

Contents

1 Misc	2	4 Dynamic-Programming	8	7 Math	17
1.1 Xor-Basis	2	4.1 SOS-DP	8	7.1 Burnside's-Lemma	17
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	5 Geometry	8	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	18
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	9	7.10 錯排公式	19
1.11 random int	3	5.7 Geometry	10	7.11 Quick-Pow	19
2 Convolution	3	6 Graph	12	7.12 二元一次方程式	19
2.1 FFT	3	6.1 Find-Bridge	12	7.13 Josephus	19
2.2 FFT-2	4	6.2 Find-AP	12	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
3 Data-Structure	5	6.5 Bridge BCC	13	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	20
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	8 String	21
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     for (int i=0 ; i<n ; i++){
23         if (i<rev[i]){
24             swap(a[i], a[rev[i]]);
25         }
26     }
27
28     for (int k=1 ; k<n ; k*=2){
29         for (int i=0 ; i<n ; i+=2*k){
30             for (int j=0 ; j<k ; j++){
31                 int z = rt[j+k]*a[i+j+k]%MOD, &ai = a[i+j];
32                 ai[j+k] = ai-z+(z>ai ? MOD : 0);
33                 ai += (ai+z>MOD ? z-MOD : z);
34             }
35         }
36     }
37 }
38
39 // 0b0e99
40 vector<int> polyMul(vector<int> &a, vector<int> &b){
41     if (a.empty() || b.empty()) return {};
42     int s = a.size()+b.size()-1, B = 32-__builtin_clz(s), n = 1<<B;
43     int inv = qp(n, MOD-2);
44
45     vector<int> L(a), R(b), out(n);
46     L.resize(n), R.resize(n);
47     NTT(L), NTT(R);
48     for (int i=0 ; i<n ; i++){
49         out[-i&(n-1)] = L[i]*R[i]%MOD*inv%MOD;
50     }
51     NTT(out);
52
53     out.resize(s);
54     return out;
55 }

```

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]*y[0]);
37                 a[i+j+k] = a[i+j]-z;
38                 a[i+j] += z;
39             }
40         }
41     }
42     return;
43 }
44
45 // d3c65e
46 vector<int> PolyMul(vector<int> a, vector<int> b){
47     if (a.empty() || b.empty()) return {};
48
49     vector<int> res(a.size()+b.size()-1);
50     int B = 32-__builtin_clz(res.size()), n = (1<<B), cut =
51         int(sqrt(MOD));
52     vector<cd> L(n), R(n), outs(n), outl(n);
53
54     for (int i=0 ; i<a.size() ; i++){
55         L[i] = cd((int) a[i]/cut, (int)a[i]%cut);
56     }
57     for (int i=0 ; i<b.size() ; i++){
58         R[i] = cd((int) b[i]/cut, (int)b[i]%cut);
59     }
60     FFT(L);
61     FFT(R);
62     for (int i=0 ; i<n ; i++){
63         int j = -i&(n-1);
64         outl[j] = (L[i]+conj(L[j])) * R[i]/(2.0*n);
65         outs[j] = (L[i]-conj(L[j])) * R[i]/(2.0*n)/1i;
66     }
67     FFT(outl);
68     FFT(outs);
69     for (int i=0 ; i<res.size() ; i++){
70         int av = (int)(real(outl[i])+0.5), cv = (int)(imag(
71             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,
    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 };
34
35 // 判斷向量正負：1=正數, 0=0, -1=負數
36 int sign(double a){
37     if (abs(a)<EPS) return 0;
38     else return (a>0 ? 1 : -1);
39 }
40
41 // 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 }
40
41 // 逆時針極角排序
42 bool operator<(point &b) {return (x*b.y > b.x*y); }
43
44 T operator*(point b) {return x * b.x + y * b.y; }
45 T operator^(point b) {return x * b.y - y * b.x; }
46 T abs2() {return (*this) * (*this); }
47
48 // 旋轉 Arg(b) 的角度 (小心溢位)
49 point rotate(point b) {return {x*b.x - y*b.y, x*b.y + y*b
50     .x}; }
51
52 // 判斷數值正負：{1:正數,0:零,-1:負數}
53 int sign(int x) {return (x >= 0) ? ((bool)x) : -1; }
54 int sign(double x) {
55     return (abs(x) < 1e-9) ? 0 : (x > 0 ? 1 : -1);
56 }
57
58 // 判斷 ab 到 ac 的方向：{1:逆時鐘,0:重疊,-1:順時鐘}
59 int ori(point<auto> a, point<auto> b, point<auto> c) {
60     return sign((b-a)^(c-a));
61 }
62
63 ostream& operator<<(ostream& os, point<auto> p) {
64     os << "(" << p.x << ", " << p.y << ")";
65     return os;
66 }
67
68 template<typename T>
69 struct line {
70     point<T> p1, p2;
71     // ax + by + c = 0
72     T a, b, c; // |a|, |b| ≤ 2C, |c| ≤ 8C²
73     line(const point<T> &x, const point<T> &y) : p1(x), p2(y)
74     {
75         build();
76     }
77     void build() {
78         a = p1.y - p2.y;
79         b = p2.x - p1.x;
80         c = (-a*p1.x)-b*p1.y;
81     }
82     // 判斷點和有向直線的關係：{1:左邊,0:在線上,-1:右邊}
83     int ori(point<T> &p) {
84         return sign((p2-p1) ^ (p-p1));
85     }
86 }

