

Contents

1 Misc	2	3.8 Segment Tree Persistent	7	6.19 Theorem	17
1.1 Custom Set PQ Sort	2	3.9 Sparse Table	7	7 Math	18
1.2 Default Code New	2	3.10 Treap	7	7.1 CRT m Coprime	18
1.3 Default Code Old	2	3.11 Trie	8	7.2 CRT m Not Coprime	18
1.4 Enumerate Subset	2	4 Dynamic-Programming	8	7.3 Josephus Problem	18
1.5 Fast Input	2	4.1 Digit DP	8	7.4 Lagrange any x	18
1.6 OEIS	2	4.2 Knaspack On Tree	8	7.5 Lagrange continuous x	18
1.7 Xor Basis	2	4.3 SOS DP	9	7.6 Lucas's Theorem	19
1.8 random int	3	4.4 Integer Partition	9	7.7 Matrix	19
1.9 Python	3	5 Geometry	9	7.8 Matrix 01	19
1.10 diff	3	5.1 Geometry Struct	9	7.9 Miller Rabin	19
1.11 hash command	3	5.2 Pick's Theorem	10	7.10 Pollard Rho	20
1.12 run	3	6 Graph	11	7.11 Polynomial	20
1.13 setup	3	6.1 2-SAT	11	7.12 數論分塊	21
2 Convolution	3	6.2 Augment Path	11	7.13 最大質因數	21
2.1 FFT any mod	3	6.3 Bridge BCC	11	7.14 歐拉公式	21
2.2 FFT new	4	6.4 Cut BCC	12	7.15 Burnside's Lemma	21
2.3 FFT short	4	6.5 Dinic	12	7.16 Catalan Number	21
2.4 FWT	4	6.6 Dominator Tree	12	7.17 Matrix Tree Theorem	21
2.5 NTT mod 998244353	4	6.7 Enumerate Triangle	13	7.18 Stirling's formula	21
2.6 Min Convolution Concave Concave	5	6.8 Find Bridge	13	7.19 Theorem	21
3 Data-Structure	5	6.9 HLD	13	7.20 二元一次方程式	21
3.1 BIT	5	6.10 Kosaraju	13	7.21 歐拉定理	21
3.2 Disjoint Set Persistent	5	6.11 Kuhn Munkres	14	7.22 錯排公式	21
3.3 PBDS GP Hash Table	5	6.12 LCA	14	8 String	22
3.4 PBDS Order Set	5	6.13 MCMF	15	8.1 Hash	22
3.5 Segment Tree Add Set	5	6.14 Tarjan	15	8.2 KMP	22
3.6 Segment Tree Li Chao Line	6	6.15 Tarjan Find AP	15	8.3 Manacher	22
3.7 Segment Tree Li Chao Segment	6	6.16 Tree Isomorphism	16	8.4 Min Rotation	22
		6.17 圓方樹	16	8.5 Suffix Array	22
		6.18 最大權閉合圖	17	8.6 Z Algorithm	23
				8.7 k-th Substring1	23

Misc

1.1 Custom Set PQ Sort

```

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

```

1.2 Default Code New

```

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

```

1.3 Default Code Old

```

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

```

```

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

```

1.4 Enumerate Subset

```

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

```

1.5 Fast Input

```

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

```

```

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

```

1.6 OEIS

```

1 // 若一個線性遞迴有 k 項 · 給他恰好 2*k 個項可以求出線性遞迴
2 // f915c2
3 template <typename T>
4 vector<T> BerlekampMassey(vector<T> a) {
5     auto scalarProduct = [](vector<T> v, T c) {
6         for (T &x: v) x *= c;
7         return v;
8     };
9     vector<T> s, best;
10    int bestPos = 0;
11    for (size_t i = 0; i < a.size(); i++) {
12        T error = a[i];
13        for (size_t j = 0; j < s.size(); j++) error -= s[j] *
14            a[i-1-j];
15        if (error == 0) continue;
16        if (s.empty()) {
17            s.resize(i + 1);
18            bestPos = i;
19            best.push_back(1 / error);
20            continue;
21        }
22        vector<T> fix = scalarProduct(best, error);
23        fix.insert(fix.begin(), i - bestPos - 1, 0);
24        if (fix.size() >= s.size()) {
25            best = scalarProduct(s, - 1 / error);
26            best.insert(best.begin(), 1 / error);
27            bestPos = i;
28            s.resize(fix.size());
29        }
30        for (size_t j = 0; j < fix.size(); j++)
31            s[j] += fix[j];
32    }
33    return s;

