

# 1 Misc

## 1.1 Xor-Basis

```

1 vector<int> basis;
2 void add_vector(int x){
3     for (auto v : basis){
4         x=min(x, x^v);
5     }
6     if (x) basis.push_back(x);
7 }
8
9 // 給一數字集合 S · 求能不能 XOR 出 x
10 bool check(int x){
11     for (auto v : basis){
12         x=min(x, x^v);
13     }
14     return x;
15 }
16
17 // 給一數字集合 S · 求能 XOR 出多少數字
18 // 答案等於 2^{basis 的大小}
19
20 // 給一數字集合 S · 求 XOR 出最大的數字
21 int get_max(){
22     int ans=0;
23     for (auto v : basis){
24         ans=max(ans, ans^v);
25     }
26     return ans;
27 }

```

## 1.2 Default-Code

```

1 #include <bits/stdc++.h>
2 #define int long long
3 #define ALL(x) x.begin(), x.end()
4 #define SZ(x) ((int)x.size())
5 #define fastio ios::sync_with_stdio(0), cin.tie(0);
6 using namespace std;
7
8 #ifdef LOCAL
9 #define cout cout << "\033[0;32m"
10 #define cerr cerr << "\033[0;31m"
11 #define endl "\n" << "\033[0m"
12 #else
13 #pragma GCC optimize("O3,unroll-loops")
14 #pragma GCC target("avx,avx2,sse,sse2,sse3,sse4,popcnt")
15 #define endl "\n"
16 #endif
17
18 const int MAX_N = 5e5+10;
19 const int INF = 2e18;
20
21 void solve1(){
22
23     return;
24 }
25
26 signed main(){

```

```

27
28     fastio;
29
30     int t = 1;
31     while (t--){
32         solve1();
33     }
34
35     return 0;
36 }

```

## 1.3 Radix-Sort

```

1 // 值域限制: 0 ~ 1073741823(2^30-1)
2 inline void radix_sort(vector<int> &a, int n){
3     static int cnt[32768] = {0};
4     vector<int> tmpa(n);
5     for(int i = 0; i < n; ++i)
6         ++cnt[a[i] & 32767];
7     for(int i = 1; i < 32768; ++i)
8         cnt[i] += cnt[i-1];
9     static int temp;
10    for(int i = n-1; i >= 0; --i){
11        temp = a[i] & 32767;
12        --cnt[temp];
13        tmpa[cnt[temp]] = a[i];
14    }
15
16    static int cnt2[32768] = {0};
17    for(int i = 0; i < n; ++i)
18        ++cnt2[(tmpa[i]>>15)];
19    for(int i = 1; i < 32768; ++i)
20        cnt2[i] += cnt2[i-1];
21
22    for(int i = n-1; i >= 0; --i){
23        temp = (tmpa[i]>>15);
24        --cnt2[temp];
25        a[cnt2[temp]] = tmpa[i];
26    }
27    return;
28 }

```

## 1.4 Set-Pq-Sort

```

1 // priority_queue
2 struct cmp{
3     bool operator () (Data a, Data b){
4         return a.x<b.x;
5     }
6 };
7 priority_queue<Data, vector<Data>, cmp> pq;
8
9 // set
10 struct Data{
11     int x;
12
13     bool operator < (const Data &b){
14         return x<b.x;
15     }
16 };

```

## 1.5 2-SAT

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 struct TWO_SAT {
5     int n, N;
6     vector<vector<int>> G, rev_G;
7     deque<bool> used;
8     vector<int> order, comp;
9     deque<bool> assignment;
10    void init(int _n) {
11        n = _n;
12        N = _n * 2;
13        G.resize(N + 5);
14        rev_G.resize(N + 5);
15    }
16    void dfs1(int v) {
17        used[v] = true;
18        for (int u : G[v]) {
19            if (!used[u])
20                dfs1(u);
21        }
22        order.push_back(v);
23    }
24    void dfs2(int v, int cl) {
25        comp[v] = cl;
26        for (int u : rev_G[v]) {
27            if (comp[u] == -1)
28                dfs2(u, cl);
29        }
30    }
31    bool solve() {
32        order.clear();
33        used.assign(N, false);
34        for (int i = 0; i < N; ++i) {
35            if (!used[i])
36                dfs1(i);
37        }
38        comp.assign(N, -1);
39        for (int i = 0, j = 0; i < N; ++i) {
40            int v = order[N - i - 1];
41            if (comp[v] == -1)
42                dfs2(v, j++);
43        }
44        assignment.assign(n, false);
45        for (int i = 0; i < N; i += 2) {
46            if (comp[i] == comp[i + 1])
47                return false;
48            assignment[i / 2] = (comp[i] > comp[i + 1]);
49        }
50        return true;
51    }
52    void add_disjunction(int a, bool na, int b, bool nb) { //
53        // A or B
54        // na means whether a is negative or not
55        // nb means whether b is negative or not
56        a = 2 * a ^ na;
57        b = 2 * b ^ nb;
58        int neg_a = a ^ 1;
59        int neg_b = b ^ 1;
60        G[neg_a].push_back(b);
61        G[neg_b].push_back(a);
62        rev_G[b].push_back(neg_a);
63        rev_G[a].push_back(neg_b);

```

```

63     return;
64 }
65 void get_result(vector<int>& res) {
66     res.clear();
67     for (int i = 0; i < n; i++)
68         res.push_back(assignment[i]);
69 }
70 };
71 /* CSES Giant Pizza
72 3 5
73 + 1 + 2
74 - 1 + 3
75
76 - + + + -
77 */
78 int main() {
79     int n, m;
80     cin >> n >> m;
81     TWO_SAT E;
82     E.init(m);
83
84     char c1, c2;
85     int in1, in2;
86     for (int i = 0; i < n; i++) {
87         cin >> c1 >> in1;
88         cin >> c2 >> in2;
89         E.add_disjunction(in1 - 1, c1 == '-', in2 - 1, c2
90             == '-');
91     }
92
93     bool able = E.solve();
94     if (able) {
95         vector<int> ans;
96         E.get_result(ans);
97         for (int i : ans)
98             cout << (i == true ? '+' : '-') << ' ';
99         cout << '\n';
100     } else {
101         cout << "IMPOSSIBLE\n";
102     }
103
104     return 0;
105 }

```

## 1.6 Enumerate-Subset

```

1 // 時間複雜度  $O(3^n)$ 
2 // 枚舉每個 mask 的子集
3 for (int mask=0 ; mask<(1<n) ; mask++){
4     for (int s=mask ; s>=0 ; s=(s-1)&m){
5         // s 是 mask 的子集
6         if (s==0) break;
7     }
8 }

```

## 1.7 Fast-Input

```

1 // fast IO
2 // 6f8879

```

```

3 inline char readchar(){
4     static char buffer[BUFSIZ], * now = buffer + BUFSIZ, *
5         end = buffer + BUFSIZ;
6     if (now == end)
7     {
8         if (end < buffer + BUFSIZ)
9             return EOF;
10        end = (buffer + fread(buffer, 1, BUFSIZ, stdin));
11        now = buffer;
12    }
13    return *now++;
14 }
15 inline int nextint(){
16     int x = 0, c = readchar(), neg = false;
17     while (('0' > c || c > '9') && c != '-' && c != EOF) c =
18         readchar();
19     if (c == '-') neg = true, c = readchar();
20     while ('0' <= c && c <= '9') x = (x<<3) + (x<<1) + (c^'0')
21         , c = readchar();
22     if (neg) x = -x;
23     return x; // returns 0 if EOF
24 }

```

## 1.8 setup

```

1 se nu rnu bs=2 sw=4 ts=4 hls ls=2 si acd bo=all mouse=a
2
3 :inoremap " ""<Esc>i
4 :inoremap {<CR> {<CR><Esc>ko
5 :inoremap [{ {<Esc>i
6
7 function! F(...)
8     execute '!./%:r < ./' . a:1
9 endfunction
10 command! -nargs=* R call F(<f-args>)
11
12 map <F8> :w<bar>!g++ "% -o %:r -std=c++17 -Wall -Wextra -
13     Wshadow -O2 -DLOCAL -g -fsanitize=undefined,address<CR>
14 map <F9> :!./%:r<CR>
15 map <F10> :!./%:r < ./%:r.in<CR>
16
17 ca hash w !cpp -dD -P -fpreprocessed \\\ tr -d "[:space:]" \\\
18     md5sum \\\ cut -c-6
19
20 " i+<esc>25A---+<esc>
21 " o|<esc>25A |<esc>
22 " "ggVGyG35pGdd

```

## 2 Convlnution

### 2.1 FFT

```

1 typedef complex<double> cd;
2 const double PI = acos(-1);
3
4 void FFT(vector<cd> &a, bool inv){
5
6     int n = a.size();

```

```

7
8     for (int i=1, j=0 ; i<n ; i++){
9         int bit = (n>>1);
10        for ( ; j&bit ; bit>>=1){
11            j ^= bit;
12        }
13        j ^= bit;
14        if (i<j){
15            swap(a[i], a[j]);
16        }
17    }
18
19    for (int len=2 ; len<=n ; len<=1){
20        cd wlen = polar(1.0, (inv ? 2 : -2)*PI/len);
21
22        for (int i=0 ; i<n ; i+=len){
23            cd w(1);
24            for (int j=0 ; j<len/2 ; j++){
25                cd u = a[i+j];
26                cd v = a[i+j+len/2]*w;
27                a[i+j] = u+v;
28                a[i+j+len/2] = u-v;
29                w *= wlen;
30            }
31        }
32    }
33
34    if (inv){
35        for (auto &x : a){
36            x /= n;
37        }
38    }
39
40    return;
41 }
42
43 vector<cd> polyMul(vector<cd> a, vector<cd> b){
44     int sa = a.size(), sb = b.size(), n = 1;
45
46     while (n<sa+sb-1) n *= 2;
47     a.resize(n);
48     b.resize(n);
49     vector<cd> c(n);
50
51     FFT(a, 0);
52     FFT(b, 0);
53     for (int i=0 ; i<n ; i++) c[i] = a[i]*b[i];
54     FFT(c, 1);
55
56     c.resize(sa+sb-1);
57
58     return c;
59 }

```

### 2.2 FFT-2

```

1 typedef complex<double> cd;
2
3 void FFT(vector<cd> &a) {
4     int n = a.size(), L = 31-__builtin_clz(n);
5     vector<complex<long double>> R(2, 1);
6     vector<cd> rt(2, 1);
7     for (int k=2 ; k<n ; k*=2){

```