```

5.5 Geometry-Struct

```

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

```

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) {

```

```

3  #define in(a, b, c) ( ori(a, b, c) \
4      ? 0 : (sign((a-c)*(b-c)) <= 0) )
5  int s1 = ori(a, b, c);
6  int s2 = ori(a, b, d);
7  int s3 = ori(c, d, a);
8  int s4 = ori(c, d, b);
9  if (in(a, b, c) || in(a, b, d) || in(c, d, a) || in(c, d,
10     b)) return 1;
11  return (s1 * s2 < 0) && (s3 * s4 < 0);
12  #undef in
}

```

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{//向量長度的平方
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     T btw(const point<T> &p)const{//點投影落在線段上<=0
48         return (p1-p).dot(p2-p);
49     }
50     bool point_on_segment(const point<T>&p)const{//點是否在線段
51         //上
52         return ori(p)==0&&btw(p)<=0;
53     }
54     T dis2(const point<T> &p,bool is_segment=0)const{//點跟直線
55         //線段的距離平方
56         point<T> v=p2-p1,v1=p-p1;
57         if(is_segment){
58             point<T> v2=p-p2;
59             if(v.dot(v1)<=0)return v1.abs2();
60             if(v.dot(v2)>=0)return v2.abs2();
61         }
62         T tmp=v.cross(v1);
63         return tmp*tmp/v.abs2();
64     }
65     T seg_dis2(const line<T> &l)const{//兩線段距離平方
66         return min({dis2(l.p1,1),dis2(l.p2,1),l.dis2(p1,1),l.dis2
67             (p2,1)});
68     }
69     point<T> projection(const point<T> &p)const{//點對直線的投
70         //影
71         point<T> n=(p2-p1).normal();
72         return p-n*(p-p1).dot(n)/n.abs2();
73     }
74     point<T> mirror(const point<T> &p)const{
75         //點對直線的鏡射，要先呼叫pton轉成一般式
76         point<T> R;
77         T d=a*a+b*b;
78         R.x=(b*b*p.x-a*a*p.x-2*a*b*p.y-2*a*c)/d;
79         R.y=(a*a*p.y-b*b*p.y-2*a*b*p.x-2*b*c)/d;
80         return R;
81     }
82     bool equal(const line &l)const{//直線相等
83         return ori(l.p1)==0&&ori(l.p2)==0;
84     }
85     bool parallel(const line &l)const{
86         return (p1-p2).cross(l.p1-l.p2)==0;
87     }
88     bool cross_seg(const line &l)const{
89         return (p2-p1).cross(l.p1-p1)*(p2-p1).cross(l.p2-p1)<=0;
90         //直線是否交線段
91     }
92     int line_intersect(const line &l)const{//直線相交情況，-1無
93         //限多點，1交於一點，0不相交
94         return parallel(l)?(ori(l.p1)==0?-1:0):1;
95     }
96     int seg_intersect(const line &l)const{
97         T c1=ori(l.p1), c2=ori(l.p2);
98         T c3=l.ori(p1), c4=l.ori(p2);
99         if(c1==0&&c2==0){//共線
100             bool b1=btw(l.p1)>=0,b2=btw(l.p2)>=0;
101             T a3=l.btw(p1),a4=l.btw(p2);
102             if(b1&&b2&&a3==0&&a4>=0) return 2;
103             if(b1&&b2&&a3>=0&&a4==0) return 3;
104             if(b1&&b2&&a3>=0&&a4>=0) return 0;
105             return -1;//無限交點
106         }else if(c1*c2<=0&&c3*c4<=0)return 1;
107         return 0;//不相交
108     }
109     point<T> line_intersection(const line &l)const{//直線交點*/

```

```

104     point<T> a=p2-p1,b=l.p2-l.p1,s=l.p1-p1;
105     //if(a.cross(b)==0)return INF;
106     return p1+a*(s.cross(b)/a.cross(b));
107 }
108 point<T> seg_intersection(const line &l)const{//線段交點
109     int res=seg_intersect(l);
110     if(res<=0) assert(0);
111     if(res==2) return p1;
112     if(res==3) return p2;
113     return line_intersection(l);
114 }
115 };
116 template<typename T>
117 struct polygon{
118     polygon(){}
119     vector<point<T> > p;//逆時針順序
120     T area()const{//面積
121         T ans=0;
122         for(int i=p.size()-1,j=0;j<(int)p.size();i=j++){
123             ans+=p[i].cross(p[j]);
124         }
125         return ans/2;
126     }
127     point<T> center_of_mass()const{//重心
128         T cx=0,cy=0,w=0;
129         for(int i=p.size()-1,j=0;j<(int)p.size();i=j++){
130             T a=p[i].cross(p[j]);
131             cx+=(p[i].x+p[j].x)*a;
132             cy+=(p[i].y+p[j].y)*a;
133             w+=a;
134         }
135         return point<T>(cx/3/w,cy/3/w);
136     }
137     char ahas(const point<T> &t)const{//點是否在簡單多邊形內，
138         //是的話回傳1、在邊上回傳-1、否則回傳0
139         bool c=0;
140         for(int i=0,j=p.size()-1;i<p.size();j=i++){
141             if(line<T>(p[i],p[j]).point_on_segment(t))return -1;
142             else if((p[i].y>t.y)!(p[j].y>t.y)&&
143                 t.x<(p[j].x-p[i].x)*(t.y-p[i].y)/(p[j].y-p[i].y)+p[i].x
144                 )
145                 c=!c;
146             return c;
147         }
148     }
149     char point_in_convex(const point<T>&x)const{
150         int l=1,r=(int)p.size()-2;
151         while(l<r){//點是否在凸多邊形內，是的話回傳1、在邊上回傳
152             //1、否則回傳0
153             int mid=(l+r)/2;
154             T a1=(p[mid]-p[0]).cross(x-p[0]);
155             T a2=(p[mid+1]-p[0]).cross(x-p[0]);
156             if(a1>0&&a2<=0){
157                 T res=(p[mid+1]-p[mid]).cross(x-p[mid]);
158                 return res>0?1:(res>0?-1:0);
159             }else if(a1<0)r=mid-1;
160             else l=mid+1;
161         }
162         return 0;
163     }
164     vector<T> getA()const{//凸包邊對x軸的夾角
165         vector<T>res;//一定是遞增的
166         for(size_t i=0;i<p.size();++i)
167             res.push_back((p[(i+1)%p.size()]-p[i]).getA());
168         return res;
169     }

