

Contents

1 Misc	2	3.8 Segment Tree Persistent	7	6.19 Theorem	18
1.1 Custom Set PQ Sort	2	3.9 Sparse Table	7	7 Math	19
1.2 Default Code New	2	3.10 Treap	8	7.1 CRT	19
1.3 Default Code Old	2	3.11 Treap2	8	7.2 Josephus Problem	19
1.4 Enumerate Subset	2	3.12 Trie	9	7.3 Lagrange any x	19
1.5 Fast Input	2	4 Dynamic-Programming	9	7.4 Lagrange continuous x	19
1.6 OEIS	2	4.1 Digit DP	9	7.5 Lucas's Theorem	20
1.7 Xor Basis	2	4.2 Knaspack On Tree	9	7.6 Matrix	20
1.8 random int	3	4.3 SOS DP	9	7.7 Matrix 01	20
1.9 Python	3	4.4 Integer Partition	9	7.8 Miller Rabin	20
1.10 diff	3	5 Geometry	10	7.9 Pollard Rho	21
1.11 hash command	3	5.1 Geometry Struct	10	7.10 Polynomial	21
1.12 run	3	5.2 Pick's Theorem	11	7.11 josephus	22
1.13 setup	3	6 Graph	11	7.12 數論分塊	22
2 Convolution	3	6.1 2-SAT	11	7.13 最大質因數	22
2.1 FFT any mod	3	6.2 Augment Path	12	7.14 歐拉公式	22
2.2 FFT new	4	6.3 Cut BCC	12	7.15 Burnside's Lemma	22
2.3 FFT short	4	6.4 Dinic	12	7.16 Catalan Number	22
2.4 FWT	4	6.5 Dominator Tree	13	7.17 Matrix Tree Theorem	22
2.5 NTT mod 998244353	4	6.6 EdgeBCC	13	7.18 Stirling's formula	22
2.6 Min Convolution Concave Concave	5	6.7 Enumerate Triangle	14	7.19 Theorem	22
3 Data-Structure	5	6.8 EnumeratePlanarFace	14	7.20 二元一次方程式	22
3.1 BIT	5	6.9 HLD	14	7.21 歐拉定理	23
3.2 Disjoint Set Persistent	5	6.10 Kosaraju	14	7.22 錯排公式	23
3.3 PBDS GP Hash Table	5	6.11 Kuhn Munkres	15	8 String	23
3.4 PBDS Order Set	5	6.12 LCA	15	8.1 Hash	23
3.5 Segment Tree Add Set	5	6.13 MCMF	16	8.2 KMP	23
3.6 Segment Tree Li Chao Line	6	6.14 Tarjan	16	8.3 Manacher	23
3.7 Segment Tree Li Chao Segment	6	6.15 Tarjan Find AP	16	8.4 Min Rotation	23
		6.16 Tree Isomorphism	17	8.5 Suffix Array	23
		6.17 圓方樹	17	8.6 Z Algorithm	24
		6.18 最大權閉合圖	18	8.7 k-th Substring1	24

Misc

1.1 Custom Set PQ Sort

```

1 // priority_queue · 務必檢查相等的 case · 給所有元素一個排序的
  依據
2 struct cmp{
3     bool operator () (Data a, Data b){
4         return a.x<b.x;
5     }
6 };
7 priority_queue<Data, vector<Data>, cmp> pq;
8
9 // set · 務必檢查相等的 case · 給所有元素一個排序的依據
10 auto cmp = [](int a, int b) {
11     return a > b;
12 };
13 set<int, decltype(cmp)> s = {1, 2, 3, 4, 5};
14 cout << *s.begin() << '\n';

```

1.2 Default Code New

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define int long long
4
5 const int MAX_N = 5e5 + 10;
6 const int INF = 2e18;
7
8 void solve(){
9
10 }
11
12 signed main(){
13     ios::sync_with_stdio(0), cin.tie(0);
14
15     int t = 1;
16     while (t--){
17         solve();
18     }
19
20     return 0;
21 }

```

1.3 Default Code Old

```

1 #include <bits/stdc++.h>
2 #define int long long
3 #define ALL(x) x.begin(), x.end()
4 #define SZ(x) ((int)x.size())
5 #define fastio ios::sync_with_stdio(0), cin.tie(0);
6 using namespace std;
7
8 #ifdef LOCAL
9 #define cout cout << "\033[0;32m"
10 #define cerr cerr << "\033[0;31m"
11 #define endl endl << "\033[0m"

```

```

12 #else
13 #pragma GCC optimize("O3,unroll-loops")
14 #pragma GCC target("avx,avx2,sse,sse2,sse3,sse4,popcnt")
15 #define endl "\n"
16 #endif
17
18 const int MAX_N = 5e5+10;
19 const int INF = 2e18;
20
21 void solve1(){
22
23     return;
24 }
25
26 signed main(){
27
28     fastio;
29
30     int t = 1;
31     while (t--){
32         solve1();
33     }
34
35     return 0;
36 }

```

1.4 Enumerate Subset

```

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

```

1.5 Fast Input

```

1 // fast IO
2 // 6f8879
3 inline char readchar(){
4     static char buffer[BUFSIZ], *now = buffer + BUFSIZ, *
5     end = buffer + BUFSIZ;
6     if (now == end)
7     {
8         if (end < buffer + BUFSIZ)
9             return EOF;
10        end = (buffer + fread(buffer, 1, BUFSIZ, stdin));
11        now = buffer;
12    }
13    return *now++;
14 }
15 inline int nextint(){
16     int x = 0, c = readchar(), neg = false;
17     while(('0' > c || c > '9') && c!='-' && c!=EOF) c =
18         readchar();
19     if(c == '-') neg = true, c = readchar();
20     while('0' <= c && c <= '9') x = (x<<3) + (x<<1) + (c^'0')
21     , c = readchar();

```

```

19     if(neg) x = -x;
20     return x; // returns 0 if EOF
21 }

```

1.6 OEIS

```

1 // 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
2 // f915c2
3 template <typename T>
4 vector<T> BerlekampMassey(vector<T> a) {
5     auto scalarProduct = [](vector<T> v, T c) {
6         for (T &x: v) x *= c;
7         return v;
8     };
9     vector<T> s, best;
10    int bestPos = 0;
11    for (size_t i = 0; i < a.size(); i++) {
12        T error = a[i];
13        for (size_t j = 0; j < s.size(); j++) error -= s[j] *
14            a[i-1-j];
15        if (error == 0) continue;
16        if (s.empty()) {
17            s.resize(i + 1);
18            bestPos = i;
19            best.push_back(1 / error);
20            continue;
21        }
22        vector<T> fix = scalarProduct(best, error);
23        fix.insert(fix.begin(), i - bestPos - 1, 0);
24        if (fix.size() >= s.size()) {
25            best = scalarProduct(s, - 1 / error);
26            best.insert(best.begin(), 1 / error);
27            bestPos = i;
28            s.resize(fix.size());
29        }
30        for (size_t j = 0; j < fix.size(); j++)
31            s[j] += fix[j];
32    }
33    return s;

```

1.7 Xor Basis

```

1 vector<int> basis;
2 void add_vector(int x){
3     for (auto v : basis){
4         x=min(x, x^v);
5     }
6     if (x) basis.push_back(x);
7 }
8
9 // 給一數字集合 S · 求能不能 XOR 出 x
10 bool check(int x){
11     for (auto v : basis){
12         x=min(x, x^v);
13     }
14     return 0;
15 }
16

```

```

17 // 給一數字集合 S · 求能 XOR 出多少數字
18 // 答案等於 2^{basis 的大小}
19
20 // 給一數字集合 S · 求 XOR 出最大的數字
21 int get_max(){
22     int ans=0;
23     for (auto v : basis){
24         ans=max(ans, ans^v);
25     }
26     return ans;
27 }

```

1.8 random int

```

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 }

```

1.9 Python

```

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

```

1.10 diff

```

1 set -e
2 g++ ac.cpp -o ac
3 g++ wa.cpp -o wa
4 for ((i=0;;i++))
5 do

```

```

6     echo "$i"
7     python3 gen.py > input
8     ./ac < input > ac.out
9     ./wa < input > wa.out
10    diff ac.out wa.out || break
11 done

```

1.11 hash command

```

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

```

1.12 run

```

1 from os import *
2
3 f = "pA"
4
5 while 1:
6     i = input("input: ")
7     system("clear")
8     p = listdir(".")
9     if i != "":
10        f = i
11        print(f"file = {f}")
12        if system(f"g++ {f}.cpp -std=c++17 -Wall -Wextra -Wshadow
          -O2 -D LOCAL -g -fsanitize=undefined,address -o {f}
          "):
13            print("CE")
14            continue
15
16        for x in sorted(p):
17            if f in x and ".in" in x:
18                print(x)
19                if system(f"./{f} < {x}"):
20                    print("RE")
21                print()

```

1.13 setup

```

1 se nu rnu bs=2 sw=4 ts=4 hls ls=2 si acd bo=all mouse=a
2
3 :inoremap " ""<Esc>i
4 :inoremap {<CR> {<CR><Esc>ko
5 :inoremap [{ {<Esc>i
6
7 function! F(...)
8     execute '!./%:r < ./' . a:1
9 endfunction
10 command! -nargs=* R call F(<f-args>)
11
12 map <F7> :w<bar>!g++ "%" -o %:r -std=c++17 -Wall -Wextra -
  Wshadow -O2 -DLOCAL -g -fsanitize=undefined,address<CR>
13 map <F8> :!./%:r<CR>
14 map <F9> :!./%:r < ./%:r.in<CR>
15

```

```

16 ca hash w !cpp -dD -P -fpreprocessed \ | tr -d "[:space:]" \ |
  md5sum \ | cut -c-6
17
18 " i+<esc>25A---+<esc>
19 " o|<esc>25A |<esc>
20 " ggVGyG35pGdd

```

2 Convolution

2.1 FFT any mod

