

# Contents

<b>1 Misc</b>	<b>2</b>	<b>4 Dynamic-Programming</b>	<b>8</b>	<b>7 Math</b>	<b>18</b>
1.1 Xor-Basis	2	4.1 SOS-DP	8	7.1 Burnside's-Lemma	18
1.2 Default-Code	2	4.2 Digit-DP	8	7.2 線性篩	18
1.3 Radix-Sort	2	4.3 整數拆分	8	7.3 Lucas's-Theorem	18
1.4 Set-Pq-Sort	2	<b>5 Geometry</b>	<b>8</b>	7.4 Matrix	18
1.5 2-SAT	2	5.1 Pick's-Theorem	8	7.5 最大質因數	18
1.6 Enumerate-Subset	3	5.2 Point-In-Polygon	8	7.6 中國剩餘定理 ( m 不互質 )	18
1.7 Fast-Input	3	5.3 Convex-Hull	8	7.7 歐拉公式	19
1.8 setup	3	5.4 Point-Struct	9	7.8 歐拉定理	19
1.9 run	3	5.5 Geometry-Struct	9	7.9 Fraction	19
1.10 default2	3	5.6 Segment-Intersection	10	7.10 錯排公式	19
1.11 random int	3	5.7 Geometry	10	7.11 Quick-Pow	19
<b>2 Convolution</b>	<b>3</b>	<b>6 Graph</b>	<b>12</b>	7.12 二元一次方程式	20
2.1 FFT	3	6.1 Find-Bridge	12	7.13 Josephus	20
2.2 FFT-2	4	6.2 Find-AP	13	7.14 數論分塊	20
2.3 NTT-998244353	4	6.3 HLD	13	7.15 Pollard-Rho	20
2.4 FFT-mod	4	6.4 Tree-Isomorphism	13	7.16 中國剩餘定理 ( m 互質 )	20
<b>3 Data-Structure</b>	<b>5</b>	6.5 Bridge BCC	14	7.17 Catalan	20
3.1 GP-Hash-Table	5	6.6 Cut BCC	14	7.18 數論定理	20
3.2 Sparse-Table	5	6.7 圖方樹	14	7.19 Miller-Rabin	20
3.3 Order-Set	5	6.8 SCC 與縮點	15	7.20 Stirling's formula	20
3.4 BIT	5	6.9 Dinic	15	7.21 Lagrange any x	21
3.5 Persistent-Segment-Tree	5	6.10 Dijkstra	16	7.22 Matrix-01	21
3.6 Trie	6	6.11 定理	16	7.23 Matrix-Tree-Theorem	21
3.7 LC-Segment-Tree	6	6.12 MCMF	16	7.24 Lagrange continuous x	21
3.8 Persistent-Disjoint-Set	7	6.13 Dinic with double	17	<b>8 String</b>	<b>21</b>
3.9 Add-Set-Segment-Tree	7	6.14 最大權閉合圖	17	8.1 Hash	21
3.10 Treap	7	6.15 tarjan	17	8.2 Manacher	22
				8.3 Z-Function	22
				8.4 KMP	22
				8.5 Suffix-Array	22
				8.6 Min-Rotation	23

# 1 Misc

## 1.1 Xor-Basis

```

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

```

## 1.2 Default-Code

```

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

```

## 1.3 Radix-Sort

```

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

```

## 1.4 Set-Pq-Sort

```

1 // priority_queue · 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 struct Data{
11     int x;
12
13     bool operator < (const Data &b) const {
14         return x<b.x;

```

```

15     }
16 };

```

## 1.5 2-SAT

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 struct TWO_SAT {
5     int n, N;
6     vector<vector<int>> G, rev_G;
7     deque<bool> used;
8     vector<int> order, comp;
9     deque<bool> assignment;
10    void init(int _n) {
11        n = _n;
12        N = _n * 2;
13        G.resize(N + 5);
14        rev_G.resize(N + 5);
15    }
16    void dfs1(int v) {
17        used[v] = true;
18        for (int u : G[v]) {
19            if (!used[u])
20                dfs1(u);
21        }
22        order.push_back(v);
23    }
24    void dfs2(int v, int c1) {
25        comp[v] = c1;
26        for (int u : rev_G[v]) {
27            if (comp[u] == -1)
28                dfs2(u, c1);
29        }
30    }
31    bool solve() {
32        order.clear();
33        used.assign(N, false);
34        for (int i = 0; i < N; ++i) {
35            if (!used[i])
36                dfs1(i);
37        }
38        comp.assign(N, -1);
39        for (int i = 0, j = 0; i < N; i += 2) {
40            int v = order[N - i - 1];
41            if (comp[v] == -1)
42                dfs2(v, j++);
43        }
44        assignment.assign(n, false);
45        for (int i = 0; i < N; i += 2) {
46            if (comp[i] == comp[i + 1])
47                return false;
48            assignment[i / 2] = (comp[i] > comp[i + 1]);
49        }
50        return true;
51    }
52    void add_disjunction(int a, bool na, int b, bool nb) { //
53        // A or B
54        // na means whether a is negative or not
55        // nb means whether b is negative or not
56        a = 2 * a ^ na;
57        b = 2 * b ^ nb;
58        int neg_a = a ^ 1;

```

```

58     int neg_b = b ^ 1;
59     G[neg_a].push_back(b);
60     G[neg_b].push_back(a);
61     rev_G[b].push_back(neg_a);
62     rev_G[a].push_back(neg_b);
63     return;
64 }
65 void get_result(vector<int>& res) {
66     res.clear();
67     for (int i = 0; i < n; i++)
68         res.push_back(assignment[i]);
69 }
70 };
71 /* CSES Giant Pizza
72 3 5
73 + 1 + 2
74 - 1 + 3
75
76 - + + + -
77 */
78 int main() {
79     int n, m;
80     cin >> n >> m;
81     TWO_SAT E;
82     E.init(m);
83
84     char c1, c2;
85     int inp1, inp2;
86     for (int i = 0; i < n; i++) {
87         cin >> c1 >> inp1;
88         cin >> c2 >> inp2;
89         E.add_disjunction(inp1 - 1, c1 == '-', inp2 - 1, c2
90             == '-');
91     }
92     bool able = E.solve();
93     if (able) {
94         vector<int> ans;
95         E.get_result(ans);
96         for (int i : ans)
97             cout << (i == true ? '+' : '-') << ' ';
98         cout << '\n';
99     } else {
100         cout << "IMPOSSIBLE\n";
101     }
102
103     return 0;
104 }

```

## 1.6 Enumerate-Subset

```

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

```

## 1.7 Fast-Input

```

1 // fast IO
2 // 6f8879
3 inline char readchar(){
4     static char buffer[BUFSIZ], * now = buffer + BUFSIZ, *
5         end = buffer + BUFSIZ;
6     if (now == end)
7     {
8         if (end < buffer + BUFSIZ)
9             return EOF;
10        end = (buffer + fread(buffer, 1, BUFSIZ, stdin));
11        now = buffer;
12    }
13    return *now++;
14 }
15 inline int nextint(){
16     int x = 0, c = readchar(), neg = false;
17     while(('0' > c || c > '9') && c!='-' && c!=EOF) c =
18         readchar();
19     if(c == '-') neg = true, c = readchar();
20     while('0' <= c && c <= '9') x = (x<<3) + (x<<1) + (c^'0')
21         , c = readchar();
22     if(neg) x = -x;
23     return x; // returns 0 if EOF
24 }

```

## 1.8 setup

```

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

```

## 1.9 run

```

1 import os
2 p = os.listdir(".")
3 f = input("input: ")
4

```

```

5 if os.system(f"g++ {f}.cpp -std=c++17 -Wall -Wextra -Wshadow
6     -O2 -DLOCAL -g -fsanitize=undefined,address -o {f}") !=
7     0:
8     print("CE")
9     exit(1)
10
11 for x in p:
12     if x[:len(f)]==f and x[-3:]!=".in":
13         print(x)
14         if os.system(f"./{f} < {x}")!=0:
15             print("RE")
16             exit(1)
17         print()

```

## 1.10 default2

```

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 }

```

## 1.11 random int

```

1 mt19937 seed(chrono::steady_clock::now().time_since_epoch().
2     count());
3 int rng(int l, int r){
4     return uniform_int_distribution<int>(l, r)(seed);
5 }

```

# 2 Convolution

## 2.1 FFT

```

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

```

```

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

```

## 2.2 FFT-2

```

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

```

```

7      for (int k=2 ; k<n ; k*=2){
8          R.resize(n);
9          rt.resize(n);
10         auto x = polar(1.0L, acos(-1.0L) / k);
11         for (int i=k ; i<2*k ; i++){
12             rt[i] = R[i] = (i&1 ? R[i/2]*x : R[i/2]);
13         }
14     }
15
16     vector<int> rev(n);
17     for (int i=0 ; i<n ; i++){
18         rev[i] = (rev[i/2] | (i&1)<<L)/2;
19     }
20
21     for (int i=0 ; i<n ; i++){
22         if (i<rev[i]) swap(a[i], a[rev[i]]);
23     }
24
25     for (int k=1 ; k<n ; k*=2){
26         for (int i=0 ; i<n ; i+=2*k){
27             for (int j=0 ; j<k ; j++){
28                 auto x = (double *)&rt[j+k];
29                 auto y = (double *)&a[i+j+k];
30                 cd z(x[0]*y[0] - x[1]*y[1], x[0]*y[1] + x[1]*y[0]);
31                 a[i+j+k] = a[i+j]-z;
32                 a[i+j] += z;
33             }
34         }
35     }
36
37     return;
38 }
39
40 vector<double> PolyMul(const vector<double> a, const vector<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;
62 }

```

## 2.3 NTT-998244353