```

1.7 Xor Basis

```

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

```

```

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

```

1.8 random int

```

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

```

1.9 Python

```

1 sys.setrecursionlimit(100000)
2
3 sys.set_int_max_str_digits(10000)

```

1.10 diff

```

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

```

1.11 hash command

```

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

```

1.12 run

```

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

```

1.13 setup

```

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

```

2 Convolution

2.1 FFT any mod

```

1 /*
2 修改 const int MOD = 998244353 更改要取餘的數字
3 PolyMul(a, b) 回傳多項式乘法的結果 (c_k = \sum_{i+j=k} a_i b_j
4     mod MOD)

```

```

4
5 大約可以支援 5e5 · a_i, b_i 皆在 MOD 以下的非負整數
6 */
7 const int MOD = 998244353;
8 typedef complex<double> cd;
9
10 // b9c90a
11 void FFT(vector<cd> &a) {
12     int n = a.size(), L = 31-__builtin_clz(n);
13     vector<complex<long double>> R(2, 1);
14     vector<cd> rt(2, 1);
15     for (int k=2; k<n; k*=2){
16         R.resize(n);
17         rt.resize(n);
18         auto x = polar(1.0L, acos(-1.0L) / k);
19         for (int i=k; i<2*k; i++){
20             rt[i] = R[i] = (i&1 ? R[i/2]*x : R[i/2]);
21         }
22     }
23
24     vector<int> rev(n);
25     for (int i=0; i<n; i++){
26         rev[i] = (rev[i/2] | (i&1)<<L)/2;
27     }
28     for (int i=0; i<n; i++){
29         if (i<rev[i]) swap(a[i], a[rev[i]]);
30     }
31     for (int k=1; k<n; k*=2){
32         for (int i=0; i<n; i+=2*k){
33             for (int j=0; j<k; j++){
34                 auto x = (double *)&rt[j+k];
35                 auto y = (double *)&a[i+j+k];
36                 cd z(x[0]*y[0] - x[1]*y[1], x[0]*y[1] + x[1]*
37                     y[0]);
38                 a[i+j+k] = a[i+j]-z;
39                 a[i+j] += z;
40             }
41         }
42     }
43     return;
44 }
45
46 // d3c65e
47 vector<int> PolyMul(vector<int> a, vector<int> b){
48     if (a.empty() || b.empty()) return {};
49
50     vector<int> res(a.size()+b.size()-1);
51     int B = 32-__builtin_clz(res.size()), n = (1<<B), cut =
52         int(sqrt(MOD));
53     vector<cd> L(n), R(n), outs(n), outl(n);
54
55     for (int i=0; i<a.size(); i++){
56         L[i] = cd((int) a[i]/cut, (int) a[i]%cut);
57     }
58     for (int i=0; i<b.size(); i++){
59         R[i] = cd((int) b[i]/cut, (int) b[i]%cut);
60     }
61     FFT(L);
62     FFT(R);
63     for (int i=0; i<n; i++){
64         int j = -i&(n-1);
65         outl[j] = (L[i]+conj(L[j])) * R[i]/(2.0*n);
66         outs[j] = (L[i]-conj(L[j])) * R[i]/(2.0*n)/1i;
67     }
68     FFT(outl);

```

```

67 FFT(outs);
68 for (int i=0 ; i<res.size() ; i++){
69     int av = (int)(real(outl[i])+0.5), cv = (int)(imag(
70         outs[i])+0.5);
71     int bv = (int)(imag(outl[i])+0.5) + (int)(real(outs[i]
72         ])+0.5);
73     res[i] = ((av%MOD*cut+bv) % MOD*cut+cv) % MOD;
74 }
75 return res;
76 }

