Contents

1 Misc

1.1 Default Code [24a798]

```cpp
#include <bits/stdc++.h>
using namespace std;
#define int long long
#define debug(a...) cerr << #a << " = ", dout(a)
void dout() { cerr << "\n"; }
template <typename A, typename... B>
void dout(A a, B... b) { cerr << a << ' ', dout(b...); }

void solve(){

}

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

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

    return 0;
}
```

1.2 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()
```

1.3 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;
};
```

```cpp
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);
```

1.4 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;
}
```

1.5 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;
    }
}
```

1.6 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
}
```

1.7 OEIS [ec45dc]

```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) {
```

```cpp
    auto scalarProduct = [](vector<T> v, T c) {
        for (T &x: v) x *= c;
        return v;
    };
    vector<T> s, best;
    int bestPos = 0;
    for (int i=0 ; i<a.size() ; i++){
        T error = a[i];
        for (int j=0 ; j<s.size() ; j++) error -= s[j] * a[i
            -1-j];
        if (error == 0) continue;
        if (s.empty()) {
            s.resize(i + 1);
            bestPos = i;
            best.push_back(1 / error);
            continue;
        }
        vector<T> fix = scalarProduct(best, error);
        fix.insert(fix.begin(), i - bestPos - 1, 0);
        if (fix.size() >= s.size()) {
            best = scalarProduct(s, - 1 / error);
            best.insert(best.begin(), 1 / error);
            bestPos = i;
            s.resize(fix.size());
        }
        for (int j = 0; j < fix.size(); j++) s[j] += fix[j];
    }
    reverse(s.begin(), s.end());
    return s;
}
```

1.8 Pragma [09d13e]

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

1.9 Xor Basis [840136]

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

// 給一數字集合 S，求能不能 XOR 出 x
bool check(int x){
    for (auto v : basis){
        x=min(x, x^v);
    }
    return 0;
}

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

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

```
27 | }
```

1.10 random int [9cc603]

```cpp
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 | }
```

1.11 OEIS

```python
1 | from fractions import Fraction
2 |
3 | def BerlekampMassey(a: list[Fraction]) -> list[Fraction]:
4 |     def scale(v: list[Fraction], c: Fraction) -> list[
         Fraction]:
5 |         return [x * c for x in v]
6 |
7 |     s: list[Fraction] = []
8 |     best: list[Fraction] = []
9 |     bestPos = 0
10 |
11 |    for i in range(len(a)):
12 |        error: Fraction = a[i]
13 |        for j in range(len(s)):
14 |            error -= s[j] * a[i - 1 - j]
15 |        if error == 0:
16 |            continue
17 |
18 |        if not s:
19 |            s = [Fraction(0)] * (i + 1)
20 |            bestPos = i
21 |            best = [Fraction(1, error)]
22 |            continue
23 |
24 |        fix = scale(best, error)
25 |        fix = [Fraction(0)] * (i - bestPos - 1) + fix
26 |
27 |        if len(fix) >= len(s):
28 |            best = scale(s, Fraction(-1, error))
29 |            best.insert(0, Fraction(1, error))
30 |            bestPos = i
31 |            if len(s) < len(fix):
32 |                s += [Fraction(0)] * (len(fix) - len(s))
33 |
34 |        for j in range(len(fix)):
35 |            s[j] += fix[j]
36 |
37 |    return list(reversed(s))
38 |
39 | n = int(input())
40 | l = list(map(Fraction, input().split()))
41 | for i in range(len(l)):
42 |     coeffs = BerlekampMassey(l[:i+1])
43 |     for x in coeffs:
44 |         print(x, end=" ")
45 |     print()
```

1.12 Python

```python
1 | # Decimal
2 | from decimal import *
3 | getcontext().prec = 6
```

```python
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]))
31 |
32 | # OOP
33 | class Point:
34 |     def __init__(self, x, y):
35 |         self.x = x
36 |         self.y = y
37 |
38 |     def __add__(self, o): # use dir(int) to know operator
             name
39 |         return Point(self.x+o.x, self.y+o.y)
40 |
41 |     @property
42 |     def distance(self):
43 |         return (self.x**2 + self.y**2)**(0.5)
44 |
45 | a = Point(3, 4)
46 | print(a.distance)
47 |
48 | # Fraction
49 | from fractions import Fraction
50 | a = Fraction(Decimal(1.1))
51 | a.numerator # 分子
52 | a.denominator # 分母
```

1.13 diff

```bash
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 disable ASLR

```bash
1 | # Disable randomization of memory addresses
2 | setarch `uname -m` -R ./yourProgram
3 | setarch $(uname -m) -R
```

1.15 hash command

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

1.16 hash windows

```python
1 | def get_hash(path):
2 |     from subprocess import run, PIPE
3 |     from hashlib import md5
4 |
5 |     p = run(
6 |         ["cpp", "-dD", "-P", "-fpreprocessed", path],
7 |         stdout = PIPE,
8 |         stderr = PIPE,
9 |         text = True
10 |    )
11 |
12 |    if p.returncode != 0:
13 |        raise RuntimeError(p.stderr)
14 |
15 |    s = ''.join(p.stdout.split())
16 |    ret = md5(s.encode()).hexdigest()
17 |    return ret[:6]
18 |
19 | print(get_hash("Suffix_Array.cpp"))
```

2 Convolution

2.1 FFT any mod [234f9e]

```cpp
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    return res;
74
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    }
44 }
```

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]

```
1  // 需要增加註解
2  // min convolution
3  vector<int> mkk(vector<int> a, vector<int> b) {
4      vector<int> slope;
5      FOR (i, 1, ssize(a) - 1) slope.pb(a[i] - a[i - 1]);
6      FOR (i, 1, ssize(b) - 1) slope.pb(b[i] - b[i - 1]);
7      sort(all(slope));
8      slope.insert(begin(slope), a[0] + b[0]);
9      partial_sum(all(slope), begin(slope));
10     return slope;
11 }
```

2.6 NTT mod 998244353 [5c6335]

```
1  const int MOD = (119 << 23) + 1, ROOT = 62; // = 998244353
2  // For p < 2^30 there is also e.g. 5 << 25, 7 << 26, 479 <<
       21
3  // and 483 << 21 (same root). The last two are > 10^9.
4
5  // 9cd58a
6  void NTT(vector<int> &a) {
7      int n = a.size();
8      int L = 31-__builtin_clz(n);
9      vector<int> rt(2, 1);
10     for (int k=2, s=2 ; k<n ; k*=2, s++){
11         rt.resize(n);
12         int z[] = {1, qp(ROOT, MOD>>s)};
13         for (int i=k ; i<2*k ; i++){
14             rt[i] = rt[i/2]*z[i&1]%MOD;
15         }
16     }
17
18     vector<int> rev(n);
19     for (int i=0 ; i<n ; i++){
20         rev[i] = (rev[i/2]|(i&1)<<L)/2;
21     }
22     for (int i=0 ; i<n ; i++){
23         if (i<rev[i]){
24             swap(a[i], a[rev[i]]);
25         }
26     }
27
28     for (int k=1 ; k<n ; k*=2){
29         for (int i=0 ; i<n ; i+=2*k){
30             for (int j=0 ; j<k ; j++){
31                 int z = rt[j+k]*a[i+j+k]%MOD, &ai = a[i+j];
32                 a[i+j+k] = ai-z+(z>ai ? MOD : 0);
33                 ai += (ai+z>=MOD ? z-MOD : z);
34             }
35         }
36     }
37 }
38
39 // 0b0e99
40 vector<int> polyMul(vector<int> &a, vector<int> &b){
41     if (a.empty() || b.empty()) return {};
42     int s = a.size()+b.size()-1, B = 32-__builtin_clz(s), n =
           1<<B;
43     int inv = qp(n, MOD-2);
44
45     vector<int> L(a), R(b), out(n);
46     L.resize(n), R.resize(n);
47     NTT(L), NTT(R);
48     for (int i=0 ; i<n ; i++){
49         out[-i&(n-1)] = L[i]*R[i]%MOD*inv%MOD;
50     }
51     NTT(out);
52
53     out.resize(s);
54     return out;
55 }
```

3 Data-Structure

3.1 BIT [7ef3a9]

```
1  vector<int> BIT(MAX_SIZE);
2
3  // const int MAX_N = (1<<20)
4  int k_th(int k){ // 回傳 BIT 中第 k 小的元素 ( based-1 )
5      int res = 0;
6      for (int i=MAX_N>>1 ; i>=1 ; i>>=1)
7          if (BIT[res+i]<k)
8              k -= BIT[res+=i];
9      return res+1;
10 }
```