```

1  const int MOD = (119 << 23) + 1, ROOT = 62; // = 998244353
2  // For p < 2^30 there is also e.g. 5 << 25, 7 << 26, 479 << 21
3  // 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
23     for (int i=0 ; i<n ; i++){
24         if (i<rev[i]){
25             swap(a[i], a[rev[i]]);
26         }
27     }
28
29     for (int k=1 ; k<n ; k*=2){
30         for (int i=0 ; i<n ; i+=2*k){
31             for (int j=0 ; j<k ; j++){
32                 int z = rt[j+k]*a[i+j+k]%MOD, &ai = a[i+j];
33                 ai+j+k = ai-z+(z>ai ? MOD : 0);
34                 ai += (ai+z>MOD ? z-MOD : z);
35             }
36         }
37     }
38
39     // 0b0e99
40     vector<int> polyMul(vector<int> &a, vector<int> &b){
41         if (a.empty() || b.empty()) return {};
42         int s = a.size()+b.size()-1, B = 32-__builtin_clz(s), n = 1<<B;
43         int inv = qp(n, MOD-2);
44
45         vector<int> L(a), R(b), out(n);
46         L.resize(n), R.resize(n);
47         NTT(L), NTT(R);
48         for (int i=0 ; i<n ; i++){
49             out[-i&(n-1)] = L[i]*R[i]%MOD*inv%MOD;
50         }
51         NTT(out);
52
53         out.resize(s);
54         return out;
55     }

```

## 2.4 FFT-mod

```

1  /*
2  修改 const int MOD = 998244353 更改要取餘的數字
3  PolyMul(a, b) 回傳多項式乘法的結果 (c_k = \sum_{i+j=k} a_i*b_j 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]*
37                     y[0]);
38                 a[i+j+k] = a[i+j]-z;
39                 a[i+j] += z;
40             }
41         }
42     }
43     return;
44 }
45
46 // d3c65e
47 vector<int> PolyMul(vector<int> a, vector<int> b){
48     if (a.empty() || b.empty()) return {};
49
50     vector<int> res(a.size()+b.size()-1);
51     int B = 32-__builtin_clz(res.size()), n = (1<<B), cut =
52         int(sqrt(MOD));
53     vector<cd> L(n), R(n), outs(n), outl(n);
54
55     for (int i=0 ; i<a.size() ; i++){
56         L[i] = cd((int) a[i]/cut, (int)a[i]%cut);
57     }
58     for (int i=0 ; i<b.size() ; i++){
59         R[i] = cd((int) b[i]/cut, (int)b[i]%cut);
60     }
61     FFT(L);
62     FFT(R);
63     for (int i=0 ; i<n ; i++){
64         int j = -i&(n-1);
65         outl[j] = (L[i]+conj(L[j])) * R[i]/(2.0*n);
66         outs[j] = (L[i]-conj(L[j])) * R[i]/(2.0*n)/1i;
67     }
68     FFT(outl);
69     FFT(outs);
70     for (int i=0 ; i<res.size() ; i++){
71         int av = (int)(real(outl[i])+0.5), cv = (int)(imag(
72             outs[i])+0.5);

```

```

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

```

## 3 Data-Structure

### 3.1 GP-Hash-Table

```

1 #include <ext/pb_ds/assoc_container.hpp>
2 using namespace __gnu_pbds;
3 typedef tree<int, null_type, less<int>, rb_tree_tag,
4     tree_order_statistics_node_update> order_set;
5 struct custom_hash {
6     static uint64_t splitmix64(uint64_t x) {
7         // http://xorshift.di.unimi.it/splitmix64.c
8         x += 0x9e3779b97f4a7c15;
9         x = (x ^ (x >> 30)) * 0xbf58476d1ce4e5b9;
10        x = (x ^ (x >> 27)) * 0x94d049bb133111eb;
11        return x ^ (x >> 31);
12    }
13
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;

```

### 3.2 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
17    // 回傳 [ll, rr) 的最小值
18    int query(int ll, int rr){
19        int h = __lg(rr-ll);
20        return min(st[h][ll], st[h][rr-(1<<h)]);
21    }
22 };

```

### 3.3 Order-Set

```

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

```

### 3.4 BIT

```

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

```

### 3.5 Persistent-Segment-Tree

```

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

```

```

16     int lc = -1;
17     int rc = -1;
18     int val = 0;
19 };
20 vector<Node> arr;
21 vector<int> version;
22
23 Persistent_Segment_Tree(int sz){
24     arr.resize(32*sz);
25     version.push_back(node_cnt++);
26     return;
27 }
28
29 void pull(Node &c, Node a, Node b){
30     c.val = a.val+b.val;
31     return;
32 }
33
34 void build(vector<int> &v, int idx, int ll = 0, int rr =
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
56     if (rr-ll==1){
57         now.val = val;
58         return;
59     }
60
61     int mid = (ll+rr)/2;
62     if (pos<mid){
63         arr[node_cnt] = arr[now.lc];
64         now.lc = node_cnt;
65         node_cnt++;
66         update(pos, val, now.lc, ll, mid);
67     }else{
68         arr[node_cnt] = arr[now.rc];
69         now.rc = node_cnt;
70         node_cnt++;
71         update(pos, val, now.rc, mid, rr);
72     }
73     pull(now, arr[now.lc], arr[now.rc]);
74     return;
75 }
76
77 void update_version(int pos, int val, int ver){
78     update(pos, val, version[ver]);
79 }

```

### 3.6 Trie

```

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

```

```

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 }

```

### 3.7 LC-Segment-Tree

```

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

```

```

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

```

### 3.8 Persistent-Disjoint-Set

```

1 struct Persistent_Disjoint_Set{
2     Persistent_Segment_Tree arr, sz;
3
4     void init(int n){
5         arr.init(n);
6         vector<int> v1;
7         for (int i=0; i<n; i++){
8             v1.push_back(i);
9         }
10        arr.build(v1, 0);
11
12        sz.init(n);
13        vector<int> v2;
14        for (int i=0; i<n; i++){
15            v2.push_back(1);
16        }
17        sz.build(v2, 0);
18    }
19
20    int find(int a){
21        int res = arr.query_version(a, a+1, arr.version.size()
22            (-1).val);
23        if (res==a) return a;
24        return find(res);
25    }
26
27    bool unite(int a, int b){
28        a = find(a);
29        b = find(b);
30
31        if (a!=b){
32
33            int sz1 = sz.query_version(a, a+1, arr.version.
34                size()-1).val;
35            int sz2 = sz.query_version(b, b+1, arr.version.
36                size()-1).val;
37
38            if (sz1<sz2){
39                arr.update_version(a, b, arr.version.size()
40                    (-1));
41                sz.update_version(b, sz1+sz2, arr.version.
42                    size()-1);
43            }else{
44                arr.update_version(b, a, arr.version.size()
45                    (-1));
46                sz.update_version(a, sz1+sz2, arr.version.
47                    size()-1);
48            }
49        }
50    }
51 };

```

```

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

```

### 3.9 Add-Set-Segment-Tree

```

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

```

```

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

```

### 3.10 Treap

```

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

```



```

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

```

## 4 Dynamic-Programming

### 4.1 SOS-DP

```

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

```

### 4.2 Digit-DP

```

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

```

```

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

```

### 4.3 整數拆分

$dp[i][x]$  = 要將整數  $x$  拆成  $i$  堆的「組合數」

$dp[i+1][x+1] += dp[i][x]$  (創造新的一堆)  
 $dp[i][x+i] += dp[i][x]$  (把每一堆都增加1)

## 5 Geometry

### 5.1 Pick's-Theorem

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

### 5.2 Point-In-Polygon

```

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

```

### 5.3 Convex-Hull



```

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

```

## 5.4 Point-Struct

```

1 const int EPS = 1e-6;
2
3 struct Point{
4     Point x, y;
5
6     Point(Point _x = 0, Point _y = 0){
7         x = _x;
8         y = _y;
9     }
10
11     // 純量乘、除法
12     Point operator * (Point a){return {a*x, a*y}};
13     Point operator / (Point a){return {a/x, a/y}};
14
15     // 向量加、減法
16     Point operator + (Point a){return {x+a.x, y+a.y}};
17     Point operator - (Point a){return {x-a.x, y-a.y}};
18
19     // 內積、外積
20     double operator * (Point a){return x*a.x+y*a.y};
21     double operator ^ (Point a){return x*a.y-y*a.x};
22
23     // bool operator < (const Point &a) const {return (x*a.y<
24         a.x*y);} // 極角排序 (順時鐘)
25     bool operator < (const Point &a) const {return x==a.x ? y
26         <a.y : x<a.x;}
27     bool operator == (const Point &a) const {return x==a.x &&
28         y==a.y;}
29
30     double dis(Point a){return sqrtl(abs(x-a.x)*abs(x-a.x)+
31         abs(y-a.y)*abs(y-a.y));}
32
33     // 判斷向量正負：1=正數, 0=0, -1=負數
34     int sign(double a){
35         if (abs(a)<EPS) return 0;
36         else return (a>0 ? 1 : -1);
37     }
38
39     // 判斷 ab 到 ac 的方向：1=逆時鐘, 0=重疊, -1=順時鐘

```

```

37 int ori(Point a, Point b, Point c){
38     return sign((b-a)^(c-a));
39 }

