Contents

1　Misc

1.1　Note

開始寫題目之前，請做下面的事：

- 在「開始寫任何題目之前」，應該要先自己看過「所有」範例測資的邏輯和演算法是否有跟範例輸出對上

- 如果你覺得別人的某段程式碼有錯誤，就應該直接講出來

- +2～+3 後就開始生測資跟對拍（根據寫 generator 跟 checker 的時間決定）

寫程式請遵照以下原則：

- 準確使用註解分段程式碼

  - declare
  - init
  - input
  - process / queries
  - output

- 陣列若可以開到最大，則使用常數宣告大小

上傳之前，請依序檢查以下資訊：

1. 是否開啟 IO 優化
2. 是否有 t 筆輸入但忘了輸入
3. 是否有初始化容器
4. 題目範圍有沒有開到最大
5. 跑過所有範例測資，並嚴格確認是否正確

1.2　Default Code [481325]

```cpp
#include <bits/stdc++.h>
using namespace std;
#define int long long
#define debug(HEHE...) std::cerr << #HEHE << " = ", dout(HEHE)
void dout() { std::cerr << "\n"; }
template <typename T, typename... U>
void dout(T t, U... u) { std::cerr << t << ' ', dout(u...); }

void solve(){

}

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

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

    return 0;
}
```

1.3　Run

```python
from os import *

f = "pA"

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

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

1.4　Custom Set PQ Sort [d4df55]

```cpp
// 所有自訂的結構體，務必檢查相等的 case，給所有元素一個排序的依據
struct my_struct{
    int val;
    my_struct(int _val) : val(_val) {}
};

auto cmp = [](my_struct a, my_struct b) {
    return a.val > b.val;
};

set<my_struct, decltype(cmp)> ss({1, 2, 3}, cmp);
priority_queue<my_struct, vector<my_struct>, decltype(cmp)>
    pq(cmp, {1, 2, 3});
map<my_struct, my_struct, decltype(cmp)> mp({{1, 4}, {2, 5},
    {3, 6}}, cmp);
```

1.5　Dynamic Bitset [c78aa8]

```cpp
const int MAXN = 2e5 + 5;
template <int len = 1>
void solve(int n) {
    if (n > len) {
        solve<min(len*2, MAXN)>(n);
        return;
    }
    bitset<len> a;
}
```

1.6　Enumerate Subset [a13e46]

```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;
    }
}
```

1.7　Fast Input [6f8879]

```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
}
```

1.8　OEIS [f915c2]

```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);
```

```
19          continue;
20        }
21        vector<T> fix = scalarProduct(best, error);
22        fix.insert(fix.begin(), i - bestPos - 1, 0);
23        if (fix.size() >= s.size()) {
24            best = scalarProduct(s, - 1 / error);
25            best.insert(best.begin(), 1 / error);
26            bestPos = i;
27            s.resize(fix.size());
28        }
29        for (size_t j = 0; j < fix.size(); j++)
30            s[j] += fix[j];
31    }
32    return s;
33 }
```

1.9   Pragma [09d13e]

```
1 #pragma GCC optimize("O3,unroll-loops")
2 #pragma GCC target("avx,avx2,sse,sse2,sse3,sse4,popcnt")
```

1.10   Xor Basis [840136]

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

1.11   random int [9cc603]

```
1 mt19937 seed(chrono::steady_clock::now().time_since_epoch().
    count());
2 int rng(int l, int r){
```

```
3     return uniform_int_distribution<int>(l, r)(seed);
4 }
```

1.12   Python

```
1 # Decimal
2 from decimal import *
3 getcontext().prec = 6
4
5 # system setting
6 sys.setrecursionlimit(100000)
7 sys.set_int_max_str_digits(10000)
8
9 # turtle
10 from turtle import *
11
12 N = 3000000010
13 setworldcoordinates(-N, -N, N, N)
14 hideturtle()
15 speed(100)
16
17 def draw_line(a, b, c, d):
18     teleport(a, b)
19     goto(c, d)
20
21 def write_dot(x, y, text, diff=1): # diff = 文字的偏移
22     teleport(x, y)
23     dot(5, "red")
24
25     teleport(x+N/100*diff, y+N/100*diff)
26     write(text, font=("Arial", 5, "bold"))
27
28 # usage
29 draw_line(*a[i], *(a[i-1]))
30 write_dot(*a[i], str(a[i]))
```

1.13   diff

```
1 set -e
2 g++ ac.cpp -o ac
3 g++ wa.cpp -o wa
4 for ((i=0;;i++))
5 do
6     echo "$i"
7     python3 gen.py > input
8     ./ac < input > ac.out
9     ./wa < input > wa.out
10     diff ac.out wa.out || break
11 done
```

1.14   hash command

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

1.15   setup

```
1 se nu rnu bs=2 sw=4 ts=4 hls ls=2 si acd bo=all mouse=a
2
3 :inoremap " ""<Esc>i
4 :inoremap {<CR> {<CR>}<Esc>ko
5 :inoremap {{ {}<ESC>i
6
7 function! F(...)
8     execute '!./%:r < ./' . a:1
9 endfunction
10 command! -nargs=* R call F(<f-args>)
11
12 map <F7> :w<bar>!g++ "%" -o %:r -std=c++17 -Wall -Wextra -
    Wshadow -O2 -DLOCAL -g -fsanitize=undefined,address<CR>
13 map <F8> :!./%:r<CR>
14 map <F9> :!./%:r < ./%:r.in<CR>
15
16 ca hash w !cpp -dD -P -fpreprocessed \| tr -d "[:space:]" \|
    md5sum \| cut -c-6
17
18 " i+<esc>25A---+<esc>
19 " o|<esc>25A    |<esc>
20 " "ggVGyG35pGdd
```

2   Convolution

2.1   FFT any mod [234f9e]

```
1 /*
2 修改 const int MOD = 998244353 更改要取餘的數字
3 PolyMul(a, b) 回傳多項式乘法的結果 ( c_k = \sum_{i+j} a_i+b_j
    mod MOD )
4
5 大約可以支援 5e5，a_i, b_i 皆在 MOD 以下的非負整數
6 */
7 const int MOD = 998244353;
8 typedef complex<double> cd;
9
10 // b9c90a
11 void FFT(vector<cd> &a) {
12     int n = a.size(), L = 31-__builtin_clz(n);
13     vector<complex<long double>> R(2, 1);
14     vector<cd> rt(2, 1);
15     for (int k=2 ; k<n ; k*=2){
16         R.resize(n);
17         rt.resize(n);
18         auto x = polar(1.0L, acos(-1.0L) / k);
19         for (int i=k ; i<2*k ; i++){
20             rt[i] = R[i] = (i&1 ? R[i/2]*x : R[i/2]);
21         }
22     }
23
24     vector<int> rev(n);
25     for (int i=0 ; i<n ; i++){
26         rev[i] = (rev[i/2] | (i&1)<<L)/2;
27     }
28     for (int i=0 ; i<n ; i++){
29         if (i<rev[i]) swap(a[i], a[rev[i]]);
```

```
30        }
31        for (int k=1 ; k<n ; k*=2){
32            for (int i=0 ; i<n ; i+=2*k){
33                for (int j=0 ; j<k ; j++){
34                    auto x = (double *)&rt[j+k];
35                    auto y = (double *)&a[i+j+k];
36                    cd z(x[0]*y[0] - x[1]*y[1], x[0]*y[1] + x[1]*
                         y[0]);
37                    a[i+j+k] = a[i+j]-z;
38                    a[i+j] += z;
39                }
40            }
41        }
42        return;
43 }
44
45 // d3c65e
46 vector<int> PolyMul(vector<int> a, vector<int> b){
47     if (a.empty() || b.empty()) return {};
48
49     vector<int> res(a.size()+b.size()-1);
50     int B = 32-__builtin_clz(res.size()), n = (1<<B), cut =
           int(sqrt(MOD));
51     vector<cd> L(n), R(n), outs(n), outl(n);
52
53     for (int i=0 ; i<a.size() ; i++){
54         L[i] = cd((int) a[i]/cut, (int)a[i]%cut);
55     }
56     for (int i=0 ; i<b.size() ; i++){
57         R[i] = cd((int) b[i]/cut, (int)b[i]%cut);
58     }
59     FFT(L);
60     FFT(R);
61     for (int i=0 ; i<n ; i++){
62         int j = -i&(n-1);
63         outl[j] = (L[i]+conj(L[j])) * R[i]/(2.0*n);
64         outs[j] = (L[i]-conj(L[j])) * R[i]/(2.0*n)/1i;
65     }
66     FFT(outl);
67     FFT(outs);
68     for (int i=0 ; i<res.size() ; i++){
69         int av = (int)(real(outl[i])+0.5), cv = (int)(imag(
               outs[i])+0.5);
70         int bv = (int)(imag(outl[i])+0.5) + (int)(real(outs[i
               ])+0.5);
71         res[i] = ((av%MOD*cut+bv) % MOD*cut+cv) % MOD;
72     }
73
74     return res;
75 }
```

2.2 FFT new [c95bb8]

```
1 typedef complex<double> cd;
2
3 // b9c90a
4 void FFT(vector<cd> &a) {
5     int n = a.size(), L = 31-__builtin_clz(n);
6     vector<complex<long double>> R(2, 1);
7     vector<cd> rt(2, 1);
8     for (int k=2 ; k<n ; k*=2){
9         R.resize(n);
10        rt.resize(n);
```

```
11        auto x = polar(1.0L, acos(-1.0L) / k);
12        for (int i=k ; i<2*k ; i++){
13            rt[i] = R[i] = (i&1 ? R[i/2]*x : R[i/2]);
14        }
15    }
16
17    vector<int> rev(n);
18    for (int i=0 ; i<n ; i++){
19        rev[i] = (rev[i/2] | (i&1)<<L)/2;
20    }
21    for (int i=0 ; i<n ; i++){
22        if (i<rev[i]) swap(a[i], a[rev[i]]);
23    }
24    for (int k=1 ; k<n ; k*=2){
25        for (int i=0 ; i<n ; i+=2*k){
26            for (int j=0 ; j<k ; j++){
27                auto x = (double *)&rt[j+k];
28                auto y = (double *)&a[i+j+k];
29                cd z(x[0]*y[0] - x[1]*y[1], x[0]*y[1] + x[1]*
                     y[0]);
30                a[i+j+k] = a[i+j]-z;
31                a[i+j] += z;
32            }
33        }
34    }
35    return;
36 }
37
38 // 39029d
39 vector<double> PolyMul(const vector<double> a, const vector<
       double> b){
40     if (a.empty() || b.empty()) return {};
41     vector<double> res(a.size()+b.size()-1);
42     int L = 32 - __builtin_clz(res.size()), n = 1 << L;
43     vector<cd> in(n), out(n);
44
45     copy(a.begin(), a.end(), begin(in));
46     for (int i=0 ; i<b.size() ; i++){
47         in[i].imag(b[i]);
48     }
49     FFT(in);
50     for (cd& x : in) x *= x;
51     for (int i=0 ; i<n ; i++){
52         out[i] = in[-i & (n - 1)] - conj(in[i]);
53     }
54     FFT(out);
55
56     for (int i=0 ; i<res.size() ; i++){
57         res[i] = imag(out[i]) / (4 * n);
58     }
59
60     return res;
61 }
```

2.3 FFT short [70c01a]

