Contents

1 Misc

1.1 Custom Set PQ Sort

```cpp
// priority_queue，務必檢查相等的 case，給所有元素一個排序的依據
struct cmp{
    bool operator () (Data a, Data b){
        return a.x<b.x;
    }
};
priority_queue<Data, vector<Data>, cmp> pq;

// set，務必檢查相等的 case，給所有元素一個排序的依據
auto cmp = [](int a, int b) {
    return a > b;
};
set<int, decltype(cmp)> s = {1, 2, 3, 4, 5};
cout << *s.begin() << '\n';
```

1.2 Default Code New

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

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

void solve(){

}

signed main(){
    ios::sync_with_stdio(0), cin.tie(0);

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

    return 0;
}
```

1.3 Default Code Old

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

#ifdef LOCAL
#define cout cout << "\033[0;32m"
#define cerr cerr << "\033[0;31m"
#define endl endl << "\033[0m"
#else
#pragma GCC optimize("O3,unroll-loops")
#pragma GCC target("avx,avx2,sse,sse2,sse3,sse4,popcnt")
#define endl "\n"
#endif

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

void solve1(){

    return;
}

signed main(){

    fastio;

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

    return 0;
}
```

1.4 Enumerate Subset

```cpp
// 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;
    }
}
```

1.5 Fast Input

```cpp
// fast IO
// 6f8879
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
}
```

1.6 OEIS

```cpp
// 若一個線性遞迴有 k 項，給他恰好 2*k 個項可以求出線性遞迴
// f915c2
template <typename T>
vector<T> BerlekampMassey(vector<T> a) {
    auto scalarProduct = [](vector<T> v, T c) {
        for (T &x: v) x *= c;
        return v;
    };
    vector<T> s, best;
    int bestPos = 0;
    for (size_t i = 0; i < a.size(); i++) {
        T error = a[i];
        for (size_t j = 0; j < s.size(); j++) error -= s[j] * a[i-1-j];
        if (error == 0) continue;
        if (s.empty()) {
            s.resize(i + 1);
            bestPos = i;
            best.push_back(1 / error);
            continue;
        }
        vector<T> fix = scalarProduct(best, error);
        fix.insert(fix.begin(), i - bestPos - 1, 0);
        if (fix.size() >= s.size()) {
            best = scalarProduct(s, - 1 / error);
            best.insert(best.begin(), 1 / error);
            bestPos = i;
            s.resize(fix.size());
        }
        for (size_t j = 0; j < fix.size(); j++)
            s[j] += fix[j];
    }
    return s;
}
```

1.7 Xor Basis

```cpp
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 0;
}
```

```cpp
// 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;
}
```

1.8 random int

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

1.9 Python

```python
sys.setrecursionlimit(100000)

sys.set_int_max_str_digits(10000)
```

1.10 diff

```sh
set -e
g++ ac.cpp -o ac
g++ wa.cpp -o wa
for ((i=0;;i++))
do
    echo "$i"
    python3 gen.py > input
    ./ac < input > ac.out
    ./wa < input > wa.out
    diff ac.out wa.out || break
done
```

1.11 hash command

```sh
cat file.cpp  | cpp -dD -P -fpreprocessed | tr -d "[:space:]"
      | md5sum | cut -c-6
```

1.12 run

```python
import os

f = "pA"

while 1:
    i = input("input: ")
    p = os.listdir(".")
    if i != "":
        f = i
    print(f"file = {f}")
    if os.system(f"g++ {f}.cpp -std=c++17 -Wall -Wextra -
        Wshadow -O2 -D LOCAL -g -fsanitize=undefined,address
         -o {f}"):
        print("CE")
        continue
    os.system("clear")

    for x in sorted(p):
        if f in x and ".in" in x:
            print(x)
            if os.system(f"./{f} < {x}"):
                print("RE")
            print()
```

1.13 setup

```
se nu rnu bs=2 sw=4 ts=4 hls ls=2 si acd bo=all mouse=a

:inoremap " ""<Esc>i
:inoremap {<CR> {<CR>}<Esc>ko
:inoremap {{ {}<ESC>i

function! F(...)
  execute '!./%:r < ./' . a:1
endfunction
command! -nargs=* R call F(<f-args>)

map <F7> :w<bar>!g++ "%" -o %:r -std=c++17 -Wall -Wextra -
    Wshadow -O2 -DLOCAL -g -fsanitize=undefined,address<CR>
map <F8> :!./%:r<CR>
map <F9> :!./%:r < ./%:r.in<CR>

ca hash w !cpp -dD -P -fpreprocessed \| tr -d "[:space:]" \|
    md5sum \| cut -c-6

" i+<esc>25A---+<esc>
" o|<esc>25A   |<esc>
" "ggVGyG35pGdd
```

2 Convolution

2.1 FFT any mod

```cpp
/*
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;

// b9c90a
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;
}

// d3c65e
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;
}
```

2.2 FFT new

```
typedef complex<double> cd;

// b9c90a
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;
}

// 39029d
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;
}
```

2.3 FFT short

```
#define int long long

using Cplx = complex<double>;
const double pi = acos(-1);
const int mod = 998244353, g = 3;
int power(int a, int b) {
    int res = 1;
    while (b) {
        if (b & 1) res = res * a % mod;
        a = a * a % mod;
        b >>= 1;
    }
    return res;
}
int inv(int x) { return power(x, mod - 2); }
// FFT use Cplx, NTT use ll
void FFT(vector<int> &a, int n, int op) {
    // n must be 2^k
    vector<int> R(n);
    FOR (i, 0, n - 1)
        R[i] = R[i/2]/2 + (i&1)*(n/2);
    FOR (i, 0, n - 1)
        if (i < R[i]) swap(a[i], a[R[i]]);
    for (int m = 2; m <= n; m *= 2) {
        // Cplx w1({cos(2*pi/m), sin(2*pi/m)*op});
        int w1 = power(g, (mod-1)/m * op + mod-1);
        for (int i = 0; i < n; i += m) {
            // Cplx wk({1, 0});
            int wk = 1;
            FOR (k, 0, m / 2 - 1) {
                auto x = a[i+k], y = a[i+k+m/2] * wk % mod;
                a[i+k] = (x+y) % mod;
                a[i+k+m/2] = (x-y+mod) % mod;
                wk = wk * w1 % mod;
            }
        }
    }
    if (op == -1)
        FOR (i, 0, n - 1) {
            // a[i] = a[i] / n;
            a[i] = a[i] * inv(n) % mod;
        }
}
```

2.4 FWT

```
// 已經把 mint 刪掉，需要增加註解
vector<int> xor_convolution(vector<int> a, vector<int> b, int
     k) {
    if (k == 0) {
        return vector<int>{a[0] * b[0]};
    }
    vector<int> aa(1 << (k - 1)), bb(1 << (k - 1));
    FOR (i, 0, (1 << (k - 1)) - 1) {
        aa[i] = a[i] + a[i + (1 << (k - 1))];
        bb[i] = b[i] + b[i + (1 << (k - 1))];
    }
    vector<int> X = xor_convolution(aa, bb, k - 1);
    FOR (i, 0, (1 << (k - 1)) - 1) {
        aa[i] = a[i] - a[i + (1 << (k - 1))];
        bb[i] = b[i] - b[i + (1 << (k - 1))];
    }
    vector<int> Y = xor_convolution(aa, bb, k - 1);
    vector<int> c(1 << k);
    FOR (i, 0, (1 << (k - 1)) - 1) {
        c[i                 ] = (X[i] + Y[i]) / 2;
        c[i + (1 << (k - 1))] = (X[i] - Y[i]) / 2;
    }
    return c;
};
```

2.5 NTT mod 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.

// 9cd58a
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);
            }
        }
    }
}

// 0b0e99
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;
}
```

2.6 Min Convolution Concave Concave

```
// 需要增加註解
// min convolution
vector<int> mkk(vector<int> a, vector<int> b) {
    vector<int> slope;
    FOR (i, 1, ssize(a) - 1) slope.pb(a[i] - a[i - 1]);
    FOR (i, 1, ssize(b) - 1) slope.pb(b[i] - b[i - 1]);
    sort(all(slope));
    slope.insert(begin(slope), a[0] + b[0]);
    partial_sum(all(slope), begin(slope));
    return slope;
}
```

3 Data-Structure

3.1 BIT