3.2 Disjoint Set Persistent [447002]

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

3.3 PBDS GP Hash Table [866cf6]

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

3.4 PBDS Order Set [231774]

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

3.5 Segment Tree Add Set [bb1898]

```
1  // [ll, rr), based-0
2  // 使用前記得 init(陣列大小), build(陣列名稱)
3  // add(ll, rr): 區間修改
4  // set(ll, rr): 區間賦值
5  // query(ll, rr): 區間求和 / 求最大值
6  struct SegmentTree{
7      struct node{
8          int add_tag = 0;
9          int set_tag = 0;
```

```
10        int sum = 0;
11        int ma = 0;
12    };
13
14    vector<node> arr;
15
16    SegmentTree(int n){
17        arr.resize(n<<2);
18    }
19
20    node pull(node A, node B){
21        node C;
22        C.sum = A.sum+B.sum;
23        C.ma = max(A.ma, B.ma);
24        return C;
25    }
26
27     // cce0c8
28    void push(int idx, int ll, int rr){
29        if (arr[idx].set_tag!=0){
30            arr[idx].sum = (rr-ll)*arr[idx].set_tag;
31            arr[idx].ma = arr[idx].set_tag;
32            if (rr-ll>1){
33                arr[idx*2+1].add_tag = 0;
34                arr[idx*2+1].set_tag = arr[idx].set_tag;
35                arr[idx*2+2].add_tag = 0;
36                arr[idx*2+2].set_tag = arr[idx].set_tag;
37            }
38            arr[idx].set_tag = 0;
39        }
40        if (arr[idx].add_tag!=0){
41            arr[idx].sum += (rr-ll)*arr[idx].add_tag;
42            arr[idx].ma += arr[idx].add_tag;
43            if (rr-ll>1){
44                arr[idx*2+1].add_tag += arr[idx].add_tag;
45                arr[idx*2+2].add_tag += arr[idx].add_tag;
46            }
47            arr[idx].add_tag = 0;
48        }
49    }
50
51    void build(vector<int> &v, int idx = 0, int ll = 0, int
          rr = n){
52        if (rr-ll==1){
53            arr[idx].sum = v[ll];
54            arr[idx].ma = v[ll];
55        }else{
56            int mid = (ll+rr)/2;
57            build(v, idx*2+1, ll, mid);
58            build(v, idx*2+2, mid, rr);
59            arr[idx] = pull(arr[idx*2+1], arr[idx*2+2]);
60        }
61    }
62
63    void add(int ql, int qr, int val, int idx = 0, int ll =
          0, int rr =n){
64        push(idx, ll, rr);
65        if (rr<=ql || qr<=ll) return;
66        if (ql<=ll && rr<=qr){
67            arr[idx].add_tag += val;
68            push(idx, ll, rr);
69            return;
70        }
71        int mid = (ll+rr)/2;
72        add(ql, qr, val, idx*2+1, ll, mid);
73        add(ql, qr, val, idx*2+2, mid, rr);
```

```
74            arr[idx]=pull(arr[idx*2+1], arr[idx*2+2]);
75    }
76
77    void set(int ql, int qr, int val, int idx=0, int ll=0,
          int rr=n){
78        push(idx, ll, rr);
79        if (rr<=ql || qr<=ll) return;
80        if (ql<=ll && rr<=qr){
81            arr[idx].add_tag = 0;
82            arr[idx].set_tag = val;
83            push(idx, ll, rr);
84            return;
85        }
86        int mid = (ll+rr)/2;
87        set(ql, qr, val, idx*2+1, ll, mid);
88        set(ql, qr, val, idx*2+2, mid, rr);
89        arr[idx] = pull(arr[idx*2+1], arr[idx*2+2]);
90    }
91
92    node query(int ql, int qr, int idx = 0, int ll = 0, int
          rr = n){
93        push(idx, ll, rr);
94        if (rr<=ql || qr<=ll) return node();
95        if (ql<=ll && rr<=qr) return arr[idx];
96
97        int mid = (ll+rr)/2;
98        return pull(query(ql, qr, idx*2+1, ll, mid), query(ql
          , qr, idx*2+2, mid, rr));
99    }
100 } ST;
```

3.6 Segment Tree Li Chao Line [45b8ba]

```
1  /*
2  全部都是 0-based
3
4  宣告
5  LC_Segment_Tree st(n);
6
7  函式：
8  update({a, b})：插入一條 y=ax+b 的全域直線
9  query(x)：查詢所有直線在位置 x 的最小值
10 */
11 const int MAX_V = 1e6+10; // 值域最大值
12
13 struct LC_Segment_Tree{
14    struct Node{ // y = ax+b
15        int a = 0;
16        int b = INF;
17
18        int y(int x){
19            return a*x+b;
20        }
21    };
22    vector<Node> arr;
23
24    LC_Segment_Tree(int n = 0){
25        arr.resize(4*n);
26    }
27
28    void update(Node val, int idx = 0, int ll = 0, int rr =
          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    int query(int x, int idx = 0, int ll = 0, int rr = MAX_V)
          {
49        if (rr-ll==0) return INF;
50        if (rr-ll==1){
51            return arr[idx].y(ll);
52        }
53
54        int mid = (ll+rr)/2;
55        if (x<mid){
56            return min(arr[idx].y(x), query(x, idx*2+1, ll,
                mid));
57        }else{
58            return min(arr[idx].y(x), query(x, idx*2+2, mid,
                rr));
59        }
60    }
61 };
```

3.7 Segment Tree Li Chao Segment [2cb0a4]

```
1  /*
2  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
42         int mid = (ll+rr)/2;
43         now.lc = node_cnt++;
44         now.rc = node_cnt++;
45         build(v, now.lc, ll, mid);
46         build(v, now.rc, mid, rr);
47         pull(now, arr[now.lc], arr[now.rc]);
48         return;
49     }
50
51     void update(int pos, int val, int idx, int ll = 0, int rr
           = n){
52         auto &now = arr[idx];
53
54         if (rr-ll==1){
55             now.val = val;
56             return;
57         }
58
59         int mid = (ll+rr)/2;
60         if (pos<mid){
61             arr[node_cnt] = arr[now.lc];
62             now.lc = node_cnt;
63             node_cnt++;
64             update(pos, val, now.lc, ll, mid);
65         }else{
66             arr[node_cnt] = arr[now.rc];
67             now.rc = node_cnt;
68             node_cnt++;
69             update(pos, val, now.rc, mid, rr);
70         }
71         pull(now, arr[now.lc], arr[now.rc]);
72         return;
73     }
74
75     void update_version(int pos, int val, int ver){
76         update(pos, val, version[ver]);
77     }
78
79     Node query(int ql, int qr, int idx, int ll = 0, int rr =
           n){
80         auto &now = arr[idx];
81
82         if (ql<=ll && rr<=qr) return now;
83         if (rr<=ql || qr<=ll) return Node();
84
85         int mid = (ll+rr)/2;
86
87         Node ret;
88         pull(ret, query(ql, qr, now.lc, ll, mid), query(ql,
               qr, now.rc, mid, rr));
89         return ret;
90     }
91
92     Node query_version(int ql, int qr, int ver){
93         return query(ql, qr, version[ver]);
94     }
95
96     void clone_version(int ver){
97         version.push_back(node_cnt);
98         arr[node_cnt] = arr[version[ver]];
99         node_cnt++;
100    }
101 };
```

3.9 Sparse Table [31f22a]

```
1  struct SparseTable{
2      vector<vector<int>> st;
3      void build(vector<int> v){
4          int h = __lg(v.size());
5          st.resize(h+1);
6          st[0] = v;
7
8          for (int i=1 ; i<=h ; i++){
9              int gap = (1<<(i-1));
10             for (int j=0 ; j+gap<st[i-1].size() ; j++){
11                 st[i].push_back(min(st[i-1][j], st[i-1][j+gap
                       ]));
12             }
13         }
14     }
15
16     // 回傳 [ll, rr) 的最小值
17     int query(int ll, int rr){
18         int h = __lg(rr-ll);
19         return min(st[h][ll], st[h][rr-(1<<h)]);
20     }
21 };
```

3.10 Treap2 [3b0cca]

```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], total[MAX_N];
// tag 為不包含自己 (僅要給子樹) 的資訊
int nodeCnt = 0;
int& new_node(int v){
    nodeCnt++;
    val[nodeCnt] = v;
    total[nodeCnt] = v;
    sz[nodeCnt] = 1;
    pri[nodeCnt] = rand();
    return nodeCnt;
}

void apply(int x, int V){
    val[x] += V;
    tag[x] += V;
    total[x] += V*sz[x];
}

void push(int x){
    if (tag[x]){
        if (lc[x]) apply(lc[x], tag[x]);
        if (rc[x]) apply(rc[x], tag[x]);
    }
    tag[x] = 0;
}
int pull(int x){
    if (x){
        fa[x] = 0;
        sz[x] = 1+sz[lc[x]]+sz[rc[x]];
        total[x] = val[x]+total[lc[x]]+total[rc[x]];
        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]);
    return ret;
}

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

3.11 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));
    }
```

```
32    // 已經搜尋完畢，紀錄答案並回傳
33    return dp[pos][pre][limit][lead] = ans;
34 }
35
36 // 回傳 [0, n] 有多少數字符合條件
37 long long find_answer(long long n){
38    memset(dp, -1, sizeof(dp));
39    string tmp = to_string(n);
40
41    return memorize_search(tmp, 0, 0, true, true);
42 }
43
44 int main(){
45
46    // input
47    cin >> l >> r;
48
49    // output - 計算 [l, r] 有多少數字任意兩個位數都不相同
50    cout << find_answer(r)-find_answer(l-1) << "\n";
51
52    return 0;
53 }
```