```
1 #define int long long
2
3 using Cplx = complex<double>;
4 const double pi = acos(-1);
5 const int mod = 998244353, g = 3;
6 int power(int a, int b) {
7     int res = 1;
```

```
8     while (b) {
9         if (b & 1) res = res * a % mod;
10        a = a * a % mod;
11        b >>= 1;
12    }
13    return res;
14 }
15 int inv(int x) { return power(x, mod - 2); }
16 // FFT use Cplx, NTT use LL
17 void FFT(vector<int> &a, int n, int op) {
18     // n must be 2^k
19     vector<int> R(n);
20     FOR (i, 0, n - 1)
21         R[i] = R[i/2]/2 + (i&1)*(n/2);
22     FOR (i, 0, n - 1)
23         if (i < R[i]) swap(a[i], a[R[i]]);
24     for (int m = 2; m <= n; m *= 2) {
25         // Cplx w1({cos(2*pi/m), sin(2*pi/m)*op});
26         int w1 = power(g, (mod-1)/m * op + mod-1);
27         for (int i = 0; i < n; i += m) {
28             // Cplx wk({1, 0});
29             int wk = 1;
30             FOR (k, 0, m / 2 - 1) {
31                 auto x = a[i+k], y = a[i+k+m/2] * wk % mod;
32                 a[i+k] = (x+y) % mod;
33                 a[i+k+m/2] = (x-y+mod) % mod;
34                 wk = wk * w1 % mod;
35             }
36         }
37     }
38     if (op == -1)
39         FOR (i, 0, n - 1) {
40             // a[i] = a[i] / n;
41             a[i] = a[i] * inv(n) % mod;
42         }
43 }
```

2.4 FWT [832aa5]

```
1 // 已經把 mint 刪掉，需要增加註解
2 vector<int> xor_convolution(vector<int> a, vector<int> b, int
       k) {
3     if (k == 0) {
4         return vector<int>{a[0] * b[0]};
5     }
6     vector<int> aa(1 << (k - 1)), bb(1 << (k - 1));
7     FOR (i, 0, (1 << (k - 1)) - 1) {
8         aa[i] = a[i] + a[i + (1 << (k - 1))];
9         bb[i] = b[i] + b[i + (1 << (k - 1))];
10    }
11    vector<int> X = xor_convolution(aa, bb, k - 1);
12    FOR (i, 0, (1 << (k - 1)) - 1) {
13        aa[i] = a[i] - a[i + (1 << (k - 1))];
14        bb[i] = b[i] - b[i + (1 << (k - 1))];
15    }
16    vector<int> Y = xor_convolution(aa, bb, k - 1);
17    vector<int> c(1 << k);
18    FOR (i, 0, (1 << (k - 1)) - 1) {
19        c[i              ] = (X[i] + Y[i]) / 2;
20        c[i + (1 << (k - 1))] = (X[i] - Y[i]) / 2;
21    }
22    return c;
23 };
```

2.5 Min Convolution Concave Concave [ffb28d]

```cpp
// 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;
}
```

2.6 NTT mod 998244353 [5c6335]

```cpp
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);
```

```cpp
    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;
}
```

3 Data-Structure

3.1 BIT [7ef3a9]

```cpp
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 [447002]

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

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

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

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

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

```cpp
        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 [866cf6]

```cpp
#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 [231774]

```cpp
/*
.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>
```

```cpp
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 [bb1898]

```cpp
// [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{
```

```cpp
            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 [45b8ba]

```cpp
/*
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
*/
```

```cpp
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 [2cb0a4]

```cpp
/*
全部都是 0-based
```

```
3  /*
4  宣告
5  LC_Segment_Tree st(n);
6
7  函式：
8  update_segment({a, b}, ql, qr)：在 [ql, qr) 插入一條 y=ax+b
         的線段
9  query(x)：查詢所有直線在位置 x 的最小值
10 */
11 const int MAX_V = 1e6+10; // 值域最大值
12
13 struct LC_Segment_Tree{
14     struct Node{ // y = ax+b
15         int a = 0;
16         int b = INF;
17
18         int y(int x){
19             return a*x+b;
20         }
21     };
22     vector<Node> arr;
23
24     LC_Segment_Tree(int n = 0){
25         arr.resize(4*n);
26     }
27
28     void update(Node val, int idx = 0, int ll = 0, int rr =
         MAX_V){
29         if (rr-ll==0) return;
30         if (rr-ll<=1){
31             if (val.y(ll)<arr[idx].y(ll)){
32                 arr[idx] = val;
33             }
34             return;
35         }
36
37         int mid = (ll+rr)/2;
38         if (arr[idx].a > val.a) swap(arr[idx], val); // 原本
             的線斜率要比較小
39         if (arr[idx].y(mid) < val.y(mid)){ // 交點在左邊
40             update(val, idx*2+1, ll, mid);
41         }else{ // 交點在右邊
42             swap(arr[idx], val); // 在左子樹中，新線比舊線還
                 要好
43             update(val, idx*2+2, mid, rr);
44         }
45         return;
46     }
47
48     // 在 [ql, qr) 加上一條 val 的線段
49     void update_segment(Node val, int ql, int qr, int idx =
         0, int ll = 0, int rr = MAX_V){
50         if (rr-ll==0) return;
51         if (rr<=ql || qr<=ll) return;
52         if (ql<=ll && rr<=qr){
53             update(val, idx, ll, rr);
54             return;
55         }
56
57         int mid = (ll+rr)/2;
58         update_segment(val, ql, qr, idx*2+1, ll, mid);
59         update_segment(val, ql, qr, idx*2+2, mid, rr);
60         return;
61     }
```

```
62
63     int query(int x, int idx = 0, int ll = 0, int rr = MAX_V)
         {
64         if (rr-ll==0) return INF;
65         if (rr-ll==1){
66             return arr[idx].y(ll);
67         }
68
69         int mid = (ll+rr)/2;
70         if (x<mid){
71             return min(arr[idx].y(x), query(x, idx*2+1, ll,
                 mid));
72         }else{
73             return min(arr[idx].y(x), query(x, idx*2+2, mid,
                 rr));
74         }
75     }
76 };
```

3.8   Segment Tree Persistent [3b5aa9]

```
1  /*
2  全部都是 0-based
3
4  宣告
5  Persistent_Segment_Tree st(n+q);
6  st.build(v, 0);
7
8  函式：
9  update_version(pos, val, ver)：對版本 ver 的 pos 位置改成 val
10 query_version(ql, qr, ver)：對版本 ver 查詢 [ql, qr) 的區間和
11 clone_version(ver)：複製版本 ver 到最新的版本
12 */
13 struct Persistent_Segment_Tree{
14     int node_cnt = 0;
15     struct Node{
16         int lc = -1;
17         int rc = -1;
18         int val = 0;
19     };
20     vector<Node> arr;
21     vector<int> version;
22
23     Persistent_Segment_Tree(int sz){
24         arr.resize(32*sz);
25         version.push_back(node_cnt++);
26         return;
27     }
28
29     void pull(Node &c, Node a, Node b){
30         c.val = a.val+b.val;
31         return;
32     }
33
34     void build(vector<int> &v, int idx, int ll = 0, int rr =
         n){
35         auto &now = arr[idx];
36
37         if (rr-ll==1){
38             now.val = v[ll];
39             return;
40         }
```

```
41         int mid = (ll+rr)/2;
42         now.lc = node_cnt++;
43         now.rc = node_cnt++;
44         build(v, now.lc, ll, mid);
45         build(v, now.rc, mid, rr);
46         pull(now, arr[now.lc], arr[now.rc]);
47         return;
48     }
49
50     void update(int pos, int val, int idx, int ll = 0, int rr
         = n){
51         auto &now = arr[idx];
52
53         if (rr-ll==1){
54             now.val = val;
55             return;
56         }
57
58         int mid = (ll+rr)/2;
59         if (pos<mid){
60             arr[node_cnt] = arr[now.lc];
61             now.lc = node_cnt;
62             node_cnt++;
63             update(pos, val, now.lc, ll, mid);
64         }else{
65             arr[node_cnt] = arr[now.rc];
66             now.rc = node_cnt;
67             node_cnt++;
68             update(pos, val, now.rc, mid, rr);
69         }
70         pull(now, arr[now.lc], arr[now.rc]);
71         return;
72     }
73
74     void update_version(int pos, int val, int ver){
75         update(pos, val, version[ver]);
76     }
77
78     Node query(int ql, int qr, int idx, int ll = 0, int rr =
         n){
79         auto &now = arr[idx];
80
81         if (ql<=ll && rr<=qr) return now;
82         if (rr<=ql || qr<=ll) return Node();
83
84         int mid = (ll+rr)/2;
85
86         Node ret;
87         pull(ret, query(ql, qr, now.lc, ll, mid), query(ql,
             qr, now.rc, mid, rr));
88         return ret;
89     }
90
91     Node query_version(int ql, int qr, int ver){
92         return query(ql, qr, version[ver]);
93     }
94
95     void clone_version(int ver){
96         version.push_back(node_cnt);
97         arr[node_cnt] = arr[version[ver]];
98         node_cnt++;
99     }
100 };
```

3.9 Sparse Table [31f22a]

```cpp
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 [5851f5]

```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);
```

```cpp
        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 Treap2 [1bf328]

```cpp
// 1-based．請注意 MAX_N 是否足夠大
int root = 0;
int lc[MAX_N], rc[MAX_N];
int pri[MAX_N], val[MAX_N];
int sz[MAX_N], tag[MAX_N], fa[MAX_N];
int new_node(int v){
    static int nodeCnt = 0;
    nodeCnt++;
    val[nodeCnt] = v;
    sz[nodeCnt] = 1;
    pri[nodeCnt] = rand();
    return nodeCnt;
}

void push(int x){
    if (tag[x]){
        if (lc[x]) tag[lc[x]] ^= 1;
        if (rc[x]) tag[rc[x]] ^= 1;
    }
    tag[x] = 0;
}

int pull(int x){
    if (x){
        fa[x] = 0;
        sz[x] = 1+sz[lc[x]]+sz[rc[x]];
```

```cpp
        if (lc[x]) fa[lc[x]] = x;
        if (rc[x]) fa[rc[x]] = x;
    }
    return x;
}

int merge(int a, int b){
    if (!a or !b) return a|b;
    push(a), push(b);

    if (pri[a]>pri[b]){
        rc[a] = merge(rc[a], b);
        return pull(a);
    }else{
        lc[b] = merge(a, lc[b]);
        return pull(b);
    }
}

// [1, k] [k+1, n]
void split(int x, int k, int &a, int &b) {
    if (!x) return a = b = 0, void();
    push(x);
    if (sz[lc[x]] >= k) {
        split(lc[x], k, a, lc[x]);
        b = x;
        pull(a); pull(b);
    }else{
        split(rc[x], k - sz[lc[x]] - 1, rc[x], b);
        a = x;
        pull(a); pull(b);
    }
}

// functions
// 回傳 x 在 Treap 中的位置
int get_pos(int x){
    vector<int> sta;
    while (fa[x]){
        sta.push_back(x);
        x = fa[x];
    }
    while (sta.size()){
        push(x);
        x = sta.back();
        sta.pop_back();
    }
    push(x);

    int res = sz[x] - sz[rc[x]];
    while (fa[x]){
        if (rc[fa[x]]==x){
            res += sz[fa[x]]-sz[x];
        }
        x = fa[x];
    }
    return res;
}