```
vector<int> BIT(MAX_SIZE);

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

3.2 Disjoint Set Persistent

```
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;
    }
};
```

3.3 PBDS 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;
```

3.4 PBDS 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
不能在 #define int long long 後 #include 檔案
*/

#include <ext/pb_ds/assoc_container.hpp>
#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;
```

3.5 Segment Tree Add Set

```
// [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 SegmentTree{
    struct node{
        int add_tag = 0;
        int set_tag = 0;
        int sum = 0;
        int ma = 0;
    };

    vector<node> arr;

    SegmentTree(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;
    }
```

```
 // cce0c8
void push(int idx, int ll, int rr){
    if (arr[idx].set_tag!=0){
        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){
        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;
```

3.6 Segment Tree Li Chao Line

```
/*
全部都是 0-based

宣告
LC_Segment_Tree st(n);

函式：
update({a, b})：插入一條 y=ax+b 的全域直線
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==0) return;
        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==0) return INF;
        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));
        }
    }
};
```

3.7 Segment Tree Li Chao Segment

```
/*
全部都是 0-based

宣告
LC_Segment_Tree st(n);

函式：
update_segment({a, b}, ql, qr)：在 [ql, qr) 插入一條 y=ax+b 的線段
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==0) return;
        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;
    }

    // 在 [ql, qr) 加上一條 val 的線段
    void update_segment(Node val, int ql, int qr, int idx = 0, int ll = 0, int rr = MAX_V){
        if (rr-ll==0) return;
        if (rr<=ql || qr<=ll) return;
        if (ql<=ll && rr<=qr){
            update(val, idx, ll, rr);
            return;
        }

        int mid = (ll+rr)/2;
        update_segment(val, ql, qr, idx*2+1, ll, mid);
        update_segment(val, ql, qr, idx*2+2, mid, rr);
        return;
    }

    int query(int x, int idx = 0, int ll = 0, int rr = MAX_V){
        if (rr-ll==0) return INF;
        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));
        }
    }
};
```

3.8 Segment Tree Persistent

```
/*
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++;
    }
};
```

3.9 Sparse Table

```
struct SparseTable{
    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 query(int ll, int rr){
        int h = __lg(rr-ll);
        return min(st[h][ll], st[h][rr-(1<<h)]);
    }
};
```

3.10 Treap

```cpp
struct Treap{
    Treap *l = nullptr, *r = nullptr;
    int pri = rand(), val = 0, sz = 1;

    Treap(int _val){
        val = _val;
    }
};

int size(Treap *t){return t ? t->sz : 0;}
void pull(Treap *t){
    t->sz = size(t->l)+size(t->r)+1;
}

Treap* merge(Treap *a, Treap *b){
    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;
    }

}

pair<Treap*, Treap*> split(Treap *&t, int k){ // 1-based <前 k 個元素, 其他元素>
    if (!t) return {};
    if (size(t->l)>=k){
        auto pa = split(t->l, k);
        t->l = pa.second;
        pull(t);
        return {pa.first, t};
    }else{
        auto pa = split(t->r, k-size(t->l)-1);
        t->r = pa.first;
        pull(t);
        return {t, pa.second};
    }
}

// functions
Treap* build(vector<int> v){
    Treap* ret = nullptr;
    for (int i=0 ; i<v.size() ; i++){
        ret = merge(ret, new Treap(v[i]));
    }
    return ret;
}

array<Treap*, 3> cut(Treap *t, int l, int r){ // 1-based <前 1~l-1 個元素, l~r 個元素, r+1 個元素>
    array<Treap*, 3> ret;
    tie(ret[1], ret[2]) = split(t, r);
    tie(ret[0], ret[1]) = split(ret[1], l-1);
    return ret;
}

void print(Treap *t, bool flag = true){
    if (t->l!=0) print(t->l, false);
    cout << t->val;
    if (t->r!=0) print(t->r, false);
    if (flag) cout << endl;
}
```

3.11 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;
```

4 Dynamic-Programming

4.1 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;
}
```

4.2 Knaspack On Tree

```cpp
// 需要重構、需要增加註解
#include <bits/stdc++.h>
#define F first
#define S second
#define all(x) begin(x), end(x)
using namespace std;
```

```cpp
#define chmax(a, b) (a) = (a) < (b) ? (b) : (a)
#define chmin(a, b) (a) = (a) < (b) ? (a) : (b)

#define ll long long

#define FOR(i, a, b) for (int i = a; i <= b; i++)

int N, W, cur;
vector<int> w, v, sz;
vector<vector<int>> adj, dp;

void dfs(int x) {
    sz[x] = 1;
    for (int i : adj[x]) dfs(i), sz[x] += sz[i];
    cur++;
    // choose x
    for (int i=w[x] ; i<=W ; i++){
        dp[cur][i] = dp[cur - 1][i - w[x]] + v[x];
    }
    // not choose x
    for (int i=0 ; i<=W ; i++){
        chmax(dp[cur][i], dp[cur - sz[x]][i]);
    }
}

signed main() {
    cin >> N >> W;
    adj.resize(N + 1);
    w.assign(N + 1, 0);
    v.assign(N + 1, 0);
    sz.assign(N + 1, 0);
    dp.assign(N + 2, vector<int>(W + 1, 0));
    for (int i=1 ; i<=N ; i++){
        int p; cin >> p;
        adj[p].push_back(i);
    }

    for (int i=1 ; i<=N ; i++) cin >> w[i];
    for (int i=1 ; i<=N ; i++) cin >> v[i];
    dfs(0);
    cout << dp[N + 1][W] << '\n';
}
```

4.3 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)];
        }
    }
}
```

4.4 Integer Partition

$dp[i][x]$ = 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 Geometry Struct

```cpp
// 判斷數值正負：{1:正數,0:零,-1:負數}
int sign(long long x) {return (x >= 0) ? ((bool)x) : -1; }
int sign(double x) {
    return (abs(x) < 1e-9) ? 0 : (x > 0 ? 1 : -1);
}

template<typename T>
struct point {
    T x, y;
    point() {}
    point(const T &x, const T &y) : x(x), y(y) {}

    point operator+(point b) {return {x+b.x, y+b.y}; }
    point operator-(point b) {return {x-b.x, y-b.y}; }
    point operator*(T b) {return {x*b, y*b}; }
    point operator/(T b) {return {x/b, y/b}; }
    bool operator==(point b) {return x==b.x && y==b.y; }
    // 逆時針極角排序
    bool operator<(point &b) {return (x*b.y > b.x*y); }
    friend ostream& operator<<(ostream& os, point p) {
        os << "(" << p.x << ", " << p.y << ")";
        return os;
    }
    // 判斷 ab 到 ac 的方向：{1:逆時鐘,0:重疊,-1:順時鐘}
    friend int ori(point a, point b, point c) {
        return sign((b-a)^(c-a));
    }
    friend bool btw(point a, point b, point c) {
        return ori(a, b, c) == 0 && sign((a-c)*(b-c)) <= 0;
    }
    // 判斷線段 ab, cd 是否相交
    friend bool banana(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 (btw(a, b, c) || btw(a, b, d) || btw(c, d, a) ||
            btw(c, d, b)) return 1;
        return (s1 * s2 < 0) && (s3 * s4 < 0);
    }

    T operator*(point b) {return x * b.x + y * b.y; }
    T operator^(point b) {return x * b.y - y * b.x; }
    T abs2() {return (*this) * (*this); }