```

## 5.5 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
29     T operator*(point b) {return x * b.x + y * b.y; }
30     T operator^(point b) {return x * b.y - y * b.x; }
31     T abs2() {return (*this) * (*this); }
32
33     // 旋轉 Arg(b) 的角度 (小心溢位)
34     point rotate(point b) {return {x*b.x - y*b.y, x*b.y + y*b
35         .x}; }
36
37 template<typename T>
38 struct line {
39     point<T> p1, p2;
40     // ax + by + c = 0
41     T a, b, c; // |a|, |b| ≤ 2C, |c| ≤ 8C²
42     line() {}
43     line(const point<T> &x, const point<T> &y) : p1(x), p2(y)
44     {
45         build();
46     }
47     void build() {
48         a = p1.y - p2.y;
49         b = p2.x - p1.x;
50         c = (-a*p1.x)-b*p1.y;
51     }
52     // 判斷點和有向直線的關係：{1:左邊,0:在線上,-1:右邊}
53     int ori(point<T> &p) {
54         return sign((p2-p1) ^ (p-p1));
55     }
56 }

```

```

55 // 判斷直線斜率是否相同
56 bool parallel(line &l) {
57     return ((p1-p2) ^ (l.p1-l.p2)) == 0;
58 }
59 // 兩直線交點
60 point<long double> line_intersection(line &l) {
61     using P = point<long double>;
62     point<T> a = p2-p1, b = l.p2-l.p1, s = l.p1-p1;
63     return P(p1.x, p1.y) + P(a.x, a.y) * (((long double)(s^b)
64         ) / (a^b));
65 };
66
67 template<typename T>
68 struct polygon {
69     vector<point<T>> v;
70     polygon() {}
71     polygon(const vector<point<T>> &u) : v(u) {}
72     // simple 為 true 的時候會回傳任意三點不共線的凸包
73     void make_convex_hull(int simple) {
74         auto cmp = [&](point<T> &p, point<T> &q) {
75             return (p.x == q.x) ? (p.y < q.y) : (p.x < q.x);
76         };
77         simple = (bool)simple;
78         sort(v.begin(), v.end(), cmp);
79         v.resize(unique(v.begin(), v.end()) - v.begin());
80         vector<point<T>> hull;
81         for (int t = 0; t < 2; ++t){
82             int sz = hull.size();
83             for (auto &i:v) {
84                 while (hull.size() >= sz+2 && ori(hull[hull.
85                     size()-2], hull.back(), i) < simple) {
86                     hull.pop_back();
87                 }
88                 hull.push_back(i);
89             }
90             hull.pop_back();
91             reverse(v.begin(), v.end());
92             swap(hull, v);
93         }
94         // 可以在有 n 個點的簡單多邊形內，用 O(n) 的時間回傳：
95         // {1：在多邊形內, 0：在多邊形上, -1：在多邊形外}
96         int in_polygon(point<T> a){
97             #define in(a, b, c) ( ori(a, b, c) \
98                 ? 0 : (sign((a-c)*(b-c)) <= 0) )
99             const T MAX_POS = (1e9 + 5); // [記得修改] 座標的最大
100                 值
101             point<T> pre = v.back(), b(MAX_POS, a.y + 1);
102             int cnt = 0;
103
104             for (auto &i:v) {
105                 if (in(pre, i, a)) return 0;
106                 if (banana(a, b, pre, i)) cnt++;
107                 pre = i;
108             }
109
110             return cnt%2 ? 1 : -1;
111             #undef in
112         }
113         friend int halfplane_intersection(vector<line<T>> &s,
114             polygon<T> &p) {
115             #define neg(p) ((p.y == 0 ? p.x : p.y) < 0)
116             auto angle_cmp = [&](line<T> &A, line<T> &B) {
117                 point<T> a = A.p2-A.p1, b = B.p2-B.p1;

```

```

116     return neg(a) < neg(b) || (neg(a) == neg(b) && (a
117         ^b) > 0);
118 };
119 #undef neg
120 sort(s.begin(), s.end(), angle_cmp); // 線段左側為該
121     線段半平面
122 int L, R, n = s.size();
123 vector<point<T>> px(n);
124 vector<line<T>> q(n);
125 q[L = R = 0] = s[0];
126 for(int i = 1; i < n; ++i) {
127     while(L < R && s[i].ori(px[R-1]) <= 0) --R;
128     while(L < R && s[i].ori(px[L]) <= 0) ++L;
129     q[++R] = s[i];
130     if(q[R].parallel(q[R-1])) {
131         --R;
132         if(q[R].ori(s[i].p1) > 0) q[R] = s[i];
133     }
134     if(L < R) px[R-1] = q[R-1].line_intersection(q[R
135         ]);
136 }
137 while(L < R && q[L].ori(px[R-1]) <= 0) --R;
138 P.v.clear();
139 if(R - L <= 1) return 0;
140 px[R] = q[R].line_intersection(q[L]);
141 for(int i = L; i <= R; ++i) P.v.push_back(px[i]);
142 return R - L + 1;
143 }
144 };

```

## 5.6 Segment-Intersection

```

1 // 判斷線段 ab, cd 是否相交
2 bool banana(point<auto> a, point<auto> b, point<auto> c,
3     point<auto> d) {
4     #define in(a, b, c) ( ori(a, b, c) \
5         ? 0 : (sign((a-c)*(b-c)) <= 0) )
6     int s1 = ori(a, b, c);
7     int s2 = ori(a, b, d);
8     int s3 = ori(c, d, a);
9     int s4 = ori(c, d, b);
10    if (in(a, b, c) || in(a, b, d) || in(c, d, a) || in(c, d,
11        b)) return 1;
12    return (s1 * s2 < 0) && (s3 * s4 < 0);
13    #undef in
14 }

```

## 5.7 Geometry

```

1 const double PI=atan2(0.0,-1.0);
2 template<typename T>
3 struct point{
4     T x,y;
5     point(){}
6     point(const T&x,const T&y):x(x),y(y){}
7     point operator+(const point &b)const{
8         return point(x+b.x,y+b.y); }
9     point operator-(const point &b)const{
10        return point(x-b.x,y-b.y); }

```

```

11 point operator*(const T &b)const{
12     return point(x*b,y*b); }
13 point operator/(const T &b)const{
14     return point(x/b,y/b); }
15 bool operator==(const point &b)const{
16     return x==b.x&&y==b.y; }
17 T dot(const point &b)const{
18     return x*b.x+y*b.y; }
19 T cross(const point &b)const{
20     return x*b.y-y*b.x; }
21 point normal()const{//求法向量
22     return point(-y,x); }
23 T abs2()const{//向量长度的平方
24     return dot(*this); }
25 T rad(const point &b)const{//兩向量的弧度
26     return fabs(atan2(fabs(cross(b)),dot(b))); }
27 T getA()const{//對x軸的弧度
28     T A=atan2(y,x);{//超過180度會變負的
29         if(A<=-PI/2)A+=PI*2;
30         return A;
31     }
32 };
33 template<typename T>
34 struct line{
35     line(){}
36     point<T> p1,p2;
37     T a,b,c;//ax+by+c=0
38     line(const point<T>&x,const point<T>&y):p1(x),p2(y){}
39     void pton()const{//轉成一般式
40         a=p1.y-p2.y;
41         b=p2.x-p1.x;
42         c=-a*p1.x-b*p1.y;
43     }
44     T ori(const point<T> &p)const{//點和有向直線的關係 · >0左
45         邊、=0在線上<0右邊
46         return (p2-p1).cross(p-p1);
47     }
48     T btw(const point<T> &p)const{//點投影落在線段上<=0
49         return (p1-p).dot(p2-p);
50     }
51     bool point_on_segment(const point<T>&p)const{//點是否在線段
52         上
53         return ori(p)==0&&btw(p)<=0;
54     }
55     T dis2(const point<T> &p,bool is_segment=0)const{//點跟直線
56         /線段的距離平方
57         point<T> v=p2-p1,v1=p-p1;
58         if(is_segment){
59             point<T> v2=p-p2;
60             if(v.dot(v1)<=0)return v1.abs2();
61             if(v.dot(v2)>=0)return v2.abs2();
62         }
63         T tmp=v.cross(v1);
64         return tmp*tmp/v.abs2();
65     }
66     T seg_dis2(const line<T> &l)const{//兩線段距離平方
67         return min({dis2(l.p1,1),dis2(l.p2,1),l.dis2(p1,1),l.dis2
68             (p2,1)});
69     }
70     point<T> projection(const point<T> &p)const{//點對直線的投
71         影
72         point<T> n=(p2-p1).normal();
73         return p-n*(p-p1).dot(n)/n.abs2();

```

```

69 }
70 point<T> mirror(const point<T> &p)const{
71     //點對直線的鏡射 · 要先呼叫pton轉成一般式
72     point<T> R;
73     T d=a*b+b*b;
74     R.x=(b*b*p.x-a*a*p.x-2*a*b*p.y-2*a*c)/d;
75     R.y=(a*a*p.y-b*b*p.y-2*a*b*p.x-2*b*c)/d;
76     return R;
77 }
78 bool equal(const line &l)const{//直線相等
79     return ori(l.p1)==0&&ori(l.p2)==0;
80 }
81 bool parallel(const line &l)const{
82     return (p1-p2).cross(l.p1-l.p2)==0;
83 }
84 bool cross_seg(const line &l)const{
85     return (p2-p1).cross(l.p1-p1)*(p2-p1).cross(l.p2-p1)<=0;
86     //直線是否交線段
87 }
88 int line_intersect(const line &l)const{//直線相交情況 · -1無
89     限多點、1交於一點、0不相交
90     return parallel(l)?(ori(l.p1)==0?-1:0):1;
91 }
92 int seg_intersect(const line &l)const{
93     T c1=ori(l.p1), c2=ori(l.p2);
94     T c3=l.ori(p1), c4=l.ori(p2);
95     if(c1==0&&c2==0){//共線
96         bool b1=btw(l.p1)>=0,b2=btw(l.p2)>=0;
97         T a3=1.btw(p1),a4=1.btw(p2);
98         if(b1&&b2&&a3==0&&a4>=0) return 2;
99         if(b1&&b2&&a3>=0&&a4==0) return 3;
100        if(b1&&b2&&a3>=0&&a4>=0) return 0;
101        return -1;//無限交點
102    }else if(c1*c2<=0&&c3*c4<=0)return 1;
103    return 0;//不相交
104 }
105 point<T> line_intersection(const line &l)const{/*直線交點*/
106     point<T> a=p2-p1,b=l.p2-l.p1,s=l.p1-p1;
107     //if(a.cross(b)==0)return INF;
108     return p1+a*(s.cross(b)/a.cross(b));
109 }
110 point<T> seg_intersection(const line &l)const{//線段交點
111     int res=seg_intersect(l);
112     if(res<=0) assert(0);
113     if(res==2) return p1;
114     if(res==3) return p2;
115     return line_intersection(l);
116 }
117 };
118 template<typename T>
119 struct polygon{
120     polygon(){}
121     vector<point<T>> p;//逆時針順序
122     T area()const{//面積
123         T ans=0;
124         for(int i=p.size()-1,j=0;j<(int)p.size();i=j++){
125             ans+=p[i].cross(p[j]);
126         }
127         return ans/2;
128     }
129     point<T> center_of_mass()const{//重心
130         T cx=0,cy=0,w=0;
131         for(int i=p.size()-1,j=0;j<(int)p.size();i=j++){
132             T a=p[i].cross(p[j]);
133             cx+=(p[i].x+p[j].x)*a;

```