```

```

165 bool line_intersect(const vector<T>&A, const line<T> &l) 219
166     const{//O(LogN)
167     int f1=upper_bound(A.begin(),A.end(),(l.p1-l.p2).getA())- 220
168         A.begin();
169     int f2=upper_bound(A.begin(),A.end(),(l.p2-l.p1).getA())- 221
170         A.begin();
171     return l.cross_seg(line<T>(p[f1],p[f2])); 222
172 }
173 polygon cut(const line<T> &l) const{//凸包對直線切割，得到直 223
174     線L左側的凸包
175     polygon ans; 224
176     for(int n=p.size(),i=n-1,j=0;j<n;i=j++){ 225
177         if(l.ori(p[i])>=0){ 226
178             ans.p.push_back(p[i]); 227
179             if(l.ori(p[j])<0) 228
180                 ans.p.push_back(l.line_intersection(line<T>(p[i],p[j]),l)); 229
181         }else if(l.ori(p[j])>0) 230
182             ans.p.push_back(l.line_intersection(line<T>(p[i],p[j]),l)); 231
183         } 232
184     } 233
185     return ans; 234
186 } 235
187 static bool monotone_chain_cmp(const point<T>& a, const 236
188     point<T>& b){{//凸包排序函數
189     return (a.x<b.x)|| (a.x==b.x&&a.y<b.y); 237
190 } 238
191 void monotone_chain(vector<point<T> > &s){{//凸包 239
192     sort(s.begin(),s.end(),monotone_chain_cmp); 240
193     p.resize(s.size()+1); 241
194     int m=0; 242
195     for(size_t i=0;i<s.size();i++){ 243
196         while(m>=2&&(p[m-1]-p[m-2]).cross(s[i]-p[m-2])<=0)--m; 244
197         p[m++]=s[i]; 245
198     } 246
199     for(int i=s.size()-2,t=m+1;i>=0;--i){ 247
200         while(m>=t&&(p[m-1]-p[m-2]).cross(s[i]-p[m-2])<=0)--m; 248
201         p[m++]=s[i]; 249
202     } 250
203     if(s.size()>1)--m; 251
204     p.resize(m); 252
205 } 253
206 T diam(){//直徑 254
207     int n=p.size(),t=1; 255
208     T ans=0;p.push_back(p[0]); 256
209     for(int i=0;i<n;i++){ 257
210         point<T> now=p[i+1]-p[i]; 258
211         while(now.cross(p[t+1]-p[i])>now.cross(p[t]-p[i]))t=(t 259
212             +1)%n; 260
213         ans=max(ans,(p[i]-p[t]).abs2()); 261
214     } 262
215     return p.pop_back(),ans; 263
216 } 264
217 T min_cover_rectangle(){//最小覆蓋矩形 265
218     int n=p.size(),t=1,r=1,l; 266
219     if(n<3)return 0;//也可以做最小周長矩形
220     T ans=1e99;p.push_back(p[0]);
221     for(int i=0;i<n;i++){
222         point<T> now=p[i+1]-p[i];
223         while(now.cross(p[t+1]-p[i])>now.cross(p[t]-p[i]))t=(t
224             +1)%n;
225         while(now.dot(p[r+1]-p[i])>now.dot(p[r]-p[i]))r=(r+1)%n
226         ;
227         if(!i)l=r;
228     }
229     while(now.dot(p[l+1]-p[i])<=now.dot(p[l]-p[i]))l=(l+1)%
230         n;
231     T d=now.abs2();
232     T tmp=now.cross(p[t]-p[i])*(now.dot(p[r]-p[i])-now.dot(
233         p[l]-p[i]))/d;
234     ans=min(ans,tmp);
235     }
236     return p.pop_back(),ans;
237 }
238 T dis2(polygon &p1){{//凸包最近距離平方
239     vector<point<T> > &P=p,&Q=p1.p;
240     int n=P.size(),m=Q.size(),l=0,r=0;
241     for(int i=0;i<n;++i)if(P[i].y<P[l].y)l=i;
242     for(int i=0;i<m;++i)if(Q[i].y<Q[r].y)r=i;
243     P.push_back(P[0]),Q.push_back(Q[0]);
244     T ans=1e99;
245     for(int i=0;i<n;++i){
246         while((P[l]-P[l+1]).cross(Q[r+1]-Q[r])<0)r=(r+1)%m;
247         ans=min(ans,line<T>(P[l],P[l+1]).seg_dis2(line<T>(Q[r],
248             Q[r+1])));
249         l=(l+1)%n;
250     }
251     return P.pop_back(),Q.pop_back(),ans;
252 }
253 static char sign(const point<T>&t){
254     return (t.y==0?t.x:t.y)<0;
255 }
256 static bool angle_cmp(const line<T>& A, const line<T>& B){
257     point<T> a=A.p2-A.p1,b=B.p2-B.p1;
258     return sign(a)<sign(b)|| (sign(a)==sign(b)&&a.cross(b)>0);
259 }
260 int halfplane_intersection(vector<line<T> > &s){{//半平面交
261     sort(s.begin(),s.end(),angle_cmp);//線段左側為該線段半平
262     面
263     int L,R,n=s.size();
264     vector<point<T> > px(n);
265     vector<line<T> > q(n);
266     q[L=R=0]=s[0];
267     for(int i=1;i<n;++i){
268         while(L<R&&s[i].ori(px[R-1])<=0)--R;
269         while(L<R&&s[i].ori(px[L])<=0)++L;
270         q[++R]=s[i];
271         if(q[R].parallel(q[R-1])){
272             --R;
273             if(q[R].ori(s[i].p1)>0)q[R]=s[i];
274         }
275         if(L<R)px[R-1]=q[R-1].line_intersection(q[R]);
276     }
277     while(L<R&&q[L].ori(px[R-1])<=0)--R;
278     p.clear();
279     if(R-L<=1)return 0;
280     px[R]=q[R].line_intersection(q[L]);
281     for(int i=L;i<R;++i)p.push_back(px[i]);
282     return R-L+1;
283 }
284 }
285 template<typename T>
286 struct triangle{
287     point<T> a,b,c;
288     triangle(){
289         triangle(const point<T> &a, const point<T> &b, const point<T>
290             &c):a(a),b(b),c(c){
291     T area()const{
292         T t=(b-a).cross(c-a)/2;
293         return t>0?t:-t;
294     }
295 }
296 }
297 point<T> barycenter()const{//重心
298     return (a+b+c)/3;
299 }
300 point<T> circumcenter()const{//外心
301     static line<T> u,v;
302     u.p1=(a+b)/2;
303     u.p2=point<T>(u.p1.x-a.y+b.y,u.p1.y+a.x-b.x);
304     v.p1=(a+c)/2;
305     v.p2=point<T>(v.p1.x-a.y+c.y,v.p1.y+a.x-c.x);
306     return u.line_intersection(v);
307 }
308 point<T> incenter()const{//內心
309     T A=sqrt((b-c).abs2()),B=sqrt((a-c).abs2()),C=sqrt((a-b).
310         abs2());
311     return point<T>(A*a.x+B*b.x+C*c.x,A*a.y+B*b.y+C*c.y)/(A+B
312         +C);
313 }
314 point<T> perpercenter()const{//垂心
315     return barycenter()*3-circumcenter()*2;
316 }
317 };
318 template<typename T>
319 struct point3D{
320     T x,y,z;
321     point3D(){
322     point3D(const T&x, const T&y, const T&z):x(x),y(y),z(z){}
323     point3D operator+(const point3D &b)const{
324         return point3D(x+b.x,y+b.y,z+b.z);
325     }
326     point3D operator-(const point3D &b)const{
327         return point3D(x-b.x,y-b.y,z-b.z);
328     }
329     point3D operator*(const T &b)const{
330         return point3D(x*b,y*b,z*b);
331     }
332     point3D operator/(const T &b)const{
333         return point3D(x/b,y/b,z/b);
334     }
335     bool operator==(const point3D &b)const{
336         return x==b.x&&y==b.y&&z==b.z;
337     }
338     T dot(const point3D &b)const{
339         return x*b.x+y*b.y+z*b.z;
340     }
341     point3D cross(const point3D &b)const{
342         return point3D(y*b.z-z*b.y,z*b.x-x*b.z,x*b.y-y*b.x);
343     }
344     T abs2()const{//向量長度的平方
345         return dot(*this);
346     }
347     T area2(const point3D &b)const{//和b、原點圍成面積的平方
348         return cross(b).abs2()/4;
349     }
350 };
351 template<typename T>
352 struct line3D{
353     point3D<T> p1,p2;
354     line3D(){
355     line3D(const point3D<T> &p1, const point3D<T> &p2):p1(p1),p2
356         (p2){
357     T dis2(const point3D<T> &p, bool is_segment=0)const{//點跟直
358         線/線段的距離平方
359         point3D<T> v=p2-p1,v1=p-p1;
360         if(is_segment){
361             point3D<T> v2=p-p2;
362             if(v.dot(v1)<=0)return v1.abs2();
363             if(v.dot(v2)>=0)return v2.abs2();
364         }
365         point3D<T> tmp=v.cross(v1);
366         return tmp.abs2()/v.abs2();
367     }
368 }