    // 旋轉 Arg(b) 的角度 ( 小心溢位 )
    point rotate(point b) {return {x*b.x - y*b.y, x*b.y + y*b.x}; }
};

template<typename T>
struct line {
    point<T> p1, p2;
    // ax + by + c = 0
    T a, b, c; // |a|, |b| ≤ 2C, |c| ≤ 8C²
    line() {}
    line(const point<T> &x,const point<T> &y) : p1(x), p2(y){
        build();
    }
    void build() {
    a = p1.y - p2.y;
    b = p2.x - p1.x;
    c = (-a*p1.x)-b*p1.y;
    }
    // 判斷點和有向直線的關係：{1:左邊,0:在線上,-1:右邊}
  int ori(point<T> &p) {
    return sign((p2-p1) ^ (p-p1));
  }
  // 判斷直線斜率是否相同
  bool parallel(line &l) {
    return ((p1-p2) ^ (l.p1-l.p2)) == 0;
  }
  // 兩直線交點
    point<long double> line_intersection(line &l) {
        using P = point<long double>;
    point<T> a = p2-p1, b = l.p2-l.p1, s = l.p1-p1;
    return P(p1.x,p1.y) + P(a.x,a.y) * (((long double)(s^b))
         / (a^b));
  }
};

template<typename T>
struct polygon {
    vector<point<T>> v;
    polygon() {}
    polygon(const vector<point<T>> &u) : v(u) {}
    // simple 為 true 的時候會回傳任意三點不共線的凸包
    void make_convex_hull(int simple) {
        auto cmp = [&](point<T> &p, point<T> &q) {
            return (p.x == q.x) ? (p.y < q.y) : (p.x < q.x);
        };
        simple = (bool)simple;
        sort(v.begin(), v.end(), cmp);
        v.resize(unique(v.begin(), v.end()) - v.begin());
        vector<point<T>> hull;
        for (int t = 0; t < 2; ++t){
            int sz = hull.size();
            for (auto &i:v) {
                while (hull.size() >= sz+2 && ori(hull[hull.
                     size()-2], hull.back(), i) < simple) {
                    hull.pop_back();
                }
                hull.push_back(i);
            }
            hull.pop_back();
            reverse(v.begin(), v.end());
        }
        swap(hull, v);
    }
//  可以在有 n 個點的簡單多邊形內，用 O(n) 判斷一個點：
//  {1 : 在多邊形內, 0 : 在多邊形上, -1 : 在多邊形外}
    int in_polygon(point<T> a){
        const T MAX_POS = 1e9 + 5; // [記得修改] 座標的最大值
        point<T> pre = v.back(), b(MAX_POS, a.y + 1);
        int cnt = 0;

        for (auto &i:v) {
            if (btw(pre, i, a)) return 0;
            if (banana(a, b, pre, i)) cnt++;
            pre = i;
        }

        return cnt%2 ? 1 : -1;
```

```
    }
/// 警告：以下所有凸包專用的函式都只接受逆時針排序且任三點不共線的凸包 ///
//  可以在有 n 個點的凸包內，用 O(log n) 判斷一個點：
//  {1 : 在凸包內, 0 : 在凸包邊上, -1 : 在凸包外}
    int in_convex(point<T> p) {
        int n = v.size();
        int a = ori(v[0], v[1], p), b = ori(v[0], v[n-1], p);
        if (a < 0 || b > 0) return -1;
        if (btw(v[0], v[1], p)) return 0;
        if (btw(v[0], v[n - 1], p)) return 0;
        int l = 1, r = n - 1, mid;
        while (l + 1 < r) {
            mid = (l + r) >> 1;
            if (ori(v[0], v[mid], p) >= 0) l = mid;
            else r = mid;
        }
        int k = ori(v[l], v[r], p);
        if (k <= 0) return k;
        return 1;
    }
//  凸包專用的環狀二分搜，回傳 0-based index
    int cycle_search(auto &f) {
        int n = v.size(), l = 0, r = n;
        bool rv = f(1, 0);
        while (r - l > 1) {
            int m = (l + r) / 2;
            if (f(0, m) ? rv: f(m, (m + 1) % n)) r = m;
            else l = m;
        }
        return f(l, r % n) ? l : r % n;
    }
//  可以在有 n 個點的凸包內，用 O(log n) 判斷一條直線：
//  {1 : 穿過凸包, 0 : 剛好切過凸包, -1 : 沒碰到凸包}
    int line_cut_convex(line<T> L) {
        point<T> p(L.a, L.b); //  記得 L 要 build
        auto gt = [&](int neg) {
            auto f = [&](int x, int y) {
                return sign((v[x] - v[y]) * p) == neg;
            };
            return -(v[cycle_search(f)] * p);
        };
        T x = gt(1), y = gt(-1);
        if (L.c < x || y < L.c) return -1;
        return not (L.c == x || L.c == y);
    }
//  可以在有 n 個點的凸包內，用 O(log n) 判斷一個線段：
//  {1 : 存在一個凸包上的邊可以把這個線段切成兩半,
//   0 : 有碰到凸包但沒有任何凸包上的邊可以把它切成兩半,
//  -1 : 沒碰到凸包}
/// 除非線段兩端點都不在凸包邊上，否則此函數回傳 0 的時候不一定表示線段沒有通過凸包內部 ///
    int segment_across_convex(line<T> L) {
        point<T> p(L.a, L.b); //  記得 L 要 build
        auto gt = [&](int neg) {
            auto f = [&](int x, int y) {
                return sign((v[x] - v[y]) * p) == neg;
            };
            return cycle_search(f);
        };
        int i = gt(1), j = gt(-1), n = v.size();
        T x = -(v[i] * p), y = -(v[j] * p);
        if (L.c < x || y < L.c) return -1;
        if (L.c == x || L.c == y) return 0;

        if (i > j) swap(i, j);
        auto g = [&](int x, int lim) {
            int now = 0, nxt;
            for (int i = 1 << __lg(lim); i > 0; i /= 2) {
                if (now + i > lim) continue;
                nxt = (x + i) % n;
                if (L.ori(v[x]) * L.ori(v[nxt]) >= 0) {
                    x = nxt;
                    now += i;
                }
            } //   ↓ BE CAREFUL
            return -(ori(v[x], v[(x + 1) % n], L.p1) * ori(v[x], v[(x + 1) % n], L.p2));
        };
        return max(g(i, j - i), g(j, n - (j - i)));
    }
//  可以在有 n 個點的凸包內，用 O(log n) 判斷一個線段：
//  {1 : 線段上存在某一點位於凸包內部（邊上不算）,
//   0 : 線段上存在某一點碰到凸包的邊但線段上任一點均不在凸包內部,
//  -1 : 線段完全在凸包外面}
    int segment_pass_convex_interior(line<T> L) {
        if (in_convex(L.p1) == 1 || in_convex(L.p2) == 1)
            return 1;
        point<T> p(L.a, L.b); //  記得 L 要 build
        auto gt = [&](int neg) {
            auto f = [&](int x, int y) {
                return sign((v[x] - v[y]) * p) == neg;
            };
            return cycle_search(f);
        };
        int i = gt(1), j = gt(-1), n = v.size();
        T x = -(v[i] * p), y = -(v[j] * p);
        if (L.c < x || y < L.c) return -1;
        if (L.c == x || L.c == y) return 0;

        if (i > j) swap(i, j);
        auto g = [&](int x, int lim) {
            int now = 0, nxt;
            for (int i = 1 << __lg(lim); i > 0; i /= 2) {
                if (now + i > lim) continue;
                nxt = (x + i) % n;
                if (L.ori(v[x]) * L.ori(v[nxt]) > 0) {
                    x = nxt;
                    now += i;
                }
            } //   ↓ BE CAREFUL
            return -(ori(v[x], v[(x + 1) % n], L.p1) * ori(v[x], v[(x + 1) % n], L.p2));
        };
        int ret = max(g(i, j - i), g(j, n - (j - i)));
        return (ret == 0) ? (in_convex(L.p1) == 0 &&
            in_convex(L.p2) == 0) : ret;
    }
//  回傳點過凸包的兩條切線的切點的 0-based index（不保證兩條切線的順逆時針關係）
    pair<int,int> convex_tangent_point(point<T> p) {
        int n = v.size(), z = -1, edg = -1;
        auto gt = [&](int neg) {
            auto check = [&](int x) {
                if (v[x] == p) z = x;
                if (btw(v[x], v[(x + 1) % n], p)) edg = x;
                if (btw(v[(x + n - 1) % n], v[x], p)) edg = (x + n - 1) % n;
            };
            auto f = [&](int x, int y) {
                check(x); check(y);
                return ori(p, v[x], v[y]) == neg;
            };
            return cycle_search(f);
        };
        int x = gt(1), y = gt(-1);
        if (z != -1) {
            return {(z + n - 1) % n, (z + 1) % n};
        }
        else if (edg != -1) {
            return {edg, (edg + 1) % n};
        }
        else {
            return {x, y};
        }
    }
    friend int halfplane_intersection(vector<line<T>> &s,
        polygon<T> &P) {
        #define neg(p) ((p.y == 0 ? p.x : p.y) < 0)
        auto angle_cmp = [&](line<T> &A, line<T> &B) {
            point<T> a = A.p2-A.p1, b = B.p2-B.p1;
            return neg(a) < neg(b) || (neg(a) == neg(b) && (a^b) > 0);
        };
        #undef neg
        sort(s.begin(), s.end(), angle_cmp); // 線段左側為該線段半平面
        int L, R, n = s.size();
        vector<point<T>> px(n);
        vector<line<T>> q(n);
        q[L = R = 0] = s[0];
        for(int i = 1; i < n; ++i) {
            while(L < R && s[i].ori(px[R-1]) <= 0) --R;
            while(L < R && s[i].ori(px[L])   <= 0) ++L;
            q[++R] = s[i];
            if(q[R].parallel(q[R-1])) {
                --R;
                if(q[R].ori(s[i].p1) > 0) q[R] = s[i];
            }
            if(L < R) px[R-1] = q[R-1].line_intersection(q[R]);
        }
        while(L < R && q[L].ori(px[R-1]) <= 0) --R;
        P.v.clear();
        if(R - L <= 1) return 0;
        px[R] = q[R].line_intersection(q[L]);
        for(int i = L; i <= R; ++i) P.v.push_back(px[i]);
        return R - L + 1;
    }
};
```

5.2 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

6 Graph

6.1 2-SAT