```

131     cy+=(p[i].y+p[j].y)*a;
132     w+=a;
133 }
134 return point<T>(cx/3/w,cy/3/w);
135 }
136 char ahas(const point<T>& t)const{//點是否在簡單多邊形內
137     是的話回傳1、在邊上回傳-1、否則回傳0
138     bool c=0;
139     for(int i=0,j=p.size()-1;i<p.size();j=++i)
140         if(line<T>(p[i],p[j]).point_on_segment(t))return -1;
141     else if((p[i].y>t.y)!=p[j].y>t.y)&&
142         t.x<(p[j].x-p[i].x)*(t.y-p[i].y)/(p[j].y-p[i].y)+p[i].x)
143         c=!c;
144     return c;
145 }
146 char point_in_convex(const point<T>&x)const{
147     int l=1,r=(int)p.size()-2;
148     while(l<r){//點是否在凸多邊形內，是的話回傳1、在邊上回傳
149         -1、否則回傳0
150         int mid=(l+r)/2;
151         T a1=(p[mid]-p[0]).cross(x-p[0]);
152         T a2=(p[mid+1]-p[0]).cross(x-p[0]);
153         if(a1>0&&a2<=0){
154             T res=(p[mid+1]-p[mid]).cross(x-p[mid]);
155             return res>0?1:(res>0?-1:0);
156         }else if(a1<0)r=mid-1;
157         else l=mid+1;
158     }
159     return 0;
160 }
161 vector<T> getA()const{//凸包邊對x軸的夾角
162     vector<T>res;//一定是遞增的
163     for(size_t i=0;i<p.size();++i)
164         res.push_back((p[(i+1)%p.size()]-p[i]).getA());
165     return res;
166 }
167 bool line_intersect(const vector<T>&A,const line<T>&l)
168     const{//O(LogN)
169     int f1=upper_bound(A.begin(),A.end(),(l.p1-l.p2).getA())-
170         A.begin();
171     int f2=upper_bound(A.begin(),A.end(),(l.p2-l.p1).getA())-
172         A.begin();
173     return l.cross_seg(line<T>(p[f1],p[f2]));
174 }
175 polygon cut(const line<T>&l)const{//凸包對直線切割，得到直
176     線l左側的凸包
177     polygon ans;
178     for(int n=p.size(),i=n-1,j=0;j<n;i=j++){
179         if(l.ori(p[i])>=0){
180             ans.push_back(p[i]);
181             if(l.ori(p[j])<0)
182                 ans.push_back(l.line_intersection(line<T>(p[i],p[j]),l));
183         }else if(l.ori(p[j])>0)
184             ans.push_back(l.line_intersection(line<T>(p[i],p[j]),l));
185     }
186     return ans;
187 }
188 static bool monotone_chain_cmp(const point<T>&a,const
189     point<T>&b){//凸包排序函數
190     return (a.x<b.x)||((a.x==b.x&&a.y<b.y));
191 }
192 void monotone_chain(vector<point<T>>&s){//凸包
193     sort(s.begin(),s.end(),monotone_chain_cmp);
194     p.resize(s.size()+1);
195     int m=0;
196     for(size_t i=0;i<s.size();++i){
197         while(m>2&&(p[m-1]-p[m-2]).cross(s[i]-p[m-2])<=0)--m;
198         p[m++]=s[i];
199     }
200     for(int i=s.size()-2,t=m+1;i>=0;--i){
201         while(m>=t&&(p[m-1]-p[m-2]).cross(s[i]-p[m-2])<=0)--m;
202         p[m++]=s[i];
203     }
204     if(s.size()>1)--m;
205     p.resize(m);
206 }
207 T diam()const{//直徑
208     int n=p.size(),t=1;
209     T ans=0;p.push_back(p[0]);
210     for(int i=0;i<n;i++){
211         point<T> now=p[i+1]-p[i];
212         while(now.cross(p[t+1]-p[i])>now.cross(p[t]-p[i]))t=(t
213             +1)%n;
214         ans=max(ans,(p[i]-p[t]).abs2());
215     }
216     return p.pop_back(),ans;
217 }
218 T min_cover_rectangle()const{//最小覆蓋矩形
219     int n=p.size(),t=1,r=1,l;
220     if(n<3)return 0;//也可以做最小周長矩形
221     T ans=1e99;p.push_back(p[0]);
222     for(int i=0;i<n;i++){
223         point<T> now=p[i+1]-p[i];
224         while(now.cross(p[t+1]-p[i])>now.cross(p[t]-p[i]))t=(t
225             +1)%n;
226         while(now.dot(p[r+1]-p[i])>now.dot(p[r]-p[i]))r=(r+1)%n;
227         if(!i)l=r;
228         while(now.dot(p[l+1]-p[i])<=now.dot(p[l]-p[i]))l=(l+1)%n;
229         T d=now.abs2();
230         T tmp=now.cross(p[t]-p[i])*(now.dot(p[r]-p[i])-now.dot(
231             p[l]-p[i]))/d;
232         ans=min(ans,tmp);
233     }
234     return p.pop_back(),ans;
235 }
236 T dis2(polygon &p1){const{//凸包最近距離平方
237     vector<point<T>>&P=p,&Q=p1.p;
238     int n=P.size(),m=Q.size(),l=0,r=0;
239     for(int i=0;i<n;++i)if(P[i].y<P[l].y)l=i;
240     for(int i=0;i<m;++i)if(Q[i].y<Q[r].y)r=i;
241     P.push_back(P[0]),Q.push_back(Q[0]);
242     T ans=1e99;
243     for(int i=0;i<n;++i){
244         while((P[l]-P[l+1]).cross(Q[r+1]-Q[r])<0)r=(r+1)%m;
245         ans=min(ans,line<T>(P[l],P[l+1]).seg_dis2(line<T>(Q[r],
246             Q[r+1])));
247         l=(l+1)%n;
248     }
249     return P.pop_back(),Q.pop_back(),ans;
250 }
251 static char sign(const point<T>&t){
252     return (t.y==0?t.x:t.y)<0;
253 }
254 static bool angle_cmp(const line<T>&A,const line<T>&B){
255     point<T> a=A.p2-A.p1,b=B.p2-B.p1;
256     return sign(a)<sign(b)||((sign(a)==sign(b)&&a.cross(b)>0));
257 }
258 int halfplane_intersection(vector<line<T>>&s){//半平面交
259     sort(s.begin(),s.end(),angle_cmp);//線段左側為該線段半平面
260     int L,R,n=s.size();
261     vector<point<T>> px(n);
262     vector<line<T>> q(n);
263     q[L=R=0]=s[0];
264     for(int i=1;i<n;++i){
265         while(L<R&&s[i].ori(px[R-1])<=0)--R;
266         while(L<R&&s[i].ori(px[L])<=0)++L;
267         q[++R]=s[i];
268         if(q[R].parallel(q[R-1])){
269             --R;
270             if(q[R].ori(s[i].p1)>0)q[R]=s[i];
271         }
272         if(L<R)px[R-1]=q[R-1].line_intersection(q[R]);
273     }
274     while(L<R&&q[L].ori(px[R-1])<=0)--R;
275     p.clear();
276     if(R-L<=1)return 0;
277     px[R]=q[R].line_intersection(q[L]);
278     for(int i=L;i<R;++i)p.push_back(px[i]);
279     return R-L+1;
280 }
281 template<typename T>
282 struct triangle{
283     point<T> a,b,c;
284     triangle(){}
285     triangle(const point<T>&a,const point<T>&b,const point<T>&c):a(a),b(b),c(c){}
286     T area()const{
287         T t=(b-a).cross(c-a)/2;
288         return t>0?t:-t;
289     }
290     point<T> barycenter()const{//重心
291         return (a+b+c)/3;
292     }
293     point<T> circumcenter()const{//外心
294         static line<T> u,v;
295         u.p1=(a+b)/2;
296         u.p2=point<T>(u.p1.x-a.y+b.y,u.p1.y+a.x-b.x);
297         v.p1=(a+c)/2;
298         v.p2=point<T>(v.p1.x-a.y+c.y,v.p1.y+a.x-c.x);
299         return u.line_intersection(v);
300 }
301 point<T> incenter()const{//內心
302     T A=sqrt((b-c).abs2()),B=sqrt((a-c).abs2()),C=sqrt((a-b).abs2());
303     return point<T>(A*a.x+B*b.x+C*c.x,A*a.y+B*b.y+C*c.y)/(A+B+C);
304 }
305 point<T> perpercenter()const{//垂心
306     return barycenter()*3-circumcenter()*2;
307 }
308 };
309 template<typename T>
310 struct point3D{
311     T x,y,z;
312     point3D(){}
313     point3D(const T&x,const T&y,const T&z):x(x),y(y),z(z){}
314     point3D operator+(const point3D &b)const{

```