// 1-based <前 [1, l-1] 個元素, [l, r] 個元素, [r+1, n] 個元
    素>
array<int, 3> cut(int x, int l, int r){
    array<int, 3> ret;
    split(x, r, ret[1], ret[2]);
    split(ret[1], l-1, ret[0], ret[1]);
```

```cpp
        return ret;
    }
}

void print(int x){
    push(x);
    if (lc[x]) print(lc[x]);
    cerr << val[x] << " ";
    if (rc[x]) print(rc[x]);
}
```

3.12　Trie [b6475c]

```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 [133f00]

```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 [df69b1]

```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;

#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 [8dfa8b]

```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 [d9966f]

```
1  using ld = double;
2
3  // 判斷數值正負：{1:正數,0:零,-1:負數}
4  int sign(long long x) {return (x >= 0) ? ((bool)x) : -1; }
5  int sign(ld x) {return (abs(x) < 1e-9) ? 0 : (x>0 ? 1 : -1);}
6
7  template<typename T>
8  struct point {
9      T x, y;
10     point() {}
11     point(const T &x, const T &y) : x(x), y(y) {}
12     explicit operator point<ld>() {return point<ld>(x, y); }
13     // A [6357c4]
14     point operator+(point b) {return {x+b.x, y+b.y}; }
15     point operator-(point b) {return {x-b.x, y-b.y}; }
16     point operator*(T b) {return {x*b, y*b}; }
17     point operator/(T b) {return {x/b, y/b}; }
18     bool operator==(point b) {return x==b.x && y==b.y; }
19
20     T operator*(point b) {return x * b.x + y * b.y; }
21     T operator^(point b) {return x * b.y - y * b.x; }
22     // B [c415da]
23     // 逆時針極角排序
24     bool side() const{return (y == 0) ? (x > 0) : (y < 0); }
25     bool operator<(const point &b) const {
26         return side() == b.side() ?
27             (x*b.y > b.x*y) : side() < b.side();
28     }
29     friend ostream& operator<<(ostream& os, point p) {
30         return os << "(" << p.x << ", " << p.y << ")";
31     }
32     // 判斷 ab 到 ac 的方向：{1:逆時鐘,0:重疊,-1:順時鐘}
33     friend int ori(point a, point b, point c) {
34         return sign((b-a)^(c-a));
35     }
36     friend bool btw(point a, point b, point c) {
37         return ori(a, b, c) == 0 && sign((a-c)*(b-c)) <= 0;
38     }
39     // 判斷線段 ab, cd 是否相交
40     friend bool banana(point a, point b, point c, point d) {
41         if (btw(a, b, c) || btw(a, b, d)
42             || btw(c, d, a) || btw(c, d, b)) return true;
43         int u = ori(a, b, c) * ori(a, b, d);
44         int v = ori(c, d, a) * ori(c, d, b);
45         return u < 0 && v < 0;
46     } // C [09fd7c]
47     // 判斷 "射線 ab" 與 "線段 cd" 是否相交
48     friend bool rayHitSeg(point a,point b,point c,point d) {
49         if (a == b) return btw(c, d, a); // Special case
50         if (((a - b) ^ (c - d)) == 0) {
51             return btw(a, c, b) || btw(a, d, b) || banana(a,
                 b, c, d);
52         }
53         point u = b - a, v = d - c, s = c - a;
54         return sign(s ^ v) * sign(u ^ v) >= 0 && sign(s ^ u)
55             * sign(u ^ v) >= 0 && abs(s ^ u) <= abs(u ^ v);
56     } // D [db541a]
57     // 旋轉 Arg(b) 的角度（小心溢位）
58     point rotate(point b){return {x*b.x-y*b.y, x*b.y+y*b.x};}
59     // 回傳極座標角度，值域：[-π, +π]
60     friend ld Arg(point b) {
61         return (b.x != 0 || b.y != 0) ? atan2(b.y, b.x) : 0;
62     }
63     friend T abs2(point b) {return b * b; }
64 };
65
66 template<typename T>
67 struct line {
68     /*--------------- E [683239] ---------------*/
69     point<T> p1, p2;
70     // ax + by + c = 0
71     T a, b, c; // |a|, |b| ≤ 2C, |c| ≤ 8C²
72     line() {}
73     line(const point<T> &x,const point<T> &y) : p1(x), p2(y){
74         build();
75     }
76     void build() {
77         a = p1.y - p2.y;
78         b = p2.x - p1.x;
79         c = (-a*p1.x)-b*p1.y;
80     } // E [683239]
81     // 判斷點和有向直線的關係：{1:左邊,0:在線上,-1:右邊}
82     int ori(point<T> &p) {
83         return sign((p2-p1) ^ (p-p1));
84     }
85     // 判斷直線斜率是否相同
86     bool parallel(line &l) {
87         return ((p1-p2) ^ (l.p1-l.p2)) == 0;
88     }
89     // 兩直線交點
90     point<ld> line_intersection(line &l) {
91         using P = point<ld>;
92         point<T> u = p2-p1, v = l.p2-l.p1, s = l.p1-p1;
93         return P(p1) + P(u) * ((ld(s^v)) / (u^v));
94     }
95 };
96
97 template<typename T>
98 struct polygon {
99     vector<point<T>> v;
100    polygon() {}
101    polygon(const vector<point<T>> &u) : v(u) {}
102    // simple 為 true 的時候會回傳任意三點不共線的凸包
103    void make_convex_hull(int simple) {
104        auto cmp = [&](point<T> &p, point<T> &q) {
105            return (p.x == q.x) ? (p.y < q.y) : (p.x < q.x);
106        };
107        simple = (bool)simple;
108        sort(v.begin(), v.end(), cmp);
109        v.resize(unique(v.begin(), v.end()) - v.begin());
110        if (v.size() <= 1) return;
111        vector<point<T>> hull;
112        for (int t = 0; t < 2; ++t){
113            int sz = hull.size();
114            for (auto &i:v) {
115                while (hull.size() >= sz+2 && ori(hull[hull.
                     size()-2], hull.back(), i) < simple) {
116                    hull.pop_back();
117                }
118                hull.push_back(i);
119            }
120            hull.pop_back();
121            reverse(v.begin(), v.end());
122        }
123        swap(hull, v);
124    } // F [2bb3ef]
125    // 可以在有 n 個點的簡單多邊形內，用 O(n) 判斷一個點：
126    // {1：在多邊形內，0：在多邊形上，-1：在多邊形外}
127    int in_polygon(point<T> a){
128        const T MAX_POS = 1e9 + 5; // [記得修改] 座標的最大值
129        point<T> pre = v.back(), b(MAX_POS, a.y + 1);
130        int cnt = 0;
131
132        for (auto &i:v) {
133            if (btw(pre, i, a)) return 0;
134            if (banana(a, b, pre, i)) cnt++;
135            pre = i;
136        }
137
138        return cnt%2 ? 1 : -1;
139    } // G [f11340]
140    /// 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 ///
141    // 可以在有 n 個點的凸包內，用 O(log n) 判斷一個點：
142    // {1：在凸包內，0：在凸包邊上，-1：在凸包外}
143    int in_convex(point<T> p) {
144        int n = v.size();
145        int a = ori(v[0], v[1], p), b = ori(v[0], v[n-1], p);
146        if (a < 0 || b > 0) return -1;
147        if (btw(v[0], v[1], p)) return 0;
148        if (btw(v[0], v[n - 1], p)) return 0;
149        int l = 1, r = n - 1, mid;
150        while (l + 1 < r) {
151            mid = (l + r) >> 1;
152            if (ori(v[0], v[mid], p) >= 0) l = mid;
153            else r = mid;
154        }
155        int k = ori(v[l], v[r], p);
156        if (k <= 0) return k;
157        return 1;
158    } // H [e64f1e]
159    // 凸包專用的環狀二分搜，回傳 0-based index
160    int cycle_search(auto &f) {
161        int n = v.size(), l = 0, r = n;
162        if (n == 1) return 0;
163        bool rv = f(1, 0);
164        while (r - l > 1) {
165            int m = (l + r) / 2;
166            if (f(0, m) ? rv: f(m, (m + 1) % n)) r = m;
167            else l = m;
168        }
169        return f(l, r % n) ? l : r % n;
170    } // I [fe2f51]
171    // 可以在有 n 個點的凸包內，用 O(log n) 判斷一條直線：
172    // {1：穿過凸包，0：剛好切過凸包，-1：沒碰到凸包}
173    int line_cut_convex(line<T> L) {
174        L.build();
175        point<T> p(L.a, L.b);
```