```

2.2 FFT new

```

1 typedef complex<double> cd;
2
3 // b9c90a
4 void FFT(vector<cd> &a) {
5     int n = a.size(), L = 31-__builtin_clz(n);
6     vector<complex<long double>> R(2, 1);
7     vector<cd> rt(2, 1);
8     for (int k=2 ; k<n ; k*=2){
9         R.resize(n);
10        rt.resize(n);
11        auto x = polar(1.0L, acos(-1.0L) / k);
12        for (int i=k ; i<2*k ; i++){
13            rt[i] = R[i] = (i&1 ? R[i/2]*x : R[i/2]);
14        }
15    }
16
17    vector<int> rev(n);
18    for (int i=0 ; i<n ; i++){
19        rev[i] = (rev[i/2] | (i&1)<<L)/2;
20    }
21    for (int i=0 ; i<n ; i++){
22        if (i<rev[i]) swap(a[i], a[rev[i]]);
23    }
24    for (int k=1 ; k<n ; k*=2){
25        for (int i=0 ; i<n ; i+=2*k){
26            for (int j=0 ; j<k ; j++){
27                auto x = (double *)&rt[j+k];
28                auto y = (double *)&a[i+j+k];
29                cd z(x[0]*y[0] - x[1]*y[1], x[0]*y[1] + x[1]*
30                    y[0]);
31                a[i+j+k] = a[i+j]-z;
32                a[i+j] += z;
33            }
34        }
35    }
36    return;
37
38 // 39029d
39 vector<double> PolyMul(const vector<double> a, const vector<
40     double> b){
41     if (a.empty() || b.empty()) return {};
42     vector<double> res(a.size()+b.size()-1);
43     int L = 32 - __builtin_clz(res.size()), n = 1 << L;
44     vector<cd> in(n), out(n);
45
46     copy(a.begin(), a.end(), begin(in));
47     for (int i=0 ; i<b.size() ; i++){
48         in[i].imag(b[i]);

```

```

48 }
49 FFT(in);
50 for (cd& x : in) x *= x;
51 for (int i=0 ; i<n ; i++){
52     out[i] = in[-i & (n - 1)] - conj(in[i]);
53 }
54 FFT(out);
55
56 for (int i=0 ; i<res.size() ; i++){
57     res[i] = imag(out[i]) / (4 * n);
58 }
59
60 return res;
61 }

```

2.3 FFT short

```

1 #define int long long
2
3 using Cplx = complex<double>;
4 const double pi = acos(-1);
5 const int mod = 998244353, g = 3;
6 int power(int a, int b) {
7     int res = 1;
8     while (b) {
9         if (b & 1) res = res * a % mod;
10        a = a * a % mod;
11        b >>= 1;
12    }
13    return res;
14 }
15 int inv(int x) { return power(x, mod - 2); }
16 // FFT use Cplx, NTT use LL
17 void FFT(vector<int> &a, int n, int op) {
18     // n must be 2^k
19     vector<int> R(n);
20     FOR (i, 0, n - 1)
21         R[i] = R[i/2]/2 + (i&1)*(n/2);
22     FOR (i, 0, n - 1)
23         if (i < R[i]) swap(a[i], a[R[i]]);
24     for (int m = 2; m <= n; m *= 2) {
25         // Cplx w1({cos(2*pi/m), sin(2*pi/m)*op});
26         int w1 = power(g, (mod-1)/m * op + mod-1);
27         for (int i = 0; i < n; i += m) {
28             // Cplx wk({1, 0});
29             int wk = 1;
30             FOR (k, 0, m / 2 - 1) {
31                 auto x = a[i+k], y = a[i+k+m/2] * wk % mod;
32                 a[i+k] = (x+y) % mod;
33                 a[i+k+m/2] = (x-y+mod) % mod;
34                 wk = wk * w1 % mod;
35             }
36         }
37     }
38     if (op == -1)
39         FOR (i, 0, n - 1) {
40             // a[i] = a[i] / n;
41             a[i] = a[i] * inv(n) % mod;
42         }
43 }

```

2.4 FWT

```

1 // 已經把 mint 刪掉 · 需要增加註解
2 vector<int> xor_convolution(vector<int> a, vector<int> b, int
3     k) {
4     if (k == 0) {
5         return vector<int>{a[0] * b[0]};
6     }
7     vector<int> aa(1 << (k - 1)), bb(1 << (k - 1));
8     FOR (i, 0, (1 << (k - 1)) - 1) {
9         aa[i] = a[i] + a[i + (1 << (k - 1))];
10        bb[i] = b[i] + b[i + (1 << (k - 1))];
11    }
12    vector<int> X = xor_convolution(aa, bb, k - 1);
13    FOR (i, 0, (1 << (k - 1)) - 1) {
14        aa[i] = a[i] - a[i + (1 << (k - 1))];
15        bb[i] = b[i] - b[i + (1 << (k - 1))];
16    }
17    vector<int> Y = xor_convolution(aa, bb, k - 1);
18    vector<int> c(1 << k);
19    FOR (i, 0, (1 << (k - 1)) - 1) {
20        c[i] = (X[i] + Y[i]) / 2;
21        c[i + (1 << (k - 1))] = (X[i] - Y[i]) / 2;
22    }
23    return c;
24 }