```

```

338 pair<point3D<T>,point3D<T> > closest_pair(const line3D<T> & 393
      1)const{ 394
339     point3D<T> v1=(p1-p2),v2=(l.p1-l.p2); 395
340     point3D<T> N=v1.cross(v2),ab(p1-l.p1); 396
341     //if(N.abs2()==0)return NULL; 平行或重合 397
342     T tmp=N.dot(ab),ans=tmp*tmp/N.abs2(); 最近點對距離 398
343     point3D<T> d1=p2-p1,d2=l.p2-l.p1,D=d1.cross(d2),G=l.p1-p1 399
      ; 400
344     T t1=(G.cross(d2)).dot(D)/D.abs2(); 401
345     T t2=(G.cross(d1)).dot(D)/D.abs2(); 402
346     return make_pair(p1+d1*t1,l.p1+d2*t2); 403
347 } 404
348 bool same_side(const point3D<T> &a,const point3D<T> &b) 405
      const{ 406
349     return (p2-p1).cross(a-p1).dot((p2-p1).cross(b-p1))>0; 407
350 } 408
351 }; 409
352 template<typename T> 410
353 struct plane{ 411
354     point3D<T> p0,n; 平面上的點和法向量 412
355     plane(){} 413
356     plane(const point3D<T> &p0,const point3D<T> &n):p0(p0),n(n) 414
      {} 415
357     T dis2(const point3D<T> &p)const{ 點到平面距離的平方 416
358     T tmp=(p-p0).dot(n); 417
359     return tmp*tmp/n.abs2(); 418
360 } 419
361 point3D<T> projection(const point3D<T> &p)const{ 420
362     return p-n*(p-p0).dot(n)/n.abs2(); 421
363 } 422
364 point3D<T> line_intersection(const line3D<T> &l1)const{ 423
365     T tmp=n.dot(l.p2-l.p1); 等於0表示平行或重合該平面 424
366     return l.p1+(l.p2-l.p1)*(n.dot(p0-l.p1)/tmp); 425
367 } 426
368 line3D<T> plane_intersection(const plane &p1)const{ 427
369     point3D<T> e=n.cross(p1.n),v=n.cross(e); 428
370     T tmp=p1.n.dot(v); 等於0表示平行或重合該平面 429
371     point3D<T> q=p0+(v*(p1.n.dot(p1.p0-p0))/tmp); 430
372     return line3D<T>(q,q+e); 431
373 } 432
374 }; 433
375 template<typename T> 434
376 struct triangle3D{ 435
377     point3D<T> a,b,c; 436
378     triangle3D(){} 437
379     triangle3D(const point3D<T> &a,const point3D<T> &b,const 438
      point3D<T> &c):a(a),b(b),c(c){} 439
380     bool point_in(const point3D<T> &p)const{ 點在該平面上的投 440
      影在三角形中 441
381     return line3D<T>(b,c).same_side(p,a)&&line3D<T>(a,c). 442
      same_side(p,b)&&line3D<T>(a,b).same_side(p,c); 443
382 } 444
383 }; 445
384 template<typename T> 446
385 struct tetrahedron{ 四面體 447
386     point3D<T> a,b,c,d; 448
387     tetrahedron(){} 449
388     tetrahedron(const point3D<T> &a,const point3D<T> &b,const 450
      point3D<T> &c,const point3D<T> &d):a(a),b(b),c(c),d(d) 451
      {} 452
389     T volume6()const{ 體積的六倍 453
390     return (d-a).dot((b-a).cross(c-a)); 454
391 } 455
392 point3D<T> centroid()const{ 456