```

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 vector<int> res(a.size()+b.size()-1);
49 int B = 32-__builtin_clz(res.size()), n = (1<<B), cut =
50     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(
70         outs[i])+0.5);
71     int bv = (int)(imag(outl[i])+0.5) + (int)(real(outs[i]
72         )+0.5);
73     res[i] = ((av%MOD*cut+bv) % MOD*cut+cv) % MOD;
74 }
75 return res;
76 }

```

2.2 FFT new

```

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(1.0/L, acos(-1.0/L) / 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]*
30            y[0]);
31        a[i+j+k] = a[i+j]-z;
32        a[i+j] += z;
33    }
34 }
35 return;
36 }
37
38 // 39029d
39 vector<double> PolyMul(const vector<double> a, const vector<
40     double> b){
41     if (a.empty() || b.empty()) return {};
42     vector<double> res(a.size()+b.size()-1);
43     int L = 32 - __builtin_clz(res.size()), n = 1 << L;
44     vector<cd> in(n), out(n);
45
46     copy(a.begin(), a.end(), begin(in));
47     for (int i=0 ; i<b.size() ; i++){
48         in[i].imag(b[i]);
49     }
50     FFT(in);
51     for (cd& x : in) x *= x;
52     for (int i=0 ; i<n ; i++){
53         out[i] = in[-i & (n - 1)] - conj(in[i]);
54     }
55     FFT(out);
56
57     for (int i=0 ; i<res.size() ; i++){
58         res[i] = imag(out[i]) / (4 * n);
59     }
60
61     return res;

```

2.3 FFT short

```

1 #define int long long
2
3 using Cplx = complex<double>;
4 const double pi = acos(-1);
5 const int mod = 998244353, g = 3;
6 int power(int a, int b) {
7     int res = 1;
8     while (b) {
9         if (b & 1) res = res * a % mod;
10        a = a * a % mod;
11        b >>= 1;
12    }
13    return res;
14 }
15 int inv(int x) { return power(x, mod - 2); }
16 // FFT use Cplx, NTT use LL
17 void FFT(vector<int> &a, int n, int op) {
18     // n must be 2^k
19     vector<int> R(n);
20     FOR (i, 0, n - 1)
21         R[i] = R[i/2]/2 + (i&1)*(n/2);
22     FOR (i, 0, n - 1)
23         if (i < R[i]) swap(a[i], a[R[i]]);
24     for (int m = 2; m <= n; m *= 2) {
25         // Cplx w1({cos(2*pi/m), sin(2*pi/m)*op});

```

```

26     int w1 = power(g, (mod-1)/m * op + mod-1);
27     for (int i = 0; i < n; i += m) {
28         // Cplx wk({1, 0});
29         int wk = 1;
30         FOR (k, 0, m / 2 - 1) {
31             auto x = a[i+k], y = a[i+k+m/2] * wk % mod;
32             a[i+k] = (x+y) % mod;
33             a[i+k+m/2] = (x-y+mod) % mod;
34             wk = wk * w1 % mod;
35         }
36     }
37 }
38 if (op == -1)
39     FOR (i, 0, n - 1) {
40         // a[i] = a[i] / n;
41         a[i] = a[i] * inv(n) % mod;
42     }
43 }

```

2.4 FWT

```

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

```

2.5 NTT mod 998244353

```

1 const int MOD = (119 << 23) + 1, ROOT = 62; // = 998244353
2 // For p < 2^30 there is also e.g. 5 << 25, 7 << 26, 479 <<
3 // 21
4 // and 483 << 21 (same root). The last two are > 10^9.
5 // 9cd58a
6 void NTT(vector<int> &a) {
7     int n = a.size();
8     int L = 31-__builtin_clz(n);
9     vector<int> rt(2, 1);
10    for (int k=2, s=2; k<n ; k*=2, s++){
11        rt.resize(n);

```

```

12     int z[] = {1, qp(ROOT, MOD>>s)};
13     for (int i=k ; i<2*k ; i++){
14         rt[i] = rt[i/2]*z[i&1]%MOD;
15     }
16 }
17
18 vector<int> rev(n);
19 for (int i=0 ; i<n ; i++){
20     rev[i] = (rev[i/2]|(i&1)<<L)/2;
21 }
22 for (int i=0 ; i<n ; i++){
23     if (i<rev[i]){
24         swap(a[i], a[rev[i]]);
25     }
26 }
27
28 for (int k=1 ; k<n ; k*=2){
29     for (int i=0 ; i<n ; i+=2*k){
30         for (int j=0 ; j<k ; j++){
31             int z = rt[j+k]*a[i+j+k]%MOD, &ai = a[i+j];
32             a[i+j+k] = ai-z+(z>ai ? MOD : 0);
33             ai += (ai+z>MOD ? z-MOD : z);
34         }
35     }
36 }
37 }
38
39 // 0b0e99
40 vector<int> polyMul(vector<int> &a, vector<int> &b){
41     if (a.empty() || b.empty()) return {};
42     int s = a.size()+b.size()-1, B = 32-__builtin_clz(s), n =
43         1<<B;
44     int inv = qp(n, MOD-2);
45
46     vector<int> L(a), R(b), out(n);
47     L.resize(n), R.resize(n);
48     NTT(L), NTT(R);
49     for (int i=0 ; i<n ; i++){
50         out[-i&(n-1)] = L[i]*R[i]%MOD*inv%MOD;
51     }
52     NTT(out);
53
54     out.resize(s);
55     return out;
56 }

```

2.6 Min Convolution Concave Concave

```

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 }

```

3 Data-Structure

3.1 BIT

```

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

```

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

```

```

39                arr.update_version(b, a, arr.version.size()
40                    (-1);
41                sz.update_version(a, sz1+sz2, arr.version.
42                    size()-1);
43            }
44            return true;
45        }
46        return false;
47    }
48 };

```

3.3 PBDS GP Hash Table

```

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

```

3.4 PBDS Order Set

```

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

```

3.5 Segment Tree Add Set

```

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        if (rr-ll==1) return;
65        if (ql<=ll && rr<=qr){
66            arr[idx].add_tag += val;
67            push(idx, ll, rr);
68        }

```

```

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        int rr=n){
79            push(idx, ll, rr);
80            if (rr<=ql || qr<=ll) return;
81            if (ql<=ll && rr<=qr){
82                arr[idx].add_tag = 0;
83                arr[idx].set_tag = val;
84                push(idx, ll, rr);
85                return;
86            }
87            int mid = (ll+rr)/2;
88            set(ql, qr, val, idx*2+1, ll, mid);
89            set(ql, qr, val, idx*2+2, mid, rr);
90            arr[idx] = pull(arr[idx*2+1], arr[idx*2+2]);
91        }
92
93        node query(int ql, int qr, int idx = 0, int ll = 0, int rr = n){
94            rr = n){
95                push(idx, ll, rr);
96                if (rr<=ql || qr<=ll) return node();
97                if (ql<=ll && rr<=qr) return arr[idx];
98
99                int mid = (ll+rr)/2;
100               return pull(query(ql, qr, idx*2+1, ll, mid), query(ql, qr, idx*2+2, mid, rr));
101           }
102       } ST;

```

3.6 Segment Tree Li Chao Line

```

1 /*
2 全部都是  $\theta$ -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

```

1 /*
2 全部都是  $\theta$ -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 =
29     MAX_V){
30     if (rr-ll==0) return;
31     if (rr-ll<=1){
32         if (val.y(ll)<arr[idx].y(ll)){
33             arr[idx] = val;
34         }
35         return;
36     }
37     int mid = (ll+rr)/2;
38     if (arr[idx].a > val.a) swap(arr[idx], val); // 原本
39     // 的線斜率要比較小
40     if (arr[idx].y(mid) < val.y(mid)){ // 交點在左邊
41         update(val, idx*2+1, ll, mid);
42     }else{ // 交點在右邊
43         swap(arr[idx], val); // 在左子樹中，新線比舊線還
44         // 要好
45         update(val, idx*2+2, mid, rr);
46     }
47     return;
48 }
49 // 在 [ql, qr] 加上一條 val 的線段
50 void update_segment(Node val, int ql, int qr, int idx =
51     0, int ll = 0, int rr = MAX_V){
52     if (rr-ll==0) return;
53     if (rr<=ql || qr<=ll) return;
54     if (ql<=ll && rr<=qr){
55         update(val, idx, ll, rr);
56         return;
57     }
58     int mid = (ll+rr)/2;
59     update_segment(val, ql, qr, idx*2+1, ll, mid);
60     update_segment(val, ql, qr, idx*2+2, mid, rr);
61     return;
62 }
63
64 int query(int x, int idx = 0, int ll = 0, int rr = MAX_V)
65 {
66     if (rr-ll==0) return INF;
67     if (rr-ll==1){
68         return arr[idx].y(ll);
69     }
70     int mid = (ll+rr)/2;
71     if (x<mid){
72         return min(arr[idx].y(x), query(x, idx*2+1, ll,

```

```

73         return min(arr[idx].y(x), query(x, idx*2+2, mid,
74             rr));
75     }
76 };

```

3.8 Segment Tree Persistent

```

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 =
35         n){
36         auto &now = arr[idx];
37
38         if (rr-ll==1){
39             now.val = v[ll];
40             return;
41         }
42
43         int mid = (ll+rr)/2;
44         now.lc = node_cnt++;
45         now.rc = node_cnt++;
46         build(v, now.lc, ll, mid);
47         build(v, now.rc, mid, rr);
48         pull(now, arr[now.lc], arr[now.rc]);
49         return;
50     }
51
52     void update(int pos, int val, int idx, int ll = 0, int rr
53         = n){
54         auto &now = arr[idx];

```