```

8     R.resize(n);
9     rt.resize(n);
10    auto x = polar(1.0L, acos(-1.0L) / k);
11    for (int i=k ; i<2*k ; i++){
12        rt[i] = R[i] = (i&1 ? R[i/2]*x : R[i/2]);
13    }
14 }
15
16 vector<int> rev(n);
17 for (int i=0 ; i<n ; i++){
18     rev[i] = (rev[i/2] | (i&1)<<(L)/2);
19 }
20 for (int i=0 ; i<n ; i++){
21     if (i<rev[i]) swap(a[i], a[rev[i]]);
22 }
23 for (int k=1 ; k<n ; k*=2){
24     for (int i=0 ; i<n ; i+=2*k){
25         for (int j=0 ; j<k ; j++){
26             auto x = (double *)&rt[j+k];
27             auto y = (double *)&a[i+j+k];
28             cd z(x[0]*y[0] - x[1]*y[1], x[0]*y[1] + x[1]*
29                 y[0]);
30             a[i+j+k] = a[i+j]-z;
31             a[i+j] += z;
32         }
33     }
34 }
35 return;
36 }
37 vector<double> PolyMul(const vector<double> a, const vector<
38     double> b){
39     if (a.empty() || b.empty()) return {};
40     vector<double> res(a.size()+b.size()-1);
41     int L = 32 - __builtin_clz(res.size()), n = 1 << L;
42     vector<cd> in(n), out(n);
43
44     copy(a.begin(), a.end(), begin(in));
45     for (int i=0 ; i<b.size() ; i++){
46         in[i].imag(b[i]);
47     }
48     FFT(in);
49     for (cd& x : in) x *= x;
50     for (int i=0 ; i<n ; i++){
51         out[i] = in[-i & (n - 1)] - conj(in[i]);
52     }
53     FFT(out);
54
55     for (int i=0 ; i<res.size() ; i++){
56         res[i] = imag(out[i]) / (4 * n);
57     }
58
59     return res;
60 }

```

## 2.3 NTT-998244353

```

1 const int MOD = (119 << 23) + 1, ROOT = 62; // = 998244353
2 // For p < 2^30 there is also e.g. 5 << 25, 7 << 26, 479 <<
3 // 21
4 // and 483 << 21 (same root). The last two are > 10^9.
5 // 9cd58a

```

```

6 void NTT(vector<int> &a) {
7     int n = a.size();
8     int L = 31-__builtin_clz(n);
9     vector<int> rt(2, 1);
10    for (int k=2, s=2 ; k<n ; k*=2, s++){
11        rt.resize(n);
12        int z[] = {1, qp(ROOT, MOD>>s)};
13        for (int i=k ; i<2*k ; i++){
14            rt[i] = rt[i/2]*z[i&1]%MOD;
15        }
16    }
17
18    vector<int> rev(n);
19    for (int i=0 ; i<n ; i++){
20        rev[i] = (rev[i/2] | (i&1)<<(L)/2);
21    }
22    for (int i=0 ; i<n ; i++){
23        if (i<rev[i]){
24            swap(a[i], a[rev[i]]);
25        }
26    }
27
28    for (int k=1 ; k<n ; k*=2){
29        for (int i=0 ; i<n ; i+=2*k){
30            for (int j=0 ; j<k ; j++){
31                int z = rt[j+k]*a[i+j+k]%MOD, &ai = a[i+j];
32                a[i+j+k] = ai-z+(z>ai ? MOD : 0);
33                ai += (ai+z>MOD ? z-MOD : z);
34            }
35        }
36    }
37 }
38
39 // 0b0e99
40 vector<int> polyMul(vector<int> &a, vector<int> &b){
41     if (a.empty() || b.empty()) return {};
42     int s = a.size()+b.size()-1, B = 32-__builtin_clz(s), n = 1<<B;
43     int inv = qp(n, MOD-2);
44
45     vector<int> L(a), R(b), out(n);
46     L.resize(n), R.resize(n);
47     NTT(L), NTT(R);
48     for (int i=0 ; i<n ; i++){
49         out[-i&(n-1)] = L[i]*R[i]%MOD*inv%MOD;
50     }
51     NTT(out);
52
53     out.resize(s);
54     return out;
55 }

```

## 2.4 FFT-mod

```

1 /*
2 修改 const int MOD = 998244353 更改要取餘的數字
3 PolyMul(a, b) 回傳多項式乘法的結果 (c_k = \sum_{i+j=k} a_i b_j
4     mod MOD)
5
6 大約可以支援 5e5 · a_i, b_i 皆在 MOD 以下的非負整數
7 */
8 const int MOD = 998244353;

```

```

8 typedef complex<double> cd;
9
10 // b9c90a
11 void FFT(vector<cd> &a) {
12     int n = a.size(), L = 31-__builtin_clz(n);
13     vector<complex<long double>> R(2, 1);
14     vector<cd> rt(2, 1);
15     for (int k=2 ; k<n ; k*=2){
16         R.resize(n);
17         rt.resize(n);
18         auto x = polar(1.0L, acos(-1.0L) / k);
19         for (int i=k ; i<2*k ; i++){
20             rt[i] = R[i] = (i&1 ? R[i/2]*x : R[i/2]);
21         }
22     }
23
24     vector<int> rev(n);
25     for (int i=0 ; i<n ; i++){
26         rev[i] = (rev[i/2] | (i&1)<<(L)/2);
27     }
28     for (int i=0 ; i<n ; i++){
29         if (i<rev[i]) swap(a[i], a[rev[i]]);
30     }
31     for (int k=1 ; k<n ; k*=2){
32         for (int i=0 ; i<n ; i+=2*k){
33             for (int j=0 ; j<k ; j++){
34                 auto x = (double *)&rt[j+k];
35                 auto y = (double *)&a[i+j+k];
36                 cd z(x[0]*y[0] - x[1]*y[1], x[0]*y[1] + x[1]*
37                     y[0]);
38                 a[i+j+k] = a[i+j]-z;
39                 a[i+j] += z;
40             }
41         }
42     }
43     return;
44 }
45
46 // d3c65e
47 vector<int> PolyMul(vector<int> a, vector<int> b){
48     if (a.empty() || b.empty()) return {};
49
50     vector<int> res(a.size()+b.size()-1);
51     int B = 32-__builtin_clz(res.size()), n = (1<<B), cut =
52         int(sqrt(MOD));
53     vector<cd> L(n), R(n), outs(n), outl(n);
54
55     for (int i=0 ; i<a.size() ; i++){
56         L[i] = cd((int) a[i]/cut, (int) a[i]%cut);
57     }
58     for (int i=0 ; i<b.size() ; i++){
59         R[i] = cd((int) b[i]/cut, (int) b[i]%cut);
60     }
61     FFT(L);
62     FFT(R);
63     for (int i=0 ; i<n ; i++){
64         outl[j] = (L[i]+conj(L[j])) * R[i]/(2.0*n);
65         outs[j] = (L[i]-conj(L[j])) * R[i]/(2.0*n)/1i;
66     }
67     FFT(outl);
68     FFT(outs);
69     for (int i=0 ; i<res.size() ; i++){
70         int av = (int)(real(outl[i])+0.5), cv = (int)(imag(
71             outs[i])+0.5);

```

```

70     int bv = (int)(imag(outl[i])+0.5) + (int)(real(outs[i]
71           ])+0.5);
72     res[i] = ((av%MOD*cut+bv) % MOD*cut+cv) % MOD;
73 }
74 return res;
75 }

```

## 3 Data-Structure

### 3.1 GP-Hash-Table

```

1 #include <ext/pb_ds/assoc_container.hpp>
2 using namespace __gnu_pbds;
3 typedef tree<int,null_type,less<int>,rb_tree_tag,
4   tree_order_statistics_node_update> order_set;
5 struct custom_hash {
6     static uint64_t splitmix64(uint64_t x) {
7         // http://xorshift.di.unimi.it/splitmix64.c
8         x += 0x9e3779b97f4a7c15;
9         x = (x ^ (x >> 30)) * 0xbf58476d1ce4e5b9;
10        x = (x ^ (x >> 27)) * 0x94d049bb133111eb;
11        return x ^ (x >> 31);
12    }
13    size_t operator()(uint64_t x) const {
14        static const uint64_t FIXED_RANDOM = chrono::
15        steady_clock::now().time_since_epoch().count();
16        return splitmix64(x + FIXED_RANDOM);
17    };
18 };
19 gp_hash_table<int, int, custom_hash> ss;

```

### 3.2 Sparse-Table

```

1 vector<vector<int>>> st;
2 void build(vector<int> v){
3     int h = __lg(v.size());
4     st.resize(h+1);
5     st[0] = v;
6
7     for (int i=1; i<=h; i++){
8         int gap = (1<<(i-1));
9         for (int j=0; j+gap<st[i-1].size(); j++){
10             st[i].push_back(min(st[i-1][j], st[i-1][j+gap]));
11         }
12     }
13 }
14
15 // 回傳 [ll, rr) 的最小值
16 int RMQ(int ll, int rr){
17     int h = __lg(rr-ll);
18     return min(st[h][ll], st[h][rr-(1<<h)]);
19 }

```

### 3.3 Order-Set

```

1 /*
2  .find_by_order(k) 回傳第 k 小的值 (based-0)
3  .order_of_key(k) 回傳有多少元素比 k 小
4  不能在 #define int long long 後 #include 檔案
5  */
6
7 #include <ext/pb_ds/assoc_container.hpp>
8 #include <ext/pb_ds/tree_policy.hpp>
9 using namespace __gnu_pbds;
10 typedef tree<int,null_type,less<int>,rb_tree_tag,
   tree_order_statistics_node_update> order_set;

```

### 3.4 BIT

```

1 vector<int> BIT(MAX_SIZE);
2 void update(int pos, int val){
3     for (int i=pos; i<MAX_SIZE; i+=i&-i){
4         BIT[i]+=val;
5     }
6 }
7
8 int query(int pos){
9     int ret=0;
10    for (int i=pos; i>0; i-=i&-i){
11        ret+=BIT[i];
12    }
13    return ret;
14 }
15
16 // const int MAX_N = (1<<20)
17 int k_th(int k){ // 回傳 BIT 中第 k 小的元素 (based-1)
18     int res = 0;
19     for (int i=MAX_N>>1; i>=1; i>>=1){
20         if (bit[res+i]<k)
21             k -= bit[res+i];
22         res+=i;
23 }

```

### 3.5 Persistent-Segment-Tree

```

1 /*
2  全部都是 0-based
3
4  宣告
5  Persistent_Segment_Tree st(n+q);
6  st.build(v, 0);
7
8  函式：
9  update_version(pos, val, ver)：對版本 ver 的 pos 位置改成 val
10 query_version(ql, qr, ver)：對版本 ver 查詢 [ql, qr) 的區間和
11 clone_version(ver)：複製版本 ver 到最新的版本
12 */
13 struct Persistent_Segment_Tree{
14     int node_cnt = 0;
15     struct Node{

```