4.2 Knaspack On Tree [df69b1]

```
1  // 需要重構、需要增加註解
2  #include <bits/stdc++.h>
3  #define F first
4  #define S second
5  #define all(x) begin(x), end(x)
6  using namespace std;
7
8  #define chmax(a, b) (a) = (a) < (b) ? (b) : (a)
9  #define chmin(a, b) (a) = (a) < (b) ? (a) : (b)
10
11 #define ll long long
12
13 #define FOR(i, a, b) for (int i = a; i <= b; i++)
14
15 int N, W, cur;
16 vector<int> w, v, sz;
17 vector<vector<int>> adj, dp;
18
19 void dfs(int x) {
20    sz[x] = 1;
21    for (int i : adj[x]) dfs(i), sz[x] += sz[i];
22    cur++;
23    // choose x
24    for (int i=w[x] ; i<=W ; i++){
25        dp[cur][i] = dp[cur - 1][i - w[x]] + v[x];
26    }
27    // not choose x
28    for (int i=0 ; i<=W ; i++){
29        chmax(dp[cur][i], dp[cur - sz[x]][i]);
30    }
31 }
32
33 signed main() {
34    cin >> N >> W;
35    adj.resize(N + 1);
36    w.assign(N + 1, 0);
37    v.assign(N + 1, 0);
38    sz.assign(N + 1, 0);
39    dp.assign(N + 2, vector<int>(W + 1, 0));
40    for (int i=1 ; i<=N ; i++){
41        int p; cin >> p;
42        adj[p].push_back(i);
43    }
44
45    for (int i=1 ; i<=N ; i++) cin >> w[i];
46    for (int i=1 ; i<=N ; i++) cin >> v[i];
47    dfs(0);
48    cout << dp[N + 1][W] << '\n';
49 }
```

4.3 SOS DP [8dfa8b]

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

4.4 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], Line 9 ~ 13
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], Line 14 ~ 22
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], only this function
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], only this function
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     point<T> p1, p2;
69     // ax + by + c = 0
70     T a, b, c; // |a|, |b| ≤ 2C, |c| ≤ 8C²
71     line() {}
72     line(const point<T> &x,const point<T> &y) : p1(x), p2(y){
73         build();
74     }
75     void build() {
76         a = p1.y - p2.y;
77         b = p2.x - p1.x;
78         c = (-a*p1.x)-b*p1.y;
79     } // E [683239], Line 68 ~ 79
80     // 判斷點和有向直線的關係：{1:左邊,0:在線上,-1:右邊}
81     int ori(point<T> &p) {
82         return sign((p2-p1) ^ (p-p1));
83     }
84     // 判斷直線斜率是否相同
85     bool parallel(line &l) {
86         return ((p1-p2) ^ (l.p1-l.p2)) == 0;
87     }
88     // 兩直線交點
89     point<ld> line_intersection(line &l) {
90         using P = point<ld>;
91         point<T> u = p2-p1, v = l.p2-l.p1, s = l.p1-p1;
92         return P(p1) + P(u) * ((ld(s^v)) / (u^v));
93     }
94 };
95
```

```cpp
template<typename T>
struct polygon {
    vector<point<T>> v;
    polygon() {}
    polygon(const vector<point<T>> &u) : v(u) {}
    // simple 為 true 的時候會回傳任意三點不共線的凸包
    void make_convex_hull(int simple) {
        auto cmp = [&](point<T> &p, point<T> &q) {
            return (p.x == q.x) ? (p.y < q.y) : (p.x < q.x);
        };
        simple = (bool)simple;
        sort(v.begin(), v.end(), cmp);
        v.resize(unique(v.begin(), v.end()) - v.begin());
        if (v.size() <= 1) return;
        vector<point<T>> hull;
        for (int t = 0; t < 2; ++t){
            int sz = hull.size();
            for (auto &i:v) {
                while (hull.size() >= sz+2 && ori(hull[hull.
                    size()-2], hull.back(), i) < simple) {
                    hull.pop_back();
                }
                hull.push_back(i);
            }
            hull.pop_back();
            reverse(v.begin(), v.end());
        }
        swap(hull, v);
    } // F [2bb3ef], only this function
    // 可以在有 n 個點的簡單多邊形內，用 O(n) 判斷一個點：
    // {1 : 在多邊形內, 0 : 在多邊形上, -1 : 在多邊形外}
    int in_polygon(point<T> a){
        const T MAX_POS = 1e9 + 5; // [記得修改] 座標的最大值
        point<T> pre = v.back(), b(MAX_POS, a.y + 1);
        int cnt = 0;
        for (auto &i:v) {
            if (btw(pre, i, a)) return 0;
            if (banana(a, b, pre, i)) cnt++;
            pre = i;
        }
        return cnt%2 ? 1 : -1;
    } // G [f11340], only this function
    /// 警告：以下所有凸包專用的函式都只接受逆時針排序且任意三點不
    共線的凸包 ///
    // 可以在有 n 個點的凸包內，用 O(log n) 判斷一個點：
    // {1 : 在凸包內, 0 : 在凸包邊上, -1 : 在凸包外}
    int in_convex(point<T> p) {
        int n = v.size();
        int a = ori(v[0], v[1], p), b = ori(v[0], v[n-1], p);
        if (a < 0 || b > 0) return -1;
        if (btw(v[0], v[1], p)) return 0;
        if (btw(v[0], v[n - 1], p)) return 0;
        int l = 1, r = n - 1, mid;
        while (l + 1 < r) {
            mid = (l + r) >> 1;
            if (ori(v[0], v[mid], p) >= 0) l = mid;
            else r = mid;
        }
        int k = ori(v[l], v[r], p);
        if (k <= 0) return k;
        return 1;
    } // H [e64f1e], only this function
```

```cpp
    // 凸包專用的環狀二分搜，回傳 0-based index
    int cycle_search(auto &f) {
        int n = v.size(), l = 0, r = n;
        if (n == 1) return 0;
        bool rv = f(1, 0);
        while (r - l > 1) {
            int m = (l + r) / 2;
            if (f(0, m) ? rv: f(m, (m + 1) % n)) r = m;
            else l = m;
        }
        return f(l, r % n) ? l : r % n;
    } // I [fe2f51], only this function
    // 可以在有 n 個點的凸包內，用 O(log n) 判斷一條直線：
    // {1 : 穿過凸包, 0 : 剛好切過凸包, -1 : 沒碰到凸包}
    int line_cut_convex(line<T> L) {
        L.build();
        point<T> p(L.a, L.b);
        auto gt = [&](int neg) {
            auto f = [&](int x, int y) {
                return sign((v[x] - v[y]) * p) == neg;
            };
            return -(v[cycle_search(f)] * p);
        };
        T x = gt(1), y = gt(-1);
        if (L.c < x || y < L.c) return -1;
        return not (L.c == x || L.c == y);
    } // J [b6a4c8], only this function
    // 可以在有 n 個點的凸包內，用 O(log n) 判斷一個線段：
    // {1 : 存在一個凸包上的邊可以把這個線段切成兩半,
    //  0 : 有碰到凸包但沒有任何凸包上的邊可以把它切成兩半,
    // -1 : 沒碰到凸包}
    /// 除非線段兩端點都不在凸包邊上，否則此函數回傳 0 的時候不一
    定表示線段沒有通過凸包內部 ///
    int segment_across_convex(line<T> L) {
        L.build();
        point<T> p(L.a, L.b);
        auto gt = [&](int neg) {
            auto f = [&](int x, int y) {
                return sign((v[x] - v[y]) * p) == neg;
            };
            return cycle_search(f);
        };
        int i = gt(1), j = gt(-1), n = v.size();
        T x = -(v[i] * p), y = -(v[j] * p);
        if (L.c < x || y < L.c) return -1;
        if (L.c == x || L.c == y) return 0;

        if (i > j) swap(i, j);
        auto g = [&](int x, int lim) {
            int now = 0, nxt;
            for (int i = 1 << __lg(lim); i > 0; i /= 2) {
                if (now + i > lim) continue;
                nxt = (x + i) % n;
                if (L.ori(v[x]) * L.ori(v[nxt]) >= 0) {
                    x = nxt;
                    now += i;
                }
            } //  ↓ BE CAREFUL
            return -(ori(v[x], v[(x + 1) % n], L.p1) * ori(v[
                x], v[(x + 1) % n], L.p2));
        };
        return max(g(i, j - i), g(j, n - (j - i)));
    } // K [b4f073], only this function
    // 可以在有 n 個點的凸包內，用 O(log n) 判斷一個線段：
```