```
177          auto gt = [&](int neg) {
178              auto f = [&](int x, int y) {
179                  return sign((v[x] - v[y]) * p) == neg;
180              };
181              return -(v[cycle_search(f)] * p);
182          };
183          T x = gt(1), y = gt(-1);
184          if (L.c < x || y < L.c) return -1;
185          return not (L.c == x || L.c == y);
186      } // J [b6a4c8]
187  //  可以在有 n 個點的凸包內，用 O(log n) 判斷一個線段：
188  //  {1 : 存在一個凸包上的邊可以把這個線段切成兩半，
189  //   0 : 有碰到凸包但沒有任何凸包上的邊可以把它切成兩半，
190  //  -1 : 沒碰到凸包}
191  /// 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 ///
192      int segment_across_convex(line<T> L) {
193          L.build();
194          point<T> p(L.a, L.b);
195          auto gt = [&](int neg) {
196              auto f = [&](int x, int y) {
197                  return sign((v[x] - v[y]) * p) == neg;
198              };
199              return cycle_search(f);
200          };
201          int i = gt(1), j = gt(-1), n = v.size();
202          T x = -(v[i] * p), y = -(v[j] * p);
203          if (L.c < x || y < L.c) return -1;
204          if (L.c == x || L.c == y) return 0;
205
206          if (i > j) swap(i, j);
207          auto g = [&](int x, int lim) {
208              int now = 0, nxt;
209              for (int i = 1 << __lg(lim); i > 0; i /= 2) {
210                  if (now + i > lim) continue;
211                  nxt = (x + i) % n;
212                  if (L.ori(v[x]) * L.ori(v[nxt]) >= 0) {
213                      x = nxt;
214                      now += i;
215                  }
216              } //  ↓ BE CAREFUL
217              return -(ori(v[x], v[(x + 1) % n], L.p1) * ori(v[
                   x], v[(x + 1) % n], L.p2));
218          };
219          return max(g(i, j - i), g(j, n - (j - i)));
220      } // K [b4f073]
221  //  可以在有 n 個點的凸包內，用 O(log n) 判斷一個線段：
222  //  {1 : 線段上存在某一點位於凸包內部（邊上不算），
223  //   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，
224  //  -1 : 線段完全在凸包外面}
225      int segment_pass_convex_interior(line<T> L) {
226          if (in_convex(L.p1) == 1 || in_convex(L.p2) == 1)
                   return 1;
227          L.build();
228          point<T> p(L.a, L.b);
229          auto gt = [&](int neg) {
230              auto f = [&](int x, int y) {
231                  return sign((v[x] - v[y]) * p) == neg;
232              };
233              return cycle_search(f);
234          };
235          int i = gt(1), j = gt(-1), n = v.size();
236          T x = -(v[i] * p), y = -(v[j] * p);
237          if (L.c < x || y < L.c) return -1;
238          if (L.c == x || L.c == y) return 0;
239
240          if (i > j) swap(i, j);
241          auto g = [&](int x, int lim) {
242              int now = 0, nxt;
243              for (int i = 1 << __lg(lim); i > 0; i /= 2) {
244                  if (now + i > lim) continue;
245                  nxt = (x + i) % n;
246                  if (L.ori(v[x]) * L.ori(v[nxt]) > 0) {
247                      x = nxt;
248                      now += i;
249                  }
250              } //  ↓ BE CAREFUL
251              return -(ori(v[x], v[(x + 1) % n], L.p1) * ori(v[
                   x], v[(x + 1) % n], L.p2));
252          };
253          int ret = max(g(i, j - i), g(j, n - (j - i)));
254          return (ret == 0) ? (in_convex(L.p1) == 0 &&
                   in_convex(L.p2) == 0) : ret;
255      } // L [5f45ca]
256  //  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）
257      pair<int,int> convex_tangent_point(point<T> p) {
258          int n = v.size(), z = -1, edg = -1;
259          auto gt = [&](int neg) {
260              auto check = [&](int x) {
261                  if (v[x] == p) z = x;
262                  if (btw(v[x], v[(x + 1) % n], p)) edg = x;
263                  if (btw(v[(x + n - 1) % n], v[x], p)) edg = (
                       x + n - 1) % n;
264              };
265              auto f = [&](int x, int y) {
266                  check(x); check(y);
267                  return ori(p, v[x], v[y]) == neg;
268              };
269              return cycle_search(f);
270          };
271          int x = gt(1), y = gt(-1);
272          if (z != -1) {
273              return {(z + n - 1) % n, (z + 1) % n};
274          }
275          else if (edg != -1) {
276              return {edg, (edg + 1) % n};
277          }
278          else {
279              return {x, y};
280          }
281      } // M [a6f66b]
282      friend int halfplane_intersection(vector<line<T>> &s,
          polygon<T> &P) {
283          auto angle_cmp = [&](line<T> &A, line<T> &B) {
284              point<T> a = A.p2-A.p1, b = B.p2-B.p1;
285              return (a < b);
286          };
287          sort(s.begin(), s.end(), angle_cmp); // 線段左側為該
              線段半平面
288          int L, R, n = s.size();
289          vector<point<T>> px(n);
290          vector<line<T>> q(n);
291          q[L = R = 0] = s[0];
292          for(int i = 1; i < n; ++i) {
293              while(L < R && s[i].ori(px[R-1]) <= 0) --R;
294              while(L < R && s[i].ori(px[L])   <= 0) ++L;
295              q[++R] = s[i];
296              if(q[R].parallel(q[R-1])) {
297                  --R;
298                  if(q[R].ori(s[i].p1) > 0) q[R] = s[i];
299              }
300              if(L<R) px[R-1] = q[R-1].line_intersection(q[R]);
301          }
302          while(L < R && q[L].ori(px[R-1]) <= 0) --R;
303          P.v.clear();
304          if(R - L <= 1) return 0;
305          px[R] = q[R].line_intersection(q[L]);
306          for(int i = L; i <= R; ++i) P.v.push_back(px[i]);
307          return R - L + 1;
308      } // N [102d48]
309  };
310
311  struct Cir {
312      point<ld> o; ld r;
313      friend ostream& operator<<(ostream& os, Cir c) {
314          return os << "(x" << "+-"[c.o.x >= 0] << abs(c.o.x)
                  << ")^2 + (y" << "+-"[c.o.y >= 0] << abs(c.o.y)
                  << ")^2 = " << c.r * c.r;
315      }
316      bool covers(Cir b) {
317          return sqrt((ld)abs2(o - b.o)) + b.r <= r;
318      }
319      vector<point<ld>> Cir_intersect(Cir c) {
320          ld d2 = abs2(o - c.o), d = sqrt(d2);
321          if (d < max(r, c.r) - min(r, c.r) || d > r + c.r)
                  return {};
322          auto sqdf = [&](ld x, ld y) { return x*x - y*y; };
323          point<ld> u = (o + c.o) / 2 + (o - c.o) * (sqdf(c.r,
                  r) / (2 * d2));
324          ld A = sqrt(sqdf(r + d, c.r) * sqdf(c.r, d - r));
325          point<ld> v = (c.o - o).rotate({0,1}) * A / (2 * d2);
326          if (sign(v.x) == 0 && sign(v.y) == 0) return {u};
327          return {u - v, u + v};
328      } // O [330a1c]
329      auto point_tangent(point<ld> p) {
330          vector<point<ld>> res;
331          ld d_sq = abs2(p - o);
332          if (sign(d_sq - r * r) <= 0) {
333              res.pb(p + (p - o).rotate({0, 1}));
334          } else if (d_sq > r * r) {
335              ld s = d_sq - r * r;
336              point<ld> v = p + (o - p) * s / d_sq;
337              point<ld> u = (o - p).rotate({0, 1}) * sqrt(s) *
                      r / d_sq;
338              res.pb(v + u);
339              res.pb(v - u);
340          }
341          return res;
342      } // P [0067e6]
343  };
```

5.2   Geometry Struct 3D [4a50c9]

```
1  using ld = long double;
2
3  // 判斷數值正負：{1:正數,0:零,-1:負數}
4  int sign(long long x) {return (x >= 0) ? ((bool)x) : -1; }
5  int sign(ld x) {return (abs(x) < 1e-9) ? 0 : (x>0 ? 1 : -1);}
6
7  template<typename T>
```

```cpp
struct pt3 {
    T x, y, z;
    pt3(){}
    pt3(const T &x, const T &y, const T &z):x(x),y(y),z(z){}
    explicit operator pt3<ld>() {return pt3<ld>(x, y, z); }

    pt3 operator+(pt3 b) {return {x+b.x, y+b.y, z+b.z}; }
    pt3 operator-(pt3 b) {return {x-b.x, y-b.y, z-b.z}; }
    pt3 operator*(T b) {return {x * b, y * b, z * b}; }
    pt3 operator/(T b) {return {x / b, y / b, z / b}; }
    bool operator==(pt3 b){return x==b.x&&y==b.y&&z==b.z;}

    T operator*(pt3 b) {return x*b.x+y*b.y+z*b.z; }
    pt3 operator^(pt3 b) {
        return pt3(y * b.z - z * b.y,
                   z * b.x - x * b.z,
                   x * b.y - y * b.x);
    }
    friend T abs2(pt3 b) {return b * b; }
    friend T len (pt3 b) {return sqrt(abs2(b)); }
    friend ostream& operator<<(ostream& os, pt3 p) {
        return os << "(" << p.x << ", " <<
                     p.y << ", " << p.z << ")";
    }
};

template<typename T>
struct face {
    int a, b, c; // 三角形在 vector 裡面的 index
    pt3<T> q;    // 面積向量（朝外）
};

/// 警告；v 在過程中可能被修改，回傳的 face 以修改後的為準
// O(n²)，最多只有 2n-4 個面
// 當凸包退化時會回傳空的凸包，否則回傳凸包上的每個面
template<typename T>
vector<face<T>> hull3(vector<pt3<T>> &v) {
    int n = v.size();
    if (n < 3) return {};
    // don't use "==" when you use ld
    sort(all(v), [&](pt3<T> &p, pt3<T> &q) {
        return sign(p.x - q.x) ? (p.x < q.x) :
          (sign(p.y - q.y) ? p.y < q.y : p.z < q.z);
    });
    v.resize(unique(v.begin(), v.end()) - v.begin());
    for (int i = 2; i <= n; ++i) {
        if (i == n) return {};
        if (sign(len(((v[1] - v[0]) ^ (v[i] - v[0]))))) {
            swap(v[2], v[i]);
            break;
        }
    }
    pt3<T> tmp_q = (v[1] - v[0]) ^ (v[2] - v[0]);
    for (int i = 3; i <= n; ++i) {
        if (i == n) return {};
        if (sign((v[i] - v[0]) * tmp_q)) {
            swap(v[3], v[i]);
            break;
        }
    }

    vector<face<T>> f;
    vector<vector<int>> dead(n, vector<int>(n, true));
    auto add_face = [&](int a, int b, int c) {
        f.emplace_back(a, b, c, (v[b] - v[a]) ^ (v[c] - v[a])
            );
        dead[a][b] = dead[b][c] = dead[c][a] = false;
    };
    add_face(0, 1, 2);
    add_face(0, 2, 1);

    for (int i = 3; i < n; ++i) {
        vector<face<T>> f2;
        for (auto &[a, b, c, q] : f) {
            if (sign((v[i] - v[a]) * q) > 0)
                dead[a][b] = dead[b][c] = dead[c][a] = true;
            else f2.emplace_back(a, b, c, q);
        }
        f.clear();
        for (face<T> &F : f2) {
            int arr[3] = {F.a, F.b, F.c};
            for (int j = 0; j < 3; ++j) {
                int a = arr[j], b = arr[(j + 1) % 3];
                if (dead[b][a]) add_face(b, a, i);
            }
        }
        f.insert(f.end(), all(f2));
    }
    return f;
}
```

5.3 Pick's Theorem

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

6 Graph

6.1 2-SAT [5a6317]

```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)
```

```cpp
                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;
    }
    // A or B 都是 0-based
    void add_disjunction(int a, bool na, int b, bool nb) {
        // na is true => ~a, na is false => a
        // nb is true => ~b, nb is false => b
        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 [f8a5dd]

```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);
    }
```

```
16    // bb03e2
17    bool dfs1(int now){
18        visx[now] = stamp;
19
20        for (auto x : G[now]){
21            if (my[x]==-1){
22                mx[now] = x;
23                my[x] = now;
24                return true;
25            }
26        }
27        for (auto x : G[now]){
28            if (visx[my[x]]!=stamp && dfs1(my[x])){
29                mx[now] = x;
30                my[x] = now;
31                return true;
32            }
33        }
34        return false;
35    }
36
37    vector<pair<int, int>> find_max_matching(){
38        vector<pair<int, int>> ret;
39
40        while (true){
41            stamp++;
42            int tmp = 0;
43            for (int i=0 ; i<n ; i++){
44                if (mx[i]==-1 && dfs1(i)) tmp++;
45            }
46            if (tmp==0) break;
47        }
48
49        for (int i=0 ; i<n ; i++){
50            if (mx[i]!=-1){
51                ret.push_back({i, mx[i]});
52            }
53        }
54        return ret;
55    }
56
57    // 645577
58    void dfs2(int now){
59        visx[now] = true;
60
61        for (auto x : G[now]){
62            if (my[x]!=-1 && visy[x]==false){
63                visy[x] = true;
64                dfs2(my[x]);
65            }
66        }
67    }
68
69    // 要先執行 find_max_matching 一次
70    vector<pair<int, int>> find_min_vertex_cover(){
71        fill(visx.begin(), visx.end(), false);
72        fill(visy.begin(), visy.end(), false);
73
74        vector<pair<int, int>> ret;
75        for (int i=0 ; i<n ; i++){
76            if (mx[i]==-1) dfs2(i);
77        }
78
79        for (int i=0 ; i<n ; i++){
80            if (visx[i]==false) ret.push_back({1, i});
81        }
82        for (int i=0 ; i<m ; i++){
83            if (visy[i]==true) ret.push_back({2, i});
84        }
85
86        return ret;
87    }
88 };
```