```

55         if (rr-ll==1){
56             now.val = val;
57             return;
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 =
80         n){
81         auto &now = arr[idx];
82
83         if (ql<=ll && rr<=qr) return now;
84         if (rr<=ql || qr<=ll) return Node();
85
86         int mid = (ll+rr)/2;
87
88         Node ret;
89         pull(ret, query(ql, qr, now.lc, ll, mid), query(ql,
90             qr, now.rc, mid, rr));
91         return ret;
92     }
93
94     Node query_version(int ql, int qr, int ver){
95         return query(ql, qr, version[ver]);
96     }
97
98     void clone_version(int ver){
99         version.push_back(node_cnt);
100         arr[node_cnt] = arr[version[ver]];
101         node_cnt++;
102     }
103 };

```

3.9 Sparse Table

```

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 Treap

```

1 struct Treap{
2     Treap *l = nullptr, *r = nullptr;
3     int pri = rand(), val = 0, sz = 1;
4
5     Treap(int _val){
6         val = _val;
7     }
8 };
9
10 int size(Treap *t){return t ? t->sz : 0;}
11 void pull(Treap *t){
12     t->sz = size(t->l)+size(t->r)+1;
13 }
14
15 Treap* merge(Treap *a, Treap *b){
16     if (!a || !b) return a ? a : b;
17
18     if (a->pri>b->pri){
19         a->r = merge(a->r, b);
20         pull(a);
21         return a;
22     }else{
23         b->l = merge(a, b->l);
24         pull(b);
25         return b;
26     }
27 }
28
29 pair<Treap*, Treap*> split(Treap *&t, int k){ // 1-based <前
30     k 個元素, 其他元素>
31     if (!t) return {};
32     if (size(t->l)>=k){
33         auto pa = split(t->l, k);
34         t->l = pa.second;
35         pull(t);
36         return {pa.first, t};
37     }else{
38         auto pa = split(t->r, k-size(t->l)-1);
39         t->r = pa.first;
40         pull(t);
41         return {t, pa.second};
42     }
43 }
44
45 // functions

```

```

46 Treap* build(vector<int> v){
47     Treap* ret = nullptr;
48     for (int i=0 ; i<v.size() ; i++){
49         ret = merge(ret, new Treap(v[i]));
50     }
51     return ret;
52 }
53
54 array<Treap*, 3> cut(Treap *t, int l, int r){ // 1-based <前
55     1~l-1 個元素, l~r 個元素, r+1 個元素>
56     array<Treap*, 3> ret;
57     tie(ret[1], ret[2]) = split(t, r);
58     tie(ret[0], ret[1]) = split(ret[1], l-1);
59     return ret;
60 }
61
62 void print(Treap *t, bool flag = true){
63     if (t->l!=0) print(t->l, false);
64     cout << t->val;
65     if (t->r!=0) print(t->r, false);
66     if (flag) cout << endl;
67 }

```

3.11 Treap2

```

1 // 1-based · 請注意 MAX_N 是否足夠大
2 int root = 0;
3 int lc[MAX_N], rc[MAX_N];
4 int pri[MAX_N], val[MAX_N];
5 int sz[MAX_N], tag[MAX_N], fa[MAX_N];
6 int new_node(int v){
7     static int nodeCnt = 0;
8     nodeCnt++;
9     val[nodeCnt] = v;
10    sz[nodeCnt] = 1;
11    pri[nodeCnt] = rand();
12    return nodeCnt;
13 }
14
15 void push(int x){
16     if (tag[x]){
17         if (lc[x]) tag[lc[x]] ^= 1;
18         if (rc[x]) tag[rc[x]] ^= 1;
19     }
20     tag[x] = 0;
21 }
22
23 int pull(int x){
24     if (x){
25         fa[x] = 0;
26         sz[x] = 1+sz[lc[x]]+sz[rc[x]];
27         if (lc[x]) fa[lc[x]] = x;
28         if (rc[x]) fa[rc[x]] = x;
29     }
30     return x;
31 }
32
33 int merge(int a, int b){
34     if (!a or !b) return a|b;
35     push(a), push(b);
36
37     if (pri[a]>pri[b]){
38         rc[a] = merge(rc[a], b);

```

```

38     return pull(a);
39 }else{
40     lc[b] = merge(a, lc[b]);
41     return pull(b);
42 }
43 }
44
45 // [1, k] [k+1, n]
46 void split(int x, int k, int &a, int &b) {
47     if (!x) return a = b = 0, void();
48     push(x);
49     if (sz[lc[x]] >= k) {
50         split(lc[x], k, a, lc[x]);
51         b = x;
52         pull(a); pull(b);
53     }else{
54         split(rc[x], k - sz[lc[x]] - 1, rc[x], b);
55         a = x;
56         pull(a); pull(b);
57     }
58 }
59
60 // functions
61 // 回傳 x 在 Treap 中的位置
62 int get_pos(int x){
63     vector<int> sta;
64     while (fa[x]){
65         sta.push_back(x);
66         x = fa[x];
67     }
68     while (sta.size()){
69         push(x);
70         x = sta.back();
71         sta.pop_back();
72     }
73     push(x);
74
75     int res = sz[x] - sz[rc[x]];
76     while (fa[x]){
77         if (rc[fa[x]]==x){
78             res += sz[fa[x]]-sz[x];
79         }
80         x = fa[x];
81     }
82     return res;
83 }
84
85 // 1-based <前 [1, l-1] 個元素, [l, r] 個元素, [r+1, n] 個元素>
86 array<int, 3> cut(int x, int l, int r){
87     array<int, 3> ret;
88     split(x, r, ret[1], ret[2]);
89     split(ret[1], l-1, ret[0], ret[1]);
90     return ret;
91 }
92
93 void print(int x){
94     push(x);
95     if (lc[x]) print(lc[x]);
96     cerr << val[x] << " ";
97     if (rc[x]) print(rc[x]);
98 }

```


3.12 Trie

```

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

```

4 Dynamic-Programming

4.1 Digit DP

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 long long l, r;
5 long long dp[20][10][2][2]; // dp[pos][pre][limit] = 後 pos
6 // 位 · pos 前一位是 pre · (是/否) 有上界 · (是/否) 有前綴零
7 // 的答案數量
8
9 long long memorize_search(string &s, int pos, int pre, bool
10 limit, bool lead){
11
12     // 已經被找過了 · 直接回傳值

```

```

10     if (dp[pos][pre][limit][lead]!=-1) return dp[pos][pre][
11         limit][lead];
12
13     // 已經搜尋完畢 · 紀錄答案並回傳
14     if (pos==(int)s.size()){
15         return dp[pos][pre][limit][lead] = 1;
16     }
17
18     // 枚舉目前的位數數字是多少
19     long long ans = 0;
20     for (int now=0 ; now<=(limit ? s[pos]-'0' : 9) ; now++){
21         if (now==pre){
22
23             // 1~9 絕對不能連續出現
24             if (pre!=0) continue;
25
26             // 如果已經不在前綴零的範圍內 · 0 不能連續出現
27             if (lead==false) continue;
28
29             ans += memorize_search(s, pos+1, now, limit&(now==(s[
30                 pos]-'0')), lead&(now==0));
31         }
32
33         // 已經搜尋完畢 · 紀錄答案並回傳
34         return dp[pos][pre][limit][lead] = ans;
35     }
36
37     // 回傳 [0, n] 有多少數字符合條件
38     long long find_answer(long long n){
39         memset(dp, -1, sizeof(dp));
40         string tmp = to_string(n);
41
42         return memorize_search(tmp, 0, 0, true, true);
43     }
44
45     int main(){
46
47         // input
48         cin >> l >> r;
49
50         // output - 計算 [l, r] 有多少數字任意兩個位數都不相同
51         cout << find_answer(r)-find_answer(l-1) << "\n";
52
53         return 0;

```

4.2 Knapsack On Tree

```

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

```

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