```cpp
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 都是 0-based
        // 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]);
    }
};
```

6.2 Augment Path

```cpp
struct AugmentPath{
    int n, m;
    vector<vector<int>> G;
    vector<int> mx, my;
    vector<int> visx, visy;
    int stamp;

    AugmentPath(int _n, int _m) : n(_n), m(_m), G(n), mx(n, -1), my(m, -1), visx(n), visy(n){
        stamp = 0;
    }

    void add(int x, int y){
        G[x].push_back(y);
    }

    // bb03e2
    bool dfs1(int now){
        visx[now] = stamp;

        for (auto x : G[now]){
            if (my[x]==-1){
                mx[now] = x;
                my[x] = now;
                return true;
            }
        }
        for (auto x : G[now]){
            if (visx[my[x]]!=stamp && dfs1(my[x])){
                mx[now] = x;
                my[x] = now;
                return true;
            }
        }
        return false;
    }

    vector<pair<int, int>> find_max_matching(){
        vector<pair<int, int>> ret;

        while (true){
            stamp++;
            int tmp = 0;
            for (int i=0 ; i<n ; i++){
                if (mx[i]==-1 && dfs1(i)) tmp++;
            }
            if (tmp==0) break;
        }

        for (int i=0 ; i<n ; i++){
            if (mx[i]!=-1){
                ret.push_back({i, mx[i]});
            }
        }
        return ret;
    }

    // 645577
    void dfs2(int now){
        visx[now] = true;

        for (auto x : G[now]){
            if (my[x]!=-1 && visy[x]==false){
                visy[x] = true;
                dfs2(my[x]);
            }
        }
    }

    // 要先執行 find_max_matching 一次
    vector<pair<int, int>> find_min_vertex_cover(){
        fill(visx.begin(), visx.end(), false);
        fill(visy.begin(), visy.end(), false);

        vector<pair<int, int>> ret;
        for (int i=0 ; i<n ; i++){
            if (mx[i]==-1) dfs2(i);
        }

        for (int i=0 ; i<n ; i++){
            if (visx[i]==false) ret.push_back({1, i});
        }
        for (int i=0 ; i<m ; i++){
            if (visy[i]==true) ret.push_back({2, i});
        }

        return ret;
    }
};
```

6.3 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();
    }
}
```

6.4 Cut BCC

```
#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]);
        }
    }
}
```

6.5 Dinic

```
// 一般圖：O(EV²)
// 二分圖：O(EV)
struct Flow{
    using T = int; // 可以換成別的型別
    struct Edge{
        int v; T rc; int rid;
    };
    vector<vector<Edge>> G;
    void add(int u, int v, T c){
        G[u].push_back({v, c, G[v].size()});
        G[v].push_back({u, 0, G[u].size()-1});
    }
    vector<int> dis, it;

    Flow(int n){
        G.resize(n);
        dis.resize(n);
        it.resize(n);
    }

    // ce56d6
    T dfs(int u, int t, T f){
        if (u == t || f == 0) return f;
        for (int &i=it[u] ; i<G[u].size() ; i++){
            auto &[v, rc, rid] = G[u][i];
            if (dis[v]!=dis[u]+1) continue;
            T df = dfs(v, t, min(f, rc));
            if (df <= 0) continue;
            rc -= df;
            G[v][rid].rc += df;
            return df;
        }
        return 0;
    }

    // e22e39
    T flow(int s, int t){
        T ans = 0;
        while (true){
            fill(dis.begin(), dis.end(), INF);
            queue<int> q;
            q.push(s);
            dis[s] = 0;

            while (q.size()){
                int u = q.front(); q.pop();
                for (auto [v, rc, rid] : G[u]){
                    if (rc <= 0 || dis[v] < INF) continue;
                    dis[v] = dis[u] + 1;
                    q.push(v);
                }
            }
            if (dis[t]==INF) break;

            fill(it.begin(), it.end(), 0);
            while (true){
                T df = dfs(s, t, INF);
                if (df <= 0) break;
                ans += df;
            }
        }
        return ans;
    }

    //  the code below constructs minimum cut
    void dfs_mincut(int now, vector<bool> &vis){
        vis[now] = true;
        for (auto &[v, rc, rid] : G[now]){
            if (vis[v] == false && rc > 0){
                dfs_mincut(v, vis);
            }
        }
    }

    vector<pair<int, int>> construct(int n, int s, vector<
        pair<int,int>> &E){
        // E is G without capacity
        vector<bool> vis(n);
        dfs_mincut(s, vis);
        vector<pair<int, int>> ret;
        for (auto &[u, v] : E){
            if (vis[u] == true && vis[v] == false){
                ret.emplace_back(u, v);
            }
        }
        return ret;
    }
};
```

6.6 Dominator Tree

```
/*
全部都是 0-based
G 要是有向無權圖
一開始要初始化 G(N, root)，代表有 N 個節點，根是 root
用完之後要 build
G[i] = i 的 idom，也就是從 root 走到 i 時，一定要走到的點且離
    i 最近
*/
struct DominatorTree{
    int N;
    vector<vector<int>> G;
    vector<vector<int>> buckets, rg;
    // dfn[x] = the DFS otder of x
    // rev[x] = the vertex with DFS order x
    // par[x] = the parent of x
    vector<int> dfn, rev, par;
    vector<int> sdom, dom, idom;
    vector<int> fa, val;
    int stamp;
    int root;

    int operator [] (int x){
        return idom[x];
    }

    DominatorTree(int _N, int _root) :
        N(_N),
        G(N), buckets(N), rg(N),
        dfn(N, -1), rev(N, -1), par(N, -1),
        sdom(N, -1), dom(N, -1), idom(N, -1),
        fa(N, -1), val(N, -1)
    {
        stamp = 0;
        root = _root;
    }

```

```
    void add_edge(int u, int v){
        G[u].push_back(v);
    }

    void dfs(int x){
        rev[dfn[x] = stamp] = x;
        fa[stamp] = sdom[stamp] = val[stamp] = stamp;
        stamp++;

        for (int u : G[x]){
            if (dfn[u]==-1){
                dfs(u);
                par[dfn[u]] = dfn[x];
            }
            rg[dfn[u]].push_back(dfn[x]);
        }
    }

    int eval(int x, bool first){
        if (fa[x]==x) return !first ? -1 : x;
        int p = eval(fa[x], false);

        if (p==-1) return x;
        if (sdom[val[x]]>sdom[val[fa[x]]]) val[x] = val[fa[x
            ]];
        fa[x] = p;

        return !first ? p : val[x];
    }

    void link(int x, int y){
        fa[x] = y;
    }

    void build(){
        dfs(root);

        for (int x=stamp-1 ; x>=0 ; x--){
            for (int y : rg[x]){
                sdom[x] = min(sdom[x], sdom[eval(y, true)]);
            }
            if (x>0) buckets[sdom[x]].push_back(x);
            for (int u : buckets[x]){
                int p = eval(u, true);
                if (sdom[p]==x) dom[u] = x;
                else dom[u] = p;
            }
            if (x>0) link(x, par[x]);
        }

        idom[root] = root;
        for (int x=1 ; x<stamp ; x++){
            if (sdom[x]!=dom[x]) dom[x] = dom[dom[x]];
        }
        for (int i=1 ; i<stamp ; i++) idom[rev[i]] = rev[dom[
            i]];
    }
};
```