```cpp
    // {1 : 線段上存在某一點位於凸包內部（邊上不算），
    //  0 : 線段上存在某一點碰到凸包的邊但線段上任一點均不在凸
    內部,
    // -1 : 線段完全在凸包外面}
    int segment_pass_convex_interior(line<T> L) {
        if (in_convex(L.p1) == 1 || in_convex(L.p2) == 1)
            return 1;
        L.build();
        point<T> p(L.a, L.b);
        auto gt = [&](int neg) {
            auto f = [&](int x, int y) {
                return sign((v[x] - v[y]) * p) == neg;
            };
            return cycle_search(f);
        };
        int i = gt(1), j = gt(-1), n = v.size();
        T x = -(v[i] * p), y = -(v[j] * p);
        if (L.c < x || y < L.c) return -1;
        if (L.c == x || L.c == y) return 0;

        if (i > j) swap(i, j);
        auto g = [&](int x, int lim) {
            int now = 0, nxt;
            for (int i = 1 << __lg(lim); i > 0; i /= 2) {
                if (now + i > lim) continue;
                nxt = (x + i) % n;
                if (L.ori(v[x]) * L.ori(v[nxt]) > 0) {
                    x = nxt;
                    now += i;
                }
            } //  ↓ BE CAREFUL
            return -(ori(v[x], v[(x + 1) % n], L.p1) * ori(v[
                x], v[(x + 1) % n], L.p2));
        };
        int ret = max(g(i, j - i), g(j, n - (j - i)));
        return (ret == 0) ? (in_convex(L.p1) == 0 &&
            in_convex(L.p2) == 0) : ret;
    } // L [5f45ca], only this function
    // 回傳過凸包的兩條切線的切點的 0-based index（不保證兩條
    切線的順逆時針關係）
    pair<int,int> convex_tangent_point(point<T> p) {
        int n = v.size(), z = -1, edg = -1;
        auto gt = [&](int neg) {
            auto check = [&](int x) {
                if (v[x] == p) z = x;
                if (btw(v[x], v[(x + 1) % n], p)) edg = x;
                if (btw(v[(x + n - 1) % n], v[x], p)) edg = (
                    x + n - 1) % n;
            };
            auto f = [&](int x, int y) {
                check(x); check(y);
                return ori(p, v[x], v[y]) == neg;
            };
            return cycle_search(f);
        };
        int x = gt(1), y = gt(-1);
        if (z != -1) {
            return {(z + n - 1) % n, (z + 1) % n};
        }
        else if (edg != -1) {
            return {edg, (edg + 1) % n};
        }
        else {
            return {x, y};
        }
    }
```

```cpp
280      } // M [a6f66b], only this function
281      friend int halfplane_intersection(vector<line<T>> &s,
            polygon<T> &P) {
282          auto angle_cmp = [&](line<T> &A, line<T> &B) {
283              point<T> a = A.p2-A.p1, b = B.p2-B.p1;
284              return (a < b);
285          };
286          sort(s.begin(), s.end(), angle_cmp); // 線段左側為該
                   線段半平面
287          int L, R, n = s.size();
288          vector<point<T>> px(n);
289          vector<line<T>> q(n);
290          q[L = R = 0] = s[0];
291          for(int i = 1; i < n; ++i) {
292              while(L < R && s[i].ori(px[R-1]) <= 0) --R;
293              while(L < R && s[i].ori(px[L])   <= 0) ++L;
294              q[++R] = s[i];
295              if(q[R].parallel(q[R-1])) {
296                  --R;
297                  if(q[R].ori(s[i].p1) > 0) q[R] = s[i];
298              }
299              if(L<R) px[R-1] = q[R-1].line_intersection(q[R]);
300          }
301          while(L < R && q[L].ori(px[R-1]) <= 0) --R;
302          P.v.clear();
303          if(R - L <= 1) return 0;
304          px[R] = q[R].line_intersection(q[L]);
305          for(int i = L; i <= R; ++i) P.v.push_back(px[i]);
306          return R - L + 1;
307      } // N [102d48], only this function
308 };
309
310 struct Cir {
311     point<ld> o; ld r;
312     friend ostream& operator<<(ostream& os, Cir c) {
313         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;
314     }
315     bool covers(Cir b) {
316         return sqrt((ld)abs2(o - b.o)) + b.r <= r;
317     }
318     vector<point<ld>> Cir_intersect(Cir c) {
319         ld d2 = abs2(o - c.o), d = sqrt(d2);
320         if (d < max(r, c.r) - min(r, c.r) || d > r + c.r)
321             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], only this function
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         }
340         return res;
341     } // P [0067e6], only this function
342 };
```

5.2 Geometry Struct 3D [4a50c9]

```cpp
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>
8  struct pt3 {
9      T x, y, z;
10     pt3(){}
11     pt3(const T &x, const T &y, const T &z):x(x),y(y),z(z){}
12     explicit operator pt3<ld>() {return pt3<ld>(x, y, z); }
13
14     pt3 operator+(pt3 b) {return {x+b.x, y+b.y, z+b.z}; }
15     pt3 operator-(pt3 b) {return {x-b.x, y-b.y, z-b.z}; }
16     pt3 operator*(T b) {return {x * b, y * b, z * b}; }
17     pt3 operator/(T b) {return {x / b, y / b, z / b}; }
18     bool operator==(pt3 b){return x==b.x&&y==b.y&&z==b.z;}
19
20     T operator*(pt3 b) {return x*b.x+y*b.y+z*b.z; }
21     pt3 operator^(pt3 b) {
22         return pt3(y * b.z - z * b.y,
23                    z * b.x - x * b.z,
24                    x * b.y - y * b.x);
25     }
26     friend T abs2(pt3 b) {return b * b; }
27     friend T len (pt3 b) {return sqrt(abs2(b)); }
28     friend ostream& operator<<(ostream& os, pt3 p) {
29         return os << "(" << p.x << ", " <<
                          p.y << ", " << p.z << ")";
30     }
31   }
32 };
33
34 template<typename T>
35 struct face {
36     int a, b, c; // 三角形在 vector 裡面的 index
37     pt3<T> q;       // 面積向量（朝外）
38 };
39
40 /// 警告；v 在過程中可能被修改，回傳的 face 以修改後的為準
41 // O(n²)，最多只有 2n-4 個面
42 // 當凸包退化時會回傳空的凸包，否則回傳凸包上的每個面
43 template<typename T>
44 vector<face<T>> hull3(vector<pt3<T>> &v) {
45     int n = v.size();
46     if (n < 3) return {};
47     // don't use "==" when you use ld
48     sort(all(v), [&](pt3<T> &p, pt3<T> &q) {
49         return sign(p.x - q.x) ? (p.x < q.x) :
50             (sign(p.y - q.y) ? p.y < q.y : p.z < q.z);
51     });
52     v.resize(unique(v.begin(), v.end()) - v.begin());
53     for (int i = 2; i <= n; ++i) {
54         if (i == n) return {};
55         if (sign(len(((v[1] - v[0]) ^ (v[i] - v[0]))))) {
56             swap(v[2], v[i]);
57             break;
58         }
59     }
60     pt3<T> tmp_q = (v[1] - v[0]) ^ (v[2] - v[0]);
61     for (int i = 3; i <= n; ++i) {
62         if (i == n) return {};
63         if (sign((v[i] - v[0]) * tmp_q)) {
64             swap(v[3], v[i]);
65             break;
66         }
67     }
68
69     vector<face<T>> f;
70     vector<vector<int>> dead(n, vector<int>(n, true));
71     auto add_face = [&](int a, int b, int c) {
72         f.emplace_back(a, b, c, (v[b] - v[a]) ^ (v[c] - v[a])
               );
73         dead[a][b] = dead[b][c] = dead[c][a] = false;
74     };
75     add_face(0, 1, 2);
76     add_face(0, 2, 1);
77
78     for (int i = 3; i < n; ++i) {
79         vector<face<T>> f2;
80         for (auto &[a, b, c, q] : f) {
81             if (sign((v[i] - v[a]) * q) > 0)
82                 dead[a][b] = dead[b][c] = dead[c][a] = true;
83             else f2.emplace_back(a, b, c, q);
84         }
85         f.clear();
86         for (face<T> &F : f2) {
87             int arr[3] = {F.a, F.b, F.c};
88             for (int j = 0; j < 3; ++j) {
89                 int a = arr[j], b = arr[(j + 1) % 3];
90                 if (dead[b][a]) add_face(b, a, i);
91             }
92         }
93         f.insert(f.end(), all(f2));
94     }
95     return f;
96 } // 15ef50
```

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
1  struct TWO_SAT {
2      int n, N;
3      vector<vector<int>> G, rev_G;
4      deque<bool> used;
5      vector<int> order, comp;
6      deque<bool> assignment;
7      void init(int _n) {
8          n = _n;
9          N = _n * 2;
10         G.resize(N + 5);
11         rev_G.resize(N + 5);
12     }
13     void dfs1(int v) {
14         used[v] = true;
15         for (int u : G[v])
16             if (!used[u])
```

```cpp
            dfs1(u);
        }
        order.push_back(v);
    }
    void dfs2(int v, int cl) {
        comp[v] = cl;
        for (int u : rev_G[v]) {
            if (comp[u] == -1)
                dfs2(u, cl);
        }
    }
    bool solve() {
        order.clear();
        used.assign(N, false);
        for (int i = 0; i < N; ++i) {
            if (!used[i])
                dfs1(i);
        }
        comp.assign(N, -1);
        for (int i = 0, j = 0; i < N; ++i) {
            int v = order[N - i - 1];
            if (comp[v] == -1)
                dfs2(v, j++);
        }
        assignment.assign(n, false);
        for (int i = 0; i < N; i += 2) {
            if (comp[i] == comp[i + 1])
                return false;
            assignment[i / 2] = (comp[i] > comp[i + 1]);
        }
        return true;
    }
    // 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;
```