```

4.4 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

```

1 // 判斷數值正負：{1:正數,0:零,-1:負數}
2 int sign(long long x) {return (x >= 0) ? ((bool)x) : -1; }
3 int sign(double x) {
4     return (abs(x) < 1e-9) ? 0 : (x > 0 ? 1 : -1);
5 }
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
13    point operator+(point b) {return {x+b.x, y+b.y}; }
14    point operator-(point b) {return {x-b.x, y-b.y}; }
15    point operator*(T b) {return {x*b, y*b}; }
16    point operator/(T b) {return {x/b, y/b}; }
17    bool operator==(point b) {return x==b.x && y==b.y; }
18    // 逆時針極角排序
19    bool operator<(point &b) {return (x*b.y > b.x*y); }
20    friend ostream& operator<<(ostream& os, point p) {
21        os << "(" << p.x << ", " << p.y << ")";
22        return os;
23    }
24    // 判斷 ab 到 ac 的方向：{1:逆時鐘,0:重疊,-1:順時鐘}
25    friend int ori(point a, point b, point c) {
26        return sign((b-a)^(c-a));
27    }
28    friend bool btw(point a, point b, point c) {
29        return ori(a, b, c) == 0 && sign((a-c)*(b-c)) <= 0;
30    }
31    // 判斷線段 ab, cd 是否相交
32    friend bool banana(point a, point b, point c, point d) {
33        int s1 = ori(a, b, c);
34        int s2 = ori(a, b, d);
35        int s3 = ori(c, d, a);
36        int s4 = ori(c, d, b);
37        if (btw(a, b, c) || btw(a, b, d) || btw(c, d, a) ||
38            btw(c, d, b)) return 1;
39        return (s1 * s2 < 0) && (s3 * s4 < 0);
40    }
41    T operator*(point b) {return x * b.x + y * b.y; }
42    T operator^(point b) {return x * b.y - y * b.x; }
43    T abs2() {return (*this) * (*this); }
44
45    // 旋轉 Arg(b) 的角度 (小心溢位)
46    point rotate(point b) {return {x*b.x - y*b.y, x*b.y + y*b
47        .x}; }
48
49 template<typename T>
50 struct line {
51    point<T> p1, p2;
52    // ax + by + c = 0
53    T a, b, c; // |a|, |b| ≤ 2C, |c| ≤ 8C²
54    line() {}
55    line(const point<T> &x, const point<T> &y) : p1(x), p2(y){
56        build();
57    }
58
59    void build() {
60        a = p1.y - p2.y;
61        b = p2.x - p1.x;
62        c = (-a*p1.x)-b*p1.y;
63    }
64    // 判斷點和有向直線的關係：{1:左邊,0:在線上,-1:右邊}
65    int ori(point<T> &p) {
66        return sign((p2-p1) ^ (p-p1));
67    }
68    // 判斷直線斜率是否相同
69    bool parallel(line &l) {
70        return ((p1-p2) ^ (l.p1-l.p2)) == 0;
71    }
72    // 兩直線交點
73    point<long double> line_intersection(line &l) {
74        using P = point<long double>;
75        point<T> a = p2-p1, b = l.p2-l.p1, s = l.p1-p1;
76        return P(p1.x,p1.y) + P(a.x,a.y) * (((long double)(s^b))
77            / (a^b));
78    }
79 };
80
81 template<typename T>
82 struct polygon {
83    vector<point<T>> v;
84    polygon() {}
85    polygon(const vector<point<T>> &u) : v(u) {}
86    // simple 為 true 的時候會回傳任意三點不共線的凸包
87    void make_convex_hull(int simple) {
88        auto cmp = [&](point<T> &p, point<T> &q) {
89            return (p.x == q.x) ? (p.y < q.y) : (p.x < q.x);
90        };
91        simple = (bool)simple;
92        sort(v.begin(), v.end(), cmp);
93        v.resize(unique(v.begin(), v.end()) - v.begin());
94        vector<point<T>> hull;
95        for (int t = 0; t < 2; ++t){
96            int sz = hull.size();
97            for (auto &i:v) {
98                while (hull.size() >= sz+2 && ori(hull[hull.
99                    size()-2], hull.back(), i) < simple) {
100                    hull.pop_back();
101                }
102                hull.push_back(i);
103            }
104            hull.pop_back();
105            reverse(v.begin(), v.end());
106        }
107        swap(hull, v);
108    }
109    // 可以在有 n 個點的簡單多邊形內 · 用 O(n) 判斷一個點：
110    // {1 : 在多邊形內, 0 : 在多邊形上, -1 : 在多邊形外}
111    int in_polygon(point<T> a){
112        const T MAX_POS = 1e9 + 5; // [記得修改] 座標的最大值
113        point<T> pre = v.back(), b(MAX_POS, a.y + 1);
114        int cnt = 0;
115
116        for (auto &i:v) {
117            if (btw(pre, i, a)) return 0;
118            if (banana(a, b, pre, i)) cnt++;
119            pre = i;
120        }
121
122        return cnt%2 ? 1 : -1;
123    }
124
125    // 警告：以下所有凸包專用的函式都只接受逆時針排序且任三點不
126    // 共線的凸包 ///
127    // 可以在有 n 個點的凸包內 · 用 O(Log n) 判斷一個點：
128    // {1 : 在凸包內, 0 : 在凸包邊上, -1 : 在凸包外}
129    int in_convex(point<T> p) {
130        int n = v.size();
131        int a = ori(v[0], v[1], p), b = ori(v[0], v[n-1], p);
132        if (a < 0 || b > 0) return -1;
133        if (btw(v[0], v[1], p)) return 0;
134        if (btw(v[0], v[n-1], p)) return 0;
135        int l = 1, r = n - 1, mid;
136        while (l + 1 < r) {
137            mid = (l + r) >> 1;
138            if (ori(v[0], v[mid], p) >= 0) l = mid;
139            else r = mid;
140        }
141        int k = ori(v[l], v[r], p);
142        if (k <= 0) return k;
143        return 1;
144    }
145    // 凸包專用的環狀二分搜 · 回傳 0-based index
146    int cycle_search(auto &f) {
147        int n = v.size(), l = 0, r = n;
148        bool rv = f(1, 0);
149        while (r - l > 1) {
150            int m = (l + r) / 2;
151            if (f(0, m) ? rv : f(m, (m + 1) % n)) r = m;
152            else l = m;
153        }
154        return f(1, r % n) ? l : r % n;
155    }
156    // 可以在有 n 個點的凸包內 · 用 O(Log n) 判斷一條直線：
157    // {1 : 穿過凸包, 0 : 剛好切過凸包, -1 : 沒碰到凸包}
158    int line_cut_convex(line<T> L) {
159        L.build();
160        point<T> p(L.a, L.b);
161        auto gt = [&](int neg) {
162            auto f = [&](int x, int y) {
163                return sign((v[x] - v[y]) * p) == neg;
164            };
165            return -(v[cycle_search(f)] * p);
166        };
167        T x = gt(1), y = gt(-1);
168        if (L.c < x || y < L.c) return -1;
169        return not (L.c == x || L.c == y);
170    }
171    // 可以在有 n 個點的凸包內 · 用 O(Log n) 判斷一個線段：
172    // {1 : 存在一個凸包上的邊可以把這個線段切成兩半,
173    // 0 : 有碰到凸包但沒有任何凸包上的邊可以把它切成兩半,
174    // -1 : 沒碰到凸包}
175    // 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
176    // 定表示線段沒有通過凸包內部 ///
177    int segment_across_convex(line<T> L) {
178        L.build();
179        point<T> p(L.a, L.b);
180        auto gt = [&](int neg) {
181            auto f = [&](int x, int y) {
182                return sign((v[x] - v[y]) * p) == neg;
183            };
184            return cycle_search(f);
185        };
186        int i = gt(1), j = gt(-1), n = v.size();
187        T x = -(v[i] * p), y = -(v[j] * p);

```

```

182 if (L.c < x || y < L.c) return -1;
183 if (L.c == x || L.c == y) return 0;
184
185 if (i > j) swap(i, j);
186 auto g = [&](int x, int lim) {
187     int now = 0, nxt;
188     for (int i = 1 << __lg(lim); i > 0; i /= 2) {
189         if (now + i > lim) continue;
190         nxt = (x + i) % n;
191         if (L.ori(v[x]) * L.ori(v[nxt]) >= 0) {
192             x = nxt;
193             now += i;
194         }
195     } // ↓ BE CAREFUL
196     return -(ori(v[x], v[(x + 1) % n], L.p1) * ori(v[
197         x], v[(x + 1) % n], L.p2));
198 };
199 return max(g(i, j - i), g(j, n - (j - i)));
200 }
201 // 可以在有 n 個點的凸包內，用 O(Log n) 判斷一個線段：
202 // {1 : 線段上存在某一點位於凸包內部（邊上不算），
203 // 0 : 線段上存在某一點碰到凸包的邊但線段上任一點均不在凸包
204 // 內部，
205 // -1 : 線段完全在凸包外面}
206 int segment_pass_convex_interior(line<T> L) {
207     if (in_convex(L.p1) == 1 || in_convex(L.p2) == 1)
208         return 1;
209     L.build();
210     point<T> p(L.a, L.b);
211     auto gt = [&](int neg) {
212         auto f = [&](int x, int y) {
213             return sign((v[x] - v[y]) * p) == neg;
214         };
215         return cycle_search(f);
216     };
217     int i = gt(1), j = gt(-1), n = v.size();
218     T x = -(v[i] * p), y = -(v[j] * p);
219     if (L.c < x || y < L.c) return -1;
220     if (L.c == x || L.c == y) return 0;
221
222     if (i > j) swap(i, j);
223     auto g = [&](int x, int lim) {
224         int now = 0, nxt;
225         for (int i = 1 << __lg(lim); i > 0; i /= 2) {
226             if (now + i > lim) continue;
227             nxt = (x + i) % n;
228             if (L.ori(v[x]) * L.ori(v[nxt]) > 0) {
229                 x = nxt;
230                 now += i;
231             }
232         } // ↓ BE CAREFUL
233         return -(ori(v[x], v[(x + 1) % n], L.p1) * ori(v[
234             x], v[(x + 1) % n], L.p2));
235     };
236     int ret = max(g(i, j - i), g(j, n - (j - i)));
237     return (ret == 0) ? (in_convex(L.p1) == 0 &&
238         in_convex(L.p2) == 0) : ret;
239 }
240 // 回傳點過凸包的兩條切線的切點的 θ-based index (不保證兩條
241 // 切線的順逆時針關係)
242 pair<int, int> convex_tangent_point(point<T> p) {
243     int n = v.size(), z = -1, edg = -1;
244     auto gt = [&](int neg) {
245         auto check = [&](int x) {