```

16     int lc = -1;
17     int rc = -1;
18     int val = 0;
19 };
20 vector<Node> arr;
21 vector<int> version;
22
23 Persistent_Segment_Tree(int sz){
24     arr.resize(32*sz);
25     version.push_back(node_cnt++);
26     return;
27 }
28
29 void pull(Node &c, Node a, Node b){
30     c.val = a.val+b.val;
31     return;
32 }
33
34 void build(vector<int> &v, int idx, int ll = 0, int rr =
   n){
35     auto &now = arr[idx];
36
37     if (rr-ll==1){
38         now.val = v[ll];
39         return;
40     }
41
42     int mid = (ll+rr)/2;
43     now.lc = node_cnt++;
44     now.rc = node_cnt++;
45     build(v, now.lc, ll, mid);
46     build(v, now.rc, mid, rr);
47     pull(now, arr[now.lc], arr[now.rc]);
48     return;
49 }
50
51 void update(int pos, int val, int idx, int ll = 0, int rr
   = n){
52     auto &now = arr[idx];
53
54     if (rr-ll==1){
55         now.val = val;
56         return;
57     }
58
59     int mid = (ll+rr)/2;
60     if (pos<mid){
61         arr[node_cnt] = arr[now.lc];
62         now.lc = node_cnt;
63         node_cnt++;
64         update(pos, val, now.lc, ll, mid);
65     }else{
66         arr[node_cnt] = arr[now.rc];
67         now.rc = node_cnt;
68         node_cnt++;
69         update(pos, val, now.rc, mid, rr);
70     }
71     pull(now, arr[now.lc], arr[now.rc]);
72     return;
73 }
74
75 void update_version(int pos, int val, int ver){
76     update(pos, val, version[ver]);
77 }

```

```

79 Node query(int ql, int qr, int idx, int ll = 0, int rr =
80         n){
81     auto &now = arr[idx];
82
83     if (ql<=ll && rr<=qr) return now;
84     if (rr<=ql || qr<=ll) return Node();
85
86     int mid = (ll+rr)/2;
87
88     Node ret;
89     pull(ret, query(ql, qr, now.lc, ll, mid), query(ql,
90         qr, now.rc, mid, rr));
91     return ret;
92 }
93
94 Node query_version(int ql, int qr, int ver){
95     return query(ql, qr, version[ver]);
96 }
97
98 void clone_version(int ver){
99     version.push_back(node_cnt);
100     arr[node_cnt] = arr[version[ver]];
101     node_cnt++;
102 }
103
104 };

```

### 3.6 Trie

```

1 struct Trie{
2     struct Data{
3         int nxt[2]={0, 0};
4     };
5
6     int sz=0;
7     vector<Data> arr;
8
9     void init(int n){
10         arr.resize(n);
11     }
12
13     void insert(int n){
14         int now=0;
15         for (int i=N ; i>=0 ; i--){
16             int v=(n>>i)&1;
17             if (!arr[now].nxt[v]){
18                 arr[now].nxt[v]=++sz;
19             }
20             now=arr[now].nxt[v];
21         }
22     }
23
24     int query(int n){
25         int now=0, ret=0;
26         for (int i=N ; i>=0 ; i--){
27             int v=(n>>i)&1;
28             if (arr[now].nxt[1-v]){
29                 ret+=(1<<i);
30                 now=arr[now].nxt[1-v];
31             }else if (arr[now].nxt[v]){
32                 now=arr[now].nxt[v];
33             }else{
34                 return ret;
35             }
36         }
37     }
38 }

```

### 3.7 LC-Segment-Tree

```

1 /*
2 全部都是 0-based
3
4 宣告
5 LC_Segment_Tree st(n);
6
7 函式：
8 update(val)：將一個 pair <a, b> 代表插入一條 y=ax+b 的直線
9 query(x)：查詢所有直線在位置 x 的最小值
10 */
11 const int MAX_V = 1e6+10; // 值域最大值
12
13 struct LC_Segment_Tree{
14     struct Node{ // y = ax+b
15         int a = 0;
16         int b = INF;
17
18         int y(int x){
19             return a*x+b;
20         }
21     };
22     vector<Node> arr;
23
24     LC_Segment_Tree(int n = 0){
25         arr.resize(4*n);
26     }
27
28     void update(Node val, int idx = 0, int ll = 0, int rr =
29         MAX_V){
30         if (rr-ll==1){
31             if (val.y(ll)<arr[idx].y(ll)){
32                 arr[idx] = val;
33             }
34             return;
35         }
36
37         int mid = (ll+rr)/2;
38         if (arr[idx].a > val.a) swap(arr[idx], val); // 原本
39             的線斜率要比較小
40         if (arr[idx].y(mid) < val.y(mid)){ // 交點在左邊
41             update(val, idx*2+1, ll, mid);
42         }else{ // 交點在右邊
43             swap(arr[idx], val); // 在左子樹中，新線比舊線還
44                 要好
45             update(val, idx*2+2, mid, rr);
46         }
47         return;
48     }
49
50     int query(int x, int idx = 0, int ll = 0, int rr = MAX_V)
51     {
52         if (rr-ll==1){
53             return arr[idx].y(ll);
54         }
55
56         int mid = (ll+rr)/2;
57         if (arr[idx].a > val.a) swap(arr[idx], val); // 原本
58             的線斜率要比較小
59         if (arr[idx].y(mid) < val.y(mid)){ // 交點在左邊
60             update(val, idx*2+1, ll, mid);
61         }else{ // 交點在右邊
62             swap(arr[idx], val); // 在左子樹中，新線比舊線還
63                 要好
64             update(val, idx*2+2, mid, rr);
65         }
66         return;
67     }
68 }

```

```

50 }
51
52 int mid = (ll+rr)/2;
53 if (x<mid){
54     return min(arr[idx].y(x), query(x, idx*2+1, ll,
55         mid));
56 }else{
57     return min(arr[idx].y(x), query(x, idx*2+2, mid,
58         rr));
59 }
60 }
61
62 };

```

### 3.8 Persistent-Disjoint-Set

```

1 struct Persistent_Disjoint_Set{
2     Persistent_Segment_Tree arr, sz;
3
4     void init(int n){
5         arr.init(n);
6         vector<int> v1;
7         for (int i=0 ; i<n ; i++){
8             v1.push_back(i);
9         }
10         arr.build(v1, 0);
11
12         sz.init(n);
13         vector<int> v2;
14         for (int i=0 ; i<n ; i++){
15             v2.push_back(1);
16         }
17         sz.build(v2, 0);
18     }
19
20     int find(int a){
21         int res = arr.query_version(a, a+1, arr.version.size()
22             (-1).val;
23         if (res==a) return a;
24         return find(res);
25     }
26
27     bool unite(int a, int b){
28         a = find(a);
29         b = find(b);
30
31         if (a!=b){
32             int sz1 = sz.query_version(a, a+1, arr.version.
33                 size()-1).val;
34             int sz2 = sz.query_version(b, b+1, arr.version.
35                 size()-1).val;
36
37             if (sz1<sz2){
38                 arr.update_version(a, b, arr.version.size()
39                     (-1));
40                 sz.update_version(b, sz1+sz2, arr.version.
41                     size()-1);
42             }else{
43                 arr.update_version(b, a, arr.version.size()
44                     (-1));
45                 sz.update_version(a, sz1+sz2, arr.version.
46                     size()-1);
47             }
48         }
49     }
50 }

```

```

42     return true;
43 }
44 return false;
45 }
46 };

```

### 3.9 Add-Set-Segment-Tree

```

1 // [ll, rr), based-0
2 // 使用前記得 init(陣列大小), build(陣列名稱)
3 // add(ll, rr): 區間修改
4 // set(ll, rr): 區間賦值
5 // query(ll, rr): 區間求和 / 求最大值
6 struct SegmentTree{
7     struct node{
8         int add_tag = 0;
9         int set_tag = 0;
10        int sum = 0;
11        int ma = 0;
12    };
13
14    vector<node> arr;
15
16    SegmentTree(int n){
17        arr.resize(n<<2);
18    }
19
20    node pull(node A, node B){
21        node C;
22        C.sum = A.sum+B.sum;
23        C.ma = max(A.ma, B.ma);
24        return C;
25    }
26
27    // cce0c8
28    void push(int idx, int ll, int rr){
29        if (arr[idx].set_tag!=0){
30            arr[idx].sum = (rr-ll)*arr[idx].set_tag;
31            arr[idx].ma = arr[idx].set_tag;
32            if (rr-ll>1){
33                arr[idx*2+1].add_tag = 0;
34                arr[idx*2+1].set_tag = arr[idx].set_tag;
35                arr[idx*2+2].add_tag = 0;
36                arr[idx*2+2].set_tag = arr[idx].set_tag;
37            }
38            arr[idx].set_tag = 0;
39        }
40        if (arr[idx].add_tag!=0){
41            arr[idx].sum += (rr-ll)*arr[idx].add_tag;
42            arr[idx].ma += arr[idx].add_tag;
43            if (rr-ll>1){
44                arr[idx*2+1].add_tag += arr[idx].add_tag;
45                arr[idx*2+2].add_tag += arr[idx].add_tag;
46            }
47            arr[idx].add_tag = 0;
48        }
49    }
50
51    void build(vector<int> &v, int idx = 0, int ll = 0, int
52        rr = n){
53        if (rr-ll==1){
54            arr[idx].sum = v[ll];

```

```

54        arr[idx].ma = v[ll];
55    }else{
56        int mid = (ll+rr)/2;
57        build(v, idx*2+1, ll, mid);
58        build(v, idx*2+2, mid, rr);
59        arr[idx] = pull(arr[idx*2+1], arr[idx*2+2]);
60    }
61 }
62
63 void add(int ql, int qr, int val, int idx = 0, int ll =
64     0, int rr = n){
65     push(idx, ll, rr);
66     if (rr<=ql || qr<=ll) return;
67     if (ql<=ll && rr<=qr){
68         arr[idx].add_tag += val;
69         push(idx, ll, rr);
70         return;
71     }
72     int mid = (ll+rr)/2;
73     add(ql, qr, val, idx*2+1, ll, mid);
74     add(ql, qr, val, idx*2+2, mid, rr);
75     arr[idx]=pull(arr[idx*2+1], arr[idx*2+2]);
76 }
77
78 void set(int ql, int qr, int val, int idx=0, int ll=0,
79     int rr=n){
80     push(idx, ll, rr);
81     if (rr<=ql || qr<=ll) return;
82     if (ql<=ll && rr<=qr){
83         arr[idx].add_tag = 0;
84         arr[idx].set_tag = val;
85         push(idx, ll, rr);
86         return;
87     }
88     int mid = (ll+rr)/2;
89     set(ql, qr, val, idx*2+1, ll, mid);
90     set(ql, qr, val, idx*2+2, mid, rr);
91     arr[idx] = pull(arr[idx*2+1], arr[idx*2+2]);
92 }
93
94 node query(int ql, int qr, int idx = 0, int ll = 0, int
95     rr = n){
96     push(idx, ll, rr);
97     if (rr<=ql || qr<=ll) return node();
98     if (ql<=ll && rr<=qr) return arr[idx];
99
100    int mid = (ll+rr)/2;
101    return pull(query(ql, qr, idx*2+1, ll, mid), query(ql
102        , qr, idx*2+2, mid, rr));
103 }
104 } ST;

```

### 3.10 Treap

```

1 struct Treap{
2     Treap *l = nullptr, *r = nullptr;
3     int pri = rand(), val = 0, sz = 1;
4
5     Treap(int _val){
6         val = _val;
7     }
8 };
9

```