```

2.5 NTT mod 998244353

```

1 const int MOD = (119 << 23) + 1, ROOT = 62; // = 998244353
2 // For p < 2^30 there is also e.g. 5 << 25, 7 << 26, 479 <<
3 // 21
4 // and 483 << 21 (same root). The last two are > 10^9.
5 // 9cd58a
6 void NTT(vector<int> &a) {
7     int n = a.size();
8     int L = 31-__builtin_clz(n);
9     vector<int> rt(2, 1);
10    for (int k=2, s=2 ; k<n ; k*=2, s++){
11        rt.resize(n);
12        int z[] = {1, qp(ROOT, MOD>>s)};
13        for (int i=k ; i<2*k ; i++){
14            rt[i] = rt[i/2]*z[i&1]%MOD;
15        }
16    }
17
18    vector<int> rev(n);
19    for (int i=0 ; i<n ; i++){
20        rev[i] = (rev[i/2] | (i&1)<<L)/2;
21    }
22    for (int i=0 ; i<n ; i++){
23        if (i<rev[i]){
24            swap(a[i], a[rev[i]]);
25        }
26    }
27
28    for (int k=1 ; k<n ; k*=2){
29        for (int i=0 ; i<n ; i+=2*k){
30            for (int j=0 ; j<k ; j++){
31                int z = rt[j+k]*a[i+j+k]%MOD, &ai = a[i+j];
32                a[i+j+k] = ai-z+(z>ai ? MOD : 0);

```

```

33         ai += (ai+z)>=MOD ? z-MOD : z);
34     }
35 }
36 }
37 }
38 // 0b0e99
39 vector<int> polyMul(vector<int> &a, vector<int> &b){
40     if (a.empty() || b.empty()) return {};
41     int s = a.size()+b.size()-1, B = 32-__builtin_clz(s), n =
42         1<<B;
43     int inv = qp(n, MOD-2);
44     vector<int> L(a), R(b), out(n);
45     L.resize(n), R.resize(n);
46     NTT(L), NTT(R);
47     for (int i=0; i<n; i++){
48         out[-i&(n-1)] = L[i]*R[i]%MOD*inv%MOD;
49     }
50     NTT(out);
51     out.resize(s);
52     return out;
53 }
54 }
55 }

```

2.6 Min Convolution Concave Concave

```

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

```

3 Data-Structure

3.1 BIT

```

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

```

3.2 Disjoint Set Persistent

```

1 struct Persistent_Disjoint_Set{
2     Persistent_Segment_Tree arr, sz;
3
4     void init(int n){
5         arr.init(n);
6         vector<int> v1;
7         for (int i=0; i<n; i++){
8             v1.push_back(i);
9         }
10        arr.build(v1, 0);
11
12        sz.init(n);
13        vector<int> v2;
14        for (int i=0; i<n; i++){
15            v2.push_back(1);
16        }
17        sz.build(v2, 0);
18    }
19
20    int find(int a){
21        int res = arr.query_version(a, a+1, arr.version.size()
22            (-1).val);
23        if (res==a) return a;
24        return find(res);
25    }
26
27    bool unite(int a, int b){
28        a = find(a);
29        b = find(b);
30
31        if (a!=b){
32            int sz1 = sz.query_version(a, a+1, arr.version.
33                size()-1).val;
34            int sz2 = sz.query_version(b, b+1, arr.version.
35                size()-1).val;
36
37            if (sz1<sz2){
38                arr.update_version(a, b, arr.version.size()
39                    (-1));
40                sz.update_version(b, sz1+sz2, arr.version.
41                    size()-1);
42            }else{
43                arr.update_version(b, a, arr.version.size()
44                    (-1));
45                sz.update_version(a, sz1+sz2, arr.version.
46                    size()-1);
47            }
48            return true;
49        }
50        return false;
51    }
52 }
53 };

```

3.3 PBDS GP Hash Table

```

1 #include <ext/pb_ds/assoc_container.hpp>
2 using namespace __gnu_pbds;
3 typedef tree<int, null_type, less<int>, rb_tree_tag,
4     tree_order_statistics_node_update> order_set;