```

```

240 if (v[x] == p) z = x;
241 if (btw(v[x], v[(x + 1) % n], p)) edg = x;
242 if (btw(v[(x + n - 1) % n], v[x], p)) edg = (
243     x + n - 1) % n;
244 };
245 auto f = [&](int x, int y) {
246     check(x); check(y);
247     return ori(p, v[x], v[y]) == neg;
248 };
249 return cycle_search(f);
250 };
251 int x = gt(1), y = gt(-1);
252 if (z != -1) {
253     return {(z + n - 1) % n, (z + 1) % n};
254 }
255 else if (edg != -1) {
256     return {edg, (edg + 1) % n};
257 }
258 else {
259     return {x, y};
260 }
261 }
262 friend int halfplane_intersection(vector<line<T>> &s,
263     polygon<T> &P) {
264     #define neg(p) ((p.y == 0 ? p.x : p.y) < 0)
265     auto angle_cmp = [&](line<T> &A, line<T> &B) {
266         point<T> a = A.p2 - A.p1, b = B.p2 - B.p1;
267         return neg(a) < neg(b) || (neg(a) == neg(b) && (a
268             ^ b) > 0);
269     };
270     #undef neg
271     sort(s.begin(), s.end(), angle_cmp); // 線段左側為該
272     // 線段半平面
273     int L, R, n = s.size();
274     vector<point<T>> px(n);
275     vector<line<T>> q(n);
276     q[L = R = 0] = s[0];
277     for (int i = 1; i < n; ++i) {
278         while (L < R && s[i].ori(px[R-1]) <= 0) --R;
279         while (L < R && s[i].ori(px[L]) <= 0) ++L;
280         q[++R] = s[i];
281         if (q[R].parallel(q[R-1])) {
282             --R;
283             if (q[R].ori(s[i].p1) > 0) q[R] = s[i];
284         }
285         if (L < R) px[R-1] = q[R-1].line_intersection(q[R
286             ]);
287     }
288     while (L < R && q[L].ori(px[R-1]) <= 0) --R;
289     P.v.clear();
290     if (R - L <= 1) return 0;
291     px[R] = q[R].line_intersection(q[L]);
292     for (int i = L; i <= R; ++i) P.v.push_back(px[i]);
293     return R - L + 1;
294 }
295 };

```

5.2 Pick's Theorem

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

6 Graph

6.1 2-SAT

```

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])
17                dfs1(u);
18        }
19        order.push_back(v);
20    }
21    void dfs2(int v, int c1) {
22        comp[v] = c1;
23        for (int u : rev_G[v]) {
24            if (comp[u] == -1)
25                dfs2(u, c1);
26        }
27    }
28    bool solve() {
29        order.clear();
30        used.assign(N, false);
31        for (int i = 0; i < N; ++i) {
32            if (!used[i])
33                dfs1(i);
34        }
35        comp.assign(N, -1);
36        for (int i = 0, j = 0; i < N; ++i) {
37            int v = order[N - i - 1];
38            if (comp[v] == -1)
39                dfs2(v, j++);
40        }
41        assignment.assign(n, false);
42        for (int i = 0; i < N; i += 2) {
43            if (comp[i] == comp[i + 1])
44                return false;
45            assignment[i / 2] = (comp[i] > comp[i + 1]);
46        }
47        return true;
48    }
49    // A or B 都是 θ-based
50    void add_disjunction(int a, bool na, int b, bool nb) {
51        // na is true => ~a, na is false => a
52        // nb is true => ~b, nb is false => b
53        a = 2 * a ^ na;
54        b = 2 * b ^ nb;
55        int neg_a = a ^ 1;
56        int neg_b = b ^ 1;
57        G[neg_a].push_back(b);
58        G[neg_b].push_back(a);
59        rev_G[b].push_back(neg_a);

```

```

60     rev_G[a].push_back(neg_b);
61     return;
62 }
63 void get_result(vector<int>& res) {
64     res.clear();
65     for (int i = 0; i < n; i++)
66         res.push_back(assignment[i]);
67 }
68 };

```

6.2 Augment Path

```

1 struct AugmentPath{
2     int n, m;
3     vector<vector<int>> G;
4     vector<int> mx, my;
5     vector<int> visx, visy;
6     int stamp;
7
8     AugmentPath(int _n, int _m) : n(_n), m(_m), G(n), mx(n,
9         -1), my(m, -1), visx(n), visy(n){
10         stamp = 0;
11     }
12
13     void add(int x, int y){
14         G[x].push_back(y);
15     }
16
17     // bb03e2
18     bool dfs1(int now){
19         visx[now] = stamp;
20
21         for (auto x : G[now]){
22             if (my[x]==-1){
23                 mx[now] = x;
24                 my[x] = now;
25                 return true;
26             }
27         }
28         for (auto x : G[now]){
29             if (visx[my[x]]!=stamp && dfs1(my[x])){
30                 mx[now] = x;
31                 my[x] = now;
32                 return true;
33             }
34         }
35         return false;
36     }
37
38     vector<pair<int, int>> find_max_matching(){
39         vector<pair<int, int>> ret;
40
41         while (true){
42             stamp++;
43             int tmp = 0;
44             for (int i=0 ; i<n ; i++){
45                 if (mx[i]==-1 && dfs1(i)) tmp++;
46             }
47             if (tmp==0) break;
48         }
49         for (int i=0 ; i<n ; i++){
50             if (mx[i]!=-1){

```

```

51         ret.push_back({i, mx[i]});
52     }
53 }
54     return ret;
55 }
56
57 // 645577
58 void dfs2(int now){
59     visx[now] = true;
60
61     for (auto x : G[now]){
62         if (my[x]!=-1 && visy[x]==false){
63             visy[x] = true;
64             dfs2(my[x]);
65         }
66     }
67 }
68
69 // 要先執行 find_max_matching 一次
70 vector<pair<int, int>> find_min_vertex_cover(){
71     fill(visx.begin(), visx.end(), false);
72     fill(visy.begin(), visy.end(), false);
73
74     vector<pair<int, int>> ret;
75     for (int i=0 ; i<n ; i++){
76         if (mx[i]==-1) dfs2(i);
77     }
78
79     for (int i=0 ; i<n ; i++){
80         if (visx[i]==false) ret.push_back({1, i});
81     }
82     for (int i=0 ; i<m ; i++){
83         if (visy[i]==true) ret.push_back({2, i});
84     }
85     return ret;
86 }
87 }
88 };

```

6.3 Cut BCC

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 const int N = 200005;
5 vector<int> G[N];
6 int low[N], depth[N];
7 bool vis[N];
8 vector<vector<int>> bcc;
9 stack<int> stk;
10
11 void dfs(int v, int p) {
12     stk.push(v);
13     vis[v] = true;
14     low[v] = depth[v] = (p == -1 ? 1 : depth[p] + 1);
15     for (int u : G[v]) {
16         if (u == p) continue;
17         if (!vis[u]) {
18             /// (v, u) 是樹邊
19             dfs(u, v);
20             low[v] = min(low[v], low[u]);
21             /// u 無法在不經過父邊的情況走到 v 的祖先

```

```

22         if (low[u] >= depth[v]) {
23             bcc.emplace_back();
24             while (stk.top() != u) {
25                 bcc.back().push_back(stk.top());
26                 stk.pop();
27             }
28             bcc.back().push_back(stk.top());
29             stk.pop();
30             bcc.back().push_back(v);
31         }
32     } else {
33         /// (v, u) 是回邊
34         low[v] = min(low[v], depth[u]);
35     }
36 }
37 }

```

6.4 Dinic

```

1 // 一般圖：O(EV^2)
2 // 二分圖：O(EV)
3 struct Flow{
4     using T = int; // 可以換成別的类型
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;
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     while (q.size()){
45         int u = q.front(); q.pop();
46         for (auto [v, rc, rid] : G[u]){
47             if (rc <= 0 || dis[v] < INF) continue;
48             dis[v] = dis[u] + 1;
49             q.push(v);
50         }
51     }
52     if (dis[t]==INF) break;
53
54     fill(it.begin(), it.end(), 0);
55     while (true){
56         T df = dfs(s, t, INF);
57         if (df <= 0) break;
58         ans += df;
59     }
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<
76     pair<int, int>> &E){
77     // E is G without capacity
78     vector<bool> vis(n);
79     dfs_mincut(s, vis);
80     vector<pair<int, int>> ret;
81     for (auto &[u, v] : E){
82         if (vis[u] == true && vis[v] == false){
83             ret.emplace_back(u, v);
84         }
85     }
86     return ret;
87 }

```

6.5 Dominator Tree

```

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
7  */
8  struct DominatorTree{
9      int N;
10     vector<vector<int>> G;
11     vector<vector<int>> buckets, rg;
12     // dfn[x] = the DFS order 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> sdome, 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         sdome(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] = sdome[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 (sdome[val[x]]>sdome[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);
71
72         for (int x=stamp-1 ; x>=0 ; x--){
73             for (int y : rg[x]){
74                 sdome[x] = min(sdome[x], sdome[eval(y, true)]);
75             }
76             if (x>0) buckets[sdome[x]].push_back(x);
77             for (int u : buckets[x]){
78                 int p = eval(u, true);

```

```

79                 if (sdome[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 (sdome[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 };

```

6.6 EdgeBCC

```

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),
11         bcc(n), bccId(n) {}
12
13     void add_edge(int u, int v) {
14         edge.push_back({u, v});
15         G[u].push_back({v, m});
16         G[v].push_back({u, m++});
17     }
18
19     void dfs(int now, int pre) {
20         dfn[now] = low[now] = ++dep;
21         stk.push_back(now);
22
23         for (auto [x, id] : G[now]){
24             if (!dfn[x]){
25                 dfs(x, id);
26                 low[now] = min(low[now], low[x]);
27             } else if (id!=pre){
28                 low[now] = min(low[now], dfn[x]);
29             }
30         }
31
32         if (low[now]==dfn[now]){
33             if (pre!=-1) isBridge[pre] = true;
34             int u;
35             do{
36                 u = stk.back();
37                 stk.pop_back();
38                 bcc[sz].push_back(u);
39                 bccId[u] = sz;
40             } while (u!=now);
41             sz++;
42         }
43     }
44
45     void get_bcc() {
46         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].
54                             second});
55         }
56     }
57 };

```

6.7 Enumerate Triangle

```

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

```

6.8 EnumeratePlanarFace