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

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

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

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

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

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

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

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

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

        vector<pair<int, int>> ret;
```

```cpp
        for (int i=0 ; i<n ; i++){
            if (mx[i]==-1) dfs2(i);
        }

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

        return ret;
    }
};
```

6.3 C3C4 [d00465]

```cpp
// 0-based
void C3C4(vector<int> deg, vector<array<int, 2>> edges){
    int N = deg.size();
    int M = deges.size();

    vector<int> ord(N), rk(N);
    iota(ord.begin(), ord.end(), 0);
    sort(ord.begin(), ord.end(), [&](int x, int y) { return
        deg[x] > deg[y]; });
    for (int i=0 ; i<N ; i++) rk[ord[i]] = i;

    vector<vector<int>> D(N), adj(N);
    for (auto [u, v] : e) {
        if (rk[u] > rk[v]) swap(u, v);
        D[u].emplace_back(v);
        adj[u].emplace_back(v);
        adj[v].emplace_back(u);
    }

    vector<int> vis(N);

    int c3 = 0, c4 = 0;
    for (int x : ord) { // c3
        for (int y : D[x]) vis[y] = 1;
        for (int y : D[x]) for (int z : D[y]){
            c3 += vis[z]; // xyz is C3
        }
        for (int y : D[x]) vis[y] = 0;
    }
    for (int x : ord) { // c4
        for (int y : D[x]) for (int z : adj[y])
            if (rk[z] > rk[x]) c4 += vis[z]++;
        for (int y : D[x]) for (int z : adj[y])
            if (rk[z] > rk[x]) --vis[z];
    }   // both are O(M*sqrt(M)), test @ 2022 CCPC guangzhou
    cout << c4 << "\n";
}
```

6.4 Cut BCC [2af809]

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

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

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

6.5 Dinic [961b34]

```cpp
 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;
32          }
33          return 0;
```

```cpp
34      }
35
36      // e22e39
37      T flow(int s, int t){
38          T ans = 0;
39          while (true){
40              fill(dis.begin(), dis.end(), INF);
41              queue<int> q;
42              q.push(s);
43              dis[s] = 0;
44
45              while (q.size()){
46                  int u = q.front(); q.pop();
47                  for (auto [v, rc, rid] : G[u]){
48                      if (rc <= 0 || dis[v] < INF) continue;
49                      dis[v] = dis[u] + 1;
50                      q.push(v);
51                  }
52              }
53              if (dis[t]==INF) break;
54
55              fill(it.begin(), it.end(), 0);
56              while (true){
57                  T df = dfs(s, t, INF);
58                  if (df <= 0) break;
59                  ans += df;
60              }
61          }
62          return ans;
63      }
64
65      // the code below constructs minimum cut
66      void dfs_mincut(int now, vector<bool> &vis){
67          vis[now] = true;
68          for (auto &[v, rc, rid] : G[now]){
69              if (vis[v] == false && rc > 0){
70                  dfs_mincut(v, vis);
71              }
72          }
73      }
74
75      vector<pair<int, int>> construct(int n, int s, vector<
         pair<int,int>> &E){
76          // E is G without capacity
77          vector<bool> vis(n);
78          dfs_mincut(s, vis);
79          vector<pair<int, int>> ret;
80          for (auto &[u, v] : E){
81              if (vis[u] == true && vis[v] == false){
82                  ret.emplace_back(u, v);
83              }
84          }
85          return ret;
86      }
87  };
```

6.6 Dominator Tree [52b249]

```cpp
 1  /*
 2  全部都是 0-based
 3  G 要是有向無權圖
 4  一開始要初始化 G(N, root)，代表有 N 個節點，根是 root
 5  用完之後要 build
 6  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
```

```cpp
 7  */
 8  struct DominatorTree{
 9      int N;
10      vector<vector<int>> G;
11      vector<vector<int>> buckets, rg;
12      // dfn[x] = the DFS otder of x
13      // rev[x] = the vertex with DFS order x
14      // par[x] = the parent of x
15      vector<int> dfn, rev, par;
16      vector<int> sdom, dom, idom;
17      vector<int> fa, val;
18      int stamp;
19      int root;
20
21      int operator [] (int x){
22          return idom[x];
23      }
24
25      DominatorTree(int _N, int _root) :
26          N(_N),
27          G(N), buckets(N), rg(N),
28          dfn(N, -1), rev(N, -1), par(N, -1),
29          sdom(N, -1), dom(N, -1), idom(N, -1),
30          fa(N, -1), val(N, -1)
31      {
32          stamp = 0;
33          root = _root;
34      }
35
36      void add_edge(int u, int v){
37          G[u].push_back(v);
38      }
39
40      void dfs(int x){
41          rev[dfn[x] = stamp] = x;
42          fa[stamp] = sdom[stamp] = val[stamp] = stamp;
43          stamp++;
44
45          for (int u : G[x]){
46              if (dfn[u]==-1){
47                  dfs(u);
48                  par[dfn[u]] = dfn[x];
49              }
50              rg[dfn[u]].push_back(dfn[x]);
51          }
52      }
53
54      int eval(int x, bool first){
55          if (fa[x]==x) return !first ? -1 : x;
56          int p = eval(fa[x], false);
57
58          if (p==-1) return x;
59          if (sdom[val[x]]>sdom[val[fa[x]]]) val[x] = val[fa[x
             ]];
60          fa[x] = p;
61
62          return !first ? p : val[x];
63      }
64
65      void link(int x, int y){
66          fa[x] = y;
67      }
68
69      void build(){
70          dfs(root);
```

```
72    for (int x=stamp-1 ; x>=0 ; x--){
73        for (int y : rg[x]){
74            sdom[x] = min(sdom[x], sdom[eval(y, true)]);
75        }
76        if (x>0) buckets[sdom[x]].push_back(x);
77        for (int u : buckets[x]){
78            int p = eval(u, true);
79            if (sdom[p]==x) dom[u] = x;
80            else dom[u] = p;
81        }
82        if (x>0) link(x, par[x]);
83    }
84
85    idom[root] = root;
86    for (int x=1 ; x<stamp ; x++){
87        if (sdom[x]!=dom[x]) dom[x] = dom[dom[x]];
88    }
89    for (int i=1 ; i<stamp ; i++) idom[rev[i]] = rev[dom[
       i]];
90    }
91 };
```

6.7 EdgeBCC [d09eb1]

```
1  // d09eb1
2  // 0-based · 支援重邊
3  struct EdgeBCC{
4      int n, m, dep, sz;
5      vector<vector<pair<int, int>>> G;
6      vector<vector<int>> bcc;
7      vector<int> dfn, low, stk, isBridge, bccId;
8      vector<pair<int, int>> edge, bridge;
9
10     EdgeBCC(int _n) : n(_n), m(0), sz(0), dfn(n), low(n), G(n
        ), bcc(n), bccId(n) {}
11
12     void add_edge(int u, int v) {
13         edge.push_back({u, v});
14         G[u].push_back({v, m});
15         G[v].push_back({u, m++});
16     }
17
18     void dfs(int now, int pre) {
19         dfn[now] = low[now] = ++dep;
20         stk.push_back(now);
21
22         for (auto [x, id] : G[now]){
23             if (!dfn[x]){
24                 dfs(x, id);
25                 low[now] = min(low[now], low[x]);
26             }else if (id!=pre){
27                 low[now] = min(low[now], dfn[x]);
28             }
29         }
30
31         if (low[now]==dfn[now]){
32             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]) { /// 若不在同一條鍊上
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         /// 此時已在同一條鍊上
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
9      的結果存在 result 裡
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
46     void dfs2(vector<vector<int>> &G, int now){
47         SCC_id[now] = SCC.size()-1;
48         SCC.back().push_back(now);
49         vis[now] = 1;
50         for (auto x : G[now]) if (!vis[x]) dfs2(G, x);
51     }
52
53     void compress(){
54         fill(vis.begin(), vis.end(), 0);
55         for (int i=0 ; i<N ; i++) if (!vis[i]) dfs1(G, i);
56
57         fill(vis.begin(), vis.end(), 0);
58         reverse(order.begin(), order.end());
59         for (int i=0 ; i<N ; i++){
60             if (!vis[order[i]]){
61                 SCC.push_back(vector<int>());
62                 dfs2(inv_G, order[i]);
63             }
64         }
65
66         result.resize(SCC.size());
67         sz = SCC.size();
68         for (auto [u, v] : edges){
69             if (SCC_id[u]!=SCC_id[v]) result[SCC_id[u]].
                   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
                   [i].end())-result[i].begin());
74         }
75     }
76 };
```

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),
            ly(n), slack(n), match(n), visx(n), visy(n) {}
10
11     void add(int x, int y, int w){
12         G[x][y] = max(G[x][y], w);
13     }
14
15     bool dfs(int i, bool aug){ // aug = true 表示要更新 match
16         if (visx[i]==stamp) return false;
17         visx[i] = stamp;
18
19         for (int j=0 ; j<n ; j++){
20             if (visy[j]==stamp) continue;
21             int d = lx[i]+ly[j]-G[i][j];
22
23             if (d==0){
24                 visy[j] = stamp;
25                 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(
            N), G(N){
8          LCA.resize(H, vector<int>(N, 0));
9      }
```

```
10
11    void add_edge(int u, int v){
12        M++;
13        G[u].push_back(v);
14        G[v].push_back(u);
15    }
16
17    void dfs(int now = 0, int pre = 0){
18        dep[now] = dep[pre]+1;
19        parent[now] = pre;
20        for (auto x : G[now]){
21            if (x==pre) continue;
22            dfs(x, now);
23        }
24    }
25
26    void build_LCA(int root = 0){
27        dfs();
28        for (int i=0 ; i<N ; i++) LCA[0][i] = parent[i];
29        for (int i=1 ; i<H ; i++){
30            for (int j=0 ; j<N ; j++){
31                LCA[i][j] = LCA[i-1][LCA[i-1][j]];
32            }
33        }
34    }
35
36    int jump(int u, int step){
37        for (int i=0 ; i<H ; i++){
38            if (step&(1<<i)) u = LCA[i][u];
39        }
40        return u;
41    }
42
43    int get_LCA(int u, int v){
44        if (dep[u]<dep[v]) swap(u, v);
45        u = jump(u, dep[u]-dep[v]);
46        if (u==v) return u;
47        for (int i=H-1 ; i>=0 ; i--){
48            if (LCA[i][u]!=LCA[i][v]){
49                u = LCA[i][u];
50                v = LCA[i][v];
51            }
52        }
53        return parent[u];
54    }
55 };
```