```

```

4 struct custom_hash {
5     static uint64_t splitmix64(uint64_t x) {
6         // http://xorshift.di.unimi.it/splitmix64.c
7         x += 0x9e3779b97f4a7c15;
8         x = (x ^ (x >> 30)) * 0xbf58476d1ce4e5b9;
9         x = (x ^ (x >> 27)) * 0x94d049bb133111eb;
10        return x ^ (x >> 31);
11    }
12
13    size_t operator()(uint64_t x) const {
14        static const uint64_t FIXED_RANDOM = chrono::
15            steady_clock::now().time_since_epoch().count();
16        return splitmix64(x + FIXED_RANDOM);
17    }
18 };
19 gp_hash_table<int, int, custom_hash> ss;

```

3.4 PBDS Order Set

```

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

```

3.5 Segment Tree Add Set

```

1 // [ll, rr), based-0
2 // 使用前記得 init(陣列大小), build(陣列名稱)
3 // add(ll, rr): 區間修改
4 // set(ll, rr): 區間賦值
5 // query(ll, rr): 區間求和 / 求最大值
6 struct SegmentTree{
7     struct node{
8         int add_tag = 0;
9         int set_tag = 0;
10        int sum = 0;
11        int ma = 0;
12    };
13
14    vector<node> arr;
15
16    SegmentTree(int n){
17        arr.resize(n<<2);
18    }
19
20    node pull(node A, node B){
21        node C;
22        C.sum = A.sum+B.sum;
23        C.ma = max(A.ma, B.ma);
24        return C;
25    }

```

```

26 // cce0c8
27 void push(int idx, int ll, int rr){
28     if (arr[idx].set_tag!=0){
29         arr[idx].sum = (rr-ll)*arr[idx].set_tag;
30         arr[idx].ma = arr[idx].set_tag;
31         if (rr-ll>1){
32             arr[idx*2+1].add_tag = 0;
33             arr[idx*2+1].set_tag = arr[idx].set_tag;
34             arr[idx*2+2].add_tag = 0;
35             arr[idx*2+2].set_tag = arr[idx].set_tag;
36         }
37         arr[idx].set_tag = 0;
38     }
39     if (arr[idx].add_tag!=0){
40         arr[idx].sum += (rr-ll)*arr[idx].add_tag;
41         arr[idx].ma += arr[idx].add_tag;
42         if (rr-ll>1){
43             arr[idx*2+1].add_tag += arr[idx].add_tag;
44             arr[idx*2+2].add_tag += arr[idx].add_tag;
45         }
46         arr[idx].add_tag = 0;
47     }
48 }
49
50 void build(vector<int> &v, int idx = 0, int ll = 0, int
51         rr = n){
52     if (rr-ll==1){
53         arr[idx].sum = v[ll];
54         arr[idx].ma = v[ll];
55     }else{
56         int mid = (ll+rr)/2;
57         build(v, idx*2+1, ll, mid);
58         build(v, idx*2+2, mid, rr);
59         arr[idx] = pull(arr[idx*2+1], arr[idx*2+2]);
60     }
61 }
62
63 void add(int ql, int qr, int val, int idx = 0, int ll =
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);

```

```

89     arr[idx] = pull(arr[idx*2+1], arr[idx*2+2]);
90 }
91
92 node query(int ql, int qr, int idx = 0, int ll = 0, int
93         rr = n){
94     push(idx, ll, rr);
95     if (rr<=ql || qr<=ll) return node();
96     if (ql<=ll && rr<=qr) return arr[idx];
97
98     int mid = (ll+rr)/2;
99     return pull(query(ql, qr, idx*2+1, ll, mid), query(ql
100         , qr, idx*2+2, mid, rr));