6.7 Enumerate Triangle

```
// O(m log m) 枚舉無向圖所有三角形，0-based
void Enumerate_Triangle(vector<pair<int, int>> &edge, vector<
    int> &deg){
    int n = deg.size();
    int m = edge.size();
    vector<vector<int>> G(n);

    for (int i=0 ; i<m ; i++){
        if (deg[edge[i].first] > deg[edge[i].second]) swap(
            edge[i].first, edge[i].second);
        if (deg[edge[i].first] == deg[edge[i].second] && edge
            [i].first > edge[i].second) swap(edge[i].first,
            edge[i].second);
        G[edge[i].first].push_back(edge[i].second);
    }

    vector<int> vis(n, false);
    for (int i=0 ; i<n ; i++){
        for (auto j : G[i]) vis[j] = true;
        for (auto j : G[i]){
            for (auto k : G[j]){
                if (vis[k]){
                    // i, j, k is a triangle
                }
            }
        }
        for (auto j : G[i]) vis[j] = false;
    }
}
```

6.8 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;
}
```

6.9 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;
    }
};
```

6.10 Kosaraju

```
/*
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，並把縮點後
    的結果存在 result 裡

SCC[i] = 某個 SCC 中的所有點
SCC_id[i] = 第 i 個點在第幾個 SCC
*/
```

```cpp
struct SCC_compress{
    int N, M, sz;
    vector<vector<int>> G, inv_G, result;
    vector<pair<int, int>> edges;
    vector<bool> vis;
    vector<int> order;

    vector<vector<int>> SCC;
    vector<int> SCC_id;

    SCC_compress(int _N) :
    N(_N), M(0), sz(0),
    G(N), inv_G(N),
    vis(N), SCC_id(N)
    {}

    vector<int> operator [] (int x){
        return result[x];
    }

    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]) dfs1(G, x);
        order.push_back(now);
    }

    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]) dfs2(G, x);
    }

    void compress(){
        fill(vis.begin(), vis.end(), 0);
        for (int i=0 ; i<N ; i++) if (!vis[i]) dfs1(G, i);

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

        result.resize(SCC.size());
        sz = SCC.size();
        for (auto [u, v] : edges){
            if (SCC_id[u]!=SCC_id[v]) result[SCC_id[u]].
                push_back(SCC_id[v]);
        }
        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());
        }
    }
};
```

6.11 Kuhn Munkres

```cpp
// O(n^2) 找到最大權匹配
struct KuhnMunkres{
    int n; // max(n, m)
    vector<vector<int>> G;
    vector<int> match, lx, ly, visx, visy;
    vector<int> slack;
    int stamp = 0;

    KuhnMunkres(int n) : n(n), G(n, vector<int>(n)), lx(n),
        ly(n), slack(n), match(n), visx(n), visy(n) {}

    void add(int x, int y, int w){
        G[x][y] = max(G[x][y], w);
    }

    bool dfs(int i, bool aug){ // aug = true 表示要更新 match
        if (visx[i]==stamp) return false;
        visx[i] = stamp;

        for (int j=0 ; j<n ; j++){
            if (visy[j]==stamp) continue;
            int d = lx[i]+ly[j]-G[i][j];

            if (d==0){
                visy[j] = stamp;
                if (match[j]==-1 || dfs(match[j], aug)){
                    if (aug){
                        match[j] = i;
                    }
                    return true;
                }
            }else{
                slack[j] = min(slack[j], d);
            }
        }
        return false;
    }

    bool augment(){
        for (int j=0 ; j<n ; j++){
            if (visy[j]!=stamp && slack[j]==0){
                visy[j] = stamp;
                if (match[j]==-1 || dfs(match[j], false)){
                    return true;
                }
            }
        }
        return false;
    }

    void relabel(){
        int delta = INF;
        for (int j=0 ; j<n ; j++){
            if (visy[j]!=stamp) delta = min(delta, slack[j]);
        }
        for (int i=0 ; i<n ; i++){
            if (visx[i]==stamp) lx[i] -= delta;
        }
        for (int j=0 ; j<n ; j++){
            if (visy[j]==stamp) ly[j] += delta;
            else slack[j] -= delta;
        }
    }

    int solve(){

        for (int i=0 ; i<n ; i++){
            lx[i] = 0;
            for (int j=0 ; j<n ; j++){
                lx[i] = max(lx[i], G[i][j]);
            }
        }

        fill(ly.begin(), ly.end(), 0);
        fill(match.begin(), match.end(), -1);

        for(int i = 0; i < n; i++) {
            fill(slack.begin(), slack.end(), INF);
            stamp++;
            if(dfs(i, true)) continue;

            while(augment()==false) relabel();
            stamp++;
            dfs(i, true);
        }

        int ans = 0;
        for (int j=0 ; j<n ; j++){
            if (match[j]!=-1){
                ans += G[match[j]][j];
            }
        }
        return ans;
    }
};
```

6.12 LCA

```cpp
struct Tree{
    int N, M = 0, H;
    vector<vector<int>> G;
    vector<vector<int>> LCA;
    vector<int> parent;
    vector<int> dep;

    Tree(int _N) : N(_N), H(__lg(_N)+1){
        G.resize(N);
        parent.resize(N, -1);
        dep.resize(N, 0);
        LCA.resize(H, vector<int>(N, 0));
    }

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

    void dfs(int now, int pre){ // root 的 pre 是自己
        dep[now] = dep[pre]+1;
        parent[now] = pre;
        for (auto x : G[now]){
            if (x==pre) continue;
            dfs(x, now);
        }
```

```
    }

    void build_LCA(int root = 0){
        dfs(root, root);
        for (int i=0 ; i<N ; i++) LCA[0][i] = parent[i];
        for (int i=1 ; i<H ; i++){
            for (int j=0 ; j<N ; j++){
                LCA[i][j] = LCA[i-1][LCA[i-1][j]];
            }
        }
    }

    int jump(int u, int step){
        for (int i=0 ; i<H ; i++){
            if (step&(1<<i)) u = LCA[i][u];
        }
        return u;
    }

    int get_LCA(int u, int v){
        if (dep[u]<dep[v]) swap(u, v);
        u = jump(u, dep[u]-dep[v]);
        if (u==v) return u;
        for (int i=H-1 ; i>=0 ; i--){
            if (LCA[i][u]!=LCA[i][v]){
                u = LCA[i][u];
                v = LCA[i][v];
            }
        }
        return parent[u];
    }
};
```

6.13 MCMF

```
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, capcity: c, cost: k
  void add(int v, int u, int c, int k){
    G[v].push_back({u, c, k, SZ(G[u])});
    G[u].push_back({v, 0, -k, SZ(G[v])-1});
  }

  // 3701d6
  int spfa(int s, int t){
    fill(ALL(par), -1);
    vector<int> dis(SZ(par), INF);
    vector<bool> in_q(SZ(par), 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<SZ(G[v]) ; 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];
  }

  // return <max flow, min cost>, 150093
  pair<int, int> flow(int s, int t){
    int fl = 0, cost = 0, d;
    while ((d = spfa(s, t))<INF){
      int cur = INF;
      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};
  }

  vector<pair<int, int>> construct(){
    vector<pair<int, int>> ret;
    for (int i=0 ; i<n ; i++){
      for (auto x : G[i]){
        if (x.rc==0){
          ret.push_back({i+1, x.u-n+1});
          break;
        }
      }
    }
    return ret;
  }
};
```