```

```

      return (a+b+c+d)/4; 457
    } 458
    bool point_in(const point3D<T> &p)const{ 459
      return triangle3D<T>(a,b,c).point_in(p)&&triangle3D<T>(c, 460
      d,a).point_in(p); 461
    } 462
  }; 463
  template<typename T> 464
  struct convexhull3D{ 465
    static const int MAXN=1005; 466
    struct face{ 467
      int a,b,c; 468
      face(int a,int b,int c):a(a),b(b),c(c){} 469
    }; 470
    vector<point3D<T>> pt; 471
    vector<face> ans; 472
    int fid[MAXN][MAXN]; 473
    void build(){ 474
      int n=pt.size(); 475
      ans.clear(); 476
      memset(fid,0,sizeof(fid)); 477
      ans.emplace_back(0,1,2); 注意不能共線 478
      ans.emplace_back(2,1,0); 479
      int ftop = 0; 480
      for(int i=3, ftop=1; i<n; ++i,++ftop){ 481
        vector<face> next; 482
        for(auto &f:ans){ 483
          T d=(pt[i]-pt[f.a]).dot((pt[f.b]-pt[f.a]).cross(pt[f. 484
          c]-pt[f.a])); 485
          if(d<=0) next.push_back(f); 486
          int ff=0; 487
          if(d>0) ff=ftop; 488
          else if(d<0) ff=-ftop; 489
          fid[f.a][f.b]=fid[f.b][f.c]=fid[f.c][f.a]=ff; 490
        } 491
        for(auto &f:ans){ 492
          if(fid[f.a][f.b]>0 && fid[f.a][f.b]!=fid[f.b][f.a]) 493
            next.emplace_back(f.a,f.b,i); 494
          if(fid[f.b][f.c]>0 && fid[f.b][f.c]!=fid[f.c][f.b]) 495
            next.emplace_back(f.b,f.c,i); 496
          if(fid[f.c][f.a]>0 && fid[f.c][f.a]!=fid[f.a][f.c]) 497
            next.emplace_back(f.c,f.a,i); 498
        } 499
        ans=next; 500
      } 501
    } 502
    point3D<T> centroid()const{ 503
      point3D<T> res(0,0,0); 504
      T vol=0; 505
      for(auto &f:ans){ 506
        T tmp=pt[f.a].dot(pt[f.b].cross(pt[f.c])); 507
        res=res+(pt[f.a]+pt[f.b]+pt[f.c])*tmp; 508
        vol+=tmp; 509
      } 510
      return res/(vol*4); 511
    } 512
  }; 513

```

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, 0ll); // query : l = id[topf[a]], r = id[a]
36            a = pa[topf[a]];
37        }
38        /// 此時已在同一條鍊上
39        if (depth[a] < depth[b]) swap(a, b);
40        res = max(res, 0ll); // query : l = id[b], r = id[a]
41        return res;
42    }
43 };

```

6.4 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     w[now]=max(w[now], n-s[now]);
36     if (w[now]<=n/2){
37         if (rec.first==0) rec.first=now;
38         else rec.second=now;
39     }
40 }
41
42 int dfs(Graph &g, Hash &m, int &id, int now, int pre){
43     vector<int> v;
44     for (auto x : g[now]){
45         if (x!=pre){
46             int add=dfs(g, m, id, x, now);
47             v.push_back(add);
48         }
49     }
50     sort(v.begin(), v.end());
51
52     if (m.find(v)!=m.end()){
53         return m[v];
54     }else{
55         m[v]=++id;
56         return id;
57     }
58 }
59
60 void solve1(){
61
62     // init
63     id1=0;
64     id2=0;
65     c1={0, 0};
66     c2={0, 0};
67     fill(sz1.begin(), sz1.begin()+n+1, 0);
68     fill(sz2.begin(), sz2.begin()+n+1, 0);
69     fill(we1.begin(), we1.begin()+n+1, 0);
70     fill(we2.begin(), we2.begin()+n+1, 0);
71     for (int i=1 ; i<=n ; i++){
72         g1[i].clear();
73         g2[i].clear();

```

```

74     }
75
76     }
77     m1.clear();
78     m2.clear();
79
80     // input
81     cin >> n;
82     for (int i=0 ; i<=n-1 ; i++){
83         cin >> a >> b;
84         g1[a].push_back(b);
85         g1[b].push_back(a);
86     }
87     for (int i=0 ; i<=n-1 ; i++){
88         cin >> a >> b;
89         g2[a].push_back(b);
90         g2[b].push_back(a);
91     }
92
93     // get tree centroid
94     centroid(g1, sz1, we1, c1, 1, 0);
95     centroid(g2, sz2, we2, c2, 1, 0);
96
97     // process
98     int res1=0, res2=0, res3=0;
99     if (c2.second!=0){
100         res1=dfs(g1, m1, id1, c1.first, 0);
101         m2=m1;
102         id2=id1;
103         res2=dfs(g2, m1, id1, c2.first, 0);
104         res3=dfs(g2, m2, id2, c2.second, 0);
105     }else if (c1.second!=0){
106         res1=dfs(g2, m1, id1, c2.first, 0);
107         m2=m1;
108         id2=id1;
109         res2=dfs(g1, m1, id1, c1.first, 0);
110         res3=dfs(g1, m2, id2, c1.second, 0);
111     }else{
112         res1=dfs(g1, m1, id1, c1.first, 0);
113         res2=dfs(g2, m1, id1, c2.first, 0);
114     }
115
116     // output
117     cout << (res1==res2 || res1==res3 ? "YES" : "NO") << endl;
118
119     return;
120 }
121
122 signed main(void){
123     fastio;
124
125     int t=1;
126     cin >> t;
127     while (t--){
128         solve1();
129     }
130     return 0;
131 }