```

305     return point3D(x+b.x,y+b.y,z+b.z);
306 point3D operator-(const point3D &b)const{
307     return point3D(x-b.x,y-b.y,z-b.z);
308 point3D operator*(const T &b)const{
309     return point3D(x*b,y*b,z*b);
310 point3D operator/(const T &b)const{
311     return point3D(x/b,y/b,z/b);
312 bool operator==(const point3D &b)const{
313     return x==b.x&&y==b.y&&z==b.z;
314 T dot(const point3D &b)const{
315     return x*b.x+y*b.y+z*b.z;
316 point3D cross(const point3D &b)const{
317     return point3D(y*b.z-z*b.y,z*b.x-x*b.z,x*b.y-y*b.x);
318 T abs2()const{//向量長度的平方
319     return dot(*this);
320 T area2(const point3D &b)const{//和b、原點圍成面積的平方
321     return cross(b).abs2()/4;
322 };
323 template<typename T>
324 struct line3D{
325     point3D<T> p1,p2;
326     line3D(){}
327     line3D(const point3D<T> &p1,const point3D<T> &p2):p1(p1),p2(p2){}
328     T dis2(const point3D<T> &p,bool is_segment=0)const{//點跟直
329         線/線段的距離平方
330     point3D<T> v=p2-p1,v1=p-p1;
331     if(is_segment){
332         point3D<T> v2=p-p2;
333         if(v.dot(v1)<=0)return v1.abs2();
334         if(v.dot(v2)>=0)return v2.abs2();
335     }
336     point3D<T> tmp=v.cross(v1);
337     return tmp.abs2()/v.abs2();
338 }
339 pair<point3D<T>,point3D<T>> closest_pair(const line3D<T> &
340     l)const{
341     point3D<T> v1=(p1-p2),v2=(l.p1-l.p2);
342     point3D<T> N=v1.cross(v2),ab(p1-l.p1);
343     //if(N.abs2()==0)return NULL;平行或重合
344     T tmp=N.dot(ab),ans=tmp*tmp/N.abs2();{//最近點對距離
345     point3D<T> d1=p2-p1,d2=l.p2-l.p1,D=d1.cross(d2),G=l.p1-p1
346     ;
347     T t1=(G.cross(d2)).dot(D)/D.abs2();
348     T t2=(G.cross(d1)).dot(D)/D.abs2();
349     return make_pair(p1+d1*t1,l.p1+d2*t2);
350 }
351 bool same_side(const point3D<T> &a,const point3D<T> &b)
352     const{
353     return (p2-p1).cross(a-p1).dot((p2-p1).cross(b-p1))>0;
354 }
355 };
356 template<typename T>
357 struct plane{
358     point3D<T> p0,n;//平面上的點和法向量
359     plane(){}
360     plane(const point3D<T> &p0,const point3D<T> &n):p0(p0),n(n)
361     {}
362     T dis2(const point3D<T> &p)const{//點到平面距離的平方
363     T tmp=(p-p0).dot(n);
364     return tmp*tmp/n.abs2();
365 }
366 point3D<T> projection(const point3D<T> &p)const{
367     return p-n*(p-p0).dot(n)/n.abs2();
368 }

```

```

363 }
364 point3D<T> line_intersection(const line3D<T> &l1)const{
365     T tmp=n.dot(l.p2-l.p1);//等於0表示平行或重合該平面
366     return l.p1+(l.p2-l.p1)*(n.dot(p0-l.p1)/tmp);
367 }
368 line3D<T> plane_intersection(const plane &p1)const{
369     point3D<T> e=n.cross(p1.n),v=n.cross(e);
370     T tmp=p1.n.dot(v);//等於0表示平行或重合該平面
371     point3D<T> q=p0+(v*(p1.n.dot(p1.p0-p0))/tmp);
372     return line3D<T>(q,q+e);
373 }
374 };
375 template<typename T>
376 struct triangle3D{
377     point3D<T> a,b,c;
378     triangle3D(){}
379     triangle3D(const point3D<T> &a,const point3D<T> &b,const
380         point3D<T> &c):a(a),b(b),c(c){}
381     bool point_in(const point3D<T> &p)const{//點在該平面上的投
382         影在三角形中
383     return line3D<T>(b,c).same_side(p,a)&&line3D<T>(a,c).
384         same_side(p,b)&&line3D<T>(a,b).same_side(p,c);
385 }
386 };
387 template<typename T>
388 struct tetrahedron{//四面體
389     point3D<T> a,b,c,d;
390     tetrahedron(){}
391     tetrahedron(const point3D<T> &a,const point3D<T> &b,const
392         point3D<T> &c,const point3D<T> &d):a(a),b(b),c(c),d(d)
393     {}
394     T volume6()const{//體積的六倍
395     return (d-a).dot((b-a).cross(c-a));
396 }
397 point3D<T> centroid()const{
398     return (a+b+c+d)/4;
399 }
400 bool point_in(const point3D<T> &p)const{
401     return triangle3D<T>(a,b,c).point_in(p)&&triangle3D<T>(c,
402         d,a).point_in(p);
403 }
404 };
405 template<typename T>
406 struct convexhull3D{
407     static const int MAXN=1005;
408     struct face{
409         int a,b,c;
410         face(int a,int b,int c):a(a),b(b),c(c){}
411     };
412     vector<point3D<T>> pt;
413     vector<face> ans;
414     int fid[MAXN][MAXN];
415     void build(){
416         int n=pt.size();
417         ans.clear();
418         memset(fid,0,sizeof(fid));
419         ans.emplace_back(0,1,2);//注意不能共線
420         ans.emplace_back(2,1,0);
421         int ftop = 0;
422         for(int i=3, ftop=1; i<n; ++i,++ftop){
423             vector<face> next;
424             for(auto &f:ans){
425                 T d=(pt[i]-pt[f.a]).dot((pt[f.b]-pt[f.a]).cross(pt[f.
426                     c]-pt[f.a]));
427             }
428         }
429     }
430 }

```

```

420     if(d<=0) next.push_back(f);
421     int ff=0;
422     if(d>0) ff=ftop;
423     else if(d<0) ff=-ftop;
424     fid[f.a][f.b]=fid[f.b][f.c]=fid[f.c][f.a]=ff;
425 }
426 for(auto &f:ans){
427     if(fid[f.a][f.b]>0 && fid[f.a][f.b]!=fid[f.b][f.a])
428         next.emplace_back(f.a,f.b,i);
429     if(fid[f.b][f.c]>0 && fid[f.b][f.c]!=fid[f.c][f.b])
430         next.emplace_back(f.b,f.c,i);
431     if(fid[f.c][f.a]>0 && fid[f.c][f.a]!=fid[f.a][f.c])
432         next.emplace_back(f.c,f.a,i);
433 }
434 ans=next;
435 }
436 }
437 point3D<T> centroid()const{
438     point3D<T> res(0,0,0);
439     T vol=0;
440     for(auto &f:ans){
441         T tmp=pt[f.a].dot(pt[f.b].cross(pt[f.c]));
442         res=res+(pt[f.a]+pt[f.b]+pt[f.c])*tmp;
443         vol+=tmp;
444     }
445     return res/(vol*4);
446 }
447 };

```

## 6 Graph

### 6.1 Find-Bridge

```

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

```

## 6.2 Find-AP

```

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

```

## 6.3 HLD

```

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

```

```

31 int path_query(int a, int b) {
32     int res = 0;
33     while (topf[a] != topf[b]) { /// 若不在同一條鍊上
34         if (depth[topf[a]] < depth[topf[b]]) swap(a, b);
35         res = max(res, 011); // query : l = id[topf[a]],
36         r = id[a]
37         a = pa[topf[a]];
38     }
39     /// 此時已在同一條鍊上
40     if (depth[a] < depth[b]) swap(a, b);
41     res = max(res, 011); // query : l = id[b], r = id[a]
42     return res;
43 }

```

## 6.4 Tree-Isomorphism

```

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

```

```

44     vector<int> v;
45     for (auto x : g[now]){
46         if (x!=pre){
47             int add=dfs(g, m, id, x, now);
48             v.push_back(add);
49         }
50     }
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
63     // init
64     id1=0;
65     id2=0;
66     c1={0, 0};
67     c2={0, 0};
68     fill(sz1.begin(), sz1.begin()+n+1, 0);
69     fill(sz2.begin(), sz2.begin()+n+1, 0);
70     fill(we1.begin(), we1.begin()+n+1, 0);
71     fill(we2.begin(), we2.begin()+n+1, 0);
72     for (int i=1; i<=n; i++){
73         g1[i].clear();
74         g2[i].clear();
75     }
76     m1.clear();
77     m2.clear();
78
79     // input
80     cin >> n;
81     for (int i=0; i<n-1; i++){
82         cin >> a >> b;
83         g1[a].push_back(b);
84         g1[b].push_back(a);
85     }
86     for (int i=0; i<n-1; i++){
87         cin >> a >> b;
88         g2[a].push_back(b);
89         g2[b].push_back(a);
90     }
91
92     // get tree centroid
93     centroid(g1, sz1, we1, c1, 1, 0);
94     centroid(g2, sz2, we2, c2, 1, 0);
95
96     // process
97     int res1=0, res2=0, res3=0;
98     if (c2.second!=0){
99         res1=dfs(g1, m1, id1, c1.first, 0);
100         m2=m1;
101         id2=id1;
102         res2=dfs(g2, m1, id1, c2.first, 0);
103         res3=dfs(g2, m2, id2, c2.second, 0);
104     }else if (c1.second!=0){
105         res1=dfs(g2, m1, id1, c2.first, 0);
106         m2=m1;
107         id2=id1;
108         res2=dfs(g1, m1, id1, c1.first, 0);

```