6.14 Tarjan

```
struct tarjan_SCC {
    int now_T, now_SCCs;
    vector<int> dfn, low, SCC;
    stack<int> S;
    vector<vector<int>> E;
    vector<bool> vis, in_stack;

    tarjan_SCC(int n) {
        init(n);
    }
    void init(int n) {
        now_T = now_SCCs = 0;
        dfn = low = SCC = vector<int>(n);
        E = vector<vector<int>>(n);
        S = stack<int>();
        vis = in_stack = vector<bool>(n);
    }
    void add(int u, int v) {
        E[u].push_back(v);
    }
    void build() {
        for (int i = 0; i < dfn.size(); ++i) {
            if (!dfn[i]) dfs(i);
        }
    }
    void dfs(int v) {
        now_T++;
        vis[v] = in_stack[v] = true;
        dfn[v] = low[v] = now_T;
        S.push(v);
        for (auto &i:E[v]) {
            if (!vis[i]) {
                vis[i] = true;
                dfs(i);
                low[v] = min(low[v], low[i]);
            }
            else if (in_stack[i]) {
                low[v] = min(low[v], dfn[i]);
            }
        }
        if (low[v] == dfn[v]) {
            int tmp;
            do {
                tmp = S.top();
                S.pop();
                SCC[tmp] = now_SCCs;
                in_stack[tmp] = false;
            } while (tmp != v);
            now_SCCs += 1;
        }
    }
};
```

6.15 Tarjan 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);
    }
}
```

6.16 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]);
        }
    }

    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;

}

signed main(void){
    fastio;

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

6.17 圓方樹

```
#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
#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";
    }
}
```

6.18 最大權閉合圖

```
/*
Problem:
    Given w = [w_0, w_1, ..., w_{n-1}] (which can be
    either positive or negative or 0), you can choose
    to take w_i (0 < i < n) or not, but if edge u -> v
    exists, you must take w_v if you want to take w_u
    (in other words, you can't take w_u without taking
     w_v), this function returns the maximum value(> 0)
     you can get. If you need a construction, you can
     output the minimum cut of the S(source) side.
Complexity:
    MaxFlow(n, m) (Non-Biparte:O(n²m) / Bipartite:O(m√n))
*/
int maximum_closure(vector<int> w, vector<pair<int,int>> EV)
    {
    int n = w.size(), S = n + 1, T = n + 2;
    Flow G(T + 5); // Graph/Dinic.cpp
    int sum = 0;
    for (int i = 0; i < n; ++i) {
        if (w[i] > 0) {
            G.add(S, i, w[i]);
            sum += w[i];
        }
        else if (w[i] < 0) {
            G.add(i, T, abs(w[i]));
        }
    }
    for (auto &[u, v] : EV) { // You should make sure that
         INF > Σ|w_i|
        G.add(u, v, INF);
    }
    int cut = G.flow(S, T);
    return sum - cut;
}
```

6.19 Theorem

- 任意圖
  - 不能有孤點，最大匹配 + 最小邊覆蓋 = n - 點覆蓋的補集是獨立集。最小點覆蓋 + 最大獨立集 = n
- 二分圖
  - 最小點覆蓋 = 最大匹配 = n - 最大獨立集
- 只有邊帶權的二分圖
  - 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
  - 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.

7 Math

7.1 CRT m Coprime

```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];
    int tmp;
        extgcd(M[i], m[i], inv_M[i], tmp);
        ans+=a[i]*inv_M[i]*M[i];
        ans%=p;
    }

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

7.2 CRT m Not Coprime

```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}};
}
```

7.3 Josephus Problem

```cpp
// 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;
    }
}
```

7.4 Lagrange any x

```cpp
// init: (x1, y1), (x2, y2) in a vector
struct Lagrange{
    int n;
    vector<pair<int, int>> v;

    Lagrange(vector<pair<int, int>> &_v){
        n = _v.size();
        v = _v;
    }

    // O(n^2 log MAX_A)
    int solve(int x){
        int ret = 0;
        for (int i=0 ; i<n ; i++){
            int now = v[i].second;
            for (int j=0 ; j<n ; j++){
                if (i==j) continue;
                now *= ((x-v[j].first)+MOD)%MOD;
                now %= MOD;
                now *= (qp((v[i].first-v[j].first+MOD)%MOD,
                     MOD-2)+MOD)%MOD;
                now %= MOD;
            }

            ret = (ret+now)%MOD;
        }
        return ret;
    }
};
```

7.5 Lagrange continuous x

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

const int MAX_N = 5e5 + 10;
const int mod = 1e9 + 7;

long long inv_fac[MAX_N];

inline int fp(long long x, int y) {
    int ret = 1;
    for (; y; y >>= 1) {
        ret = (y & 1) ? (ret * x % mod) : ret;
        x = x * x % mod;
    }
    return ret;
}

// TO USE THIS TEMPLATE, YOU MUST MAKE SURE THAT THE MOD
     NUMBER IS A PRIME.
struct Lagrange {
/*
    Initialize a polynomial with f(x_0), f(x_0 + 1), ..., f(
        x_0 + n).
    This determines a polynomial f(x) whose degree is at most
         n.
    Then you can call sample(x) and you get the value of f(x)
        .
    Complexity of init() and sample() are both O(n).
*/
    int m, shift; // m = n + 1
    vector<int> v, mul;
// You can use this function if you don't have inv_fac array
     already.
    void construct_inv_fac() {
        long long fac = 1;
        for (int i = 2; i < MAX_N; ++i) {
            fac = fac * i % mod;
        }
        inv_fac[MAX_N - 1] = fp(fac, mod - 2);
        for (int i = MAX_N - 1; i >= 1; --i) {
            inv_fac[i - 1] = inv_fac[i] * i % mod;
        }
    }
// You call init() many times without having a second
     instance of this struct.
    void init(int X_0, vector<int> &u) {
        v = u;
        shift = ((1 - X_0) % mod + mod) % mod;
        if (v.size() == 1) v.push_back(v[0]);
        m = v.size();
        mul.resize(m);
    }
// You can use sample(x) instead of sample(x % mod).
    int sample(int x) {
        x = ((long long)x + shift) % mod;
        x = (x < 0) ? (x + mod) : x;
        long long now = 1;
        for (int i = m; i >= 1; --i) {
            mul[i - 1] = now;
            now = now * (x - i) % mod;
        }
        int ret = 0;
        bool neg = (m - 1) & 1;
        now = 1;
        for (int i = 1; i <= m; ++i) {
            int up = now * mul[i - 1] % mod;
            int down = inv_fac[m - i] * inv_fac[i - 1] % mod;
```

```
            int tmp = ((long long)v[i - 1] * up % mod) * down % mod;
            ret += (neg && tmp) ? (mod - tmp) : (tmp);
            ret = (ret >= mod) ? (ret - mod) : ret;
            now = now * (x - i) % mod;
            neg ^= 1;
        }
        return ret;
    }
};

int main() {
    int n; cin >> n;
    vector<int> v(n);
    for (int i = 0; i < n; ++i) {
        cin >> v[i];
    }
    Lagrange L;
    L.construct_inv_fac();
    L.init(0, v);
    int x; cin >> x;
    cout << L.sample(x);
}
```

7.6 Lucas's Theorem

```
// 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);
}
```

7.7 Matrix

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

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

    vector<int> & operator [] (int i){
        return arr[i];
    }

    Matrix operator * (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]%MOD;
                    ret.arr[i][j] %= MOD;
                }
            }
        }
        return ret;
    }

    Matrix pow(int p){
        Matrix ret(n, n), mul = *this;
        for (int i=0 ; i<n ; i++){
            ret.arr[i][i] = 1;
        }

        for ( ; p ; p>>=1){
            if (p&1) ret = ret*mul;
            mul = mul*mul;
        }

        return ret;
    }


    int det(){
        vector<vector<int>> arr = this->arr;
        bool flag = false;
        for (int i=0 ; i<n ; i++){
            int target = -1;
            for (int j=i ; j<n ; j++){
                if (arr[j][i]){
                    target = j;
                    break;
                }
            }
            if (target==-1) return 0;
            if (i!=target){
                swap(arr[i], arr[target]);
                flag = !flag;
            }

            for (int j=i+1 ; j<n ; j++){
                if (!arr[j][i]) continue;
                int freq = arr[j][i]*qp(arr[i][i], MOD-2)%MOD;
                for (int k=i ; k<n ; k++){
                    arr[j][k] -= freq*arr[i][k];
                    arr[j][k] = (arr[j][k]%MOD+MOD)%MOD;
                }
            }
        }

        int ret = !flag ? 1 : MOD-1;
        for (int i=0 ; i<n ; i++){
            ret *= arr[i][i];
            ret %= MOD;
        }
        return ret;
    }
};
```