6.3　C3C4 [d00465]

```
1  // 0-based
2  void C3C4(vector<int> deg, vector<array<int, 2>> edges){
3      int N = deg.size();
4      int M = deges.size();
5
6      vector<int> ord(N), rk(N);
7      iota(ord.begin(), ord.end(), 0);
8      sort(ord.begin(), ord.end(), [&](int x, int y) { return
           deg[x] > deg[y]; });
9      for (int i=0 ; i<N ; i++) rk[ord[i]] = i;
10
11     vector<vector<int>> D(N), adj(N);
12     for (auto [u, v] : e) {
13         if (rk[u] > rk[v]) swap(u, v);
14         D[u].emplace_back(v);
15         adj[u].emplace_back(v);
16         adj[v].emplace_back(u);
17     }
18
19     vector<int> vis(N);
20
21     int c3 = 0, c4 = 0;
22     for (int x : ord) { // c3
23         for (int y : D[x]) vis[y] = 1;
24         for (int y : D[x]) for (int z : D[y]){
25             c3 += vis[z]; // xyz is C3
26         }
27         for (int y : D[x]) vis[y] = 0;
28     }
29     for (int x : ord) { // c4
30         for (int y : D[x]) for (int z : adj[y])
31             if (rk[z] > rk[x]) c4 += vis[z]++;
32         for (int y : D[x]) for (int z : adj[y])
33             if (rk[z] > rk[x]) --vis[z];
34     }  // both are O(M*sqrt(M)), test @ 2022 CCPC guangzhou
35     cout << c4 << "\n";
36 }
```

6.4　Cut BCC [2af809]

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

6.5　Dinic [961b34]

```
1  // 一般圖：O(EV²)
2  // 二分圖：O(E√V)
3  struct Flow{
4      using T = int; // 可以換成別的型別
5      struct Edge{
6          int v; T rc; int rid;
7      };
8      vector<vector<Edge>> G;
9      void add(int u, int v, T c){
10         G[u].push_back({v, c, G[v].size()});
11         G[v].push_back({u, 0, G[u].size()-1});
12     }
13     vector<int> dis, it;
14
15     Flow(int n){
16         G.resize(n);
17         dis.resize(n);
18         it.resize(n);
19     }
20
21     // ce56d6
22     T dfs(int u, int t, T f){
23         if (u == t || f == 0) return f;
24         for (int &i=it[u] ; i<G[u].size() ; i++){
25             auto &[v, rc, rid] = G[u][i];
26             if (dis[v]!=dis[u]+1) continue;
27             T df = dfs(v, t, min(f, rc));
28             if (df <= 0) continue;
29             rc -= df;
30             G[v][rid].rc += df;
31             return df;
```

```cpp
        }
        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 [52b249]

```cpp
/*
全部都是 0-based
```

```cpp
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){
```

```cpp
        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   EdgeBCC [d09eb1]

```cpp
// d09eb1
// 0-based，支援重邊
struct EdgeBCC{
    int n, m, dep, sz;
    vector<vector<pair<int, int>>> G;
    vector<vector<int>> bcc;
    vector<int> dfn, low, stk, isBridge, bccId;
    vector<pair<int, int>> edge, bridge;

    EdgeBCC(int _n) : n(_n), m(0), sz(0), dfn(n), low(n), G(n
        ), bcc(n), bccId(n) {}

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

    void dfs(int now, int pre) {
        dfn[now] = low[now] = ++dep;
        stk.push_back(now);

        for (auto [x, id] : G[now]){
            if (!dfn[x]){
                dfs(x, id);
                low[now] = min(low[now], low[x]);
            }else if (id!=pre){
                low[now] = min(low[now], dfn[x]);
            }
        }

        if (low[now]==dfn[now]){
            if (pre!=-1) isBridge[pre] = true;
```

```
33          int u;
34          do{
35              u = stk.back();
36              stk.pop_back();
37              bcc[sz].push_back(u);
38              bccId[u] = sz;
39          } while (u!=now);
40          sz++;
41      }
42   }
43
44   void get_bcc() {
45       isBridge.assign(m, 0);
46       dep = 0;
47       for (int i=0 ; i<n ; i++){
48           if (!dfn[i]) dfs(i, -1);
49       }
50
51       for (int i=0 ; i<m ; i++){
52           if (isBridge[i]){
53               bridge.push_back({edge[i].first , edge[i].
                     second});
54           }
55       }
56   }
57 };
```

6.8   EnumeratePlanarFace [e70ee1]

```
1  // 0-based
2  struct PlanarGraph{
3      int n, m, id;
4      vector<point<int>> v;
5      vector<vector<pair<int, int>>> G;
6      vector<int> conv, nxt, vis;
7
8      PlanarGraph(int n, int m, vector<point<int>> _v) :
9      n(n), m(m), id(0),
10     v(_v), G(n),
11     conv(2*m), nxt(2*m), vis(2*m) {}
12
13     void add_edge(int x, int y){
14         G[x].push_back({y, 2*id});
15         G[y].push_back({x, 2*id+1});
16         conv[2*id] = x;
17         conv[2*id+1] = y;
18         id++;
19     }
20
21     vector<int> enumerate_face(){
22         for (int i=0 ; i<n ; i++){
23             sort(G[i].begin(), G[i].end(), [&](pair<int, int>
                   a, pair<int, int> b){
24                 return (v[a.first]-v[i])<(v[b.first]-v[i]);
25             });
26
27             int sz = G[i].size(), pre = sz-1;
28             for (int j=0 ; j<sz ; j++){
29                 nxt[G[i][pre].second] = G[i][j].second^1;
30                 pre = j;
31             }
32         }
33
```

```
34         vector<int> ret;
35         for (int i=0 ; i<2*m ; i++){
36             if (vis[i]==false){
37                 int area = 0, now = i;
38                 vector<int> pt;
39
40                 while (!vis[now]){
41                     vis[now] = true;
42                     pt.push_back(conv[now]);
43                     now = nxt[now];
44                 }
45
46                 pt.push_back(pt.front());
47                 for (int i=0 ; i+1<pt.size() ; i++){
48                     area -= (v[pt[i]]^v[pt[i+1]]);
49                 }
50
51                 // pt = face boundary
52                 if (area>0){
53                     ret.push_back(area);
54                 }else{
55                     // pt is outer face
56                 }
57             }
58         }
59         return ret;
60     }
61 };
```

6.9   HLD [f57ec6]

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

```
32         int res = 0;
33         while (topf[a] != topf[b]) { /// 若不在同一條錬上
34             if (depth[topf[a]] < depth[topf[b]]) swap(a, b);
35             res = max(res, 0ll); // query : l = id[topf[a]],
                    r = id[a]
36             a = pa[topf[a]];
37         }
38         /// 此時已在同一條錬上
39         if (depth[a] < depth[b]) swap(a, b);
40         res = max(res, 0ll); // query : l = id[b], r = id[a]
41         return res;
42     }
43 };
```

6.10   Kosaraju [c7d5aa]

```
1  /* c7d5aa
2  給定一個有向圖，迴回傳縮點後的圖、SCC 的資訊
3  所有點都以 based-0 編號
4
5  函式 :
6  SCC_compress G(n): 宣告一個有 n 個點的圖
7  .add_edge(u, v): 加上一條邊 u -> v
8  .compress: O(n log n) 計算 G3、SCC、SCC_id 的資訊，並把縮點後
       的結果存在 result 裡
9
10 SCC[i] = 某個 SCC 中的所有點
11 SCC_id[i] = 第 i 個點在第幾個 SCC
12 */
13 struct SCC_compress{
14     int N, M, sz;
15     vector<vector<int>> G, inv_G, result;
16     vector<pair<int, int>> edges;
17     vector<bool> vis;
18     vector<int> order;
19
20     vector<vector<int>> SCC;
21     vector<int> SCC_id;
22
23     SCC_compress(int _N) :
24     N(_N), M(0), sz(0),
25     G(N), inv_G(N),
26     vis(N), SCC_id(N)
27     {}
28
29     vector<int> operator [] (int x){
30         return result[x];
31     }
32
33     void add_edge(int u, int v){
34         G[u].push_back(v);
35         inv_G[v].push_back(u);
36         edges.push_back({u, v});
37         M++;
38     }
39
40     void dfs1(vector<vector<int>> &G, int now){
41         vis[now] = 1;
42         for (auto x : G[now]) if (!vis[x]) dfs1(G, x);
43         order.push_back(now);
44     }
```

```
45  void dfs2(vector<vector<int>> &G, int now){
46      SCC_id[now] = SCC.size()-1;
47      SCC.back().push_back(now);
48      vis[now] = 1;
49      for (auto x : G[now]) if (!vis[x]) dfs2(G, x);
50  }
51
52  void compress(){
53      fill(vis.begin(), vis.end(), 0);
54      for (int i=0 ; i<N ; i++) if (!vis[i]) dfs1(G, i);
55
56      fill(vis.begin(), vis.end(), 0);
57      reverse(order.begin(), order.end());
58      for (int i=0 ; i<N ; i++){
59          if (!vis[order[i]]){
60              SCC.push_back(vector<int>());
61              dfs2(inv_G, order[i]);
62          }
63      }
64
65      result.resize(SCC.size());
66      sz = SCC.size();
67      for (auto [u, v] : edges){
68          if (SCC_id[u]!=SCC_id[v]) result[SCC_id[u]].
69              push_back(SCC_id[v]);
70      }
71      for (int i=0 ; i<SCC.size() ; i++){
72          sort(result[i].begin(), result[i].end());
73          result[i].resize(unique(result[i].begin(), result
74              [i].end())-result[i].begin());
75      }
76  }
77  };
```

6.11 Kuhn Munkres [e66c35]

```
1   // O(n^3) 找到最大權匹配
2   struct KuhnMunkres{
3       int n; // max(n, m)
4       vector<vector<int>> G;
5       vector<int> match, lx, ly, visx, visy;
6       vector<int> slack;
7       int stamp = 0;
8
9       KuhnMunkres(int n) : n(n), G(n, vector<int>(n)), lx(n),
10          ly(n), slack(n), match(n), visx(n), visy(n) {}
11
12      void add(int x, int y, int w){
13          G[x][y] = max(G[x][y], w);
14      }
15
16      bool dfs(int i, bool aug){ // aug = true 表示要更新 match
17          if (visx[i]==stamp) return false;
18          visx[i] = stamp;
19
20          for (int j=0 ; j<n ; j++){
21              if (visy[j]==stamp) continue;
22              int d = lx[i]+ly[j]-G[i][j];
23
24              if (d==0){
25                  visy[j] = stamp;
26                  if (match[j]==-1 || dfs(match[j], aug)){
```

```
26              if (aug){
27                  match[j] = i;
28              }
29              return true;
30          }
31          }else{
32              slack[j] = min(slack[j], d);
33          }
34          }
35          return false;
36      }
37
38      bool augment(){
39          for (int j=0 ; j<n ; j++){
40              if (visy[j]!=stamp && slack[j]==0){
41                  visy[j] = stamp;
42                  if (match[j]==-1 || dfs(match[j], false)){
43                      return true;
44                  }
45              }
46          }
47          return false;
48      }
49
50      void relabel(){
51          int delta = INF;
52          for (int j=0 ; j<n ; j++){
53              if (visy[j]!=stamp) delta = min(delta, slack[j]);
54          }
55          for (int i=0 ; i<n ; i++){
56              if (visx[i]==stamp) lx[i] -= delta;
57          }
58          for (int j=0 ; j<n ; j++){
59              if (visy[j]==stamp) ly[j] += delta;
60              else slack[j] -= delta;
61          }
62      }
63
64      int solve(){
65
66          for (int i=0 ; i<n ; i++){
67              lx[i] = 0;
68              for (int j=0 ; j<n ; j++){
69                  lx[i] = max(lx[i], G[i][j]);
70              }
71          }
72
73          fill(ly.begin(), ly.end(), 0);
74          fill(match.begin(), match.end(), -1);
75
76          for(int i = 0; i < n; i++) {
77              fill(slack.begin(), slack.end(), INF);
78              stamp++;
79              if(dfs(i, true)) continue;
80
81              while(augment()==false) relabel();
82              stamp++;
83              dfs(i, true);
84          }
85
86          int ans = 0;
87          for (int j=0 ; j<n ; j++){
88              if (match[j]!=-1){
89                  ans += G[match[j]][j];
90              }
91          }
```

```
92          return ans;
93      }
94  };
```