```

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>
24                a, pair<int, int> b){
25                return (v[a.first]-v[i])<(v[b.first]-v[i]);
26            });
27
28            int sz = G[i].size(), pre = sz-1;
29            for (int j=0 ; j<sz ; j++){
30                nxt[G[i][pre].second] = G[i][j].second^1;
31                pre = j;
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

```

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 +
11        1), mxson(_n + 1), topf(_n + 1), id(_n + 1) {}
12    void dfs1(int v = 1, int p = -1) {
13        pa[v] = p; sz[v] = 1; mxson[v] = 0;
14        depth[v] = (p == -1 ? 0 : depth[p] + 1);
15        for (int u : G[v]) {
16            if (u == p) continue;
17            dfs1(u, v);
18            sz[v] += sz[u];
19            if (sz[u] > sz[mxson[v]]) mxson[v] = u;
20        }
21    }
22    void dfs2(int v = 1, int top = 1) {
23        id[v] = ++idcnt;
24        topf[v] = top;
25        if (mxson[v]) dfs2(mxson[v], top);
26        for (int u : G[v]) {
27            if (u == mxson[v] || u == pa[v]) continue;
28            dfs2(u, u);
29        }
30    }
31    // query 為區間資料結構
32    int path_query(int a, int b) {
33        int res = 0;
34        while (topf[a] != topf[b]) { /// 若不在同一條鍊上
35            if (depth[topf[a]] < depth[topf[b]]) swap(a, b);
36            res = max(res, 0ll); // query : l = id[topf[a]],
37            r = id[a];
38            a = pa[topf[a]];
39        }
40        /// 此時已在同一條鍊上
41        if (depth[a] < depth[b]) swap(a, b);
42        res = max(res, 0ll); // query : l = id[b], r = id[a]
43        return res;
44    }
45 };

```

6.10 Kosaraju

```

1 /* c7d5aa
2 給定一個有向圖，迴傳傳縮點後的圖、SCC 的資訊
3 所有點都以 based-0 編號
4
5 函式：
6 SCC_compress G(n): 宣告一個有 n 個點的圖
7 .add_edge(u, v): 加上一條邊 u -> v
8 .compress: O(n log n) 計算 G3、SCC、SCC_id 的資訊，並把縮點後
9 的結果存在 result 裡
10
11 SCC[i] = 某個 SCC 中的所有點
12 SCC_id[i] = 第 i 個點在第幾個 SCC
13 */
14 struct SCC_compress{
15     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]].
70             push_back(SCC_id[v]);
71     }
72     for (int i=0; i<SCC.size(); i++){
73         sort(result[i].begin(), result[i].end());
74         result[i].resize(unique(result[i].begin(), result
75             [i].end())-result[i].begin());
76     }
77 }

```

6.11 Kuhn Munkres

```

1 // O(n^3) 找到最大權匹配
2 struct KuhnMunkres{
3     int n; // max(n, m)
4     vector<vector<int>> G;
5     vector<int> match, lx, ly, visx, visy;
6     vector<int> slack;
7     int stamp = 0;
8
9     KuhnMunkres(int n) : n(n), G(n, vector<int>(n)), lx(n),
10        ly(n), slack(n), match(n), visx(n), visy(n) {}
11
12 void add(int x, int y, int w){
13     G[x][y] = max(G[x][y], w);
14 }
15
16 bool dfs(int i, bool aug){ // aug = true 表示要更新 match
17     if (visx[i]==stamp) return false;
18     visx[i] = stamp;
19
20     for (int j=0; j<n; j++){
21         if (visy[j]==stamp) continue;
22         int d = lx[i]+ly[j]-G[i][j];
23
24         if (d==0){
25             visy[j] = stamp;
26             if (match[j]==-1 || dfs(match[j], aug)){
27                 if (aug){
28                     match[j] = i;
29                 }
30                 return true;
31             }
32         }else{
33             slack[j] = min(slack[j], d);
34         }
35     }
36     return false;
37 }
38
39 bool augment(){
40     for (int j=0; j<n; j++){
41         if (visy[j]!=stamp && slack[j]==0){
42             visy[j] = stamp;
43             if (match[j]==-1 || dfs(match[j], false)){
44                 return true;
45             }
46         }
47     }
48     return false;
49 }
50
51 void relabel(){
52     int delta = INF;
53     for (int j=0; j<n; j++){
54         if (visy[j]!=stamp) delta = min(delta, slack[j]);
55     }
56     for (int i=0; i<n; i++){
57         if (visx[i]==stamp) lx[i] -= delta;
58     }
59     for (int j=0; j<n; j++){
60         if (visy[j]==stamp) ly[j] += delta;
61         else slack[j] -= delta;
62     }
63 }

```

```

63 int solve(){
64
65     for (int i=0; i<n; i++){
66         lx[i] = 0;
67         for (int j=0; j<n; j++){
68             lx[i] = max(lx[i], G[i][j]);
69         }
70     }
71
72     fill(ly.begin(), ly.end(), 0);
73     fill(match.begin(), match.end(), -1);
74
75     for (int i = 0; i < n; i++) {
76         fill(slack.begin(), slack.end(), INF);
77         stamp++;
78         if (dfs(i, true)) continue;
79
80         while(augment()==false) relabel();
81         stamp++;
82         dfs(i, true);
83     }
84
85     int ans = 0;
86     for (int j=0; j<n; j++){
87         if (match[j]!=-1){
88             ans += G[match[j]][j];
89         }
90     }
91     return ans;
92 }
93
94 };

```

6.12 LCA

```

1 struct Tree{
2     int N, M = 0, H;
3     vector<vector<int>> G;
4     vector<vector<int>> LCA;
5     vector<int> parent;
6     vector<int> dep;
7
8     Tree(int _N) : N(_N), H(__lg(_N)+1){
9         G.resize(N);
10        parent.resize(N, -1);
11        dep.resize(N, 0);
12        LCA.resize(H, vector<int>(N, 0));
13    }
14
15 void add_edge(int u, int v){
16     M++;
17     G[u].push_back(v);
18     G[v].push_back(u);
19 }
20
21 void dfs(int now, int pre){ // root 的 pre 是自己
22     dep[now] = dep[pre]+1;
23     parent[now] = pre;
24     for (auto x : G[now]){
25         if (x==pre) continue;
26         dfs(x, now);
27     }
28 }

```



```

29
30 void build_LCA(int root = 0){
31     dfs(root, root);
32     for (int i=0; i<N; i++) LCA[0][i] = parent[i];
33     for (int i=1; i<H; i++){
34         for (int j=0; j<N; j++){
35             LCA[i][j] = LCA[i-1][LCA[i-1][j]];
36         }
37     }
38 }
39
40 int jump(int u, int step){
41     for (int i=0; i<H; i++){
42         if (step&(1<<i)) u = LCA[i][u];
43     }
44     return u;
45 }
46
47 int get_LCA(int u, int v){
48     if (dep[u]<dep[v]) swap(u, v);
49     u = jump(u, dep[u]-dep[v]);
50     if (u==v) return u;
51     for (int i=H-1; i>=0; i--){
52         if (LCA[i][u]!=LCA[i][v]){
53             u = LCA[i][u];
54             v = LCA[i][v];
55         }
56     }
57     return parent[u];
58 }
59 };

```

6.13 MCMF

```

1 struct Flow {
2     struct Edge {
3         int u, rc, k, rv;
4     };
5
6     vector<vector<Edge>> G;
7     vector<int> par, par_eid;
8     Flow(int n) : G(n+1), par(n+1), par_eid(n+1) {}
9
10    // v->u, capacity: c, cost: k
11    void add(int v, int u, int c, int k){
12        G[v].push_back({u, c, k, SZ(G[u])});
13        G[u].push_back({v, 0, -k, SZ(G[v])-1});
14    }
15
16    // 3701d6
17    int spfa(int s, int t){
18        fill(ALL(par), -1);
19        vector<int> dis(SZ(par), INF);
20        vector<bool> in_q(SZ(par), false);
21        queue<int> Q;
22        dis[s] = 0;
23        in_q[s] = true;
24        Q.push(s);
25
26        while (!Q.empty()){
27            int v = Q.front();
28            Q.pop();
29            in_q[v] = false;

```

```

30     for (int i=0; i<SZ(G[v]); i++){
31         auto [u, rc, k, rv] = G[v][i];
32         if (rc>0 && dis[v]+k<dis[u]){
33             dis[u] = dis[v]+k;
34             par[u] = v;
35             par_eid[u] = i;
36             if (!in_q[u]) Q.push(u);
37             in_q[u] = true;
38         }
39     }
40 }
41
42 return dis[t];
43 }
44
45 // return <max flow, min cost>, 150093
46 pair<int, int> flow(int s, int t){
47     int fl = 0, cost = 0, d;
48     while ((d = spfa(s, t))<INF){
49         int cur = INF;
50         for (int v=t; v!=s; v=par[v]){
51             cur = min(cur, G[par[v]][par_eid[v]].rc);
52             fl += cur;
53             cost += d*cur;
54             for (int v=t; v!=s; v=par[v]){
55                 G[par[v]][par_eid[v]].rc -= cur;
56                 G[v][G[par[v]][par_eid[v]].rv].rc += cur;
57             }
58         }
59         return {fl, cost};
60     }
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

```

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

```

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