```

6.5 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下走到 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             }
29         } else {
30             /// (v, u) 是回邊
31             low[v] = min(low[v], depth[u]);
32         }
33     }
34     bcc.back().push_back(stk.top());
35     stk.pop();
36 }

```

```

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 <<
15     qwe << ' '; cerr << endl;
16 #define deb1 cerr << "deb1" << endl;
17 #define deb2 cerr << "deb2" << endl;
18 #define deb3 cerr << "deb3" << endl;
19 #define deb4 cerr << "deb4" << endl;
20 #define deb5 cerr << "deb5" << endl;
21 #define bye exit(0);
22 using namespace std;
23
24 const int mxn = (int)(2e5) + 10;
25 const int mxlg = 17;
26 int last_special_node = (int)(1e5) + 1;
27 vector<int> E[mxn], F[mxn];
28
29 struct edg{
30     int fr, to;
31     edg(int _fr, int _to){
32         fr = _fr;
33         to = _to;
34     }
35 };
36 ostream& operator<<(ostream& os, edg x){os << x.fr << "--" <<
37     x.to;}
38 vector<edg> EV;
39
40 void tarjan(int v, int par, stack<int>& S){
41     static vector<int> dfn(mxn), low(mxn);
42     static vector<bool> to_add(mxn);
43     static int nowT = 0;
44
45     int childs = 0;
46     nowT += 1;
47     dfn[v] = low[v] = nowT;
48     for(auto &ne:E[v]){
49         int i = EV[ne].to;
50         if(i == par) continue;
51         if(!dfn[i]){
52             S.push(ne);
53             tarjan(i, v, S);
54             childs += 1;
55             low[v] = min(low[v], low[i]);
56         } else {
57             /// (v, u) 是回邊
58             low[v] = min(low[v], depth[i]);
59         }
60     }
61     if(low[v] == depth[v]){
62         bcc.emplace_back();
63         while (stk.top() != v) {
64             bcc.back().push_back(stk.top());
65             stk.pop();
66         }
67         bcc.back().push_back(stk.top());
68         stk.pop();
69     }
70 }

```

```

48     if(i == par) continue;
49     if(!dfn[i]){
50         S.push(ne);
51         tarjan(i, v, S);
52         childs += 1;
53         low[v] = min(low[v], low[i]);
54     }
55     if(par >= 0 && low[i] >= dfn[v]){
56         vector<int> bcc;
57         int tmp;
58         do{
59             tmp = S.top(); S.pop();
60             if(!to_add[EV[tmp].fr]){
61                 to_add[EV[tmp].fr] = true;
62                 bcc.pb(EV[tmp].fr);
63             }
64             if(!to_add[EV[tmp].to]){
65                 to_add[EV[tmp].to] = true;
66                 bcc.pb(EV[tmp].to);
67             }
68         }while(tmp != ne);
69         for(auto &j:bcc){
70             to_add[j] = false;
71             F[last_special_node].pb(j);
72             F[j].pb(last_special_node);
73         }
74         last_special_node += 1;
75     }
76 }
77 else{
78     low[v] = min(low[v], dfn[i]);
79     if(dfn[i] < dfn[v]){ /// edge i--v will be visited
80         twice at here, but we only need one.
81         S.push(ne);
82     }
83 }
84 }
85 }
86
87 int dep[mxn], jmp[mxn][mxlg];
88 void dfs_lca(int v, int par, int depth){
89     dep[v] = depth;
90     for(auto &i:F[v]){
91         if(i == par) continue;
92         jmp[i][0] = v;
93         dfs_lca(i, v, depth + 1);
94     }
95 }
96
97 inline void build_lca(){
98     jmp[1][0] = 1;
99     dfs_lca(1, -1, 1);
100     lp(j,1,mxlg){
101         lp(i,1,mxn){
102             jmp[i][j] = jmp[jmp[i][j-1]][j-1];
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     while(x != y){
116         lp(j,0,mxlg){
117             if((diff >> j) & 1){
118                 x = jmp[x][j];
119             }
120         }
121     }
122 }

```

```

113     }
114 }
115 if(x == y) return x;
116
117 for(int j = mxlg - 1; j >= 0; j--){
118     if(jmp[x][j] != jmp[y][j]){
119         x = jmp[x][j];
120         y = jmp[y][j];
121     }
122 }
123 return jmp[x][0];
124 }
125
126 inline bool can_reach(int fr, int to){
127     if(dep[to] > dep[fr]) return false;
128
129     int diff = dep[fr] - dep[to];
130     lp(j, 0, mxlg){
131         if((diff >> j) & 1){
132             fr = jmp[fr][j];
133         }
134     }
135     return fr == to;
136 }
137
138 int main(){
139     ios::sync_with_stdio(false); cin.tie(0);
140     // freopen("test_input.txt", "r", stdin);
141     int n, m, q; cin >> n >> m >> q;
142     lp(i, 0, m){
143         int u, v; cin >> u >> v;
144         E[u].pb(EV.size());
145         EV.pb(edg(u, v));
146         E[v].pb(EV.size());
147         EV.pb(edg(v, u));
148     }
149     E[0].pb(EV.size());
150     EV.pb(edg(0, 1));
151     stack<int> S;
152     tarjan(0, -1, S);
153     build_lca();
154
155     lp(queries, 0, q){
156         int fr, to, relay; cin >> fr >> to >> relay;
157         if(fr == relay || to == relay){
158             cout << "NO\n";
159             continue;
160         }
161         if((can_reach(fr, relay) || can_reach(to, relay)) &&
162             dep[relay] >= dep[lca(fr, to)]){
163             cout << "NO\n";
164             continue;
165         }
166         cout << "YES\n";
167     }
168 }

```

6.8 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 // c8b146
14 struct SCC_compress{
15     int n = 0, m = 0;
16     vector<vector<int>> G, inv_G, result;
17     vector<pair<int, int>> edges;
18     vector<bool> vis;
19     vector<int> order;
20
21     vector<vector<int>> SCC;
22     vector<int> SCC_id;
23
24     SCC_compress(int _n){
25         n = _n;
26         G.resize(n);
27         inv_G.resize(n);
28         result.resize(n);
29         vis.resize(n);
30         SCC_id.resize(n);
31     }
32
33     void add_edge(int u, int v){
34         G[u].push_back(v);
35         inv_G[v].push_back(u);
36         edges.push_back({u, v});
37         m++;
38     }
39
40     void dfs1(vector<vector<int>> &G, int now){
41         vis[now] = 1;
42         for (auto x : G[now]){
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         fill(vis.begin(), vis.end(), 0);
72         reverse(order.begin(), order.end());
73         for (int i=0 ; i<n ; i++){
74             if (vis[order[i]]==0){
75                 SCC.push_back(vector<int>());
76                 dfs2(inv_G, order[i]);
77             }
78         }
79
80         for (int i=0 ; i<m ; i++){
81             if (SCC_id[edges[i].first]!=SCC_id[edges[i].second]){
82                 result[SCC_id[edges[i].first]].push_back(
83                     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
89                                     [i].end())-result[i].begin());
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 vector<vector<pair<int, int>>> G(n); // G[i] = <節點, 權重>
5 vector<int> dis(n, INF);
6 priority_queue<pair<int, int>, vector<pair<int, int>>,
7     greater<pair<int, int>>> pq;