6.12 LCA [5b6a5b]

```
1   // 1-based，可以支援森林，0 是超級源點，所有樹都要跟他建邊
2   struct Tree{
3       int N, M = 0, H;
4       vector<int> parent, dep;
5       vector<vector<int>> G, LCA;
6
7       Tree(int _N) : N(_N+1), H(__lg(N)+1), parent(N, -1), dep(
8           N), G(N){
9           LCA.resize(H, vector<int>(N, 0));
10      }
11
12      void add_edge(int u, int v){
13          M++;
14          G[u].push_back(v);
15          G[v].push_back(u);
16      }
17
18      void dfs(int now = 0, int pre = 0){
19          dep[now] = dep[pre]+1;
20          parent[now] = pre;
21          for (auto x : G[now]){
22              if (x==pre) continue;
23              dfs(x, now);
24          }
25      }
26
27      void build_LCA(int root = 0){
28          dfs();
29          for (int i=0 ; i<N ; i++) LCA[0][i] = parent[i];
30          for (int i=1 ; i<H ; i++){
31              for (int j=0 ; j<N ; j++){
32                  LCA[i][j] = LCA[i-1][LCA[i-1][j]];
33              }
34          }
35      }
36
37      int jump(int u, int step){
38          for (int i=0 ; i<H ; i++){
39              if (step&(1<<i)) u = LCA[i][u];
40          }
41          return u;
42      }
43
44      int get_LCA(int u, int v){
45          if (dep[u]<dep[v]) swap(u, v);
46          u = jump(u, dep[u]-dep[v]);
47          if (u==v) return u;
48          for (int i=H-1 ; i>=0 ; i--){
49              if (LCA[i][u]!=LCA[i][v]){
50                  u = LCA[i][u];
51                  v = LCA[i][v];
52              }
53          }
54          return parent[u];
55      }
56  };
```

6.13   MCMF [1b5a27]

```cpp
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, G[u].size()});
    G[u].push_back({v, 0, -k, G[v].size()-1});
  }
  // 3701d6
  int spfa(int s, int t){
    fill(par.begin(), par.end(), -1);
    vector<int> dis(par.size(), INF);
    vector<bool> in_q(par.size(), false);
    queue<int> Q;
    dis[s] = 0;
    in_q[s] = true;
    Q.push(s);

    while (!Q.empty()){
      int v = Q.front();
      Q.pop();
      in_q[v] = false;

      for (int i=0 ; i<G[v].size() ; i++){
        auto [u, rc, k, rv] = G[v][i];
        if (rc>0 && dis[v]+k<dis[u]){
          dis[u] = dis[v]+k;
          par[u] = v;
          par_eid[u] = i;
          if (!in_q[u]) Q.push(u);
          in_q[u] = true;
        }
      }
    }

    return dis[t];
  }

  // 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 [8b2350]

```cpp
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);
```

```cpp
            now_SCCs += 1;
        }
    }
};
```

6.15   Tarjan Find AP [1daed6]

```cpp
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 [cd2bbc]

```cpp
#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;
```

```
24  void centroid(Graph &g, vector<int> &s, vector<int> w, pair<
25       int, int> &rec, int now, int pre){
26      s[now]=1;
27      w[now]=0;
28      for (auto x : g[now]){
29          if (x!=pre){
30              centroid(g, s, w, rec, x, now);
31              s[now]+=s[x];
32              w[now]=max(w[now], s[x]);
33          }
34      }
35
36      w[now]=max(w[now], n-s[now]);
37      if (w[now]<=n/2){
38          if (rec.first==0) rec.first=now;
39          else rec.second=now;
40      }
41  }
42
43  int dfs(Graph &g, Hash &m, int &id, int now, int pre){
44      vector<int> v;
45      for (auto x : g[now]){
46          if (x!=pre){
47              int add=dfs(g, m, id, x, now);
48              v.push_back(add);
49          }
50      }
51      sort(v.begin(), v.end());
52
53      if (m.find(v)!=m.end()){
54          return m[v];
55      }else{
56          m[v]=++id;
57          return id;
58      }
59  }
60
61
62  void solve1(){
63
64      // init
65      id1=0;
66      id2=0;
67      c1={0, 0};
68      c2={0, 0};
69      fill(sz1.begin(), sz1.begin()+n+1, 0);
70      fill(sz2.begin(), sz2.begin()+n+1, 0);
71      fill(we1.begin(), we1.begin()+n+1, 0);
72      fill(we2.begin(), we2.begin()+n+1, 0);
73      for (int i=1 ; i<=n ; i++){
74          g1[i].clear();
75          g2[i].clear();
76      }
77      m1.clear();
78      m2.clear();
79
80      // input
81      cin >> n;
82      for (int i=0 ; i<n-1 ; i++){
83          cin >> a >> b;
84          g1[a].push_back(b);
85          g1[b].push_back(a);
86      }
87      for (int i=0 ; i<n-1 ; i++){
88          cin >> a >> b;
```

```
89          g2[a].push_back(b);
90          g2[b].push_back(a);
91      }
92
93      // get tree centroid
94      centroid(g1, sz1, we1, c1, 1, 0);
95      centroid(g2, sz2, we2, c2, 1, 0);
96
97      // process
98      int res1=0, res2=0, res3=0;
99      if (c2.second!=0){
100         res1=dfs(g1, m1, id1, c1.first, 0);
101         m2=m1;
102         id2=id1;
103         res2=dfs(g2, m1, id1, c2.first, 0);
104         res3=dfs(g2, m2, id2, c2.second, 0);
105     }else if (c1.second!=0){
106         res1=dfs(g2, m1, id1, c2.first, 0);
107         m2=m1;
108         id2=id1;
109         res2=dfs(g1, m1, id1, c1.first, 0);
110         res3=dfs(g1, m2, id2, c1.second, 0);
111     }else{
112         res1=dfs(g1, m1, id1, c1.first, 0);
113         res2=dfs(g2, m1, id1, c2.first, 0);
114     }
115
116     // output
117     cout << (res1==res2 || res1==res3 ? "YES" : "NO") << endl
                ;
118
119     return;
120 }
121
122 signed main(void){
123     fastio;
124
125     int t=1;
126     cin >> t;
127     while (t--){
128         solve1();
129     }
130     return 0;
131 }
```

6.17   圓方樹 [675aec]

```
1   #include <bits/stdc++.h>
2   #define lp(i,a,b) for(int i=(a);i<(b);i++)
3   #define pii pair<int,int>
4   #define pb push_back
5   #define ins insert
6   #define ff first
7   #define ss second
8   #define opa(x) cerr << #x << " = " << x << ", ";
9   #define op(x) cerr << #x << " = " << x << endl;
10  #define ops(x) cerr << x;
11  #define etr cerr << endl;
12  #define spc cerr << ' ';
13  #define BAE(x) (x).begin(), (x).end()
14  #define STL(x) cerr << #x << " : "; for(auto &qwe:x) cerr <<
        qwe << ' '; cerr << endl;
15  #define deb1 cerr << "deb1" << endl;
```

```
16  #define deb2 cerr << "deb2" << endl;
17  #define deb3 cerr << "deb3" << endl;
18  #define deb4 cerr << "deb4" << endl;
19  #define deb5 cerr << "deb5" << endl;
20  #define bye exit(0);
21  using namespace std;
22
23  const int mxn = (int)(2e5) + 10;
24  const int mxlg = 17;
25  int last_special_node = (int)(1e5) + 1;
26  vector<int> E[mxn], F[mxn];
27
28  struct edg{
29      int fr, to;
30      edg(int _fr, int _to){
31          fr = _fr;
32          to = _to;
33      }
34  };
35  ostream& operator<<(ostream& os, edg x){os << x.fr << "--" <<
        x.to;}
36  vector<edg> EV;
37
38  void tarjan(int v, int par, stack<int>& S){
39      static vector<int> dfn(mxn), low(mxn);
40      static vector<bool> to_add(mxn);
41      static int nowT = 0;
42
43      int childs = 0;
44      nowT += 1;
45      dfn[v] = low[v] = nowT;
46      for(auto &ne:E[v]){
47          int i = EV[ne].to;
48          if(i == par) continue;
49          if(!dfn[i]){
50              S.push(ne);
51              tarjan(i, v, S);
52              childs += 1;
53              low[v] = min(low[v], low[i]);
54
55              if(par >= 0 && low[i] >= dfn[v]){
56                  vector<int> bcc;
57                  int tmp;
58                  do{
59                      tmp = S.top(); S.pop();
60                      if(!to_add[EV[tmp].fr]){
61                          to_add[EV[tmp].fr] = true;
62                          bcc.pb(EV[tmp].fr);
63                      }
64                      if(!to_add[EV[tmp].to]){
65                          to_add[EV[tmp].to] = true;
66                          bcc.pb(EV[tmp].to);
67                      }
68                  }while(tmp != ne);
69                  for(auto &j:bcc){
70                      to_add[j] = false;
71                      F[last_special_node].pb(j);
72                      F[j].pb(last_special_node);
73                  }
74                  last_special_node += 1;
75              }
76          }
77      }else{
78          low[v] = min(low[v], dfn[i]);
79          if(dfn[i] < dfn[v]){ // edge i--v will be visited
                 twice at here, but we only need one.
```

```
80              S.push(ne);
81          }
82        }
83      }
84 }
85
86 int dep[mxn], jmp[mxn][mxlg];
87 void dfs_lca(int v, int par, int depth){
88      dep[v] = depth;
89      for(auto &i:F[v]){
90          if(i == par) continue;
91          jmp[i][0] = v;
92          dfs_lca(i, v, depth + 1);
93      }
94 }
95
96 inline void build_lca(){
97      jmp[1][0] = 1;
98      dfs_lca(1, -1, 1);
99      lp(j,1,mxlg){
100         lp(i,1,mxn){
101             jmp[i][j] = jmp[jmp[i][j-1]][j-1];
102         }
103     }
104 }
105
106 inline int lca(int x, int y){
107     if(dep[x] < dep[y]){ swap(x, y); }
108
109     int diff = dep[x] - dep[y];
110     lp(j,0,mxlg){
111         if((diff >> j) & 1){
112             x = jmp[x][j];
113         }
114     }
115     if(x == y) return x;
116
117     for(int j = mxlg - 1; j >= 0; j--){
118         if(jmp[x][j] != jmp[y][j]){
119             x = jmp[x][j];
120             y = jmp[y][j];
121         }
122     }
123     return jmp[x][0];
124 }
125
126 inline bool can_reach(int fr, int to){
127     if(dep[to] > dep[fr]) return false;
128
129     int diff = dep[fr] - dep[to];
130     lp(j,0,mxlg){
131         if((diff >> j) & 1){
132             fr = jmp[fr][j];
133         }
134     }
135     return fr == to;
136 }
137
138 int main(){
139     ios::sync_with_stdio(false); cin.tie(0);
140 //    freopen("test_input.txt", "r", stdin);
141     int n, m, q; cin >> n >> m >> q;
142     lp(i,0,m){
143         int u, v; cin >> u >> v;
144         E[u].pb(EV.size());
145         EV.pb(edg(u, v));
```

```
146         E[v].pb(EV.size());
147         EV.pb(edg(v, u));
148     }
149     E[0].pb(EV.size());
150     EV.pb(edg(0, 1));
151     stack<int> S;
152     tarjan(0, -1, S);
153     build_lca();
154
155     lp(queries,0,q){
156         int fr, to, relay; cin >> fr >> to >> relay;
157         if(fr == relay || to == relay){
158             cout << "NO\n";
159             continue;
160         }
161         if((can_reach(fr, relay) || can_reach(to, relay)) &&
162             dep[relay] >= dep[lca(fr, to)]){
163             cout << "NO\n";
164             continue;
165         }
166         cout << "YES\n";
167     }
168 }
```