6.13 MCMF [1b5a27]

```
1 struct Flow {
2    struct Edge {
3        int u, rc, k, rv;
4    };
5
6    vector<vector<Edge>> G;
7    vector<int> par, par_eid;
8    Flow(int n) : G(n+1), par(n+1), par_eid(n+1) {}
9
10    // v->u, capcity: c, cost: k
11    void add(int v, int u, int c, int k){
12      G[v].push_back({u, c, k, G[u].size()});
13      G[u].push_back({v, 0, -k, G[v].size()-1});
14    }
15
16    // 3701d6
```

```
17    int spfa(int s, int t){
18        fill(par.begin(), par.end(), -1);
19        vector<int> dis(par.size(), INF);
20        vector<bool> in_q(par.size(), false);
21        queue<int> Q;
22        dis[s] = 0;
23        in_q[s] = true;
24        Q.push(s);
25
26        while (!Q.empty()){
27            int v = Q.front();
28            Q.pop();
29            in_q[v] = false;
30
31            for (int i=0 ; i<G[v].size() ; i++){
32                auto [u, rc, k, rv] = G[v][i];
33                if (rc>0 && dis[v]+k<dis[u]){
34                    dis[u] = dis[v]+k;
35                    par[u] = v;
36                    par_eid[u] = i;
37                    if (!in_q[u]) Q.push(u);
38                    in_q[u] = true;
39                }
40            }
41        }
42
43        return dis[t];
44    }
45
46    // return <max flow, min cost>, 150093
47    pair<int, int> flow(int s, int t){
48        int fl = 0, cost = 0, d;
49        while ((d = spfa(s, t))<INF){
50            int cur = INF;
51            for (int v=t ; v!=s ; v=par[v])
52                cur = min(cur, G[par[v]][par_eid[v]].rc);
53            fl += cur;
54            cost += d*cur;
55            for (int v=t ; v!=s ; v=par[v]){
56                G[par[v]][par_eid[v]].rc -= cur;
57                G[v][G[par[v]][par_eid[v]].rv].rc += cur;
58            }
59        }
60        return {fl, cost};
61    }
62
63    vector<pair<int, int>> construct(){
64        vector<pair<int, int>> ret;
65        for (int i=0 ; i<n ; i++){
66            for (auto x : G[i]){
67                if (x.rc==0){
68                    ret.push_back({i+1, x.u-n+1});
69                    break;
70                }
71            }
72        }
73        return ret;
74    }
75 };
```

6.14 Tarjan [8b2350]

```
1 struct tarjan_SCC {
2    int now_T, now_SCCs;
3    vector<int> dfn, low, SCC;
```

```
4    stack<int> S;
5    vector<vector<int>> E;
6    vector<bool> vis, in_stack;
7
8    tarjan_SCC(int n) {
9        init(n);
10    }
11    void init(int n) {
12        now_T = now_SCCs = 0;
13        dfn = low = SCC = vector<int>(n);
14        E = vector<vector<int>>(n);
15        S = stack<int>();
16        vis = in_stack = vector<bool>(n);
17    }
18    void add(int u, int v) {
19        E[u].push_back(v);
20    }
21    void build() {
22        for (int i = 0; i < dfn.size(); ++i) {
23            if (!dfn[i]) dfs(i);
24        }
25    }
26    void dfs(int v) {
27        now_T++;
28        vis[v] = in_stack[v] = true;
29        dfn[v] = low[v] = now_T;
30        S.push(v);
31        for (auto &i:E[v]) {
32            if (!vis[i]) {
33                vis[i] = true;
34                dfs(i);
35                low[v] = min(low[v], low[i]);
36            }
37            else if (in_stack[i]) {
38                low[v] = min(low[v], dfn[i]);
39            }
40        }
41        if (low[v] == dfn[v]) {
42            int tmp;
43            do {
44                tmp = S.top();
45                S.pop();
46                SCC[tmp] = now_SCCs;
47                in_stack[tmp] = false;
48            } while (tmp != v);
49            now_SCCs += 1;
50        }
51    }
52 };
```

6.15 Tarjan Find AP [1daed6]

```
1 vector<int> dep(MAX_N), low(MAX_N), AP;
2 bitset<MAX_N> vis;
3
4 void dfs(int now, int pre){
5    int cnt = 0;
6    bool ap = 0;
7    vis[now] = 1;
8    low[now] = dep[now] = (now==1 ? 0 : dep[pre]+1);
9
10    for (auto x : G[now]){
11        if (x==pre){
12            continue;
13        }else if (vis[x]==0){
```

```
14            cnt++;
15            dfs(x, now);
16            low[now] = min(low[now], low[x]);
17            if (low[x]>=dep[now]) ap=1;
18        }else{
19            low[now] = min(low[now], dep[x]);
20        }
21    }
22
23    if ((now==pre && cnt>=2) || (now!=pre && ap)){
24        AP.push_back(now);
25    }
26 }
```

6.16 Tree Isomorphism [cd2bbc]

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

```
48            v.push_back(add);
49         }
50     }
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;
```

```cpp
42    int childs = 0;
43    nowT += 1;
44    dfn[v] = low[v] = nowT;
45    for(auto &ne:E[v]){
46        int i = EV[ne].to;
47        if(i == par) continue;
48        if(!dfn[i]){
49            S.push(ne);
50            tarjan(i, v, S);
51            childs += 1;
52            low[v] = min(low[v], low[i]);
53
54            if(par >= 0 && low[i] >= dfn[v]){
55                vector<int> bcc;
56                int tmp;
57                do{
58                    tmp = S.top(); S.pop();
59                    if(!to_add[EV[tmp].fr]){
60                        to_add[EV[tmp].fr] = true;
61                        bcc.pb(EV[tmp].fr);
62                    }
63                    if(!to_add[EV[tmp].to]){
64                        to_add[EV[tmp].to] = true;
65                        bcc.pb(EV[tmp].to);
66                    }
67                }while(tmp != ne);
68                for(auto &j:bcc){
69                    to_add[j] = false;
70                    F[last_special_node].pb(j);
71                    F[j].pb(last_special_node);
72                }
73                last_special_node += 1;
74            }
75        }
76        else{
77            low[v] = min(low[v], dfn[i]);
78            if(dfn[i] < dfn[v]){ // edge i--v will be visited
79                                 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]

```cpp
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 $\sum w_e \leq b_v$。
  - The maximum w-weight of a b-matching equals the minimum b-weight of vertices in a w-vertex-cover.

7 Math

7.1 CRT [682ac6]

```cpp
1 // 求出 d = gcd(a,b)，並找出 x, y 使 ax + by = d
2 tuple<int, int, int> extgcd(int a, int b){
3     if (!b) return {a, 1, 0};
4     auto [d, x, y] = extgcd(b, a%b);
5     return {d, y, x-a/b*y};
6 }
```

```
7
8   // CRT maybe need use int128
9   int CRT_m_coprime(vector<int> &a, vector<int> &m) {
10      int n = a.size(), p = 1, ans = 0;
11      vector<int> M(n), invM(n);
12
13      for (int i=0 ; i<n ; i++) p *= m[i];
14      for (int i=0 ; i<n ; i++){
15          M[i] = p/m[i];
16          auto [d, x, y] = extgcd(M[i], m[i]);
17          invM[i] = x;
18          ans += a[i]*invM[i]*M[i];
19          ans %= p;
20      }
21      return (ans%p+p)%p;
22  }
23
24  // CRT maybe need use int128
25  int CRT_m_not_coprime(vector<int> &a, vector<int> &m) {
26      int n = a.size();
27
28      for (int i=1 ; i<n ; i++){
29          int g = __gcd(m[0], m[i]);
30          if ((a[i]-a[0])%g!=0) return -1;
31
32          auto [d, x, y] = extgcd(m[0], m[i]);
33          x = (a[i]-a[0])*x/g;
34
35          a[0] = x*m[0]+a[0];
36          m[0] = m[0]*m[i]/g;
37          a[0] = (a[0]%m[0]+m[0])%m[0];
38      }
39
40      if (a[0]<0) return a[0]+m[0];
41      return a[0];
42  }
43
44  // ans = a / b (mod m)
45  // ans = ret.F + k * ret.S, k is integer
46  pair<int, int> div(int a, int b, int m) {
47      int flag = 1;
48      if (a < 0) { a = -a; flag *= -1; }
49      if (b < 0) { b = -b; flag *= -1; }
50      int t = -1, k = -1;
51      int res = extgcd_abc(b, m, a, t, k);
52      if (res == INF) return {INF, INF};
53      m = abs(m / res);
54      t = t * flag;
55      t = (t % m + m) % m;
56      return {t, m};
57  }
```