```

3.6 Segment Tree Li Chao Line

```

1 /*
2 全部都是  $\theta$ -based
3
4 宣告
5 LC_Segment_Tree st(n);
6
7 函式：
8 update({a, b})：插入一條  $y=ax+b$  的全域直線
9 query(x)：查詢所有直線在位置 x 的最小值
10 */
11 const int MAX_V = 1e6+10; // 值域最大值
12
13 struct LC_Segment_Tree{
14     struct Node{ //  $y = ax+b$ 
15         int a = 0;
16         int b = INF;
17
18         int y(int x){
19             return a*x+b;
20         }
21     };
22     vector<Node> arr;
23
24     LC_Segment_Tree(int n = 0){
25         arr.resize(4*n);
26     }
27
28     void update(Node val, int idx = 0, int ll = 0, int rr =
29         MAX_V){
30         if (rr-ll==0) return;
31         if (rr-ll==1){
32             if (val.y(ll)<arr[idx].y(ll)){
33                 arr[idx] = val;
34             }
35             return;
36         }
37
38         int mid = (ll+rr)/2;
39         if (arr[idx].a > val.a) swap(arr[idx], val); // 原本
40             的線斜率要比較小
41         if (arr[idx].y(mid) < val.y(mid)){ // 交點在左邊
42             update(val, idx*2+1, ll, mid);
43         }else{ // 交點在右邊

```

```

42         swap(arr[idx], val); // 在左子樹中，新線比舊線還
43             要好
44         update(val, idx*2+2, mid, rr);
45     }
46     return;
47 }
48
49 int query(int x, int idx = 0, int ll = 0, int rr = MAX_V)
50 {
51     if (rr-ll==0) return INF;
52     if (rr-ll==1){
53         return arr[idx].y(ll);
54     }
55
56     int mid = (ll+rr)/2;
57     if (x<mid){
58         return min(arr[idx].y(x), query(x, idx*2+1, ll,
59             mid));
60     }else{
61         return min(arr[idx].y(x), query(x, idx*2+2, mid,
62             rr));
63     }
64 }
65
66 };

```

3.7 Segment Tree Li Chao Segment

```

1 /*
2 全部都是  $\theta$ -based
3
4 宣告
5 LC_Segment_Tree st(n);
6
7 函式：
8 update_segment({a, b}, ql, qr)：在 [ql, qr) 插入一條  $y=ax+b$ 
9     的線段
10 query(x)：查詢所有直線在位置 x 的最小值
11 */
12 const int MAX_V = 1e6+10; // 值域最大值
13
14 struct LC_Segment_Tree{
15     struct Node{ //  $y = ax+b$ 
16         int a = 0;
17         int b = INF;
18
19         int y(int x){
20             return a*x+b;
21         }
22     };
23     vector<Node> arr;
24
25     LC_Segment_Tree(int n = 0){
26         arr.resize(4*n);
27     }
28
29     void update(Node val, int idx = 0, int ll = 0, int rr =
30         MAX_V){
31         if (rr-ll==0) return;
32         if (rr-ll==1){
33             if (val.y(ll)<arr[idx].y(ll)){
34                 arr[idx] = val;
35             }
36             return;
37         }
38
39         int mid = (ll+rr)/2;
40         if (arr[idx].a > val.a) swap(arr[idx], val); // 原本
41             的線斜率要比較小
42         if (arr[idx].y(mid) < val.y(mid)){ // 交點在左邊
43             update(val, idx*2+1, ll, mid);
44         }else{ // 交點在右邊

```

```

34     return;
35 }
36
37 int mid = (ll+rr)/2;
38 if (arr[idx].a > val.a) swap(arr[idx], val); // 原本
// 的線斜率要比較小
39 if (arr[idx].y(mid) < val.y(mid)){ // 交點在左邊
40     update(val, idx*2+1, ll, mid);
41 }else{ // 交點在右邊
42     swap(arr[idx], val); // 在左子樹中，新線比舊線還
// 要好
43     update(val, idx*2+2, mid, rr);
44 }
45 return;
46 }
47
48 // 在 [ql, qr] 加上一條 val 的線段
49 void update_segment(Node val, int ql, int qr, int idx =
0, int ll = 0, int rr = MAX_V){
50     if (rr-ll==0) return;
51     if (rr<=ql || qr<=ll) return;
52     if (ql<=ll && rr<=qr){
53         update(val, idx, ll, rr);
54         return;
55     }
56
57     int mid = (ll+rr)/2;
58     update_segment(val, ql, qr, idx*2+1, ll, mid);
59     update_segment(val, ql, qr, idx*2+2, mid, rr);
60     return;
61 }
62
63 int query(int x, int idx = 0, int ll = 0, int rr = MAX_V)
64 {
65     if (rr-ll==0) return INF;
66     if (rr-ll==1){
67         return arr[idx].y(ll);
68     }
69
70     int mid = (ll+rr)/2;
71     if (x<mid){
72         return min(arr[idx].y(x), query(x, idx*2+1, ll,
73         mid));
74     }else{
75         return min(arr[idx].y(x), query(x, idx*2+2, mid,
76         rr));
77     }
78 }
79
80 };