```

110     res3=dfs(g1, m2, id2, c1.second, 0);
111 }else{
112     res1=dfs(g1, m1, id1, c1.first, 0);
113     res2=dfs(g2, m1, id1, c2.first, 0);
114 }
115
116 // output
117 cout << (res1==res2 || res1==res3 ? "YES" : "NO") << endl;
118 ;
119 return;
120 }
121
122 signed main(void){
123     fastio;
124
125     int t=1;
126     cin >> t;
127     while (t--){
128         solve1();
129     }
130     return 0;
131 }

```

## 6.5 Bridge BCC

```

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

```

## 6.6 Cut BCC

```

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

```

## 6.7 圓方樹

```

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

```

```

19 #define deb5 cerr << "deb5" << endl;
20 #define bye exit(0);
21 using namespace std;
22
23 const int mxn = (int)(2e5) + 10;
24 const int mxlg = 17;
25 int last_special_node = (int)(1e5) + 1;
26 vector<int> E[mxn], F[mxn];
27
28 struct edg{
29     int fr, to;
30     edg(int _fr, int _to){
31         fr = _fr;
32         to = _to;
33     }
34 };
35 ostream& operator<<(ostream& os, edg x){os << x.fr << "--" << x.to;}
36 vector<edg> EV;
37
38 void tarjan(int v, int par, stack<int>& S){
39     static vector<int> dfn(mxn), low(mxn);
40     static vector<bool> to_add(mxn);
41     static int nowT = 0;
42
43     int child = 0;
44     nowT += 1;
45     dfn[v] = low[v] = nowT;
46     for(auto &ne:E[v]){
47         int i = EV[ne].to;
48         if(i == par) continue;
49         if(!dfn[i]){
50             S.push(ne);
51             tarjan(i, v, S);
52             child += 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     }
78     else{
79         low[v] = min(low[v], dfn[i]);
80         if(dfn[i] < dfn[v]){ // edge i--v will be visited twice at here, but we only need one.
81             S.push(ne);
82         }
83     }
84 }

```

```

83     }
84 }
85
86 int dep[mxn], jmp[mxn][mxlg];
87 void dfs_lca(int v, int par, int depth){
88     dep[v] = depth;
89     for(auto &i:F[v]){
90         if(i == par) continue;
91         jmp[i][0] = v;
92         dfs_lca(i, v, depth + 1);
93     }
94 }
95
96 inline void build_lca(){
97     jmp[1][0] = 1;
98     dfs_lca(1, -1, 1);
99     lp(j,1,mxlg){
100         lp(i,1,mxn){
101             jmp[i][j] = jmp[jmp[i][j-1]][j-1];
102         }
103     }
104 }
105
106 inline int lca(int x, int y){
107     if(dep[x] < dep[y]){ swap(x, y); }
108
109     int diff = dep[x] - dep[y];
110     lp(j,0,mxlg){
111         if((diff >> j) & 1){
112             x = jmp[x][j];
113         }
114     }
115     if(x == y) return x;
116
117     for(int j = mxlg - 1; j >= 0; j--){
118         if(jmp[x][j] != jmp[y][j]){
119             x = jmp[x][j];
120             y = jmp[y][j];
121         }
122     }
123     return jmp[x][0];
124 }
125
126 inline bool can_reach(int fr, int to){
127     if(dep[to] > dep[fr]) return false;
128
129     int diff = dep[fr] - dep[to];
130     lp(j,0,mxlg){
131         if((diff >> j) & 1){
132             fr = jmp[fr][j];
133         }
134     }
135     return fr == to;
136 }
137
138 int main(){
139     ios::sync_with_stdio(false); cin.tie(0);
140     // freopen("test_input.txt", "r", stdin);
141     int n, m, q; cin >> n >> m >> q;
142     lp(i,0,m){
143         int u, v; cin >> u >> v;
144         E[u].pb(EV.size());
145         EV.pb(edg(u, v));
146         E[v].pb(EV.size());
147         EV.pb(edg(v, u));
148     }

```

```

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

```

## 6.8 SCC 與縮點

```

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

```

```

38     }
39
40     void dfs1(vector<vector<int>> &G, int now){
41         vis[now] = 1;
42         for (auto x : G[now]){
43             if (vis[x]==0){
44                 dfs1(G, x);
45             }
46         }
47         order.push_back(now);
48         return;
49     }
50
51     void dfs2(vector<vector<int>> &G, int now){
52         SCC_id[now] = SCC.size()-1;
53         SCC.back().push_back(now);
54         vis[now] = 1;
55
56         for (auto x : G[now]){
57             if (vis[x]==0){
58                 dfs2(G, x);
59             }
60         }
61         return;
62     }
63
64     void compress(){
65         fill(vis.begin(), vis.end(), 0);
66         for (int i=0; i<n; i++){
67             if (vis[i]==0){
68                 dfs1(G, i);
69             }
70         }
71
72         fill(vis.begin(), vis.end(), 0);
73         reverse(order.begin(), order.end());
74         for (int i=0; i<n; i++){
75             if (vis[order[i]]==0){
76                 SCC.push_back(vector<int>());
77                 dfs2(inv_G, order[i]);
78             }
79         }
80
81         for (int i=0; i<m; i++){
82             if (SCC_id[edges[i].first]!=SCC_id[edges[i].second]){
83                 result[SCC_id[edges[i].first]].push_back(SCC_id[edges[i].second]);
84             }
85         }
86         for (int i=0; i<SCC.size(); i++){
87             sort(result[i].begin(), result[i].end());
88             result[i].resize(unique(result[i].begin(), result[i].end())-result[i].begin());
89         }
90     }
91 };

```

## 6.9 Dinic

```

1 // 一般圖：O(EV^2)
2 // 二分圖：O(EV)
3 struct Flow{

```



```

4 struct Edge{
5     int v, rc, rid;
6 };
7 vector<vector<Edge>> G;
8 void add(int u, int v, int c){
9     G[u].push_back({v, c, G[v].size()});
10    G[v].push_back({u, 0, G[u].size()-1});
11 }
12 vector<int> dis, it;
13
14 Flow(int n){
15     G.resize(n);
16     dis.resize(n);
17     it.resize(n);
18 }
19
20 int dfs(int u, int t, int f){
21     if (u==t || f==0) return f;
22     for (int &i=it[u] ; i<G[u].size() ; i++){
23         auto &[v, rc, rid] = G[u][i];
24         if (dis[v]!=dis[u]+1) continue;
25         int df = dfs(v, t, min(f, rc));
26         if (df<=0) continue;
27         rc -= df;
28         G[v][rid].rc += df;
29         return df;
30     }
31     return 0;
32 }
33
34 int flow(int s, int t){
35     int ans = 0;
36     while (true){
37         fill(dis.begin(), dis.end(), INF);
38         queue<int> q;
39         q.push(s);
40         dis[s] = 0;
41
42         while (q.size()){
43             int u = q.front(); q.pop();
44             for (auto [v, rc, rid] : G[u]){
45                 if (rc<=0 || dis[v]<INF) continue;
46                 dis[v] = dis[u]+1;
47                 q.push(v);
48             }
49         }
50         if (dis[t]==INF) break;
51
52         fill(it.begin(), it.end(), 0);
53         while (true){
54             int df = dfs(s, t, INF);
55             if (df<=0) break;
56             ans += df;
57         }
58     }
59     return ans;
60 }
61 // the code below constructs minimum cut
62 void dfs_mincut(int now, vector<bool> &vis){
63     vis[now] = true;
64     for (auto &[v, rc, rid] : G[now]){
65         if (vis[v]==false && rc>0){
66             dfs_mincut(v, vis);
67         }
68     }
69 }

```

```

70
71 vector<pair<int, int>> construct(int n, int s, vector<pair<
72     int, int>> &E){
73     // E is G without capacity
74     vector<bool> vis(n);
75     dfs_mincut(s, vis);
76     vector<pair<int, int>> ret;
77     for (auto &[u, v] : E){
78         if (vis[u]==true && vis[v]==false){
79             ret.emplace_back(u, v);
80         }
81     }
82     return ret;
83 }

```

## 6.10 Dijkstra

```

1 // 可以在  $O(E \log E)$  的時間複雜度解決在無負權有向圖單點源最短
2 // 路
3 const int INF = 2e18; // 要確保 INF 開的足夠大
4
5 vector<vector<pair<int, int>>> G(n); // G[i] = <節點, 權重>
6 vector<int> dis(n, INF);
7 priority_queue<pair<int, int>, vector<pair<int, int>>,
8     greater<pair<int, int>>> pq;
9 dis[s] = 0;
10 pq.push({0, s});
11
12 while (pq.size()){
13     int now_dis = pq.top().first;
14     int now_node = pq.top().second;
15     pq.pop();
16
17     if (now_dis>dis[now_node]) continue;
18
19     for (auto x : G[now_node]){
20         if (now_dis+x.second<dis[x.first]){
21             dis[x.first] = now_dis+x.second;
22             pq.push({dis[x.first], x.first});
23         }
24     }
25 }

```

## 6.11 定理

- 最小點覆蓋 = 最大匹配 =  $n$  - 最大點獨立集
  - 最小點覆蓋：選最少點讓所有的邊都有碰到一個點
  - 最大點獨立集：選最多不共邊的點
- 只有邊帶權的二分圖的定理 (可能不重要)
  - w-vertex-cover (帶權點覆蓋)：每條邊的兩個連接點被選中的次數總和至少要是  $w_e$ 。
  - w-weight matching (帶權匹配)
  - minimum vertex count of w-vertex-cover = maximum weight count of w-weight matching (一個點可以被選很多次，但邊不行)

- 點、邊都帶權的二分圖的定理 (可能不重要)

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

## 6.12 MCMF