```

10 int size(Treap *t){return t ? t->sz : 0;}
11 void pull(Treap *t){
12     t->sz = size(t->l)+size(t->r)+1;
13 }
14
15 Treap* merge(Treap *a, Treap *b){
16     if (!a || !b) return a ? a : b;
17
18     if (a->pri>b->pri){
19         a->r = merge(a->r, b);
20         pull(a);
21         return a;
22     }else{
23         b->l = merge(a, b->l);
24         pull(b);
25         return b;
26     }
27 }
28
29 pair<Treap*, Treap*> split(Treap *t, int k){ // 1-based <前
30     k 個元素, 其他元素>
31     if (!t) return {};
32     if (size(t->l)>=k){
33         auto pa = split(t->l, k);
34         t->l = pa.second;
35         pull(t);
36         return {pa.first, t};
37     }else{
38         auto pa = split(t->r, k-size(t->l)-1);
39         t->r = pa.first;
40         pull(t);
41         return {t, pa.second};
42     }
43 }
44
45 // functions
46 Treap* build(vector<int> v){
47     Treap* ret;
48     for (int i=0; i<SZ(v); i++){
49         ret = merge(ret, new Treap(v[i]));
50     }
51     return ret;
52 }
53
54 array<Treap*, 3> cut(Treap *t, int l, int r){ // 1-based <前
55     1~l-1 個元素, l~r 個元素, r+1 個元素>
56     array<Treap*, 3> ret;
57     tie(ret[1], ret[2]) = split(t, r);
58     tie(ret[0], ret[1]) = split(ret[1], l-1);
59     return ret;
60 }
61
62 void print(Treap *t, bool flag = true){
63     if (t->l!=0) print(t->l, false);
64     cout << t->val;
65     if (t->r!=0) print(t->r, false);
66     if (flag) cout << endl;
67 }

```

## 4 Dynamic-Programming

### 4.1 SOS-DP

```

1 // 總時間複雜度為  $O(n^2 \cdot 2^n)$ 
2 // 計算  $dp[i] = i$  所有 bit mask 子集的和
3 for (int i=0 ; i<n ; i++){
4     for (int mask=0 ; mask<(1<<n) ; mask++){
5         if ((mask>>i)&1){
6             dp[mask] += dp[mask^(1<<i)];
7         }
8     }
9 }

```

### 4.2 Digit-DP

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 long long l, r;
5 long long dp[20][10][2][2]; // dp[pos][pre][Limit] = 後 pos
6 // 位 · pos 前一位是 pre · (是/否) 有上界 · (是/否) 有前綴零
7 // 的答案數量
8
9 long long memorize_search(string &s, int pos, int pre, bool
10 limit, bool lead){
11
12     // 已經被找過了 · 直接回傳值
13     if (dp[pos][pre][limit][lead]!=-1) return dp[pos][pre][
14 limit][lead];
15
16     // 已經搜尋完畢 · 紀錄答案並回傳
17     if (pos==(int)s.size()){
18         return dp[pos][pre][limit][lead] = 1;
19     }
20
21     // 枚舉目前的位數數字是多少
22     long long ans = 0;
23     for (int now=0 ; now<=(limit ? s[pos]-'0' : 9) ; now++){
24         if (now==pre){
25
26             // 1~9 絕對不能連續出現
27             if (pre!=0) continue;
28
29             // 如果已經不在前綴零的範圍內 · 0 不能連續出現
30             if (lead==false) continue;
31
32             ans += memorize_search(s, pos+1, now, limit&(now==(s[
33 pos]-'0')), lead&(now==0));
34         }
35     }
36
37     // 已經搜尋完畢 · 紀錄答案並回傳
38     return dp[pos][pre][limit][lead] = ans;
39 }
40
41 // 回傳 [0, n] 有多少數字符合條件
42 long long find_answer(long long n){
43     memset(dp, -1, sizeof(dp));
44 }

```

```

39 string tmp = to_string(n);
40
41 return memorize_search(tmp, 0, 0, true, true);
42 }
43
44 int main(){
45
46     // input
47     cin >> l >> r;
48
49     // output - 計算 [l, r] 有多少數字任意兩個位數都不相同
50     cout << find_answer(r)-find_answer(l-1) << "\n";
51
52     return 0;
53 }

```

```

29 };
30
31 // 判斷向量正負：1=正數，0=0，-1=負數
32 int sign(double a){
33     if (abs(a)<EPS) return 0;
34     else return (a>0 ? 1 : -1);
35 }
36
37 // 判斷 ab 到 ac 的方向：1=逆時鐘，0=重疊，-1=順時鐘
38 int ori(point a, point b, point c){
39     return sign((b-a)^(c-a));
40 }

```

### 5.2 Line-Intersection

```

1 // c 是否在 ab 裡面
2 bool in(point a, point b, point c){
3     if (ori(a, b, c)) return 0;
4     return sign((a-c)*(b-c))<=0;
5 }
6
7 // 判斷 ab 是否跟 cd 相交
8 bool banana(point a, point b, point c, point d){
9     int s1=ori(a, b, c);
10    int s2=ori(a, b, d);
11    int s3=ori(c, d, a);
12    int s4=ori(c, d, b);
13    if (in(a, b, c) || in(a, b, d) || in(c, d, a) || in(c, d,
14        b)) return 1;
15    return (s1*s2<0) && (s3*s4<0);
16 }

```

### 5.3 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

### 5.4 Point-In-Polygon

### 4.3 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)

## 5 Geometry

### 5.1 Point-Struct

```

1 const int EPS = 1e-6;
2
3 typedef int pt;
4 struct point{
5     pt x, y;
6
7     point(pt _x = 0, pt _y = 0){
8         x = _x;
9         y = _y;
10    }
11
12    // 純量乘、除法
13    point operator * (pt a){return {a*x, a*y}};
14    point operator / (pt a){return {a/x, a/y}};
15
16    // 向量加、減法
17    point operator + (point a){return {x+a.x, y+a.y}};
18    point operator - (point a){return {x-a.x, y-a.y}};
19
20    // 內積、外積
21    double operator * (point a){return x*a.x+y*a.y};
22    double operator ^ (point a){return x*a.y-y*a.x};
23
24    // bool operator < (const point &a) const {return (x*a.y<
25        a.x*y);} // 極角排序 (順時鐘)
26    bool operator < (const point &a) const {return x==a.x ? y
27        <a.y : x<a.x;}
28    bool operator == (const point &a) const {return x==a.x &&
29        y==a.y;}
30
31    double dis(point a){return sqrtl(abs(x-a.x)*abs(x-a.x)+
32        abs(y-a.y)*abs(y-a.y));}
33 }

```

```

1 /*
2 可以在有 n 個點的簡單多邊形內 · 用  $O(n)$  的時間回傳：
3 1: 在多邊形內，0: 在多邊形上，-1: 在多邊形外
4 */
5 const int MAX_POS = 1e9+5; // [記得修改] 座標的最大值
6 int in_polygon(vector<point> &v, point a){
7     int c = v.size();
8     v.push_back(v[0]); // 已經用好循環了
9     point b = {MAX_POS, a.y+1};
10    int cnt = 0;
11
12    for (int i=0 ; i<n ; i++){
13        if (in(v[i], v[i+1], a)) return 0;
14        if (banana(a, b, v[i], v[i+1])) cnt++;
15    }
16
17    return cnt%2 ? 1 : -1;
18 }

```



## 5.5 Convex-Hull

```
1 // e0a719
2 vector<point> convex_hull(vector<point> v){
3     sort(v.begin(), v.end());
4     v.resize(unique(v.begin(), v.end())-v.begin());
5     vector<point> hull;
6     for (int _=0 ; _<2 ; _++){
7         int sz=hull.size();
8         for (int i=0 ; i<v.size() ; i++){
9             while (hull.size()==sz+2 && ori(hull[hull.size()-2], hull[hull.size()-1], v[i])<0){
10                 hull.pop_back();
11             }
12             hull.push_back(v[i]);
13         }
14         hull.pop_back();
15         reverse(v.begin(), v.end());
16     }
17     return hull;
18 }
```

```
4 void dfs(int now, int pre){
5     int cnt = 0;
6     bool ap = 0;
7     vis[now] = 1;
8     low[now] = dep[now] = (now==1 ? 0 : dep[pre]+1);
9
10    for (auto x : G[now]){
11        if (x==pre){
12            continue;
13        }else if (vis[x]==0){
14            cnt++;
15            dfs(x, now);
16            low[now] = min(low[now], low[x]);
17            if (low[x]>=dep[now]) ap=1;
18        }else{
19            low[now] = min(low[now], dep[x]);
20        }
21    }
22
23    if ((now==pre && cnt>=2) || (now!=pre && ap)){
24        AP.push_back(now);
25    }
26 }
```

```
36         a = pa[topf[a]];
37     }
38     /// 此時已在同一條鍊上
39     if (depth[a] < depth[b]) swap(a, b);
40     res = max(res, 011); // query : l = id[b], r = id[a]
41     return res;
42 }
43 };
```

## 6.4 Tree-Isomorphism

```
1 #include <bits/stdc++.h>
2 #pragma GCC optimize("O3,unroll-loops")
3 #define fastio ios::sync_with_stdio(0), cin.tie(0), cout.tie(0)
4 #define dbg(x) cerr << #x << " = " << x << endl
5 #define int long long
6 using namespace std;
7
8 // declare
9 const int MAX_SIZE = 2e5+5;
10 const int INF = 9e18;
11 const int MOD = 1e9+7;
12 const double EPS = 1e-6;
13 typedef vector<vector<int>> Graph;
14 typedef map<vector<int>, int> Hash;
15
16 int n, a, b;
17 int id1, id2;
18 pair<int, int> c1, c2;
19 vector<int> sz1(MAX_SIZE), sz2(MAX_SIZE);
20 vector<int> we1(MAX_SIZE), we2(MAX_SIZE);
21 Graph g1(MAX_SIZE), g2(MAX_SIZE);
22 Hash m1, m2;
23 int testcase=0;
24
25 void centroid(Graph &g, vector<int> &s, vector<int> &w, pair<int, int> &rec, int now, int pre){
26     s[now]=1;
27     w[now]=0;
28     for (auto x : g[now]){
29         if (x!=pre){
30             centroid(g, s, w, rec, x, now);
31             s[now]+=s[x];
32             w[now]=max(w[now], s[x]);
33         }
34     }
35
36     w[now]=max(w[now], n-s[now]);
37     if (w[now]<=n/2){
38         if (rec.first==0) rec.first=now;
39         else rec.second=now;
40     }
41 }
42
43 int dfs(Graph &g, Hash &m, int &id, int now, int pre){
44     vector<int> v;
45     for (auto x : g[now]){
46         if (x!=pre){
47             int add=dfs(g, m, id, x, now);
48             v.push_back(add);
49         }
50     }
```

## 6 Graph

### 6.1 Find-Bridge

```
1 vector<int> dep(MAX_N), low(MAX_N);
2 vector<pair<int, int>> bridge;
3 bitset<MAX_N> vis;
4
5 void dfs(int now, int pre){
6     vis[now] = 1;
7     low[now] = dep[now] = (now==1 ? 0 : dep[pre]+1);
8
9     for (auto x : G[now]){
10         if (x==pre){
11             continue;
12         }else if (vis[x]==0){
13             /// 沒有走過的節點
14             dfs(x, now);
15             low[now] = min(low[now], low[x]);
16         }else if (vis[x]==1){
17             low[now] = min(low[now], dep[x]);
18         }
19     }
20
21     if (now!=1 && low[now]==dep[now]){
22         bridge.push_back({now, pre});
23     }
24     return;
25 }
```