```

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
4 (0)
5 #define dbg(x) cerr << #x << " = " << x << endl
6 #define int long long
7 using namespace std;
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<
26     int, int> &rec, int now, int pre){
27     s[now]=1;
28     w[now]=0;
29     for (auto x : g[now]){
30         if (x!=pre){
31             centroid(g, s, w, rec, x, now);
32             s[now]+=s[x];
33             w[now]=max(w[now], s[x]);
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 void solve1(){
62     // init
63     id1=0;
64     id2=0;
65     c1={0, 0};
66     c2={0, 0};
67     fill(sz1.begin(), sz1.begin()+n+1, 0);
68     fill(sz2.begin(), sz2.begin()+n+1, 0);
69     fill(we1.begin(), we1.begin()+n+1, 0);
70     fill(we2.begin(), we2.begin()+n+1, 0);
71     for (int i=1; i<=n; i++){
72         g1[i].clear();
73         g2[i].clear();
74     }
75     m1.clear();
76     m2.clear();
77
78     // input
79     cin >> n;
80     for (int i=0; i<n-1; i++){
81         cin >> a >> b;
82         g1[a].push_back(b);
83         g1[b].push_back(a);
84     }
85     for (int i=0; i<n-1; i++){
86         cin >> a >> b;
87         g2[a].push_back(b);
88         g2[b].push_back(a);
89     }
90
91     // get tree centroid
92     centroid(g1, sz1, we1, c1, 1, 0);
93     centroid(g2, sz2, we2, c2, 1, 0);
94
95     // process
96     int res1=0, res2=0, res3=0;
97     if (c2.second!=0){
98         res1=dfs(g1, m1, id1, c1.first, 0);
99         m2=m1;
100         id2=id1;
101         res2=dfs(g2, m1, id1, c2.first, 0);
102         res3=dfs(g2, m2, id2, c2.second, 0);
103     }else if (c1.second!=0){
104         res1=dfs(g2, m1, id1, c2.first, 0);
105         m2=m1;
106         id2=id1;
107         res2=dfs(g1, m1, id1, c1.first, 0);
108         res3=dfs(g1, m2, id2, c1.second, 0);
109     }else{
110         res1=dfs(g1, m1, id1, c1.first, 0);
111         res2=dfs(g2, m1, id1, c2.first, 0);
112     }
113
114     // output
115     cout << (res1==res2 || res1==res3 ? "YES" : "NO") << endl;
116
117     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

```

1 #include <bits/stdc++.h>
2 #define lp(i,a,b) for(int i=(a);i<(b);i++)
3 #define pii pair<int,int>
4 #define pb push_back
5 #define ins insert
6 #define ff first
7 #define ss second
8 #define opa(x) cerr << #x << " = " << x << ", ";
9 #define op(x) cerr << #x << " = " << x << endl;
10 #define ops(x) cerr << x;
11 #define etr cerr << endl;
12 #define spc cerr << ' ';
13 #define BAE(x) (x).begin(), (x).end()
14 #define STL(x) cerr << #x << " : "; for(auto &qwe:x) cerr <<
15     qwe << ' '; cerr << endl;
16 #define deb1 cerr << "deb1" << endl;
17 #define deb2 cerr << "deb2" << endl;
18 #define deb3 cerr << "deb3" << endl;
19 #define deb4 cerr << "deb4" << endl;
20 #define deb5 cerr << "deb5" << endl;
21 #define bye exit(0);
22 using namespace std;
23
24 const int mxn = (int)(2e5) + 10;
25 const int mxlg = 17;
26 int last_special_node = (int)(1e5) + 1;
27 vector<int> E[mxn], F[mxn];
28
29 struct edg{
30     int fr, to;
31     edg(int _fr, int _to){
32         fr = _fr;
33         to = _to;
34     }
35 };
36 ostream& operator<<(ostream& os, edg x){os << x.fr << "--" <<
37     x.to;}
38 vector<edg> EV;
39
40 void tarjan(int v, int par, stack<int>& S){
41     static vector<int> dfn(mxn), low(mxn);
42     static vector<bool> to_add(mxn);
43     static int nowT = 0;
44
45     int child = 0;
46     nowT += 1;
47     dfn[v] = low[v] = nowT;
48     for(auto &ne:E[v]){

```

```

47     int i = EV[ne].to;
48     if(i == par) continue;
49     if(!dfn[i]){
50         S.push(ne);
51         tarjan(i, v, S);
52         childs += 1;
53         low[v] = min(low[v], low[i]);
54
55         if(par >= 0 && low[i] >= dfn[v]){
56             vector<int> bcc;
57             int tmp;
58             do{
59                 tmp = S.top(); S.pop();
60                 if(!to_add[EV[tmp].fr]){
61                     to_add[EV[tmp].fr] = true;
62                     bcc.pb(EV[tmp].fr);
63                 }
64                 if(!to_add[EV[tmp].to]){
65                     to_add[EV[tmp].to] = true;
66                     bcc.pb(EV[tmp].to);
67                 }
68             }while(tmp != ne);
69             for(auto &j:bcc){
70                 to_add[j] = false;
71                 F[last_special_node].pb(j);
72                 F[j].pb(last_special_node);
73             }
74             last_special_node += 1;
75         }
76     }
77     else{
78         low[v] = min(low[v], dfn[i]);
79         if(dfn[i] < dfn[v]){ // edge i--v will be visited
80             // twice at here, but we only need one.
81             S.push(ne);
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(edge(u, v));
146         E[v].pb(EV.size());
147         EV.pb(edge(v, u));
148     }
149     E[0].pb(EV.size());
150     EV.pb(edge(0, 1));
151     stack<int> S;
152     tarjan(0, -1, S);
153     build_lca();
154
155     lp(queries,0,q){
156         int fr, to, relay; cin >> fr >> to >> relay;
157         if(fr == relay || to == relay){
158             cout << "NO\n";
159             continue;
160         }
161         if((can_reach(fr, relay) || can_reach(to, relay)) &&
162            dep[relay] >= dep[lca(fr, to)]){
163             cout << "NO\n";
164             continue;
165         }
166         cout << "YES\n";
167     }

```

6.18 最大權閉合圖

```

1 /*
2 Problem:
3 Given  $w = [w_0, w_1, \dots, w_{n-1}]$  (which can be

```

```

4 either positive or negative or 0), you can choose
5 to take  $w_i$  ( $0 < i < n$ ) or not, but if edge  $u \rightarrow v$ 
6 exists, you must take  $w_v$  if you want to take  $w_u$ 
7 (in other words, you can't take  $w_u$  without taking
8  $w_v$ ), this function returns the maximum value( $> 0$ )
9 you can get. If you need a construction, you can
10 output the minimum cut of the  $S$ (source) side.
11 Complexity:
12     MaxFlow( $n, m$ ) (Non-Biparte: $O(n^2m)$  / Bipartite: $O(m\sqrt{n})$ )
13 */
14 int maximum_closure(vector<int> w, vector<pair<int,int>> EV)
15 {
16     int n = w.size(), S = n + 1, T = n + 2;
17     Flow G(T + 5); // Graph/Dinic.cpp
18     int sum = 0;
19     for (int i = 0; i < n; ++i) {
20         if (w[i] > 0) {
21             G.add(S, i, w[i]);
22             sum += w[i];
23         }
24         else if (w[i] < 0) {
25             G.add(i, T, abs(w[i]));
26         }
27     }
28     for (auto &[u, v] : EV) { // You should make sure that
29         INF >  $\sum w_i$ 
30         G.add(u, v, INF);
31     }
32     int cut = G.flow(S, T);
33     return sum - cut;

```

6.19 Theorem

• 任意圖

- 不能有孤點 · 最大匹配 + 最小邊覆蓋 = n - 點覆蓋的補集是獨立集 · 最小點覆蓋 + 最大獨立集 = n

• 二分圖

- 最小點覆蓋 = 最大匹配 = n - 最大獨立集

• 只有邊帶權的二分圖

- w-vertex-cover (帶權點覆蓋): 每條邊的兩個連接點被選中的次數總和至少要是 w_e 。
- w-weight matching (帶權匹配)
- minimum vertex count of w-vertex-cover = maximum weight count of w-weight matching (一個點可以被選很多次 · 但邊不行)

• 點、邊都帶權的二分圖的定理

- b-matching: 假設 v 的點權是 b_v · 那所有 v 的匹配邊 e 的權重都要滿足 $\sum w_e \leq b_v$ 。
- The maximum w-weight of a b-matching equals the minimum b-weight of vertices in a w-vertex-cover.

7 Math

7.1 CRT

```

1 // ax + by = c
2 int extgcd(int a, int b, int c, int &x, int &y) {
3     if (b == 0) {
4         if (c % a) return INF;
5         x = c / a, y = 0;
6         return abs(a);
7     }
8     int x1, y1;
9     int g = extgcd(b, a % b, c, x1, y1);
10    x = y1;
11    y = x1 - a / b * y1;
12    return g;
13 }
14
15 // 有 n 個式子 · 求解  $x \equiv a_i \pmod{m_i}$ 
16 int CRT_m_coprime(int n, vector<int> &a, vector<int> &m) {
17     int p = 1, ans = 0;
18     vector<int> M(n), inv_M(n);
19
20     for (int i = 0; i < n; i++) p *= m[i];
21     for (int i = 0; i < n; i++) {
22         M[i] = p / m[i];
23         int tmp;
24         extgcd(M[i], m[i], inv_M[i], tmp);
25         ans += a[i] * inv_M[i] * M[i];
26         ans %= p;
27     }
28     return (ans % p + p) % p;
29 }
30
31 // 對於方程組的式子兩兩求解
32 // 回傳: {是否有解, {a, m}}
33 pair<bool, pair<int, int>> CRT_m_NOT_coprime(int a1, int m1,
34     int a2, int m2) {
35     int g = __gcd(m1, m2);
36     if ((a2 - a1) % g != 0) return {0, {-1, -1}};
37
38     int x, y; extgcd(m1, m2, x, y);
39
40     x = (a2 - a1) * x / g; // 兩者不能相反
41     a1 = x * m1 + a1;
42     m1 = m1 * m2 / g;
43     a1 = (a1 % m1 + m1) % m1;
44     return {1, {a1, m1}};
45 }
46
47 // ans = a / b (mod m)
48 // ans = ret.F + k * ret.S, k is integer
49 pair<int, int> div(int a, int b, int m) {
50     int flag = 1;
51     if (a < 0) { a = -a; flag *= -1; }
52     if (b < 0) { b = -b; flag *= -1; }
53     int t = -1, k = -1;
54     int res = extgcd(b, m, a, t, k);
55     if (res == INF) return {INF, INF};
56     m = abs(m / res);
57     t = t * flag;
58     return {t % m + m} % m;
59 }

```

59 | }

7.2 Josephus Problem

```

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

```

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,
21                     MOD-2)+MOD)%MOD;
22                 now %= MOD;
23             }
24             ret = (ret+now)%MOD;
25         }
26         return ret;
27     }
28 };

```

7.4 Lagrange continuous x