```

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

```

```

55     for (int v=t ; v!=s ; v=par[v]){
56         G[par[v]][par_eid[v]].rc -= cur;
57         G[v][G[par[v]][par_eid[v]].rv].rc += cur;
58     }
59 }
60 return {f1, cost};
61 }
62
63 vector<pair<int, int>> construct(){
64     vector<pair<int, int>> ret;
65     for (int i=0 ; i<n ; i++){
66         for (auto x : G[i]){
67             if (x.rc==0){
68                 ret.push_back({i+1, x.u-n+1});
69                 break;
70             }
71         }
72     }
73     return ret;
74 }
75 };

```

### 6.13 Dinic with double

```

1  const double double_INF = 1e18;
2  const int INF = (int)(1e9 + 10);
3
4  struct Flow{
5      const double eps = 1e-9;
6      struct Edge{
7          int v; double rc; int rid;
8      };
9      vector<vector<Edge>> G;
10     void add(int u, int v, double c){
11         G[u].push_back({v, c, G[v].size()});
12         G[v].push_back({u, 0, G[u].size()-1});
13     }
14     vector<int> dis, it;
15
16     Flow(int n){
17         G.resize(n);
18         dis.resize(n);
19         it.resize(n);
20     }
21
22     double dfs(int u, int t, double f){
23         if (u == t || abs(f) < eps) 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             double df = dfs(v, t, min(f, rc));
28             if (abs(df) <= eps) continue;
29             rc -= df;
30             G[v][rid].rc += df;
31             return df;
32         }
33         return 0;
34     }
35
36     double flow(int s, int t){
37         double ans = 0;
38         while (true){
39             fill(dis.begin(), dis.end(), INF);

```

```

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

```

### 6.14 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;
34 }

```

### 6.15 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 };

```

## 7 Math

### 7.1 Burnside's-Lemma

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

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

### 7.2 線性篩

```

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

```

### 7.3 Lucas's-Theorem

```

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

```

## 7.4 Matrix

```

1 struct Matrix{
2     int n, m;
3     vector<vector<int>> arr;
4
5     Matrix(int _n, int _m){
6         n = _n;
7         m = _m;
8         arr.resize(n, vector<int>(m));
9     }
10
11     Matrix operator * (Matrix b){
12         Matrix b_t(b.m, b.n);
13         for (int i=0 ; i<b.n ; i++){
14             for (int j=0 ; j<b.m ; j++){
15                 b_t.arr[j][i] = b.arr[i][j];
16             }
17         }
18
19         Matrix ret(n, b.m);
20         for (int i=0 ; i<n ; i++){
21             for (int j=0 ; j<b.m ; j++){
22                 for (int k=0 ; k<m ; k++){
23                     ret.arr[i][j] += arr[i][k]*b_t.arr[j][k];
24                     ret.arr[i][j] %= MOD;
25                 }
26             }
27         }
28         return ret;
29     }
30
31     Matrix pow(int p){
32         Matrix ret(n, n), mul = *this;
33         for (int i=0 ; i<n ; i++){
34             ret.arr[i][i] = 1;
35         }
36
37         for ( ; p ; p>>=1){
38             if (p&1) ret = ret*mul;
39             mul = mul*mul;
40         }
41         return ret;
42     }
43 }
44
45 int det(){
46     vector<vector<int>> arr = this->arr;
47     bool flag = false;
48     for (int i=0 ; i<n ; i++){
49         int target = -1;
50         for (int j=i ; j<n ; j++){
51             if (arr[j][i]){
52                 target = j;
53                 break;
54             }
55         }
56         if (target==-1) return 0;
57         if (i!=target){
58             swap(arr[i], arr[target]);
59             flag = !flag;
60         }
61     }
62
63     for (int j=i+1 ; j<n ; j++){

```

```

64         if (!arr[j][i]) continue;
65         int freq = arr[j][i]*qp(arr[i][i], MOD-2)%MOD;
66         for (int k=i ; k<n ; k++){
67             arr[j][k] -= freq*arr[i][k];
68             arr[j][k] = (arr[j][k]%MOD+MOD)%MOD;
69         }
70     }
71 }
72
73 int ret = !flag ? 1 : MOD-1;
74 for (int i=0 ; i<n ; i++){
75     ret *= arr[i][i];
76     ret %= MOD;
77 }
78 return ret;
79 }
80 };

```

### 7.5 最大質因數

```

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

```

### 7.6 中國剩餘定理 (m 不互質)

```

1 int extgcd(int a, int b, int &x, int &y){
2     if (b==0){
3         x=1, y=0;
4         return a;
5     }
6
7     int ret=extgcd(b, a%b, y, x);
8     y-=a/b*x;
9     return ret;
10 }
11
12 // 對於方程組的式子兩兩求解
13 // {是否有解, {a, m}}
14 pair<bool, pair<int, int>> CRT(int a1, int m1, int a2, int m2)
15 {
16     int g=__gcd(m1, m2);
17     if ((a2-a1)%g!=0) return {0, {-1, -1}};
18
19     int x, y;
20     extgcd(m1, m2, x, y);
21
22     x=(a2-a1)*x/g; // 兩者不能相反
23     a1=x*m1+a1;
24     m1=m1*m2/g;
25     a1=(a1%m1+m1)%m1;

```

```

25     return {1, {a1, m1}};
26 }

```

## 7.7 歐拉公式

```

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

```

## 7.8 歐拉定理

若  $a, m$  互質，則：

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

若  $a, m$  可能是任何數，則：

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

## 7.9 Fraction

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 /// Fraction template starts ///

```

```

5 #define fraction_template_bonus_check
6 const long long ll_overflow_warning_value = (long long)(3e9);
7
8 long long gcd(long long a, long long b){
9     if(a == 0) return 0;
10    if(b == 0) return a;
11    if(a < b) return gcd(b,a);
12    return gcd(b, a%b);
13 }
14 struct frac{
15     long long a, b;
16     frac(long long _a = 0, long long _b = 1){
17         a = _a; b = _b;
18         if(b == 0){
19             cerr << "Error: division by zero\n";
20             cerr << "Called : Constructor(" << _a << ", " <<
                _b << ")\n";
21             return;
22         }
23         if(a == 0){b = 1; return;}
24         if(b < 0){a = -a; b = -b;}
25         long long gcd_ab = gcd(std::abs(a), b);
26         if(gcd_ab != 1){a /= gcd_ab; b /= gcd_ab;}
27
28         #ifdef fraction_template_bonus_check
29         if(std::abs(a) > ll_overflow_warning_value || b >
            ll_overflow_warning_value){
30             cerr << "Overflow warning : " << a << "/" << b <<
                "\n";
31         }
32         #endif // fraction_template_bonus_check
33     }
34     frac operator+(frac const &B){
35         return frac(a*(B.b)+(B.a)*b, b*(B.b));
36     }
37     frac operator-(frac const &B){
38         return frac(a*(B.b)-(B.a)*b, b*(B.b));
39     }
40     frac operator*(frac const &B){
41         return frac(a*(B.a), b*(B.b));
42     }
43     frac operator/(frac const &B){
44         return frac(a*(B.b), b*(B.a));
45     }
46
47     frac operator+=(frac const &B){
48         *this = frac(a*(B.b)+(B.a)*b, b*(B.b));
49     }
50     frac operator-=(frac const &B){
51         *this = frac(a*(B.b)-(B.a)*b, b*(B.b));
52     }
53     frac operator*=(frac const &B){
54         *this = frac(a*(B.a), b*(B.b));
55     }
56     frac operator/=(frac const &B){
57         *this = frac(a*(B.b), b*(B.a));
58     }
59
60     frac abs(){
61         a = std::abs(a);
62         return *this;
63     }
64
65     bool operator<(frac const &B){
66         return a*B.b < B.a*b;
67     }
68     bool operator<=(frac const &B){
69         return a*B.b <= B.a*b;
70     }
71     bool operator>(frac const &B){
72         return a*B.b > B.a*b;
73     }
74     bool operator>=(frac const &B){
75         return a*B.b >= B.a*b;
76     }
77     bool operator==(frac const &B){
78         return a * B.b == B.a * b;
79     }
80     bool operator!=(frac const &B){
81         return a * B.b != B.a * b;
82     }
83 }
84
85 ostream& operator<<(ostream &os, const frac& A){
86     os << A.a << "/" << A.b;
87     return os;
88 }
89
90 /// Fraction template ends ///
91
92 void test(frac A, frac B){
93     cout << "A = " << A << endl;
94     cout << "B = " << B << endl;
95     cout << endl;
96     cout << "A + B = " << A + B << endl;
97     cout << "A - B = " << A - B << endl;
98     cout << "A * B = " << A * B << endl;
99     cout << "A / B = " << A / B << endl;
100    cout << endl;
101    cout << "(A < B) = " << (A < B) << endl;
102    cout << "(A <= B) = " << (A <= B) << endl;
103    cout << "(A > B) = " << (A > B) << endl;
104    cout << "(A >= B) = " << (A >= B) << endl;
105    cout << "(A == B) = " << (A == B) << endl;
106    cout << "(A != B) = " << (A != B) << endl;
107    cout << "-----\n";
108    return;
109 }
110
111 int main(){
112     frac tmp1(-7, 2);
113     frac tmp2(5, 3);
114     test(tmp1, tmp2);
115
116     frac tmp3(-7);
117     frac tmp4(0);
118     test(tmp3, tmp4);
119     return 0;
120 }