6.18 最大權閉合圖 [6ca663]

```
1  /*
2      邊 u → v 表示選 u 就要選 v (0-based)
3      保證回傳值非負
4      構造: 從 S 開始 dfs, 不走最小割的邊,
5            所有經過的點就是要選的那些點。
6      一般圖: O(n²m) / 二分圖: O(m√n)
7  */
8  template<typename U>
9  U maximum_closure(vector<U> w, vector<pair<int,int>> EV) {
10     int n = w.size(), S = n + 1, T = n + 2;
11     Flow G(T + 5); // Graph/Dinic.cpp
12     U sum = 0;
13     for (int i = 0; i < n; ++i) {
14         if (w[i] > 0) {
15             G.add(S, i, w[i]);
16             sum += w[i];
17         }
18         else if (w[i] < 0) {
19             G.add(i, T, abs(w[i]));
20         }
21     }
22     for (auto &[u, v] : EV) { // 請務必確保 INF > Σ|w_i|
23         G.add(u, v, INF);
24     }
25     U cut = G.flow(S, T);
26     return sum - cut;
27 }
```

6.19 Theorem

- 任意圖

  - 最大匹配 + 最小邊覆蓋 = n (不能有孤點)

  - 點覆蓋的補集是獨立集,最小點覆蓋 + 最大獨立集 = n

  - $w($ 最小權點覆蓋 $) + w($ 最大權獨立集 $) = \sum w_v$

  - (帶點權的二分圖可以用最小割解,構造請參考 Augment Path.cpp)

- 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 $\sum w_e \le b_v$。

  - 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 [2b18f3]

```
1  // ax + by = c
2  int extgcd_abc(int a, int b, int c, int &x, int &y) {
3      if (b == 0) {
4          if (c % a) return INF;
5          x = c / a, y = 0;
6          return abs(a);
7      }
8      int x1, y1;
9      int g = extgcd_abc(b, a % b, c, x1, y1);
10     x = y1;
11     y = x1 - a / b * y1;
12     return g;
13 }
14 // Sorry that I don't know how to merge two functions
15 int extgcd(int a, int b, int &x, int &y) {
16     if (b == 0) {
17         x = 1, y = 0;
18         return a;
19     }
20     int ret = extgcd(b, a % b, y, x);
21     y -= a / b * x;
22     return ret;
23 }
24
25 // 有 n 個式子,求解 x ≡ a_i (mod m_i)
26 int CRT_m_coprime(int n, vector<int> &a, vector<int> &m) {
27     int p = 1, ans = 0;
28     vector<int> M(n), inv_M(n);
29
30     for (int i = 0; i < n; i++) p *= m[i];
31     for (int i = 0; i < n; i++) {
32         M[i] = p / m[i];
```

```
33        int tmp;
34        extgcd(M[i], m[i], inv_M[i], tmp);
35        ans += a[i] * inv_M[i] * M[i];
36        ans %= p;
37    }
38    return (ans % p + p) % p;
39 }
40
41 // 對於方程組的式子兩兩求解
42 // 回傳：{是否有解, {a, m}}
43 pair<bool, pair<int, int>> CRT_m_NOT_coprime(int a1, int m1,
        int a2, int m2) {
44    int g = __gcd(m1, m2);
45    if ((a2 - a1) % g != 0) return {0, {-1, -1}};
46
47    int x, y; extgcd(m1, m2, x, y);
48
49    x = (a2 - a1) * x / g; // 兩者不能相反
50    a1 = x * m1 + a1;
51    m1 = m1 * m2 / g;
52    a1 = (a1 % m1 + m1) % m1;
53    return {1, {a1, m1}};
54 }
55
56 // ans = a / b (mod m)
57 // ans = ret.F + k * ret.S, k is integer
58 pair<int, int> div(int a, int b, int m) {
59    int flag = 1;
60    if (a < 0) { a = -a; flag *= -1; }
61    if (b < 0) { b = -b; flag *= -1; }
62    int t = -1, k = -1;
63    int res = extgcd_abc(b, m, a, t, k);
64    if (res == INF) return {INF, INF};
65    m = abs(m / res);
66    t = t * flag;
67    t = (t % m + m) % m;
68    return {t, m};
69 }
```

7.2　Josephus Problem [e0ed50]

```
1 // 有 n 個人，第偶數個報數的人被刪掉，問第 k 個被踢掉的是誰
2 int solve(int n, int k){
3    if (n==1) return 1;
4    if (k<=(n+1)/2){
5        if (2*k>n) return 2*k%n;
6        else return 2*k;
7    }else{
8        int res=solve(n/2, k-(n+1)/2);
9        if (n&1) return 2*res+1;
10        else return 2*res-1;
11    }
12 }
```

7.3　Lagrange any x [1f2c26]

```
1 // init: (x1, y1), (x2, y2) in a vector
2 struct Lagrange{
3    int n;
```

```
4    vector<pair<int, int>> v;
5
6    Lagrange(vector<pair<int, int>> &_v){
7        n = _v.size();
8        v = _v;
9    }
10
11    // O(n^2 log MAX_A)
12    int solve(int x){
13        int ret = 0;
14        for (int i=0 ; i<n ; i++){
15            int now = v[i].second;
16            for (int j=0 ; j<n ; j++){
17                if (i==j) continue;
18                now *= ((x-v[j].first)+MOD)%MOD;
19                now %= MOD;
20                now *= (qp((v[i].first-v[j].first+MOD)%MOD,
                        MOD-2)+MOD)%MOD;
21                now %= MOD;
22            }
23
24            ret = (ret+now)%MOD;
25        }
26        return ret;
27    }
28 };
```

7.4　Lagrange continuous x [57536a]

```
1 #include <bits/stdc++.h>
2 using namespace std;
3
4 const int MAX_N = 5e5 + 10;
5 const int mod = 1e9 + 7;
6
7 long long inv_fac[MAX_N];
8
9 inline int fp(long long x, int y) {
10    int ret = 1;
11    for (; y; y >>= 1) {
12        ret = (y & 1) ? (ret * x % mod) : ret;
13        x = x * x % mod;
14    }
15    return ret;
16 }
17
18 // TO USE THIS TEMPLATE, YOU MUST MAKE SURE THAT THE MOD
        NUMBER IS A PRIME.
19 struct Lagrange {
20 /*
21    Initialize a polynomial with f(x_0), f(x_0 + 1), ..., f(
        x_0 + n).
22    This determines a polynomial f(x) whose degree is at most
        n.
23    Then you can call sample(x) and you get the value of f(x)
        .
24    Complexity of init() and sample() are both O(n).
25 */
26    int m, shift; // m = n + 1
27    vector<int> v, mul;
28 // You can use this function if you don't have inv_fac array
        already.
29    void construct_inv_fac() {
```

```
30        long long fac = 1;
31        for (int i = 2; i < MAX_N; ++i) {
32            fac = fac * i % mod;
33        }
34        inv_fac[MAX_N - 1] = fp(fac, mod - 2);
35        for (int i = MAX_N - 1; i >= 1; --i) {
36            inv_fac[i - 1] = inv_fac[i] * i % mod;
37        }
38    }
39 // You call init() many times without having a second
        instance of this struct.
40    void init(int X_0, vector<int> &u) {
41        v = u;
42        shift = ((1 - X_0) % mod + mod) % mod;
43        if (v.size() == 1) v.push_back(v[0]);
44        m = v.size();
45        mul.resize(m);
46    }
47 // You can use sample(x) instead of sample(x % mod).
48    int sample(int x) {
49        x = ((long long)x + shift) % mod;
50        x = (x < 0) ? (x + mod) : x;
51        long long now = 1;
52        for (int i = m; i >= 1; --i) {
53            mul[i - 1] = now;
54            now = now * (x - i) % mod;
55        }
56        int ret = 0;
57        bool neg = (m - 1) & 1;
58        now = 1;
59        for (int i = 1; i <= m; ++i) {
60            int up = now * mul[i - 1] % mod;
61            int down = inv_fac[m - i] * inv_fac[i - 1] % mod;
62            int tmp = ((long long)v[i - 1] * up % mod) * down
                % mod;
63            ret += (neg && tmp) ? (mod - tmp) : (tmp);
64            ret = (ret >= mod) ? (ret - mod) : ret;
65            now = now * (x - i) % mod;
66            neg ^= 1;
67        }
68        return ret;
69    }
70 };
71
72 int main() {
73    int n; cin >> n;
74    vector<int> v(n);
75    for (int i = 0; i < n; ++i) {
76        cin >> v[i];
77    }
78    Lagrange L;
79    L.construct_inv_fac();
80    L.init(0, v);
81    int x; cin >> x;
82    cout << L.sample(x);
83 }
```

7.5　Linear Mod Inverse [ecf71e]

```
1 // 線性求 1-based a[i] 對 p 的乘法反元素
2 vector<int> s(n+1, 1), invS(n+1), invA(n+1);
3 for (int i=1 ; i<=n ; i++) s[i] = s[i-1]*a[i]%p;
4 invS[n] = qp(s[n], p-2, p);
```

```
5  for (int i=n ; i>=1 ; i--) invS[i-1] = invS[i]*a[i]%p;
6  for (int i=1 ; i<=n ; i++) invA[i] = invS[i]*s[i-1]%p;
```