### 6.2 Find-AP

```
1 vector<int> dep(MAX_N), low(MAX_N), AP;
2 bitset<MAX_N> vis;
3
```

### 6.3 HLD

```
1 #include <bits/stdc++.h>
2 #define int long long
3 using namespace std;
4
5 const int N = 100005;
6 vector<int> G[N];
7 struct HLD {
8     vector<int> pa, sz, depth, mxson, topf, id;
9     int n, idcnt = 0;
10    HLD(int _n) : n(_n), pa(_n + 1), sz(_n + 1), depth(_n + 1), mxson(_n + 1), topf(_n + 1), id(_n + 1) {}
11    void dfs1(int v = 1, int p = -1) {
12        pa[v] = p; sz[v] = 1; mxson[v] = 0;
13        depth[v] = (p == -1 ? 0 : depth[p] + 1);
14        for (int u : G[v]) {
15            if (u == p) continue;
16            dfs1(u, v);
17            sz[v] += sz[u];
18            if (sz[u] > sz[mxson[v]]) mxson[v] = u;
19        }
20    }
21    void dfs2(int v = 1, int top = 1) {
22        id[v] = ++idcnt;
23        topf[v] = top;
24        if (mxson[v]) dfs2(mxson[v], top);
25        for (int u : G[v]) {
26            if (u == mxson[v] || u == pa[v]) continue;
27            dfs2(u, u);
28        }
29    }
30    /// query 為區間資料結構
31    int path_query(int a, int b) {
32        int res = 0;
33        while (topf[a] != topf[b]) { /// 若不在同一條鍊上
34            if (depth[topf[a]] < depth[topf[b]]) swap(a, b);
35            res = max(res, 011); // query : l = id[topf[a]], r = id[a]
```



```

51     sort(v.begin(), v.end());
52
53     if (m.find(v)!=m.end()){
54         return m[v];
55     }else{
56         m[v]++;
57         return id;
58     }
59 }
60
61 void solve1(){
62
63     // init
64     id1=0;
65     id2=0;
66     c1={0, 0};
67     c2={0, 0};
68     fill(sz1.begin(), sz1.begin()+n+1, 0);
69     fill(sz2.begin(), sz2.begin()+n+1, 0);
70     fill(we1.begin(), we1.begin()+n+1, 0);
71     fill(we2.begin(), we2.begin()+n+1, 0);
72     for (int i=1 ; i<=n ; i++){
73         g1[i].clear();
74         g2[i].clear();
75     }
76     m1.clear();
77     m2.clear();
78
79     // input
80     cin >> n;
81     for (int i=0 ; i<n-1 ; i++){
82         cin >> a >> b;
83         g1[a].push_back(b);
84         g1[b].push_back(a);
85     }
86     for (int i=0 ; i<n-1 ; i++){
87         cin >> a >> b;
88         g2[a].push_back(b);
89         g2[b].push_back(a);
90     }
91
92     // get tree centroid
93     centroid(g1, sz1, we1, c1, 1, 0);
94     centroid(g2, sz2, we2, c2, 1, 0);
95
96     // process
97     int res1=0, res2=0, res3=0;
98     if (c2.second!=0){
99         res1=dfs(g1, m1, id1, c1.first, 0);
100         m2=m1;
101         id2=id1;
102         res2=dfs(g2, m1, id1, c2.first, 0);
103         res3=dfs(g2, m2, id2, c2.second, 0);
104     }else if (c1.second!=0){
105         res1=dfs(g2, m1, id1, c2.first, 0);
106         m2=m1;
107         id2=id1;
108         res2=dfs(g1, m1, id1, c1.first, 0);
109         res3=dfs(g1, m2, id2, c1.second, 0);
110     }else{
111         res1=dfs(g1, m1, id1, c1.first, 0);
112         res2=dfs(g2, m1, id1, c2.first, 0);
113     }
114
115     // output
116

```

```

117     cout << (res1==res2 || res1==res3 ? "YES" : "NO") << endl;
118     ;
119     return;
120 }
121
122 signed main(void){
123     fastio;
124
125     int t=1;
126     cin >> t;
127     while (t--){
128         solve1();
129     }
130     return 0;
131 }

```

## 6.5 Bridge BCC

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 const int N = 200005;
5 vector <int> G[N];
6 int low[N], depth[N];
7 bool vis[N];
8 vector <vector <int>> bcc;
9 stack <int> stk;
10
11 void dfs(int v, int p) {
12     stk.push(v);
13     vis[v] = true;
14     low[v] = depth[v] = (p == -1 ? 1 : depth[p] + 1);
15     for (int u : G[v]) {
16         if (u == p) continue;
17         if (!vis[u]) {
18             /// (v, u) 是樹邊
19             dfs(u, v);
20             low[v] = min(low[v], low[u]);
21         } else {
22             /// (v, u) 是回邊
23             low[v] = min(low[v], depth[u]);
24         }
25     }
26     /// v 在不依靠父邊的情況下永遠沒辦法走到它的祖先
27     if (low[v] == depth[v]) {
28         bcc.emplace_back();
29         while (stk.top() != v) {
30             bcc.back().push_back(stk.top());
31             stk.pop();
32         }
33         bcc.back().push_back(stk.top());
34         stk.pop();
35     }
36 }

```

## 6.6 Cut BCC

```
1 #include <bits/stdc++.h>

```

```

2 using namespace std;
3
4 const int N = 200005;
5 vector <int> G[N];
6 int low[N], depth[N];
7 bool vis[N];
8 vector <vector <int>> bcc;
9 stack <int> stk;
10
11 void dfs(int v, int p) {
12     stk.push(v);
13     vis[v] = true;
14     low[v] = depth[v] = (p == -1 ? 1 : depth[p] + 1);
15     for (int u : G[v]) {
16         if (u == p) continue;
17         if (!vis[u]) {
18             /// (v, u) 是樹邊
19             dfs(u, v);
20             low[v] = min(low[v], low[u]);
21             /// u 無法在不經過父邊的情況走到 v 的祖先
22             if (low[u] >= depth[v]) {
23                 bcc.emplace_back();
24                 while (stk.top() != u) {
25                     bcc.back().push_back(stk.top());
26                     stk.pop();
27                 }
28                 bcc.back().push_back(stk.top());
29                 stk.pop();
30                 bcc.back().push_back(v);
31             }
32         } else {
33             /// (v, u) 是回邊
34             low[v] = min(low[v], depth[u]);
35         }
36     }
37 }

```

## 6.7 圓方樹

```

1 #include <bits/stdc++.h>
2 #define lp(i,a,b) for(int i=(a);i<(b);i++)
3 #define pii pair<int,int>
4 #define pb push_back
5 #define ins insert
6 #define ff first
7 #define ss second
8 #define opa(x) cerr << #x << " = " << x << ", ";
9 #define op(x) cerr << #x << " = " << x << endl;
10 #define ops(x) cerr << x;
11 #define etr cerr << endl;
12 #define spc cerr << ' ';
13 #define BAE(x) (x).begin(), (x).end()
14 #define STL(x) cerr << #x << " : "; for(auto &qwe:x) cerr <<
15     qwe << ' '; cerr << endl;
16 #define deb1 cerr << "deb1" << endl;
17 #define deb2 cerr << "deb2" << endl;
18 #define deb3 cerr << "deb3" << endl;
19 #define deb4 cerr << "deb4" << endl;
20 #define deb5 cerr << "deb5" << endl;
21 #define bye exit(0);
22 using namespace std;

```

```

23 const int mxn = (int)(2e5) + 10;
24 const int mxlg = 17;
25 int last_special_node = (int)(1e5) + 1;
26 vector<int> E[mxn], F[mxn];
27
28 struct edg{
29     int fr, to;
30     edg(int _fr, int _to){
31         fr = _fr;
32         to = _to;
33     }
34 };
35 ostream& operator<<(ostream& os, edg x){os << x.fr << "--" <<
36     x.to;}
37 vector<edg> EV;
38
39 void tarjan(int v, int par, stack<int>& S){
40     static vector<int> dfn(mxn), low(mxn);
41     static vector<bool> to_add(mxn);
42     static int nowT = 0;
43
44     int childs = 0;
45     nowT += 1;
46     dfn[v] = low[v] = nowT;
47     for(auto &ne:E[v]){
48         int i = EV[ne].to;
49         if(i == par) continue;
50         if(!dfn[i]){
51             S.push(ne);
52             tarjan(i, v, S);
53             childs += 1;
54             low[v] = min(low[v], low[i]);
55
56             if(par >= 0 && low[i] >= dfn[v]){
57                 vector<int> bcc;
58                 int tmp;
59                 do{
60                     tmp = S.top(); S.pop();
61                     if(!to_add[EV[tmp].fr]){
62                         to_add[EV[tmp].fr] = true;
63                         bcc.pb(EV[tmp].fr);
64                     }
65                     if(!to_add[EV[tmp].to]){
66                         to_add[EV[tmp].to] = true;
67                         bcc.pb(EV[tmp].to);
68                     }
69                 }while(tmp != ne);
70                 for(auto &j:bcc){
71                     to_add[j] = false;
72                     F[last_special_node].pb(j);
73                     F[j].pb(last_special_node);
74                 }
75                 last_special_node += 1;
76             }
77         }
78         else{
79             low[v] = min(low[v], dfn[i]);
80             if(dfn[i] < dfn[v]){ // edge i--v will be visited
81                 twice at here, but we only need one.
82                 S.push(ne);
83             }
84         }
85     }
86     int dep[mxn], jmp[mxn][mxlg];

```

```

87 void dfs_lca(int v, int par, int depth){
88     dep[v] = depth;
89     for(auto &i:F[v]){
90         if(i == par) continue;
91         jmp[i][0] = v;
92         dfs_lca(i, v, depth + 1);
93     }
94 }
95
96 inline void build_lca(){
97     jmp[1][0] = 1;
98     dfs_lca(1, -1, 1);
99     lp(j,1,mxlg){
100         lp(i,1,mxn){
101             jmp[i][j] = jmp[jmp[i][j-1]][j-1];
102         }
103     }
104 }
105
106 inline int lca(int x, int y){
107     if(dep[x] < dep[y]){ swap(x, y); }
108
109     int diff = dep[x] - dep[y];
110     lp(j,0,mxlg){
111         if((diff >> j) & 1){
112             x = jmp[x][j];
113         }
114     }
115     if(x == y) return x;
116
117     for(int j = mxlg - 1; j >= 0; j--){
118         if(jmp[x][j] != jmp[y][j]){
119             x = jmp[x][j];
120             y = jmp[y][j];
121         }
122     }
123     return jmp[x][0];
124 }
125
126 inline bool can_reach(int fr, int to){
127     if(dep[to] > dep[fr]) return false;
128
129     int diff = dep[fr] - dep[to];
130     lp(j,0,mxlg){
131         if((diff >> j) & 1){
132             fr = jmp[fr][j];
133         }
134     }
135     return fr == to;
136 }
137
138 int main(){
139     ios::sync_with_stdio(false); cin.tie(0);
140     // freopen("test_input.txt", "r", stdin);
141     int n, m, q; cin >> n >> m >> q;
142     lp(i,0,m){
143         int u, v; cin >> u >> v;
144         E[u].pb(EV.size());
145         EV.pb(edg(u, v));
146         E[v].pb(EV.size());
147         EV.pb(edg(v, u));
148     }
149     E[0].pb(EV.size());
150     EV.pb(edg(0, 1));
151     stack<int> S;
152     tarjan(0, -1, S);

```