```

## 7.10 錯排公式

錯排公式：( $n$  個人中，每個人皆不再原來位置的組合數)

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

## 7.11 Quick-Pow

```

1 int qp(int b, int p, int m = MOD){
2     int ret = 1;
3     for (; p ; p>>=1){
4         if (p&1) ret = ret*b%m;
5         b = b*b%m;
6     }
7     return ret;
8 }

```

## 7.12 二元一次方程式

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

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

## 7.13 Josephus

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

## 7.14 數論分塊

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

## 7.15 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 }
```

## 7.16 中國剩餘定理 (m 互質)

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

## 7.17 Catalan

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

## 7.18 數論定理

- $1 \sim x$  質數的數量  $\approx \frac{x}{\ln x}$
- $1 \sim x$  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

## 7.19 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 }
```

## 7.20 Stirling's formula

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

## 7.21 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 };

```

## 7.22 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    }
29 }

```

28 | };

## 7.23 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

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

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

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

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

## 7.24 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; 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

```

```

32         fac = fac * i % mod;
33     }
34     inv_fac[MAX_N - 1] = fp(fac, mod - 2);
35     for (int i = MAX_N - 1; i >= 1; --i) {
36         inv_fac[i - 1] = inv_fac[i] * i % mod;
37     }
38 }
39 // You call init() many times without having a second
40 // instance of this struct.
41 void init(int X_0, vector<int> &u) {
42     v = u;
43     shift = ((1 - X_0) % mod + mod) % mod;
44     if (v.size() == 1) v.push_back(v[0]);
45     m = v.size();
46     mul.resize(m);
47 }
48 // You can use sample(x) instead of sample(x % mod).
49 int sample(int x) {
50     x = ((long long)x + shift) % mod;
51     x = (x < 0) ? (x + mod) : x;
52     long long now = 1;
53     for (int i = m; i >= 1; --i) {
54         mul[i - 1] = now;
55         now = now * (x - i) % mod;
56     }
57     int ret = 0;
58     bool neg = (m - 1) & 1;
59     now = 1;
60     for (int i = 1; i <= m; ++i) {
61         int up = now * mul[i - 1] % mod;
62         int down = inv_fac[m - i] * inv_fac[i - 1] % mod;
63         int tmp = ((long long)v[i - 1] * up % mod) * down
64             % mod;
65         ret += (neg && tmp) ? (mod - tmp) : (tmp);
66         ret = (ret >= mod) ? (ret - mod) : ret;
67         now = now * (x - i) % mod;
68         neg ^= 1;
69     }
70     return ret;
71 }
72 }
73
74 int main() {
75     int n; cin >> n;
76     vector<int> v(n);
77     for (int i = 0; i < n; ++i) {
78         cin >> v[i];
79     }
80     Lagrange L;
81     L.construct_inv_fac();
82     L.init(0, v);
83     int x; cin >> x;
84     cout << L.sample(x);
85 }

```

## 8 String

### 8.1 Hash

```

1 int A = rng(1e5, 8e8);
2 const int B = 1e9+7;

```



```

3
4 struct RollingHash{
5     vector<int> Pow, Pre;
6     RollingHash(string s = ""){
7         Pow.resize(s.size());
8         Pre.resize(s.size());
9
10        for (int i=0 ; i<s.size() ; i++){
11            if (i==0){
12                Pow[i] = 1;
13                Pre[i] = s[i];
14            }else{
15                Pow[i] = Pow[i-1]*A%B;
16                Pre[i] = (Pre[i-1]*A+s[i])%B;
17            }
18        }
19
20        return;
21    }
22
23    int get(int l, int r){ // 取得 [l, r] 的數值
24        if (l==0) return Pre[r];
25        int res = (Pre[r]-Pre[l-1]*Pow[r-l+1])%B;
26        if (res<0) res += B;
27        return res;
28    }
29 };

```

## 8.2 Manacher

```

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

```

## 8.3 Z-Function

```

1 // 定義一個長度為 n 的文本為 T，則陣列 Z 的 Z[i] 代表 T[0:n]
2 // 和 T[i:n] 最長共同前綴
3 // bcfbd6
4 vector<int> z_function(string s){
5     vector<int> ret(s.size());
6     int ll = 0, rr = 0;
7
8     for (int i=1 ; i<s.size() ; i++){

```

```

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

```

## 8.4 KMP

```

1 // 給一個字串 S，定義函數 pi(i) = k 代表 S[1 ... k] = S[i-k
2 // +1 ... i]
3 // 4c61a3
4 vector<int> KMP(string &s){
5     n = SZ(s);
6     vector<int> ret(n);
7     int now = 0;
8     for (int i=1 ; i<n ; i++){
9         int j = ret[i-1];
10        while (j>0 && s[i]!=s[j]){
11            j = ret[j-1];
12        }
13        if (s[i]==s[j]) j++;
14        ret[i] = j;
15    }
16    return ret;
17 }

```

## 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     SuffixArray(string _s, int lim = 256) {
6         s = _s;
7         int n = s.size()+1, k = 0, a, b;
8         vector<int> x(s.begin(), s.end()), y(n), ws(max(n,
9             lim)), rank(n);
10        x.push_back(0);
11        sa = lcp = y;
12        iota(sa.begin(), sa.end(), 0);
13        for (int j=0, p=0 ; p<n ; j=max(1LL, j*2), lim=p) {
14            p = j;
15            iota(y.begin(), y.end(), n-j);
16            for (int i=0 ; i<n ; i++) if (sa[i] >= j) y[p++]
17                = sa[i] - j;
18            fill(ws.begin(), ws.end(), 0);
19            for (int i=0 ; i<n ; i++) ws[x[i]]++;
20            for (int i=1 ; i<lim ; i++) ws[i] += ws[i-1];
21            for (int i = n; i--;) sa[--ws[x[y[i]]]] = y[i];
22            swap(x, y), p = 1, x[sa[0]] = 0;

```

```

21        for (int i=1 ; i<n ; i++){
22            a = sa[i-1];
23            b = sa[i];
24            x[b] = (y[a] == y[b] && y[a+j] == y[b+j])
25                ? p-1 : p++;
26        }
27
28        for (int i=1 ; i<n ; i++) rank[sa[i]] = i;
29        for (int i=0, j ; i<n-1 ; lcp[rank[i++]]=k)
30            for (k && k--, j=sa[rank[i]-1] ; i+k<s.size() &&
31                j+k<s.size() && s[i+k]==s[j+k] ; k++);
32        sa.erase(sa.begin());
33        lcp.erase(lcp.begin(), lcp.begin()+2);
34    }
35
36    vector<int> pos; // pos[i] = i 這個值在 pos 的哪個地方
37    SparseTable st;
38    void init_lcp(){
39        pos.resize(sa.size());
40        for (int i=0 ; i<sa.size() ; i++){
41            pos[sa[i]] = i;
42        }
43        if (lcp.size()){
44            st.build(lcp);
45        }
46    }
47
48    // 用之前記得 init
49    // 回傳 [l1, r1] 跟 [l2, r2] 的 lcp，0-based
50    int get_lcp(int l1, int r1, int l2, int r2){
51        int pos_1 = pos[l1], len_1 = r1-l1+1;
52        int pos_2 = pos[l2], len_2 = r2-l2+1;
53        if (pos_1>pos_2){
54            swap(pos_1, pos_2);
55            swap(len_1, len_2);
56        }
57
58        if (l1==l2){
59            return min(len_1, len_2);
60        }else{
61            return min({st.query(pos_1, pos_2), len_1, len_2
62                });
63        }
64    }
65
66    // 檢查 [l1, r1] 跟 [l2, r2] 的大小關係，0-based
67    // 如果前者小於後者，就回傳 <0，相等就回傳 =0，否則回傳
68    // >0
69    int substring_cmp(int l1, int r1, int l2, int r2){
70        int len_1 = r1-l1+1;
71        int len_2 = r2-l2+1;
72        int res = get_lcp(l1, r1, l2, r2);
73
74        if (res<len_1 && res<len_2){
75            return s[l1+res]-s[l2+res];
76        }else if (len_1==res && len_2==res){
77            // 如果不需要以 index 作為次要排序參數，這裡要回
78            // 傳 0
79            return l1-l2;
80        }else{
81            return len_1==res ? -1 : 1;
82        }
83    }
84 }

```



```

81 // 對於位置在 <=p 的後綴，找離他左邊/右邊最接近位置 >p 的
    後綴的 lcp · 0-based
82 // pre[i] = s[i] 離他左邊最接近位置 >p 的後綴的 lcp · 0-
    based
83 // suf[i] = s[i] 離他右邊最接近位置 >p 的後綴的 lcp · 0-
    based
84 pair<vector<int>, vector<int>> get_left_and_right_lcp(int
    p){
85     vector<int> pre(p+1);
86     vector<int> suf(p+1);
87
88     { // build pre
89         int now = 0;
90         for (int i=0 ; i<s.size() ; i++){
91             if (sa[i]<=p){
92                 pre[sa[i]] = now;
93                 if (i<lcp.size()) now = min(now, lcp[i]);
94             }else{
95                 if (i<lcp.size()) now = lcp[i];
96             }
97         }
98     }
99     { // build suf
100         int now = 0;
101         for (int i=s.size()-1 ; i>=0 ; i--){
102             if (sa[i]<=p){
103                 suf[sa[i]] = now;
104                 if (i-1>=0) now = min(now, lcp[i-1]);
105             }else{
106                 if (i-1>=0) now = lcp[i-1];
107             }
108         }
109     }
110
111     return {pre, suf};
112 }
113 };

```

## 8.6 Min-Rotation

```

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

```