7.6　Lucas's Theorem [b37dcf]

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

7.7　Matrix [8d1a23]

```
1   struct Matrix{
2       int n, m;
3       vector<vector<int>> arr;
4
5       Matrix(int _n, int _m){
6           n = _n;
7           m = _m;
8           arr.assign(n, vector<int>(m));
9       }
10
11      vector<int> & operator [] (int i){
12          return arr[i];
13      }
14
15      Matrix operator * (Matrix b){
16          Matrix ret(n, b.m);
17          for (int i=0 ; i<n ; i++){
18              for (int j=0 ; j<b.m ; j++){
19                  for (int k=0 ; k<m ; k++){
20                      ret.arr[i][j] += arr[i][k]*b.arr[k][j]%
                           MOD;
21                      ret.arr[i][j] %= MOD;
22                  }
23              }
24          }
25          return ret;
26      }
27
28      Matrix pow(int p){
29          Matrix ret(n, n), mul = *this;
30          for (int i=0 ; i<n ; i++){
31              ret.arr[i][i] = 1;
32          }
33
34          for ( ; p ; p>>=1){
35              if (p&1) ret = ret*mul;
36              mul = mul*mul;
37          }
38
39          return ret;
40      }
41
42
43      int det(){
44          vector<vector<int>> arr = this->arr;
45          bool flag = false;
46          for (int i=0 ; i<n ; i++){
```

```
47          int target = -1;
48          for (int j=i ; j<n ; j++){
49              if (arr[j][i]){
50                  target = j;
51                  break;
52              }
53          }
54          if (target==-1) return 0;
55          if (i!=target){
56              swap(arr[i], arr[target]);
57              flag = !flag;
58          }
59
60          for (int j=i+1 ; j<n ; j++){
61              if (!arr[j][i]) continue;
62              int freq = arr[j][i]*qp(arr[i][i], MOD-2)%MOD
                   ;
63              for (int k=i ; k<n ; k++){
64                  arr[j][k] -= freq*arr[i][k];
65                  arr[j][k] = (arr[j][k]%MOD+MOD)%MOD;
66              }
67          }
68      }
69
70      int ret = !flag ? 1 : MOD-1;
71      for (int i=0 ; i<n ; i++){
72          ret *= arr[i][i];
73          ret %= MOD;
74      }
75      return ret;
76  }
77  };
```

7.8　Matrix 01 [8d542a]

```
1   const int MAX_N = (1LL<<12);
2   struct Matrix{
3       int n, m;
4       vector<bitset<MAX_N>> arr;
5
6       Matrix(int _n, int _m){
7           n = _n;
8           m = _m;
9           arr.resize(n);
10      }
11
12      Matrix operator * (Matrix b){
13          Matrix b_t(b.m, b.n);
14          for (int i=0 ; i<b.n ; i++){
15              for (int j=0 ; j<b.m ; j++){
16                  b_t.arr[j][i] = b.arr[i][j];
17              }
18          }
19
20          Matrix ret(n, b.m);
21          for (int i=0 ; i<n ; i++){
22              for (int j=0 ; j<b.m ; j++){
23                  ret.arr[i][j] = ((arr[i]&b_t.arr[j]).count()
                       &1);
24              }
25          }
26          return ret;
27      }
```

```
28  };
```

7.9　Miller Rabin [24bd0d]

```
1   // O(k log^3 n), k = llsprp.size()
2   typedef Uint unsigned long long;
3   Uint modmul(Uint a, Uint b, Uint m) {
4       int ret = a*b - m*(Uint)((long double)a*b/m);
5       return ret + m*(ret < 0) - m*(ret>=(int)m);
6   }
7
8   int qp(int b, int p, int m){
9       int ret = 1;
10      for ( ; p ; p>>=1){
11          if (p&1) ret = modmul(ret, b, m);
12          b = modmul(b, b, m);
13      }
14      return ret;
15  }
16
17  // ed23aa
18  vector<int> llsprp = {2, 325, 9375, 28178, 450775, 9780504,
         1795265022};
19  bool is_prime(int n, vector<int> sprp = llsprp){
20      if (n==2) return 1;
21      if (n<2 || n%2==0) return 0;
22
23      int t = 0;
24      int u = n-1;
25      for ( ; u%2==0 ; t++) u>>=1;
26
27      for (int i=0 ; i<sprp.size() ; i++){
28          int a = sprp[i]%n;
29          if (a==0 || a==1 || a==n-1) continue;
30          int x = qp(a, u, n);
31          if (x==1 || x==n-1) continue;
32          for (int j=0 ; j<t ; j++){
33              x = modmul(x, x, n);
34              if (x==1) return 0;
35              if (x==n-1) break;
36          }
37
38          if (x==n-1) continue;
39          return false;
40      }
41
42      return true;
43  }
```

7.10　Pollard Rho [a5daef]

```
1   mt19937 seed(chrono::steady_clock::now().time_since_epoch().
         count());
2   int rnd(int l, int r){
3       return uniform_int_distribution<int>(l, r)(seed);
4   }
5
6   // O(n^{1/4}) 回傳 1 或自己的因數,記得先判斷 n 是不是質數
       (用 Miller-Rabin )
```

```cpp
// 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 [51ca3b]

```cpp
struct Poly {
    int len, deg;
    int *a;
    // len = 2^k >= the original length
    Poly(): len(0), deg(0), a(nullptr) {}
    Poly(int _n) {
        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) {
```

```cpp
        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;
```

```cpp
        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)
```

```
168          b.a[j] = a.a[j + i] * inv1 % mod;
169        Poly res1 = (b.ln() * (n % mod)).exp() * (::power(a.a
              [i], n));
170        Poly res2(a.deg + 1);
171        FOR (j, 0, min((ll)(res1.deg), (ll)(a.deg - n * i)))
172          res2.a[j + n * i] = res1.a[j];
173        return res2;
174      }
175  };
```

7.12 josephus [0be067]

```
1  // n 個人，每 k 個人就刪除的約瑟夫遊戲
2  int josephus(int n, int k) {
3      if (n == 1)
4          return 0;
5      if (k == 1)
6          return n-1;
7      if (k > n)
8          return (josephus(n-1, k) + k) % n;
9      int cnt = n / k;
10     int res = josephus(n - cnt, k);
11     res -= n % k;
12     if (res < 0)
13         res += n;
14     else
15         res += res / (k - 1);
16     return res;
17 }
```

7.13 數論分塊 [8ccab5]

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

7.14 最大質因數 [ca5e52]

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

7.15 歐拉公式 [85f3b1]

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

7.16 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

7.17 Catalan Number

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

7.18 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

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

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

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

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

7.19 Stirling's formula

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

7.20 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

7.21 二元一次方程式

$\begin{cases} ax + by = e \\ cx + dy = f \end{cases} = \begin{cases} x = \frac{ed-bf}{ad-bc} \\ y = \frac{af-ec}{ad-bc} \end{cases}$
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。

7.22 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：
$$a^n \equiv a^{\varphi(m)+[n \bmod \varphi(m)]} \pmod{m}$$

7.23 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   AC automation [6ece7f]

```cpp
const int MAXN = 5e5 + 5;
struct ac_automation {

    int go[MAXN][26], fail[MAXN], is_end[MAXN];
    int sz;

    void add(string s) {
        int now = 0;
        for (char c : s) {
            if (!go[now][c - 'a'])
                go[now][c - 'a'] = ++sz;
            now = go[now][c - 'a'];
        }
        is_end[now]++;
    }
    vector<int> que;
    void build() {
        que.pb(0);
        for (int i = 0; i < ssize(que); i++) {
            auto u = que[i];
            FOR (c, 0, 25) {
                if (go[u][c]) {
                    int v = go[u][c];
                    fail[v] = !u ? 0 : go[fail[u]][c];
                    is_end[v] += is_end[fail[v]];
                    que.pb(v);
                }
                else {
                    go[u][c] = go[fail[u]][c];
                }
            }
        }
    }
} AC;
```

8.2   Hash [942f42]

```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;
```

```cpp
                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;
    }
};
```

8.3   KMP [e5b7ce]

```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;
}
```

8.4   Manacher [9a4b4d]

```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);
}
```

8.5   Min Rotation [9d296f]

```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;
}
```

8.6   Suffix Array [6352b3]

```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());
```

```cpp
42     for (int i=0 ; i<sa.size() ; i++){
43         pos[sa[i]] = i;
44     }
45     if (lcp.size()){
46         st.build(lcp);
47     }
48 }
49
50 // 用之前記得 init
51 // 回傳 [l1, r1] 跟 [l2, r2] 的 lcp，0-based
52 int get_lcp(int l1, int r1, int l2, int r2){
53     int pos_1 = pos[l1], len_1 = r1-l1+1;
54     int pos_2 = pos[l2], len_2 = r2-l2+1;
55     if (pos_1>pos_2){
56         swap(pos_1, pos_2);
57         swap(len_1, len_2);
58     }
59
60     if (l1==l2){
61         return min(len_1, len_2);
62     }else{
63         return min({st.query(pos_1, pos_2), len_1, len_2
                });
64     }
65 }
66
67 // 檢查 [l1, r1] 跟 [l2, r2] 的大小關係，0-based
68 // 如果前者小於後者，就回傳 <0，相等就回傳 =0，否則回傳
        >0
69 // 5b8db0
70 int substring_cmp(int l1, int r1, int l2, int r2){
71     int len_1 = r1-l1+1;
72     int len_2 = r2-l2+1;
73     int res = get_lcp(l1, r1, l2, r2);
74
75     if (res<len_1 && res<len_2){
76         return s[l1+res]-s[l2+res];
77     }else if (len_1==res && len_2==res){
78         // 如果不需要以 index 作為次要排序參數，這裡要回
                傳 0
79         return l1-l2;
80     }else{
81         return len_1==res ? -1 : 1;
82     }
83 }
84
85 // 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
86 // pre[i] = s[i] 離他左邊最接近位置 >p 的後綴的 lcp，0-
        based
87 // suf[i] = s[i] 離他右邊最接近位置 >p 的後綴的 lcp，0-
        based
88 // da12fa
89 pair<vector<int>, vector<int>> get_left_and_right_lcp(int
        p){
90     vector<int> pre(p+1);
91     vector<int> suf(p+1);
92
93     { // build pre
94         int now = 0;
95         for (int i=0 ; i<s.size() ; i++){
96             if (sa[i]<=p){
97                 pre[sa[i]] = now;
98                 if (i<lcp.size()) now = min(now, lcp[i]);
99             }else{
100                if (i<lcp.size()) now = lcp[i];
101            }
102        }
103    }
104    { // build suf
105        int now = 0;
106        for (int i=s.size()-1 ; i>=0 ; i--){
107            if (sa[i]<=p){
108                suf[sa[i]] = now;
109                if (i-1>=0) now = min(now, lcp[i-1]);
110            }else{
111                if (i-1>=0) now = lcp[i-1];
112            }
113        }
114    }
115
116    return {pre, suf};
117 }
118 };
```

8.7 Z Algorithm [bcfbd6]

```cpp
1 // 定義一個長度為 n 的文本為 T，則陣列 Z 的 Z[i] 代表 T[0:n]
        和 T[i:n] 最長共同前綴
2 // bcfbd6
3 vector<int> z_function(string s){
4     vector<int> ret(s.size());
5     int ll = 0, rr = 0;
6
7     for (int i=1 ; i<s.size() ; i++){
8         int j = 0;
9
10        if (i<rr) j = min(ret[i-ll], rr-i);
11        while (s[j]==s[i+j]) j++;
12        ret[i] = j;
13
14        if (i+j>rr){
15            ll = i;
16            rr = i+j;
17        }
18    }
19
20    ret[0] = s.size();
21    return ret;
22 }
```

8.8 k-th Substring1 [61f66b]

```cpp
1 // 回傳 s 所有子字串（完全不同）中，第 k 大的
2 string k_th_substring(string &s, int k){
3     int n = s.size();
4     SuffixArray sa(s);
5     sa.init_lcp();
6
7     int prePrefix = 0, nowRank = 0;
8     for (int i=0 ; i<n ; i++){
9         int len = n-sa[i];
10        int add = len-prePrefix;
```

```cpp
11        if (nowRank+add>=k){
12            return s.substr(sa[i], prePrefix+k-nowRank);
13        }
14
15        prePrefix = sa.lcp[i];
16        nowRank += add;
17    }
18 }
```