7.8 Matrix 01

```
const int MAX_N = (1LL<<12);
struct Matrix{
    int n, m;
    vector<bitset<MAX_N>> arr;

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

    Matrix operator * (Matrix b){
        Matrix b_t(b.m, b.n);
        for (int i=0 ; i<b.n ; i++){
            for (int j=0 ; j<b.m ; j++){
                b_t.arr[j][i] = b.arr[i][j];
            }
        }

        Matrix ret(n, b.m);
        for (int i=0 ; i<n ; i++){
            for (int j=0 ; j<b.m ; j++){
                ret.arr[i][j] = ((arr[i]&b_t.arr[j]).count()&1);
            }
        }
        return ret;
    }
};
```

7.9 Miller Rabin

```
// O(log n)
typedef Uint unsigned long long
Uint modmul(Uint a, Uint b, Uint m) {
    int ret = a*b - m*(Uint)((long double)a*b/m);
    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;
}

// ed23aa
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;
}
```

7.10 Pollard Rho

```
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)
// c1670c
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;
    }
}
```

7.11 Polynomial

```
struct Poly {
    int len, deg;
    int *a;
    // len = 2^k >= the original length
    Poly(): len(0), deg(0), a(nullptr) {}
    Poly(int _n) {
        len = 1;
        deg = _n - 1;
        while (len < _n) len <<= 1;
        a = (ll*) calloc(len, sizeof(ll));
    }
    Poly(int l, int d, int *b) {
        len = l;
        deg = d;
        a = b;
    }
    void resize(int _n) {
        int len1 = 1;
        while (len1 < _n) len1 <<= 1;
        int *res = (ll*) calloc(len1, sizeof(ll));
        for (int i = 0; i < min(len, _n); i++) {
            res[i] = a[i];
        }
        len = len1;
        deg = _n - 1;
        free(a);
        a = res;
    }
    Poly& operator=(const Poly rhs) {
        this->len = rhs.len;
        this->deg = rhs.deg;
        this->a = (ll*)realloc(this->a, sizeof(ll) * len);
        copy(rhs.a, rhs.a + len, this->a);
        return *this;
    }
    Poly operator*(Poly rhs) {
        int l1 = this->len, l2 = rhs.len;
        int d1 = this->deg, d2 = rhs.deg;
        while (l1 > 0 and this->a[l1 - 1] == 0) l1--;
        while (l2 > 0 and rhs.a[l2 - 1] == 0) l2--;
        int l = 1;
        while (l < max(l1 + l2 - 1, d1 + d2 + 1)) l <<= 1;
        int *x, *y, *res;
        x = (ll*) calloc(l, sizeof(ll));
        y = (ll*) calloc(l, sizeof(ll));
        res = (ll*) calloc(l, sizeof(ll));
        copy(this->a, this->a + l1, x);
        copy(rhs.a, rhs.a + l2, y);
        ntt.tran(l, x); ntt.tran(l, y);
        FOR (i, 0, l - 1)
            res[i] = x[i] * y[i] % mod;
        ntt.tran(l, res, true);
        free(x); free(y);
        return Poly(l, d1 + d2, res);
    }
    Poly operator+(Poly rhs) {
        int l1 = this->len, l2 = rhs.len;
        int l = max(l1, l2);
        Poly res;
        res.len = l;
        res.deg = max(this->deg, rhs.deg);
        res.a = (ll*) calloc(l, sizeof(ll));
        FOR (i, 0, l1 - 1) {
            res.a[i] += this->a[i];
            if (res.a[i] >= mod) res.a[i] -= mod;
        }
        FOR (i, 0, l2 - 1) {
            res.a[i] += rhs.a[i];
            if (res.a[i] >= mod) res.a[i] -= mod;
        }
        return res;
    }
    Poly operator-(Poly rhs) {
        int l1 = this->len, l2 = rhs.len;
        int l = max(l1, l2);
        Poly res;
        res.len = l;
        res.deg = max(this->deg, rhs.deg);
        res.a = (ll*) calloc(l, sizeof(ll));
        FOR (i, 0, l1 - 1) {
            res.a[i] += this->a[i];
            if (res.a[i] >= mod) res.a[i] -= mod;
        }
        FOR (i, 0, l2 - 1) {
            res.a[i] -= rhs.a[i];
            if (res.a[i] < 0) res.a[i] += mod;
        }
        return res;
    }
    Poly operator*(const int rhs) {
        Poly res;
        res = *this;
        FOR (i, 0, res.len - 1) {
            res.a[i] = res.a[i] * rhs % mod;
            if (res.a[i] < 0) res.a[i] += mod;
        }
        return res;
    }
    Poly(vector<int> f) {
        int _n = f.size();
        len = 1;
        deg = _n - 1;
        while (len < _n) len <<= 1;
        a = (ll*) calloc(len, sizeof(ll));
        FOR (i, 0, deg) a[i] = f[i];
    }
    Poly derivative() {
        Poly g(this->deg);
        FOR (i, 1, this->deg) {
            g.a[i - 1] = this->a[i] * i % mod;
        }
        return g;
    }
    Poly integral() {
        Poly g(this->deg + 2);
        FOR (i, 0, this->deg) {
            g.a[i + 1] = this->a[i] * ::inv(i + 1) % mod;
        }
        return g;
    }
    Poly inv(int len1 = -1) {
        if (len1 == -1) len1 = this->len;
        Poly g(1); g.a[0] = ::inv(a[0]);
        for (int l = 1; l < len1; l <<= 1) {
            Poly t; t = *this;
            t.resize(l << 1);
            t = g * g * t;
            t.resize(l << 1);
            Poly g1 = g * 2 - t;
            swap(g, g1);
        }
        return g;
    }
    Poly ln(int len1 = -1) {
        if (len1 == -1) len1 = this->len;
        auto g = *this;
        auto x = g.derivative() * g.inv(len1);
        x.resize(len1);
        x = x.integral();
        x.resize(len1);
        return x;
    }
    Poly exp() {
        Poly g(1);
        g.a[0] = 1;
```

```
        for (int l = 1; l < len; l <<= 1) {
            Poly t, g1; t = *this;
            t.resize(l << 1); t.a[0]++;
            g1 = (t - g.ln(l << 1)) * g;
            g1.resize(l << 1);
            swap(g, g1);
        }
        return g;
    }
    Poly pow(ll n) {
        Poly &a = *this;
        int i = 0;
        while (i <= a.deg and a.a[i] == 0) i++;
        if (i and (n > a.deg or n * i > a.deg)) return Poly(a.deg + 1);
        if (i == a.deg + 1) {
            Poly res(a.deg + 1);
            res.a[0] = 1;
            return res;
        }
        Poly b(a.deg - i + 1);
        int inv1 = ::inv(a.a[i]);
        FOR (j, 0, b.deg)
            b.a[j] = a.a[j + i] * inv1 % mod;
        Poly res1 = (b.ln() * (n % mod)).exp() * (::power(a.a[i], n));
        Poly res2(a.deg + 1);
        FOR (j, 0, min((ll)(res1.deg), (ll)(a.deg - n * i)))
            res2.a[j + n * i] = res1.a[j];
        return res2;
    }
};
```

7.12 數論分塊

```
/*
時間複雜度為 O(sqrt(n))
區間為 [l, r]
*/
for(int i=1 ; i<=n ; i++){
    int l = i, r = n/(n/i);
    i = r;
    ans.push_back(r);
}
```

7.13 最大質因數

```
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);
}
```

7.14 歐拉公式

```
// 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;
}
```

7.15 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

7.16 Catalan Number

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

7.17 Matrix Tree Theorem

目標：給定一張無向圖，問他的生成樹數量。
方法：先把所有自環刪掉，定義 Q 為以下矩陣

$$Q_{i,j} = \begin{cases} \deg(v_i) & \text{if } i = j \\ -(\text{邊} v_i v_j \text{ 的數量}) & \text{otherwise} \end{cases}$$