```

```

7 dis[s] = 0;
8 pq.push({0, s});
9
10 while (pq.size()){
11     int now_dis = pq.top().first;
12     int now_node = pq.top().second;
13     pq.pop();
14
15     if (now_dis>dis[now_node]) continue;
16
17     for (auto x : G[now_node]){
18         if (now_dis+x.second<dis[x.first]){
19             dis[x.first] = now_dis+x.second;
20             pq.push({dis[x.first], x.first});
21         }
22     }
23 }

```

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         for (int v=t; v!=s; v=par[v]){
57             G[par[v]][par_eid[v]].rc -= cur;
58             G[v][G[par[v]][par_eid[v]].rv].rc += cur;
59         }
60     }
61     return {fl, cost};
62 }
63
64 vector<pair<int, int>> construct(){
65     vector<pair<int, int>> ret;
66     for (int i=0; i<n; i++){
67         for (auto x : G[i]){
68             if (x.rc==0){
69                 ret.push_back({i+1, x.u-n+1});
70                 break;
71             }
72         }
73     }
74     return ret;
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            }
53            if (dis[t]==INF) break;
54
55            fill(it.begin(), it.end(), 0);
56            while (true){
57                double df = dfs(s, t, double_INF);
58                if (abs(df) <= eps) break;
59                ans += df;
60            }
61        }
62        return ans;
63    }

```

```

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 }

```

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++;
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) * 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     return ret;
28 }
29
30 Matrix pow(int p){
31     Matrix ret(n, n), mul = *this;
32     for (int i=0 ; i<n ; i++){
33         ret.arr[i][i] = 1;
34     }
35
36     for ( ; p ; p>=1){
37         if (p&1) ret = ret*mul;
38         mul = mul*mul;
39     }
40     return ret;
41 }
42
43 int det(){
44     vector<vector<int>> arr = this->arr;
45     bool flag = false;
46     for (int i=0 ; i<n ; i++){
47         int target = -1;
48         for (int j=i ; j<n ; j++){
49             if (arr[j][i]){
50                 target = j;
51                 break;
52             }
53         }
54         if (target==-1) return 0;
55         if (i!=target){
56             swap(arr[i], arr[target]);
57             flag = !flag;
58         }
59         for (int j=i+1 ; j<n ; j++){
60             if (!arr[j][i]) continue;
61             int freq = arr[j][i]*qp(arr[i][i], MOD-2)%MOD;
62             for (int k=i ; k<n ; k++){
63                 arr[j][k] -= freq*arr[i][k];
64                 arr[j][k] = (arr[j][k]%MOD+MOD)%MOD;
65             }
66         }
67     }
68     int ret = !flag ? 1 : MOD-1;
69     for (int i=0 ; i<n ; i++){
70         ret *= arr[i][i];
71         ret %= MOD;
72     }
73     return ret;
74 }
75
76 };
77
78 int ret = !flag ? 1 : MOD-1;
79 for (int i=0 ; i<n ; i++){
80     ret *= arr[i][i];
81     ret %= MOD;
82 }
83
84 };
85
86 int ret = !flag ? 1 : MOD-1;
87 for (int i=0 ; i<n ; i++){
88     ret *= arr[i][i];
89     ret %= MOD;
90 }
91
92 };
93
94 int ret = !flag ? 1 : MOD-1;
95 for (int i=0 ; i<n ; i++){
96     ret *= arr[i][i];
97     ret %= MOD;
98 }
99
100 };

```

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;
26     return {1, {a1, m1}};
27 }

```

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 bool operator<=(frac const &B){
68     return a*B.b <= B.a*b;}
69 bool operator>(frac const &B){
70     return a*B.b > B.a*b;}
71 bool operator>=(frac const &B){
72     return a*B.b >= B.a*b;}
73 bool operator==(frac const &B){
74     return a * B.b == B.a * b;}
75 bool operator!=(frac const &B){
76     return a * B.b != B.a * b;}
77
78 };
79 ostream& operator<<(ostream &os, const frac& A){
80     os << A.a << "/" << A.b;
81     return os;
82 }
83 /// Fraction template ends ///
84
85 void test(frac A, frac B){
86     cout << "A = " << A << endl;
87     cout << "B = " << B << endl;
88     cout << endl;
89     cout << "A + B = " << A + B << endl;
90     cout << "A - B = " << A - B << endl;
91     cout << "A * B = " << A * B << endl;
92     cout << "A / B = " << A / B << endl;
93     cout << endl;
94     cout << "(A < B) = " << (A < B) << endl;
95     cout << "(A <= B) = " << (A <= B) << 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 << "-----\n";
101    return;
102 }