```

153     build_lca();
154
155     lp(queries,0,q){
156         int fr, to, relay; cin >> fr >> to >> relay;
157         if(fr == relay || to == relay){
158             cout << "NO\n";
159             continue;
160         }
161         if((can_reach(fr, relay) || can_reach(to, relay)) &&
162             dep[relay] >= dep[lca(fr, to)]){
163             cout << "NO\n";
164             continue;
165         }
166         cout << "YES\n";
167     }

```

## 6.8 SCC 與縮點

```

1  /*
2  給定一個有向圖，迴傳傳縮點後的圖、SCC 的資訊
3  所有點都以 based-0 編號
4
5  函式：
6  SCC_compress G(n): 宣告一個有 n 個點的圖
7  .add_edge(u, v): 加上一條邊 u -> v
8  .compress: O(n log n) 計算 G3、SCC、SCC_id 的資訊，並把縮點後
9  的結果存在 result 裡
10
11 SCC[i] = 某個 SCC 中的所有點
12 SCC_id[i] = 第 i 個點在第幾個 SCC
13 */
14 // c8b146
15 struct SCC_compress{
16     int n = 0, m = 0;
17     vector<vector<int>> G, inv_G, result;
18     vector<pair<int, int>> edges;
19     vector<bool> vis;
20     vector<int> order;
21
22     vector<vector<int>> SCC;
23     vector<int> SCC_id;
24
25     SCC_compress(int _n){
26         n = _n;
27         G.resize(n);
28         inv_G.resize(n);
29         result.resize(n);
30         vis.resize(n);
31         SCC_id.resize(n);
32     }
33
34     void add_edge(int u, int v){
35         G[u].push_back(v);
36         inv_G[v].push_back(u);
37         edges.push_back({u, v});
38         m++;
39     }
40
41     void dfs1(vector<vector<int>> &G, int now){
42         vis[now] = 1;

```

```

42     for (auto x : G[now]){
43         if (vis[x]==0){
44             dfs1(G, x);
45         }
46     }
47     order.push_back(now);
48     return;
49 }
50
51 void dfs2(vector<vector<int>> &G, int now){
52     SCC_id[now] = SCC.size()-1;
53     SCC.back().push_back(now);
54     vis[now] = 1;
55
56     for (auto x : G[now]){
57         if (vis[x]==0){
58             dfs2(G, x);
59         }
60     }
61     return;
62 }
63
64 void compress(){
65     fill(vis.begin(), vis.end(), 0);
66     for (int i=0; i<n; i++){
67         if (vis[i]==0){
68             dfs1(G, i);
69         }
70     }
71
72     fill(vis.begin(), vis.end(), 0);
73     reverse(order.begin(), order.end());
74     for (int i=0; i<n; i++){
75         if (vis[order[i]]==0){
76             SCC.push_back(vector<int>());
77             dfs2(inv_G, order[i]);
78         }
79     }
80
81     for (int i=0; i<m; i++){
82         if (SCC_id[edges[i].first]!=SCC_id[edges[i].second]){
83             result[SCC_id[edges[i].first]].push_back(
84                 SCC_id[edges[i].second]);
85         }
86     }
87     for (int i=0; i<SCC.size(); i++){
88         sort(result[i].begin(), result[i].end());
89         result[i].resize(unique(result[i].begin(), result
90             [i].end())-result[i].begin());
91     }
92 }
93 };

```

## 6.9 Dinic

```

1 // 時間複雜度:  $O(V^2E)$ 
2 struct Flow{
3     struct Edge{
4         int v, rc, rid;
5     };
6     vector<vector<Edge>> G;
7     void add(int u, int v, int c){

```

```

8         G[u].push_back({v, c, G[v].size()});
9         G[v].push_back({u, 0, G[u].size()-1});
10    }
11    vector<int> dis, it;
12
13    Flow(int n){
14        G.resize(n);
15        dis.resize(n);
16        it.resize(n);
17    }
18
19    int dfs(int u, int t, int f){
20        if (u==t || f==0) return f;
21        for (int &i=it[u]; i<G[u].size(); i++){
22            auto &[v, rc, rid] = G[u][i];
23            if (dis[v]!=dis[u]+1) continue;
24            int df = dfs(v, t, min(f, rc));
25            if (df<=0) continue;
26            rc -= df;
27            G[v][rid].rc += df;
28            return df;
29        }
30        return 0;
31    }
32
33    int flow(int s, int t){
34        int ans = 0;
35        while (true){
36            fill(dis.begin(), dis.end(), INF);
37            queue<int> q;
38            q.push(s);
39            dis[s] = 0;
40
41            while (q.size()){
42                int u = q.front(); q.pop();
43                for (auto [v, rc, rid] : G[u]){
44                    if (rc<=0 || dis[v]<INF) continue;
45                    dis[v] = dis[u]+1;
46                    q.push(v);
47                }
48            }
49            if (dis[t]==INF) break;
50
51            fill(it.begin(), it.end(), 0);
52            while (true){
53                int df = dfs(s, t, INF);
54                if (df<=0) break;
55                ans += df;
56            }
57        }
58        return ans;
59    }
60
61    // the code below constructs minimum cut
62    void dfs_mincut(int now, vector<bool> &vis){
63        vis[now] = true;
64        for (auto &[v, rc, rid] : G[now]){
65            if (vis[v]==false && rc>0){
66                dfs_mincut(v, vis);
67            }
68        }
69    }
70
71    vector<pair<int, int>> construct(int n, int s, vector<pair<
72        int, int>> &E){
73        // E is G without capacity
74        vector<bool> vis(n);

```

```

73        dfs_mincut(s, vis);
74        vector<pair<int, int>> ret;
75        for (auto &[u, v] : E){
76            if (vis[u]==true && vis[v]==false){
77                ret.emplace_back(u, v);
78            }
79        }
80        return ret;
81    }
82 };

```

## 6.10 Dijkstra

```

1 // 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
2
3 const int INF = 2e18; // 要確保 INF 開的足夠大
4
5 vector<vector<pair<int, int>>> G(n); // G[i] = <節點, 權重>
6 vector<int> dis(n, INF);
7 priority_queue<pair<int, int>, vector<pair<int, int>>,
8     greater<pair<int, int>>> pq;
9
10 dis[s] = 0;
11 pq.push({0, s});
12
13 while (pq.size()){
14     int now_dis = pq.top().first;
15     int now_node = pq.top().second;
16     pq.pop();
17
18     if (now_dis>dis[now_node]) continue;
19
20     for (auto x : G[now_node]){
21         if (now_dis+x.second<dis[x.first]){
22             dis[x.first] = now_dis+x.second;
23             pq.push({dis[x.first], x.first});
24         }
25     }
26 }

```

## 6.11 定理

- 最小點覆蓋 = 最大匹配 =  $n$  - 最大點獨立集
  - 最小點覆蓋：選最少的點讓所有的邊都有碰到一個點
  - 最大點獨立集：選最多不共邊的點
- 只有邊帶權的二分圖的定理 (可能不重要)
  - w-vertex-cover (帶權點覆蓋)：每條邊的兩個連接點被選中的次數總和至少要是  $w_e$ 。
  - w-weight matching (帶權匹配)
  - minimum vertex count of w-vertex-cover = maximum weight count of w-weight matching (一個點可以被選很多次，但邊不行)
- 點、邊都帶權的二分圖的定理 (可能不重要)
  - b-matching：假設  $v$  的點權是  $b_v$ ，那所有  $v$  的匹配邊  $e$  的權重都要滿足  $\sum w_e \leq b_v$ 。
  - The maximum w-weight of a b-matching equals the minimum b-weight of vertices in a w-vertex-cover.

## 6.12 MCMF

```

1 struct Flow {
2     struct Edge {
3         int u, rc, k, rv;
4     };
5
6     vector<vector<Edge>> G;
7     vector<int> par, par_eid;
8     Flow(int n) : G(n+1), par(n+1), par_eid(n+1) {}
9
10    // v->u, capacity: c, cost: k
11    void add(int v, int u, int c, int k){
12        G[v].push_back({u, c, k, SZ(G[u])});
13        G[u].push_back({v, 0, -k, SZ(G[v])-1});
14    }
15
16    // 3701d6
17    int spfa(int s, int t){
18        fill(ALL(par), -1);
19        vector<int> dis(SZ(par), INF);
20        vector<bool> in_q(SZ(par), false);
21        queue<int> Q;
22        dis[s] = 0;
23        in_q[s] = true;
24        Q.push(s);
25
26        while (!Q.empty()){
27            int v = Q.front();
28            Q.pop();
29            in_q[v] = false;
30
31            for (int i=0 ; i<SZ(G[v]) ; i++){
32                auto [u, rc, k, rv] = G[v][i];
33                if (rc>0 && dis[v]+k<dis[u]){
34                    dis[u] = dis[v]+k;
35                    par[u] = v;
36                    par_eid[u] = i;
37                    if (!in_q[u]) Q.push(u);
38                    in_q[u] = true;
39                }
40            }
41        }
42
43        return dis[t];
44    }
45
46    // return <max flow, min cost>, 150093
47    pair<int, int> flow(int s, int t){
48        int f1 = 0, cost = 0, d;
49        while ((d = spfa(s, t))<INF){
50            int cur = INF;
51            for (int v=t ; v!=s ; v=par[v])
52                cur = min(cur, G[par[v]][par_eid[v]].rc);
53            f1 += cur;
54            cost += d*cur;
55            for (int v=t ; v!=s ; v=par[v]){
56                G[par[v]][par_eid[v]].rc -= cur;
57                G[v][G[par[v]][par_eid[v]].rv].rc += cur;
58            }
59        }
60        return {f1, cost};
61    }
62
63    vector<pair<int, int>> construct(){

```

```

64    vector<pair<int, int>> ret;
65    for (int i=0 ; i<n ; i++){
66        for (auto x : G[i]){
67            if (x.rc==0){
68                ret.push_back({i+1, x.u-n+1});
69                break;
70            }
71        }
72    }
73    return ret;
74 }
75 };

```

## 7 Math

### 7.1 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

### 7.2 線性篩

```

1 const int MAX_N = 5e5;
2
3 // lpf[i] = i 的最小質因數
4 vector<int> prime, lpf(MAX_N);
5
6 void prime_init(){
7     for (int i=2 ; i<MAX_N ; i++){
8         if (lpf[i]==0){
9             lpf[i]=i;
10            prime.push_back(i);
11        }
12
13        for (int j : prime){
14            if (i*j>=MAX_N) break;
15            lpf[i*j]=j;
16            if (lpf[i]==j) break;
17        }
18    }
19 }

```

### 7.3 Lucas's-Theorem