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;
                      now %= MOD;
21              }
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
```

```cpp
vector<int> s(n+1, 1), invS(n+1), invA(n+1);
for (int i=1 ; i<=n ; i++) s[i] = s[i-1]*a[i]%p;
invS[n] = qp(s[n], p-2, p);
for (int i=n ; i>=1 ; i--) invS[i-1] = invS[i]*a[i]%p;
for (int i=1 ; i<=n ; i++) invA[i] = invS[i]*s[i-1]%p;
```

7.6 Lucas's Theorem [b37dcf]

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

7.7 Matrix [8d1a23]

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

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

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

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

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

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

        return ret;
    }

    int det(){
        vector<vector<int>> arr = this->arr;
        bool flag = false;
        for (int i=0 ; i<n ; i++){
            int target = -1;
            for (int j=i ; j<n ; j++){
```

```cpp
                if (arr[j][i]){
                    target = j;
                    break;
                }
            }
            if (target==-1) return 0;
            if (i!=target){
                swap(arr[i], arr[target]);
                flag = !flag;
            }

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

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

7.8 Matrix 01 [8d542a]

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

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

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

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

7.9 Miller Rabin [24bd0d]

```cpp
// O(k log^3 n), k = llsprp.size()
typedef Uint unsigned long long;
Uint modmul(Uint a, Uint b, Uint m) {
    int ret = a*b - m*(Uint)((long double)a*b/m);
    return ret + m*(ret < 0) - m*(ret>=(int)m);
}

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

// ed23aa
vector<int> llsprp = {2, 325, 9375, 28178, 450775, 9780504,
    1795265022};
bool is_prime(int n, vector<int> sprp = llsprp){
    if (n==2) return 1;
    if (n<2 || n%2==0) return 0;

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

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

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

    return true;
}
```

7.10 Pollard Rho [a5daef]

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

// O(n^{1/4}) 回傳 1 或自己的因數,記得先判斷 n 是不是質數
// (用 Miller-Rabin)
// 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++){
```

```
17              t = ((__int128)t*t+c)%n;
18              val = (__int128)val*abs(t-s)%n;
19
20
21              if ((step % 127) == 0){
22                  int d = __gcd(val, n);
23                  if (d>1) return d;
24              }
25          }
26      }
27      int d = __gcd(val, n);
28      if (d>1) return d;
29  }
30 }
```

7.11 Polynomial [51ca3b]

```
1  struct Poly {
2      int len, deg;
3      int *a;
4      // len = 2^k >= the original length
5      Poly(): len(0), deg(0), a(nullptr) {}
6      Poly(int _n) {
7          len = 1;
8          deg = _n - 1;
9          while (len < _n) len <<= 1;
10         a = (ll*) calloc(len, sizeof(ll));
11     }
12     Poly(int l, int d, int *b) {
13         len = l;
14         deg = d;
15         a = b;
16     }
17     void resize(int _n) {
18         int len1 = 1;
19         while (len1 < _n) len1 <<= 1;
20         int *res = (ll*) calloc(len1, sizeof(ll));
21         for (int i = 0; i < min(len, _n); i++) {
22             res[i] = a[i];
23         }
24         len = len1;
25         deg = _n - 1;
26         free(a);
27         a = res;
28     }
29     Poly& operator=(const Poly rhs) {
30         this->len = rhs.len;
31         this->deg = rhs.deg;
32         this->a = (ll*)realloc(this->a, sizeof(ll) * len);
33         copy(rhs.a, rhs.a + len, this->a);
34         return *this;
35     }
36     Poly operator*(Poly rhs) {
37         int l1 = this->len, l2 = rhs.len;
38         int d1 = this->deg, d2 = rhs.deg;
39         while (l1 > 0 and this->a[l1 - 1] == 0) l1--;
40         while (l2 > 0 and rhs.a[l2 - 1] == 0) l2--;
41         int l = 1;
42         while (l < max(l1 + l2 - 1, d1 + d2 + 1)) l <<= 1;
43         int *x, *y, *res;
44         x = (ll*) calloc(l, sizeof(ll));
45         y = (ll*) calloc(l, sizeof(ll));
46         res = (ll*) calloc(l, sizeof(ll));
47         copy(this->a, this->a + l1, x);
48         copy(rhs.a, rhs.a + l2, y);
```

```
49         ntt.tran(l, x); ntt.tran(l, y);
50         FOR (i, 0, l - 1)
51             res[i] = x[i] * y[i] % mod;
52         ntt.tran(l, res, true);
53         free(x); free(y);
54         return Poly(l, d1 + d2, res);
55     }
56     Poly operator+(Poly rhs) {
57         int l1 = this->len, l2 = rhs.len;
58         int l = max(l1, l2);
59         Poly res;
60         res.len = l;
61         res.deg = max(this->deg, rhs.deg);
62         res.a = (ll*) calloc(l, sizeof(ll));
63         FOR (i, 0, l1 - 1) {
64             res.a[i] += this->a[i];
65             if (res.a[i] >= mod) res.a[i] -= mod;
66         }
67         FOR (i, 0, l2 - 1) {
68             res.a[i] += rhs.a[i];
69             if (res.a[i] >= mod) res.a[i] -= mod;
70         }
71         return res;
72     }
73     Poly operator-(Poly rhs) {
74         int l1 = this->len, l2 = rhs.len;
75         int l = max(l1, l2);
76         Poly res;
77         res.len = l;
78         res.deg = max(this->deg, rhs.deg);
79         res.a = (ll*) calloc(l, sizeof(ll));
80         FOR (i, 0, l1 - 1) {
81             res.a[i] += this->a[i];
82             if (res.a[i] >= mod) res.a[i] -= mod;
83         }
84         FOR (i, 0, l2 - 1) {
85             res.a[i] -= rhs.a[i];
86             if (res.a[i] < 0) res.a[i] += mod;
87         }
88         return res;
89     }
90     Poly operator*(const int rhs) {
91         Poly res;
92         res = *this;
93         FOR (i, 0, res.len - 1) {
94             res.a[i] = res.a[i] * rhs % mod;
95             if (res.a[i] < 0) res.a[i] += mod;
96         }
97         return res;
98     }
99     Poly(vector<int> f) {
100        int _n = f.size();
101        len = 1;
102        deg = _n - 1;
103        while (len < _n) len <<= 1;
104        a = (ll*) calloc(len, sizeof(ll));
105        FOR (i, 0, deg) a[i] = f[i];
106    }
107    Poly derivative() {
108        Poly g(this->deg);
109        FOR (i, 1, this->deg) {
110            g.a[i - 1] = this->a[i] * i % mod;
111        }
112        return g;
113    }
114    Poly integral() {
```

```
115        Poly g(this->deg + 2);
116        FOR (i, 0, this->deg) {
117            g.a[i + 1] = this->a[i] * ::inv(i + 1) % mod;
118        }
119        return g;
120    }
121    Poly inv(int len1 = -1) {
122        if (len1 == -1) len1 = this->len;
123        Poly g(1); g.a[0] = ::inv(a[0]);
124        for (int l = 1; l < len1; l <<= 1) {
125            Poly t; t = *this;
126            t.resize(l << 1);
127            t = g * g * t;
128            t.resize(l << 1);
129            Poly g1 = g * 2 - t;
130            swap(g, g1);
131        }
132        return g;
133    }
134    Poly ln(int len1 = -1) {
135        if (len1 == -1) len1 = this->len;
136        auto g = *this;
137        auto x = g.derivative() * g.inv(len1);
138        x.resize(len1);
139        x = x.integral();
140        x.resize(len1);
141        return x;
142    }
143    Poly exp() {
144        Poly g(1);
145        g.a[0] = 1;
146        for (int l = 1; l < len; l <<= 1) {
147            Poly t, g1; t = *this;
148            t.resize(l << 1); t.a[0]++;
149            g1 = (t - g.ln(l << 1)) * g;
150            g1.resize(l << 1);
151            swap(g, g1);
152        }
153        return g;
154    }
155    Poly pow(ll n) {
156        Poly &a = *this;
157        int i = 0;
158        while (i <= a.deg and a.a[i] == 0) i++;
159        if (i and (n > a.deg or n * i > a.deg)) return Poly(a
               .deg + 1);
160        if (i == a.deg + 1) {
161            Poly res(a.deg + 1);
162            res.a[0] = 1;
163            return res;
164        }
165        Poly b(a.deg - i + 1);
166        int inv1 = ::inv(a.a[i]);
167        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]