```

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 >= 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
19 // NUMBER IS A PRIME.
20 struct Lagrange {
21     /*
22      * Initialize a polynomial with f(x_0), f(x_0 + 1), ..., f(
23      * x_0 + n).
24      * This determines a polynomial f(x) whose degree is at most
25      * n.
26      * Then you can call sample(x) and you get the value of f(x)
27      * .
28      * Complexity of init() and sample() are both O(n).
29      */
30     int m, shift; // m = n + 1
31     vector<int> v, mul;
32     // You can use this function if you don't have inv_fac array
33     // already.
34     void construct_inv_fac() {
35         long long fac = 1;
36         for (int i = 2; i < MAX_N; ++i) {
37             fac = fac * i % mod;
38         }
39         inv_fac[MAX_N - 1] = fp(fac, mod - 2);
40         for (int i = MAX_N - 1; i >= 1; --i) {
41             inv_fac[i - 1] = inv_fac[i] * i % mod;
42         }
43     }
44     // You call init() many times without having a second
45     // instance of this struct.
46     void init(int X_0, vector<int> &u) {
47         v = u;
48         shift = ((1 - X_0) % mod + mod) % mod;
49         if (v.size() == 1) v.push_back(v[0]);
50         m = v.size();
51         mul.resize(m);
52     }
53     // You can use sample(x) instead of sample(x % mod).
54     int sample(int x) {
55         x = ((long long)x + shift) % mod;
56         x = (x < 0) ? (x + mod) : x;
57         long long now = 1;
58         for (int i = m; i >= 1; --i) {
59             mul[i - 1] = now;
60             now = now * (x - i) % mod;
61         }
62         int ret = 0;
63         bool neg = (m - 1) & 1;
64         now = 1;
65         for (int i = 1; i <= m; ++i) {
66             int up = now * mul[i - 1] % mod;
67             int down = inv_fac[m - i] * inv_fac[i - 1] % mod;
68             int tmp = ((long long)v[i - 1] * up % mod) * down
69                 % mod;
70             ret += (neg && tmp) ? (mod - tmp) : (tmp);
71             ret = (ret >= mod) ? (ret - mod) : ret;
72         }
73     }
74 };

```

```

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 Lucas's Theorem

```

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

```

7.6 Matrix

```

1 struct Matrix{
2     int n, m;
3     vector<vector<int>> arr;
4
5     Matrix(int _n, int _m){
6         n = _n;
7         m = _m;
8         arr.assign(n, vector<int>(m));
9     }
10
11     vector<int> & operator [] (int i){
12         return arr[i];
13     }
14
15     Matrix operator * (Matrix b){
16         Matrix ret(n, b.m);
17         for (int i=0 ; i<n ; i++){
18             for (int j=0 ; j<b.m ; j++){
19                 for (int k=0 ; k<m ; k++){
20                     ret.arr[i][j] += arr[i][k]*b.arr[k][j]%
21                         MOD;
22                     ret.arr[i][j] %= MOD;
23                 }
24             }
25         }
26         return ret;
27     }
28
29     Matrix pow(int p){
30         Matrix ret(n, n), mul = *this;

```

```

30     for (int i=0 ; i<n ; i++){
31         ret.arr[i][i] = 1;
32     }
33
34     for ( ; p ; p>>=1){
35         if (p&1) ret = ret*mul;
36         mul = mul*mul;
37     }
38
39     return ret;
40 }
41
42 int det(){
43     vector<vector<int>> arr = this->arr;
44     bool flag = false;
45     for (int i=0 ; i<n ; i++){
46         int target = -1;
47         for (int j=i ; j<n ; j++){
48             if (arr[j][i]){
49                 target = j;
50                 break;
51             }
52         }
53         if (target==-1) return 0;
54         if (i!=target){
55             swap(arr[i], arr[target]);
56             flag = !flag;
57         }
58     }
59
60     for (int j=i+1 ; j<n ; j++){
61         if (!arr[j][i]) continue;
62         int freq = arr[j][i]*qp(arr[i][i], MOD-2)%MOD
63             ;
64         for (int k=i ; k<n ; k++){
65             arr[j][k] -= freq*arr[i][k];
66             arr[j][k] = (arr[j][k]%MOD+MOD)%MOD;
67         }
68     }
69
70     int ret = !flag ? 1 : MOD-1;
71     for (int i=0 ; i<n ; i++){
72         ret *= arr[i][i];
73         ret %= MOD;
74     }
75     return ret;
76 }
77 };

```

7.7 Matrix 01

```

1 const int MAX_N = (1LL<<12);
2 struct Matrix{
3     int n, m;
4     vector<bitset<MAX_N>> arr;
5
6     Matrix(int _n, int _m){
7         n = _n;
8         m = _m;
9         arr.resize(n);
10    }
11

```

```

12    Matrix operator * (Matrix b){
13        Matrix b_t(b.m, b.n);
14        for (int i=0 ; i<b.n ; i++){
15            for (int j=0 ; j<b.m ; j++){
16                b_t.arr[j][i] = b.arr[i][j];
17            }
18        }
19
20        Matrix ret(n, b.m);
21        for (int i=0 ; i<n ; i++){
22            for (int j=0 ; j<b.m ; j++){
23                ret.arr[i][j] = (arr[i]&b_t.arr[j]).count()
24                    &1;
25            }
26        }
27        return ret;
28    }
};

```

7.8 Miller Rabin

```

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

```

```

42     }
43
44     return 1;
45 }

```

7.9 Pollard Rho

```

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

```

7.10 Polynomial

```

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 = 1;
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(1 << 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
160         .deg + 1);
161     if (i == a.deg + 1) {
162         Poly res(a.deg + 1);
163         res.a[0] = 1;
164         return res;
165     }
166     Poly b(a.deg - i + 1);
167     int inv1 = ::inv(a.a[i]);
168     FOR (j, 0, b.deg)
169         b.a[j] = a.a[j + i] * inv1 % mod;
170     Poly res1 = (b.ln() * (n % mod)).exp() * (::power(a.a
171         [i], n));
172     Poly res2(a.deg + 1);
173     FOR (j, 0, min((ll)(res1.deg), (ll)(a.deg - n * i)))
174         res2.a[j + n * i] = res1.a[j];
175     return res2;
176 }
177 };

```

7.11 josephus

```

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

```

7.12 數論分塊

```

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

```

7.13 最大質因數

```

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

```

7.14 歐拉公式

```

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

```

7.15 Burnside's Lemma

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

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

7.16 Catalan Number

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

7.17 Matrix Tree Theorem

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

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

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

方法: 先把所有自環刪掉, 定義 Q 為以下矩陣

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

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

7.18 Stirling's formula

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

7.19 Theorem

- $1 \sim x$ 質數的數量 $\approx \frac{x}{\ln x}$
- x 的因數的數量 $\approx x^{\frac{1}{3}}$
- x 的質因數的數量 $\approx \log \log x$
- p is a prime number $\Leftrightarrow (p-1)! \equiv -1 \pmod{p}$
- 每個正整數都可以表示成四個整數的平方和
- 任何大於 2 的整數都可以表示成兩個質數的和
- $n^{k-2} \cdot \prod_{i=1}^k s_i$ 個點、 k 的連通塊, 加上 $k-1$ 條邊使得變成一個連通圖的方法數, 其中每個連通塊有 s_i 個點

7.20 二元一次方程式

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

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

7.21 歐拉定理

若 a, m 互質，則：

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

若 a, m 不互質，則：

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

7.22 錯排公式

錯排公式：(n 個人中，每個人皆不再原來位置的組合數)

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

8 String

8.1 Hash

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

8.2 KMP

```
1 // 給一個字串 S，定義函數 pi(i) = k 代表 S[1 ... k] = S[i-k
  +1 ... i] (最長真前後綴)
2 // e5b7ce
3 vector<int> KMP(string &s){
4   int n = s.size();
5   vector<int> ret(n);
6   for (int i=1 ; i<n ; i++){
7     int j = ret[i-1];
8     while (j>0 && s[i]!=s[j]) j = ret[j-1];
9     j += (s[i]==s[j]);
10    ret[i] = j;
11  }
12  return ret;
13 }
```

8.3 Manacher

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

8.4 Min Rotation

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

8.5 Suffix Array

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

8.6 Z Algorithm

```

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

```

8.7 k-th Substring1

```

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 }

```

```

62     }else{
63         return min({st.query(pos_1, pos_2), len_1, len_2});
64     }
65 }
66
67 // 檢查  $[l1, r1]$  跟  $[l2, r2]$  的大小關係， $\theta$ -based
68 // 如果前者小於後者，就回傳  $<0$ ，相等就回傳  $=0$ ，否則回傳  $>0$ 
69 // 5b8db0
70 int substring_cmp(int l1, int r1, int l2, int r2){
71     int len_1 = r1-l1+1;
72     int len_2 = r2-l2+1;
73     int res = get_lcp(l1, r1, l2, r2);
74
75     if (res<len_1 && res<len_2){
76         return s[l1+res]-s[l2+res];
77     }else if (len_1==res && len_2==res){
78         // 如果不需要以 index 作為次要排序參數，這裡要回傳 0
79         return l1-l2;
80     }else{
81         return len_1==res ? -1 : 1;
82     }
83 }
84
85 // 對於位置在  $\leq p$  的後綴，找離他左邊/右邊最接近位置  $> p$  的後綴的  $Lcp$ ， $\theta$ -based
86 //  $pre[i] = s[i]$  離他左邊最接近位置  $> p$  的後綴的  $Lcp$ ， $\theta$ -based
87 //  $suf[i] = s[i]$  離他右邊最接近位置  $> p$  的後綴的  $Lcp$ ， $\theta$ -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 };

```