```

1 // 對於很大的  $C^n_m$  對質數  $p$  取模 · 只要  $p$  不大就可以用。
2 int Lucas(int n, int m, int p){
3     if (m==0) return 1;
4     return (C(n%p, m%p, p)*Lucas(n/p, m/p, p)%p);
5 }

```

## 7.4 Matrix

```

1 struct Matrix{
2     int n, m;
3     vector<vector<int>> arr;
4
5     Matrix(int _n, int _m){
6         n = _n;
7         m = _m;
8         arr.resize(n, vector<int>(m));
9     }
10
11    Matrix operator * (const Matrix B){
12        Matrix ret(n, B.m);
13
14        for (int i=0 ; i<n ; i++){
15            for (int j=0 ; j<B.m ; j++){
16                for (int k=0 ; k<m ; k++){
17                    ret.arr[i][j] += arr[i][k]*B.arr[k][j];
18                    ret.arr[i][j] %= MOD;
19                }
20            }
21        }
22
23        return ret;
24    }
25 };

```

### 7.5 最大質因數

```

1 void max_fac(int n, int &ret){
2     if (n<=ret || n<2) return;
3     if (isprime(n)){
4         ret = max(ret, n);
5         return;
6     }
7
8     int p = Pollard_Rho(n);
9     max_fac(p, ret), max_fac(n/p, ret);
10 }

```

### 7.6 中國剩餘定理 ( $m$ 不互質 )

```

1 int extgcd(int a, int b, int &x, int &y){
2     if (b==0){
3         x=1, y=0;
4         return a;
5     }
6
7     int ret=extgcd(b, a%b, y, x);
8     y-=a/b*x;
9     return ret;
10 }
11
12 // 對於方程組的式子兩兩求解
13 // {是否有解, {a, m}}
14 pair<bool, pair<int, int>> CRT(int a1, int m1, int a2, int m2){
15     return {true, {a1, m1}};
16 }

```

```

15 int g=__gcd(m1, m2);
16 if ((a2-a1)%g!=0) return {0, {-1, -1}};
17
18 int x, y;
19 extgcd(m1, m2, x, y);
20
21 x=(a2-a1)*x/g; // 兩者不能相反
22 a1=x*m1+a1;
23 m1=m1*m2/g;
24 a1=(a1%m1+m1)%m1;
25 return {1, {a1, m1}};
26 }

```

## 7.7 歐拉公式

```

1 // phi(n) = 小於 n 並與 n 互質的正整數數量。
2 // O(sqrt(n)) · 回傳 phi(n)
3 int phi(int n){
4     int ret = n;
5
6     for (int i=2 ; i*i<=n ; i++){
7         if (n%i==0){
8             while (n%i==0) n /= i;
9             ret = ret*(i-1)/i;
10        }
11    }
12    if (n>1) ret = ret*(n-1)/n;
13
14    return ret;
15 }
16
17 // O(n log n) · 回傳 1~n 的 phi 值
18 vector<int> phi_1_to_n(int n){
19     vector<int> phi(n+1);
20     phi[0]=0;
21     phi[1]=1;
22
23     for (int i=2 ; i<=n ; i++){
24         phi[i]=i-1;
25     }
26
27     for (int i=2 ; i<=n ; i++){
28         for (int j=2*i ; j<=n ; j+=i){ // 枚舉所有倍數
29             phi[j]-=phi[i];
30         }
31     }
32
33     return phi;
34 }

```

## 7.8 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：

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

## 7.9 Fraction

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 /// Fraction template starts ///
5 #define fraction_template_bonus_check
6 const long long ll_overflow_warning_value = (long long)(3e9);
7
8 long long gcd(long long a, long long b){
9     if(a == 0) return 0;
10    if(b == 0) return a;
11    if(a < b) return gcd(b,a);
12    return gcd(b, a%b);
13 }
14 struct frac{
15     long long a, b;
16     frac(long long _a = 0, long long _b = 1){
17         a = _a; b = _b;
18         if(b == 0){
19             cerr << "Error: division by zero\n";
20             cerr << "Called : Constructor(" << _a << ", " <<
                _b << ")\n";
21             return;
22         }
23         if(a == 0){b = 1; return;}
24         if(b < 0){a = -a; b = -b;}
25         long long gcd_ab = gcd(std::abs(a), b);
26         if(gcd_ab != 1){a /= gcd_ab; b /= gcd_ab;}
27
28         #ifdef fraction_template_bonus_check
29         if(std::abs(a) > ll_overflow_warning_value || b >
            ll_overflow_warning_value){
30             cerr << "Overflow warning : " << a << "/" << b <<
                "\n";
31         }
32         #endif // fraction_template_bonus_check
33     }
34     frac operator+(frac const &B){
35         return frac(a*(B.b)+(B.a)*b, b*(B.b));
36     }
37     frac operator-(frac const &B){
38         return frac(a*(B.b)-(B.a)*b, b*(B.b));
39     }
40     frac operator*(frac const &B){
41         return frac(a*(B.a), b*(B.b));
42     }
43     frac operator/(frac const &B){
44         return frac(a*(B.b), b*(B.a));
45     }
46
47     frac operator+=(frac const &B){
48         *this = frac(a*(B.b)+(B.a)*b, b*(B.b));
49     }
50     frac operator-=(frac const &B){
51         *this = frac(a*(B.b)-(B.a)*b, b*(B.b));
52     }
53     frac operator*=(frac const &B){
54         *this = frac(a*(B.a), b*(B.b));
55     }
56     frac operator/=(frac const &B){
57         *this = frac(a*(B.b), b*(B.a));
58     }
59
60     frac abs(){
61         a = std::abs(a);
62         return *this;
63     }
64
65     bool operator<(frac const &B){
66         return a*B.b < B.a*b;
67     }
68     bool operator<=(frac const &B){
69         return a*B.b <= B.a*b;
70     }
71     bool operator>(frac const &B){
72         return a*B.b > B.a*b;
73     }
74     bool operator>=(frac const &B){
75         return a*B.b >= B.a*b;
76     }
77
78     ostream& operator<<(ostream &os, const frac& A){
79         os << A.a << "/" << A.b;
80         return os;
81     }
82
83     /// Fraction template ends ///
84
85     void test(frac A, frac B){
86         cout << "A = " << A << endl;
87         cout << "B = " << B << endl;
88         cout << endl;
89         cout << "A + B = " << A + B << endl;
90         cout << "A - B = " << A - B << endl;
91         cout << "A * B = " << A * B << endl;
92         cout << "A / B = " << A / B << endl;
93         cout << endl;
94         cout << "(A < B) = " << (A < B) << endl;
95         cout << "(A <= B) = " << (A <= B) << endl;
96         cout << "(A > B) = " << (A > B) << endl;
97         cout << "(A >= B) = " << (A >= B) << endl;
98         cout << "(A == B) = " << (A == B) << endl;
99         cout << "(A != B) = " << (A != B) << endl;
100        cout << "-----\n";
101        return;
102    }
103 }

```

```

61     return a*B.b > B.a*b;
62     bool operator>=(frac const &B){
63         return a*B.b >= B.a*b;
64     }
65     bool operator==(frac const &B){
66         return a * B.b == B.a * b;
67     }
68     bool operator!=(frac const &B){
69         return a * B.b != B.a * b;
70     }
71
72     ostream& operator<<(ostream &os, const frac& A){
73         os << A.a << "/" << A.b;
74         return os;
75     }
76
77     /// Fraction template ends ///
78
79     void test(frac A, frac B){
80         cout << "A = " << A << endl;
81         cout << "B = " << B << endl;
82         cout << endl;
83         cout << "A + B = " << A + B << endl;
84         cout << "A - B = " << A - B << endl;
85         cout << "A * B = " << A * B << endl;
86         cout << "A / B = " << A / B << endl;
87         cout << endl;
88         cout << "(A < B) = " << (A < B) << endl;
89         cout << "(A <= B) = " << (A <= B) << endl;
90         cout << "(A > B) = " << (A > B) << endl;
91         cout << "(A >= B) = " << (A >= B) << endl;
92         cout << "(A == B) = " << (A == B) << endl;
93         cout << "(A != B) = " << (A != B) << endl;
94         cout << "-----\n";
95         return;
96     }
97
98     int main(){
99         frac tmp1(-7, 2);
100        frac tmp2(5, 3);
101        test(tmp1, tmp2);
102
103        frac tmp3(-7);
104        frac tmp4(0);
105        test(tmp3, tmp4);
106        return 0;
107    }

```

## 7.10 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)

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

## 7.11 Quick-Pow

```

1 int qp(int b, int p, int m = MOD){
2     int ret = 1;
3     for (; p ; p>>=1){
4         if (p&1) ret = ret*b%m;
5         b = b*b%m;

```

```

6 |     }
7 |     return ret;
8 | }

```

## 7.12 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。

## 7.13 Josephus

```

1 | // 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
2 | int solve(int n, int k){
3 |     if (n==1) return 1;
4 |     if (k<=(n+1)/2){
5 |         if (2*k>n) return 2*k%n;
6 |         else return 2*k;
7 |     }else{
8 |         int res=solve(n/2, k-(n+1)/2);
9 |         if (n&1) return 2*res+1;
10 |        else return 2*res-1;
11 |     }
12 | }

```

## 7.14 數論分塊

```

1 | /*
2 | 時間複雜度為 O(sqrt(n))
3 | 區間為 [l, r]
4 | */
5 | for(int i=1 ; i<=n ; i++){
6 |     int l = i, r = n/(n/i);
7 |     i = r;
8 |     ans.push_back(r);
9 | }

```

## 7.15 Miller-Rabin

```

1 | // O(1)
2 | typedef Uint unsigned long long
3 | Uint modmul(Uint a, Uint b, Uint m) {
4 |     int ret = a*b - m*(Uint)(1.L/m*a*b);
5 |     return ret + m*(ret < 0) - m*(ret>=(int)m);
6 | }
7 |
8 | int qp(int b, int p, int m){
9 |     int ret = 1;
10 |    for ( ; p ; p>>=1){
11 |        if (p&1){
12 |            ret = modmul(ret, b, m);
13 |        }

```

```

14 |        b = modmul(b, b, m);
15 |    }
16 |    return ret;
17 | }
18 |
19 | // ed23aa
20 | vector<int> llsprp = {2, 325, 9375, 28178, 450775, 9780504,
21 | 1795265022};
22 | bool isprime(int n, vector<int> sprp = llsprp){
23 |     if (n==2) return 1;
24 |     if (n<2 || n%2==0) return 0;
25 |
26 |     int t = 0;
27 |     int u = n-1;
28 |     for ( ; u%2==0 ; t++) u>>=1;
29 |
30 |     for (int i=0 ; i<sprp.size() ; i++){
31 |         int a = sprp[i]%n;
32 |         if (a==0 || a==1 || a==n-1) continue;
33 |         int x = qp(a, u, n);
34 |         if (x==1 || x==n-1) continue;
35 |         for (int j=0 ; j<t ; j++){
36 |             x = modmul(x, x, n);
37 |             if (x==1) return 0;
38 |             if (x==n-1) break;
39 |         }
40 |         if (x==n-1) continue;
41 |         return 0;
42 |     }
43 |
44 |     return 1;
45 | }