```

3.8 Segment Tree Persistent

```

1  /*
2  全部都是 0-based
3
4  宣告
5  Persistent_Segment_Tree st(n+q);
6  st.build(v, 0);
7
8  函式：
9  update_version(pos, val, ver)：對版本 ver 的 pos 位置改成 val

```

```

10 query_version(ql, qr, ver)：對版本 ver 查詢 [ql, qr] 的區間和
11 clone_version(ver)：複製版本 ver 到最新的版本
12 */
13 struct Persistent_Segment_Tree{
14     int node_cnt = 0;
15     struct Node{
16         int lc = -1;
17         int rc = -1;
18         int val = 0;
19     };
20     vector<Node> arr;
21     vector<int> version;
22
23     Persistent_Segment_Tree(int sz){
24         arr.resize(32*sz);
25         version.push_back(node_cnt++);
26         return;
27     }
28
29     void pull(Node &c, Node a, Node b){
30         c.val = a.val+b.val;
31         return;
32     }
33
34     void build(vector<int> &v, int idx, int ll = 0, int rr =
n){
35         auto &now = arr[idx];
36
37         if (rr-ll==1){
38             now.val = v[ll];
39             return;
40         }
41
42         int mid = (ll+rr)/2;
43         now.lc = node_cnt++;
44         now.rc = node_cnt++;
45         build(v, now.lc, ll, mid);
46         build(v, now.rc, mid, rr);
47         pull(now, arr[now.lc], arr[now.rc]);
48         return;
49     }
50
51     void update(int pos, int val, int idx, int ll = 0, int rr
= n){
52         auto &now = arr[idx];
53
54         if (rr-ll==1){
55             now.val = val;
56             return;
57         }
58
59         int mid = (ll+rr)/2;
60         if (pos<mid){
61             arr[node_cnt] = arr[now.lc];
62             now.lc = node_cnt;
63             node_cnt++;
64             update(pos, val, now.lc, ll, mid);
65         }else{
66             arr[node_cnt] = arr[now.rc];
67             now.rc = node_cnt;
68             node_cnt++;
69             update(pos, val, now.rc, mid, rr);
70         }
71         pull(now, arr[now.lc], arr[now.rc]);
72         return;

```

```

73     }
74
75     void update_version(int pos, int val, int ver){
76         update(pos, val, version[ver]);
77     }
78
79     Node query(int ql, int qr, int idx, int ll = 0, int rr =
n){
80         auto &now = arr[idx];
81
82         if (ql<=ll && rr<=qr) return now;
83         if (rr<=ql || qr<=ll) return Node();
84
85         int mid = (ll+rr)/2;
86
87         Node ret;
88         pull(ret, query(ql, qr, now.lc, ll, mid), query(ql,
89         qr, now.rc, mid, rr));
90         return ret;
91     }
92
93     Node query_version(int ql, int qr, int ver){
94         return query(ql, qr, version[ver]);
95     }
96
97     void clone_version(int ver){
98         version.push_back(node_cnt);
99         arr[node_cnt] = arr[version[ver]];
100         node_cnt++;
101     }
102 };