接著刪掉 Q 的第一個 row 跟 column，它的 determinant 就是答案。
目標：給定一張有向圖，問他的以 r 為根，可以走到所有點生成樹數量。

方法：先把所有自環刪掉，定義 Q 為以下矩陣

$$Q_{i,j} = \begin{cases} \deg_{in}(v_i) & \text{if } i = j \\ -(\text{邊} v_i v_j \text{ 的數量}) & \text{otherwise} \end{cases}$$

接著刪掉 Q 的第 r 個 row 跟 column，它的 determinant 就是答案。

7.18 Stirling's formula

$n! \approx \sqrt{2\pi n}(\frac{n}{e})^n$

7.19 Theorem

1. $1 \sim x$ 質數的數量 $\approx \frac{x}{\ln x}$
2. 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
7. $n^{k-2} \cdot \prod_{i=1}^{k} s_i$ n 個點、k 的連通塊，加上 $k-1$ 條邊使得變成一個連通圖的方法數，其中每個連通塊有 s_i 個點

7.20 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。

7.21 歐拉定理

若 a, m 互質，則：

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

若 a, m 不互質，則：

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

7.22 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)

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

8 String

8.1 Hash

```cpp
mt19937 seed(chrono::steady_clock::now().time_since_epoch().
    count());
int rng(int l, int r){
    return uniform_int_distribution<int>(l, r)(seed);
}
int A = rng(1e5, 8e8);
const int B = 1e9+7;

// 2f6192
struct RollingHash{
    vector<int> Pow, Pre;
    RollingHash(string s = ""){
        Pow.resize(s.size());
        Pre.resize(s.size());

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

        return;
    }

    int get(int l, int r){ // 取得 [l, r] 的數值
        if (l==0) return Pre[r];
        int res = (Pre[r]-Pre[l-1]*Pow[r-l+1])%B;
        if (res<0) res += B;
        return res;
    }
};
```

8.2 KMP

```cpp
// 給一個字串 S，定義函數 \pi(i) = k 代表 S[1 ... k] = S[i-k
    +1 ... i]（最長真前後綴）
// e5b7ce
vector<int> KMP(string &s){
    int n = s.size();
    vector<int> ret(n);
    for (int i=1 ; i<n ; i++){
        int j = ret[i-1];
        while (j>0 && s[i]!=s[j]) j = ret[j-1];
        j += (s[i]==s[j]);
        ret[i] = j;
    }
    return ret;
}
```

8.3 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);
}
```

8.4 Min Rotation

```cpp
// 9d296f
int minRotation(string s) {
    int a=0, N=SZ(s); s += s;
    for (int b=0 ; b<N ; b++){
        for (int k=0 ; k<N ; k++){
            if (a+k == b || s[a+k] < s[b+k]) {b += max(0LL, k
                -1); break;}
            if (s[a+k] > s[b+k]) { a = b; break; }
        }
    }
    return a;
}
```

8.5 Suffix Array

```cpp
// 注意，當 |s|=1 時，lcp 不會有值，務必測試 |s|=1 的 case
struct SuffixArray {
    string s;
    vector<int> sa, lcp;

    // 69ced9
    SuffixArray(string _s, int lim = 256) {
        s = _s;
        int n = s.size()+1, k = 0, a, b;
        vector<int> x(s.begin(), s.end()), y(n), ws(max(n,
            lim)), rank(n);
        x.push_back(0);
        sa = lcp = y;
        iota(sa.begin(), sa.end(), 0);
        for (int j=0, p=0 ; p<n ; j=max(1LL, j*2), lim=p) {
            p = j;
            iota(y.begin(), y.end(), n-j);
            for (int i=0 ; i<n ; i++) if (sa[i] >= j) y[p++]
                = sa[i] - j;
            fill(ws.begin(), ws.end(), 0);
            for (int i=0 ; i<n ; i++) ws[x[i]]++;
            for (int i=1 ; i<lim ; i++) ws[i] += ws[i - 1];
            for (int i = n; i--;) sa[--ws[x[y[i]]]] = y[i];
            swap(x, y), p = 1, x[sa[0]] = 0;
            for (int i=1 ; i<n ; i++){
                a = sa[i - 1];
                b = sa[i];
                x[b] = (y[a] == y[b] && y[a + j] == y[b + j])
                    ? p - 1 : p++;
            }
        }

        for (int i=1 ; i<n ; i++) rank[sa[i]] = i;
        for (int i=0, j ; i<n-1 ; lcp[rank[i++]]=k)
            for (k && k--, j=sa[rank[i]-1] ; i+k<s.size() &&
                j+k<s.size() && s[i+k]==s[j+k] ; k++);
        sa.erase(sa.begin());
        lcp.erase(lcp.begin(), lcp.begin()+2);
    }

    // f49583
    vector<int> pos; // pos[i] = i 這個值在 pos 的哪個地方
    SparseTable st;
    void init_lcp(){
        pos.resize(sa.size());
        for (int i=0 ; i<sa.size() ; i++){
            pos[sa[i]] = i;
        }
        if (lcp.size()){
            st.build(lcp);
        }
    }

    // 用之前記得 init
    // 回傳 [l1, r1] 跟 [l2, r2] 的 lcp，0-based
    int get_lcp(int l1, int r1, int l2, int r2){
        int pos_1 = pos[l1], len_1 = r1-l1+1;
        int pos_2 = pos[l2], len_2 = r2-l2+1;
        if (pos_1>pos_2){
            swap(pos_1, pos_2);
            swap(len_1, len_2);
        }

        if (l1==l2){
            return min(len_1, len_2);
        }else{
            return min({st.query(pos_1, pos_2), len_1, len_2
                });
        }
    }

    // 檢查 [l1, r1] 跟 [l2, r2] 的大小關係，0-based
    // 如果前者小於後者，就回傳 <0，相等就回傳 =0，否則回傳
        >0
    // 5b8db0
    int substring_cmp(int l1, int r1, int l2, int r2){
        int len_1 = r1-l1+1;
        int len_2 = r2-l2+1;
        int res = get_lcp(l1, r1, l2, r2);

        if (res<len_1 && res<len_2){
            return s[l1+res]-s[l2+res];
        }else if (len_1==res && len_2==res){
            // 如果不需要以 index 作為次要排序參數，這裡要回
                傳 0
            return l1-l2;
```

```
        }else{
            return len_1==res ? -1 : 1;
        }
    }

    // 對於位置在 <=p 的後綴，找離他左邊/右邊最接近位置 >p 的後綴的 lcp，0-based
    // pre[i] = s[i] 離他左邊最接近位置 >p 的後綴的 lcp，0-based
    // suf[i] = s[i] 離他右邊最接近位置 >p 的後綴的 lcp，0-based
    // da12fa
    pair<vector<int>, vector<int>> get_left_and_right_lcp(int p){
        vector<int> pre(p+1);
        vector<int> suf(p+1);

        { // build pre
            int now = 0;
            for (int i=0 ; i<s.size() ; i++){
                if (sa[i]<=p){
                    pre[sa[i]] = now;
                    if (i<lcp.size()) now = min(now, lcp[i]);
                }else{
                    if (i<lcp.size()) now = lcp[i];
                }
            }
        }
        { // build suf
            int now = 0;
            for (int i=s.size()-1 ; i>=0 ; i--){
                if (sa[i]<=p){
                    suf[sa[i]] = now;
                    if (i-1>=0) now = min(now, lcp[i-1]);
                }else{
                    if (i-1>=0) now = lcp[i-1];
                }
            }
        }

        return {pre, suf};
    }
};
```

8.6 Z Algorithm

```
// 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
// bcfbd6
vector<int> z_function(string s){
    vector<int> ret(s.size());
    int ll = 0, rr = 0;

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

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

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

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

8.7 k-th Substring1

```
// 回傳 s 所有子字串（完全不同）中，第 k 大的
string k_th_substring(string &s, int k){
    int n = s.size();
    SuffixArray sa(s);
    sa.init_lcp();

    int prePrefix = 0, nowRank = 0;
    for (int i=0 ; i<n ; i++){
        int len = n-sa[i];
        int add = len-prePrefix;

        if (nowRank+add>=k){
            return s.substr(sa[i], prePrefix+k-nowRank);
        }

        prePrefix = sa.lcp[i];
        nowRank += add;
    }
}
```