```cpp
// n 個人，每 k 個人就刪除的約瑟夫遊戲
int josephus(int n, int k) {
    if (n == 1)
        return 0;
    if (k == 1)
        return n-1;
    if (k > n)
        return (josephus(n-1, k) + k) % n;
    int cnt = n / k;
    int res = josephus(n - cnt, k);
    res -= n % k;
    if (res < 0)
        res += n;
    else
        res += res / (k - 1);
    return res;
}
```

7.13 數論分塊 [8ccab5]

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

7.14 最大質因數 [ca5e52]

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

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

7.15 歐拉公式 [85f3b1]

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

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

    return ret;
}

// O(n log n)，回傳 1~n 的 phi 值
vector<int> phi_1_to_n(int n){
```

```cpp
    vector<int> phi(n+1);
    phi[0]=0;
    phi[1]=1;

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

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

    return phi;
}
```

7.16 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

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

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

方法：先把所有自環刪掉，定義 Q 為以下矩陣

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

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

7.19 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 [018290]

```cpp
struct ACAutomation{
    vector<vector<int>> go;
    vector<int> fail, match, pos;
    int sz = 0; // 有效節點為 [0, sz]，開陣列的時候要小
                // 心！！！

    ACAutomation(int n) : go(n, vector<int>(26)), fail(n),
        match(n) {}

    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'];
        }
        pos.push_back(now);
    }

    void build(){
        queue<int> que;
        for (int i=0 ; i<26 ; i++){
            if (go[0][i]) que.push(go[0][i]);
        }
        while (que.size()){
            int u = que.front();
            que.pop();
            for (int i=0 ; i<26 ; i++){
                if (go[u][i]){
                    fail[go[u][i]] = go[fail[u]][i];
                    que.push(go[u][i]);
                }else go[u][i] = go[fail[u]][i];
            }
        }
    }
}
```

```cpp
33        // counting pattern
34        void buildMatch(string &s){
35            int now = 0;
36            for (char c : s){
37                now = go[now][c-'a'];
38                match[now]++;
39            }
40
41            vector<int> in(sz+1), que;
42            for (int i=1 ; i<=sz ; i++) in[fail[i]]++;
43            for (int i=1 ; i<=sz ; i++) if (in[i]==0) que.
44                push_back(i);
45            for (int i=0 ; i<que.size() ; i++){
46                int now = que[i];
47                match[fail[now]] += match[now];
48                if (--in[fail[now]]==0) que.push_back(fail[now]);
49            }
50        }
51    };
```

8.2 Hash [942f42]

```cpp
1  mt19937 seed(chrono::steady_clock::now().time_since_epoch().
2      count());
3  int rng(int l, int r){
4      return uniform_int_distribution<int>(l, r)(seed);
5  }
6  int A = rng(1e5, 8e8);
7  const int B = 1e9+7;
8
9  // 2f6192
10 struct RollingHash{
11     vector<int> Pow, Pre;
12     RollingHash(string s = ""){
13         Pow.resize(s.size());
14         Pre.resize(s.size());
15
16         for (int i=0 ; i<s.size() ; i++){
17             if (i==0){
18                 Pow[i] = 1;
19                 Pre[i] = s[i];
20             }else{
21                 Pow[i] = Pow[i-1]*A%B;
22                 Pre[i] = (Pre[i-1]*A+s[i])%B;
23             }
24         }
25
26         return;
27     }
28
29     int get(int l, int r){ // 取得 [l, r] 的數值
30         if (l==0) return Pre[r];
31         int res = (Pre[r]-Pre[l-1]*Pow[r-l+1])%B;
32         if (res<0) res += B;
33         return res;
34     }
35 };
```

8.3 KMP [7b95d6]

```cpp
1  // KMP[i] = s[0...i] 的最長共同前後綴長度，KMP[KMP[i]-1] 可以
2      跳 fail link
3  // e5b7ce
```

```cpp
3  vector<int> KMP(string &s){
4      vector<int> ret(n);
5      for (int i=1 ; i<s.size() ; i++){
6          int j = ret[i-1];
7          while (j && s[i]!=s[j]) j = ret[j-1];
8          ret[i] = j + (s[i]==s[j]);
9      }
10     return ret;
11 }
```

8.4 Manacher [9a4b4d]

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

8.5 Min Rotation [b24786]

```cpp
1  int minRotation(string s) {
2      int a = 0, n = s.size();
3      s += s;
4
5      for (int b=0 ; b<n ; b++){
6          for (int k=0 ; k<n ; k++){
7              if (a+k==b || s[a+k]<s[b+k]){
8                  b += max(0LL, k-1);
9                  break;
10             }
11             if (s[a+k]>s[b+k]){
12                 a = b;
13                 break;
14             }
15         }
16     }
17
18     return a;
19 }
```

8.6 Suffix Array [77b5fd]

```cpp
1  // 注意，當 |s|=1 時，lcp 不會有值，務必測試 |s|=1 的 case
2  struct SuffixArray {
3      string s;
4      vector<int> sa, lcp;
5
6      // 69ced9
7      // lim 要調整成字元集大小，_s 不可以有 0
8      SuffixArray(string _s, int lim = 256) {
9          s = _s;
10         int n = s.size()+1, k = 0, a, b;
11         vector<int> x(s.begin(), s.end()), y(n), ws(max(n,
12             lim)), rank(n);
13         x.push_back(0);
14         sa = lcp = y;
15         iota(sa.begin(), sa.end(), 0);
16         for (int j=0, p=0 ; p<n ; j=max(1LL, j*2), lim=p) {
17             p = j;
18             iota(y.begin(), y.end(), n-j);
19             for (int i=0 ; i<n ; i++) if (sa[i] >= j) y[p++]
20                 = sa[i] - j;
21             fill(ws.begin(), ws.end(), 0);
22             for (int i=0 ; i<n ; i++) ws[x[i]]++;
23             for (int i=1 ; i<lim ; i++) ws[i] += ws[i - 1];
24             for (int i = n; i--;) sa[--ws[x[y[i]]]] = y[i];
25             swap(x, y), p = 1, x[sa[0]] = 0;
26             for (int i=1 ; i<n ; i++){
27                 a = sa[i - 1];
28                 b = sa[i];
29                 x[b] = (y[a] == y[b] && y[a + j] == y[b + j])
30                     ? p - 1 : p++;
31             }
32         }
33
34         for (int i=1 ; i<n ; i++) rank[sa[i]] = i;
35         for (int i=0, j ; i<n-1 ; lcp[rank[i++]]=k)
36             for (k && k--, j=sa[rank[i]-1] ; i+k<s.size() &&
37                 j+k<s.size() && s[i+k]==s[j+k] ; k++);
38         sa.erase(sa.begin());
39         lcp.erase(lcp.begin(), lcp.begin()+2);
40     }
41
42     // f49583
43     vector<int> pos; // pos[i] = i 這個值在 pos 的哪個地方
44     SparseTable st;
45     void init_lcp(){
46         pos.resize(sa.size());
47         for (int i=0 ; i<sa.size() ; i++){
48             pos[sa[i]] = i;
49         }
50         if (lcp.size()){
51             st.build(lcp);
52         }
53     }
54
55     // 用之前記得 init
56     // 回傳 [l1, r1] 跟 [l2, r2] 的 lcp，0-based
57     int get_lcp(int l1, int r1, int l2, int r2){
58         int pos_1 = pos[l1], len_1 = r1-l1+1;
59         int pos_2 = pos[l2], len_2 = r2-l2+1;
60         if (pos_1>pos_2){
61             swap(pos_1, pos_2);
62             swap(len_1, len_2);
63         }
64
65         if (l1==l2){
66             return min(len_1, len_2);
67         }else{
68             return min({st.query(pos_1, pos_2), len_1, len_2
69                 });
70         }
71     }
72
73     // 檢查 [l1, r1] 跟 [l2, r2] 的大小關係，0-based
```

```
69   // 如果前者小於後者，就回傳 <0，相等就回傳 =0，否則回傳
          >0
70   // 5b8db0
71   int substring_cmp(int l1, int r1, int l2, int r2){
72       int len_1 = r1-l1+1;
73       int len_2 = r2-l2+1;
74       int res = get_lcp(l1, r1, l2, r2);
75
76       if (res<len_1 && res<len_2){
77           return s[l1+res]-s[l2+res];
78       }else if (len_1==res && len_2==res){
79           return 0;
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 [9d559a]

```
1  // z[i] 回傳 s[0...] 跟 s[i...] 的 lcp, z[0] = 0
2  vector<int> z_function(string s){
3      vector<int> z(s.size());
4      int l = -1, r = -1;
5      for (int i=1 ; i<s.size() ; i++){
6          z[i] = i>=r ? 0 : min(r-i, z[i-l]);
7          while (i+z[i]<s.size() && s[i+z[i]]==s[z[i]]) z[i]++;
8          if (i+z[i]>r) l=i, r=i+z[i];
9      }
10     return z;
11 }
```

8.8 k-th Substring1 [61f66b]

```
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;
11
12         if (nowRank+add>=k){
13             return s.substr(sa[i], prePrefix+k-nowRank);
14         }
15
16         prePrefix = sa.lcp[i];
17         nowRank += add;
18     }
19 }
```