000000007;

vector<int> myPow, myPre;
void hash_init(string s){
    myPow.resize(s.size());
    myPre.resize(s.size());
```

```cpp

    for (int i=0 ; i<s.size() ; i++){
        if (i==0){
            myPow[i] = 1;
            myPre[i] = s[i];
        }else{
            myPow[i] = myPow[i-1]*A%B;
            myPre[i] = (myPre[i-1]*A+s[i])%B;
        }
    }

    return;
}

int hash_value(int l, int r){ // 取得 s[l..r] 的數值
    if (l==0) return myPre[r];
    return ((myPre[r]-myPre[l-1]*myPow[r-l+1])%B+B)%B;
}
```

8.2 Z-Function

```cpp
vector<int> z_value;
void z_function(string s){
    z_value.resize(s.size());
    int ll = 0, rr = 0;

    for (int i=1 ; i<s.size() ; i++){
        int j = 0;

        if (i<rr) j = min(z_value[i-ll], rr-i);
        while (s[j]==s[i+j]) j++;
        z_value[i] = j;

        if (i+j>rr){
            ll = i;
            rr = i+j;
        }
    }

    z_value[0] = s.size();
    return;
}
```

8.3 Suffix-Array

```cpp
/*
s = temmie
pos = 6 5 1 4 3 2 0
rnk = 6 2 5 4 3 1 0

pos:
(空字串) -> e -> emmie -> ie -> mie -> mmie -> temmie

rnk[i] = i 在 pos 的哪個位置(第幾小的陣列)
*/

vector<int> pos, rnk;

void Radix_Sort(vector<array<int, 3>> &v){
    int n = v.size();

    for (int p=1 ; p>=0 ; p--){
        vector<int> cnt(n);
        for (auto x : v){
            cnt[x[p]]++;
        }

        vector<array<int, 3>> tmp(n);
        vector<int> ptr(n); // ptr[i] = 目前 second 是 i 的元素要放在哪個位置
        ptr[0] = 0;
        for (int i=1 ; i<n ; i++){
            ptr[i] = ptr[i-1]+cnt[i-1];
        }
        for (auto x : v){
            int i = x[p];
            tmp[ptr[i]] = x;
            ptr[i]++;
        }
        v = tmp;
    }
    return;
}

void Build_SA(string s){
    s += '$';
    int n = s.size();
    rnk.resize(n);
    pos.resize(n);

    vector<array<int, 2>> tmp(n);
    for (int i=0 ; i<n ; i++) tmp[i] = {s[i], i};
    sort(tmp.begin(), tmp.end());
    for (int i=0 ; i<n ; i++) pos[i] = tmp[i][1];
    rnk[pos[0]] = 0;
    for (int i=1 ; i<n ; i++){
        if (tmp[i][0]==tmp[i-1][0]) rnk[pos[i]] = rnk[pos[i-1]];
        else rnk[pos[i]] = rnk[pos[i-1]]+1;
    }

    for (int k=0 ; (1<<k)<n ; k++){
        vector<array<int, 3>> tmp(n);
        for (int i=0 ; i<n ; i++) tmp[i] = {rnk[i], rnk[(i+(1<<k))%n], i};
        Radix_Sort(tmp);
        for (int i=0 ; i<n ; i++) pos[i] = tmp[i][2];
        rnk[pos[0]] = 0;
        for (int i=1 ; i<n ; i++){
            if (tmp[i][0]==tmp[i-1][0] && tmp[i][1]==tmp[i-1][1]) rnk[pos[i]] = rnk[pos[i-1]];
            else rnk[pos[i]] = rnk[pos[i-1]]+1;
        }
    }
}
```

8.4 Longest-Common-Prefix-Array

```cpp

/*
rnk:
(空字串) -> e -> emmie -> ie -> mie -> mmie -> temmie

lcp[i] = 第 i 小的後綴跟 i-1 小的後綴的 lcp(based-1)
[(無意義, -1), 0, 1, 0, 0, 1, 0]
*/
vector<int> pos, rnk;
vector<int> lcp;

void Build_LCP(string s){
    int n = s.size();
    s += '$';
    lcp.resize(s.size());
    lcp[0] = -1;

    int k = 0;
    for(int i=0 ; i<n ; i++){
        int pi = rnk[i];
        int j = pos[pi-1];

        while(i+k<n && j+k<n && s[i+k]==s[j+k]) k++;

        lcp[pi] = k;
        k = max((int)0, k-1);
    }
}
```

8.5 Manacher

```cpp
string Manacher(string str) {
    string tmp = "$#";
    for(char i : str) {
        tmp += i;
        tmp += '#';
    }

    vector<int> p(tmp.size(), 0);
    int mx = 0, id = 0, len = 0, center = 0;
    for(int i=1 ; i<(int)tmp.size() ; i++) {
        p[i] = mx > i ? min(p[id*2-i], mx-i) : 1;

        while(tmp[i+p[i]] == tmp[i-p[i]]) p[i]++;
        if(mx<i+p[i]) mx = i+p[i], id = i;
        if(len<p[i]) len = p[i], center = i;
    }
    return str.substr((center-len)/2, len-1);
}
```

Contents