```

```

92 }
93
94 int main(){
95     frac tmp1(-7, 2);
96     frac tmp2(5, 3);
97     test(tmp1, tmp2);
98
99     frac tmp3(-7);
100    frac tmp4(0);
101    test(tmp3, tmp4);
102    return 0;
103 }

```

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
19             t = ((__int128)t*t+c)%n;
20             val = ((__int128)val*abs(t-s)%n;
21
22             if ((step % 127) == 0){
23                 int d = __gcd(val, n);
24                 if (d>1) return d;
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     };

```

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             fac = fac * i % mod;
38         }
39         inv_fac[MAX_N - 1] = fp(fac, mod - 2);
40         for (int i = MAX_N - 1; i >= 1; --i) {
41             inv_fac[i - 1] = inv_fac[i] * i % mod;
42         }
43     }
44     // You call init() many times without having a second
45     // instance of this struct.
46     void init(int X_0, vector<int> &u) {
47         v = u;
48         shift = ((1 - X_0) % mod + mod) % mod;
49         if (v.size() == 1) v.push_back(v[0]);
50         m = v.size();
51         mul.resize(m);
52     }
53     // You can use sample(x) instead of sample(x % mod).
54     int sample(int x) {
55         x = ((long long)x + shift) % mod;
56         x = (x < 0) ? (x + mod) : x;
57         long long now = 1;
58         for (int i = m; i >= 1; --i) {
59             mul[i - 1] = now;
60             now = now * (x - i) % mod;
61         }
62         int ret = 0;
63         bool neg = (m - 1) & 1;

```

```

58     now = 1;
59     for (int i = 1; i <= m; ++i) {
60         int up = now * mul[i - 1] % mod;
61         int down = inv_fac[m - i] * inv_fac[i - 1] % mod;
62         int tmp = ((long long)v[i - 1] * up % mod) * down
63             % mod;
64         ret += (neg && tmp) ? (mod - tmp) : (tmp);
65         ret = (ret >= mod) ? (ret - mod) : ret;
66         now = now * (x - i) % mod;
67         neg ^= 1;
68     }
69     return ret;
70 };
71
72 int main() {
73     int n; cin >> n;
74     vector<int> v(n);
75     for (int i = 0; i < n; ++i) {
76         cin >> v[i];
77     }
78     Lagrange L;
79     L.construct_inv_fac();
80     L.init(0, v);
81     int x; cin >> x;
82     cout << L.sample(x);
83 }

```

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

```

8.4 KMP

```

1 // 給一個字串  $S$ ，定義函數  $\pi(i) = k$  代表  $S[1 \dots k] = S[i-k$ 
   //  $+1 \dots i]$ 
2 // 4c61a3
3 vector<int> KMP(string &s){
4     n = SZ(s);
5     vector<int> ret(n);
6     int now = 0;
7     for (int i=1 ; i<n ; i++){
8         int j = ret[i-1];

```

```

9         while (j>0 && s[i]!=s[j]){
10             j = ret[j-1];
11         }
12         if (s[i]==s[j]) j++;
13         ret[i] = j;
14     }
15     return ret;
16 }

```

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;
23            for (int i=1 ; i<n ; i++){
24                a = sa[i-1];
25                b = sa[i];
26                x[b] = (y[a] == y[b] && y[a+j] == y[b+j])
27                    ? p-1 : p++;
28            }
29            for (int i=1 ; i<n ; i++) rank[sa[i]] = i;
30            for (int i=0, j ; i<n-1 ; lcp[rank[i++]]=(k)
31                for (k && k--, j=sa[rank[i]-1] ; i+k<s.size() &&
32                    j+k<s.size() && s[i+k]==s[j+k] ; k++);
33            sa.erase(sa.begin());
34            lcp.erase(lcp.begin(), lcp.begin()+2);
35        }
36
37        vector<int> pos; // pos[i] = i 這個值在 pos 的哪個地方
38        SparseTable st;
39        void init_lcp(){
40            pos.resize(sa.size());
41            for (int i=0 ; i<sa.size() ; i++){
42                pos[sa[i]] = i;
43            }
44            if (lcp.size()){
45                st.build(lcp);
46            }
47
48            // 用之前記得 init

```

```

49 // 回傳  $[l1, r1]$  跟  $[l2, r2]$  的  $Lcp$ ， $\theta$ -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， $\theta$ -based
67 // 如果前者小於後者，就回傳  $<0$ ，相等就回傳  $=0$ ，否則回傳
   //  $>0$ 
68 int substring_cmp(int l1, int r1, int l2, int r2){
69     int len_1 = r1-l1+1;
70     int len_2 = r2-l2+1;
71     int res = get_lcp(l1, r1, l2, r2);
72
73     if (res<len_1 && res<len_2){
74         return s[l1+res]-s[l2+res];
75     }else if (len_1==res && len_2==res){
76         // 如果不需要以 index 作為次要排序參數，這裡要回
       // 傳 0
77         return l1-l2;
78     }else{
79         return len_1==res ? -1 : 1;
80     }
81 }
82
83 // 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  $Lcp$ ， $\theta$ -based
84 // pre[i] = s[i] 離他左邊最接近位置  $>p$  的後綴的  $Lcp$ ， $\theta$ -
   // based
85 // suf[i] = s[i] 離他右邊最接近位置  $>p$  的後綴的  $Lcp$ ， $\theta$ -
   // based
86 pair<vector<int>, vector<int>> get_left_and_right_lcp(int
87     p){
88     vector<int> pre(p+1);
89     vector<int> suf(p+1);
90
91     { // build pre
92         int now = 0;
93         for (int i=0 ; i<s.size() ; i++){
94             if (sa[i]<=p){
95                 pre[sa[i]] = now;
96                 if (i<lcp.size()) now = min(now, lcp[i]);
97             }else{
98                 if (i<lcp.size()) now = lcp[i];
99             }
100         }
101     }
102     { // build suf
103         int now = 0;
104         for (int i=s.size()-1 ; i>=0 ; i--){
105             if (sa[i]<=p){
106                 suf[sa[i]] = now;
107                 if (i-1>=0) now = min(now, lcp[i-1]);
108             }
109         }
110     }
111 }

```



```

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    return a;
11 }

```