```

## 7.16 Pollard-Rho

```

1 | mt19937 seed(chrono::steady_clock::now().time_since_epoch().
2 | count());
3 | int rnd(int l, int r){
4 |     return uniform_int_distribution<int>(l, r)(seed);
5 | }
6 | // O(n^{1/4}) 回傳 1 或自己的因數，記得先判斷 n 是不是質數
7 | // (用 Miller-Rabin)
8 | // c1670c
9 | int Pollard_Rho(int n){
10 |    int s = 0, t = 0;
11 |    int c = rnd(1, n-1);
12 |
13 |    int step = 0, goal = 1;
14 |    int val = 1;
15 |
16 |    for (goal=1 ; ; goal<=1, s=t, val=1){
17 |        for (step=1 ; step<=goal ; step++){
18 |
19 |            t = ((__int128)t*t+c)%n;
20 |            val = ((__int128)val*abs(t-s)%n;
21 |
22 |            if ((step % 127) == 0){
23 |                int d = __gcd(val, n);
24 |                if (d>1) return d;

```

```

25 |     }
26 |
27 |     int d = __gcd(val, n);
28 |     if (d>1) return d;
29 | }
30 | }

```

## 7.17 中國剩餘定理 (m 互質)

```

1 | vector<int> a, m;
2 |
3 | int extgcd(int a, int b, int &x, int &y){
4 |     if (b==0){
5 |         x=1, y=0;
6 |         return a;
7 |     }
8 |
9 |     int ret=extgcd(b, a%b, y, x);
10 |    y-=a/b*x;
11 |    return ret;
12 | }
13 |
14 | // n = 有幾個式子，求解 x \equiv a_i \pmod{m_i}
15 | int CRT(int n, vector<int> &a, vector<int> &m){
16 |     int p=1, ans=0;
17 |
18 |     vector<int> M(n), inv_M(n);
19 |
20 |     for (int i=0 ; i<n ; i++) p*=m[i];
21 |     for (int i=0 ; i<n ; i++){
22 |         M[i]=p/m[i];
23 |         int tmp;
24 |         extgcd(M[i], m[i], inv_M[i], tmp);
25 |         ans+=a[i]*inv_M[i]*M[i];
26 |         ans%=p;
27 |     }
28 |
29 |     return (ans%p+p)%p;
30 | }

```

## 7.18 Catalan

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

## 7.19 數論定理

1.  $1 \sim x$  質數的數量  $\approx \frac{x}{\ln x}$
2.  $1 \sim x$  的因數的數量  $\approx x^{\frac{1}{3}}$
3.  $x$  的質因數的數量  $\approx \log \log x$
4.  $p$  is a prime number  $\Leftrightarrow (p-1)! \equiv -1 \pmod{p}$
5. 每個正整數都可以表示成四個整數的平方和
6. 任何大於 2 的整數都可以表示成兩個質數的和



## 8 String

### 8.1 Hash

```

1 mt19937 rnd(chrono::steady_clock::now().time_since_epoch().
  count());
2 int A = rnd(), B = 1000000007;
3
4 vector<int> myPow, myPre;
5 void hash_init(string s){
6     myPow.resize(s.size());
7     myPre.resize(s.size());
8
9     for (int i=0 ; i<s.size() ; i++){
10         if (i==0){
11             myPow[i] = 1;
12             myPre[i] = s[i];
13         }else{
14             myPow[i] = myPow[i-1]*A%B;
15             myPre[i] = (myPre[i-1]*A+s[i])%B;
16         }
17     }
18
19     return;
20 }
21
22 int hash_value(int l, int r){ // 取得 s[l..r] 的數值
23     if (l==0) return myPre[r];
24     return ((myPre[r]-myPre[l-1]*myPow[r-l+1])%B+B)%B;
25 }

```

### 8.2 Manacher

```

1 string Manacher(string str) {
2     string tmp = "$#";
3     for(char i : str) {
4         tmp += i;
5         tmp += '#';
6     }
7
8     vector<int> p(tmp.size(), 0);
9     int mx = 0, id = 0, len = 0, center = 0;
10    for(int i=1 ; i<(int)tmp.size() ; i++) {
11        p[i] = mx > i ? min(p[id*2-i], mx-i) : 1;
12
13        while(tmp[i+p[i]] == tmp[i-p[i]]) p[i]++;
14        if(mx<i+p[i]) mx = i+p[i], id = i;
15        if(len<p[i]) len = p[i], center = i;
16    }
17    return str.substr((center-len)/2, len-1);
18 }

```

### 8.3 Z-Function

```

1 // 定義一個長度為 n 的文本為 T · 則陣列 Z 的 Z[i] 代表 T[0:n]
  和 T[i:n] 最長共同前綴
2 // bcfbd6

```

```

3 vector<int> z_function(string s){
4     vector<int> ret(s.size());
5     int ll = 0, rr = 0;
6
7     for (int i=1 ; i<s.size() ; i++){
8         int j = 0;
9
10        if (i<rr) j = min(ret[i-ll], rr-i);
11        while (s[j]==s[i+j]) j++;
12        ret[i] = j;
13
14        if (i+j>rr){
15            ll = i;
16            rr = i+j;
17        }
18    }
19
20    ret[0] = s.size();
21    return ret;
22 }

```

### 8.4 KMP

```

1 // 給一個字串 S · 定義函數 \pi(i) = k 代表 S[1 ... k] = S[i-k
  +1 ... i]
2 // 4c61a3
3 vector<int> KMP(string &s){
4     n = SZ(s);
5     vector<int> ret(n);
6     int now = 0;
7     for (int i=1 ; i<n ; i++){
8         int j = ret[i-1];
9         while (j>0 && s[i]!=s[j]){
10            j = ret[j-1];
11        }
12        if (s[i]==s[j]) j++;
13        ret[i] = j;
14    }
15    return ret;
16 }

```

### 8.5 Suffix-Array

```

1 // 注意 · 當 |s|=1 時 · lcp 不會有值 · 務必測試 |s|=1 的 case
2 struct SuffixArray {
3     vector<int> sa, lcp;
4     SuffixArray(string& s, int lim = 256) {
5         // 49c4d2
6         int n = SZ(s)+1, k = 0, a, b;
7         vector<int> x(ALL(s)), y(n), ws(max(n, lim)), rank(n)
8         ;
9         x.push_back(0);
10        sa = lcp = y;
11        iota(ALL(sa), 0);
12        for (int j=0, p=0 ; p<n ; j=max(1LL, j*2), lim=p) {
13            p = j;
14            iota(ALL(y), n-j);
15            for (int i=0 ; i<n ; i++) if (sa[i] >= j) y[p++]
16                = sa[i] - j;
17            fill(ALL(ws), 0);

```

```

16        for (int i=0 ; i<n ; i++) ws[x[i]]++;
17        for (int i=1 ; i<lim ; i++) ws[i] += ws[i - 1];
18        for (int i = n; i--;) sa[--ws[x[y[i]]]] = y[i];
19        swap(x, y), p = 1, x[sa[0]] = 0;
20        for (int i=1 ; i<n ; i++){
21            a = sa[i - 1];
22            b = sa[i];
23            x[b] = (y[a] == y[b] && y[a + j] == y[b + j])
24                ? p - 1 : p++;
25        }
26
27        // 7181dd
28        for (int i=1 ; i<n ; i++) rank[sa[i]] = i;
29        for (int i=0, j ; i<n-1 ; lcp[rank[i+1]]=k)
30            for (k && k--, j=sa[rank[i]-1] ; i+k<SZ(s) && j+k
31                <SZ(s) && s[i+k]==s[j+k] ; k++);
32        sa.erase(sa.begin());
33        lcp.erase(lcp.begin(), lcp.begin()+2);
34    }
};

```

### 8.6 Min-Rotation

```

1 // 9d296f
2 int minRotation(string s) {
3     int a=0, N=SZ(s); s += s;
4     for (int b=0 ; b<N ; b++){
5         for (int k=0 ; k<N ; k++){
6             if (a+k == b || s[a+k] < s[b+k]) {b += max(0LL, k
7                 -1); break;}
8             if (s[a+k] > s[b+k]) {a = b; break;}
9         }
10    }
11    return a;

```

# Contents

<b>1 Misc</b>	<b>1</b>	<b>3.9 Add-Set-Segment-Tree</b>	<b>6</b>	<b>7 Math</b>	<b>12</b>
1.1 Xor-Basis	1	3.10 Treap	6	7.1 Burnside's-Lemma	12
1.2 Default-Code	1	<b>4 Dynamic-Programming</b>	<b>7</b>	7.2 線性篩	12
1.3 Radix-Sort	1	4.1 SOS-DP	7	7.3 Lucas's-Theorem	12
1.4 Set-Pq-Sort	1	4.2 Digit-DP	7	7.4 Matrix	12
1.5 2-SAT	1	4.3 整數拆分	7	7.5 最大質因數	12
1.6 Enumerate-Subset	2	<b>5 Geometry</b>	<b>7</b>	7.6 中國剩餘定理 ( m 不互質 )	12
1.7 Fast-Input	2	5.1 Point-Struct	7	7.7 歐拉公式	13
1.8 setup	2	5.2 Line-Intersection	7	7.8 歐拉定理	13
<b>2 Convlution</b>	<b>2</b>	5.3 Pick's-Theorem	7	7.9 Fraction	13
2.1 FFT	2	5.4 Point-In-Polygon	7	7.10 錯排公式	13
2.2 FFT-2	2	5.5 Convex-Hull	8	7.11 Quick-Pow	13
2.3 NTT-998244353	3	<b>6 Graph</b>	<b>8</b>	7.12 二元一次方程式	14
2.4 FFT-mod	3	6.1 Find-Bridge	8	7.13 Josephus	14
<b>3 Data-Structure</b>	<b>4</b>	6.2 Find-AP	8	7.14 數論分塊	14
3.1 GP-Hash-Table	4	6.3 HLD	8	7.15 Miller-Rabin	14
3.2 Sparse-Table	4	6.4 Tree-Isomorphism	8	7.16 Pollard-Rho	14
3.3 Order-Set	4	6.5 Bridge BCC	9	7.17 中國剩餘定理 ( m 互質 )	14
3.4 BIT	4	6.6 Cut BCC	9	7.18 Catalan	14
3.5 Persistent-Segment-Tree	4	6.7 圓方樹	9	7.19 數論定理	14
3.6 Trie	5	6.8 SCC 與縮點	10	<b>8 String</b>	<b>15</b>
3.7 LC-Segment-Tree	5	6.9 Dinic	11	8.1 Hash	15
3.8 Persistent-Disjoint-Set	5	6.10 Dijkstra	11	8.2 Manacher	15
		6.11 定理	11	8.3 Z-Function	15
		6.12 MCMF	12	8.4 KMP	15
				8.5 Suffix-Array	15
				8.6 Min-Rotation	15