```

3.9 Sparse Table

```

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

```

3.10 Treap


```

1 struct Treap{
2     Treap *l = nullptr, *r = nullptr;
3     int pri = rand(), val = 0, sz = 1;
4
5     Treap(int _val){
6         val = _val;
7     }
8 };
9
10 int size(Treap *t){return t ? t->sz : 0;}
11 void pull(Treap *t){
12     t->sz = size(t->l)+size(t->r)+1;
13 }
14
15 Treap* merge(Treap *a, Treap *b){
16     if (!a || !b) return a ? a : b;
17
18     if (a->pri>b->pri){
19         a->r = merge(a->r, b);
20         pull(a);
21         return a;
22     }else{
23         b->l = merge(a, b->l);
24         pull(b);
25         return b;
26     }
27 }
28
29 pair<Treap*, Treap*> split(Treap *&t, int k){ // 1-based <前
30     k 個元素, 其他元素>
31     if (!t) return {};
32     if (size(t->l)>=k){
33         auto pa = split(t->l, k);
34         t->l = pa.second;
35         pull(t);
36         return {pa.first, t};
37     }else{
38         auto pa = split(t->r, k-size(t->l)-1);
39         t->r = pa.first;
40         pull(t);
41         return {t, pa.second};
42     }
43 }
44
45 // functions
46 Treap* build(vector<int> v){
47     Treap* ret = nullptr;
48     for (int i=0 ; i<v.size() ; i++){
49         ret = merge(ret, new Treap(v[i]));
50     }
51     return ret;
52 }
53
54 array<Treap*, 3> cut(Treap *t, int l, int r){ // 1-based <前
55     1~l-1 個元素, l~r 個元素, r+1 個元素>
56     array<Treap*, 3> ret;
57     tie(ret[1], ret[2]) = split(t, r);
58     tie(ret[0], ret[1]) = split(ret[1], l-1);
59     return ret;
60 }
61
62 void print(Treap *t, bool flag = true){
63     if (t->l!=0) print(t->l, false);
64     cout << t->val;

```

```

64     if (t->r!=0) print(t->r, false);
65     if (flag) cout << endl;
66 }

```

3.11 Trie

```

1 struct Trie{
2     struct Data{
3         int nxt[2]={0, 0};
4     };
5
6     int sz=0;
7     vector<Data> arr;
8
9     void init(int n){
10         arr.resize(n);
11     }
12
13     void insert(int n){
14         int now=0;
15         for (int i=N ; i>=0 ; i--){
16             int v=(n>>i)&1;
17             if (!arr[now].nxt[v]){
18                 arr[now].nxt[v]=++sz;
19             }
20             now=arr[now].nxt[v];
21         }
22     }
23
24     int query(int n){
25         int now=0, ret=0;
26         for (int i=N ; i>=0 ; i--){
27             int v=(n>>i)&1;
28             if (arr[now].nxt[1-v]){
29                 ret+=(1<<i);
30                 now=arr[now].nxt[1-v];
31             }else if (arr[now].nxt[v]){
32                 now=arr[now].nxt[v];
33             }else{
34                 return ret;
35             }
36         }
37         return ret;
38     }
39 } tr;
40

```

4 Dynamic-Programming

4.1 Digit DP

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 long long l, r;
5 long long dp[20][10][2][2]; // dp[pos][pre][limit] = 後 pos
6 // 位 · pos 前一位是 pre · (是/否) 有上界 · (是/否) 有前綴零
7 // 的答案數量

```

```

6 long long memorize_search(string &s, int pos, int pre, bool
7     limit, bool lead){
8
9     // 已經被找過了 · 直接回傳值
10    if (dp[pos][pre][limit][lead]!=-1) return dp[pos][pre][
11        limit][lead];
12
13    // 已經搜尋完畢 · 紀錄答案並回傳
14    if (pos==(int)s.size()){
15        return dp[pos][pre][limit][lead] = 1;
16    }
17
18    // 枚舉目前的位數數字是多少
19    long long ans = 0;
20    for (int now=0 ; now<=(limit ? s[pos]-'0' : 9) ; now++){
21        if (now==pre){
22
23            // 1~9 絕對不能連續出現
24            if (pre!=0) continue;
25
26            // 如果已經不在前綴零的範圍內 · 0 不能連續出現
27            if (lead==false) continue;
28
29            ans += memorize_search(s, pos+1, now, limit&(now==(s[
30                pos]-'0')), lead&(now==0));
31        }
32
33        // 已經搜尋完畢 · 紀錄答案並回傳
34        return dp[pos][pre][limit][lead] = ans;
35    }
36
37    // 回傳 [0, n] 有多少數字符合條件
38    long long find_answer(long long n){
39        memset(dp, -1, sizeof(dp));
40        string tmp = to_string(n);
41
42        return memorize_search(tmp, 0, 0, true, true);
43    }
44
45    int main(){
46
47        // input
48        cin >> l >> r;
49
50        // output - 計算 [l, r] 有多少數字任意兩個位數都不相同
51        cout << find_answer(r)-find_answer(l-1) << "\n";
52
53        return 0;
54    }

```

4.2 Knapsack On Tree

```

1 // 需要重構 · 需要增加註解
2 #include <bits/stdc++.h>
3 #define F first
4 #define S second
5 #define all(x) begin(x), end(x)
6 using namespace std;
7

```



```

8 #define chmax(a, b) (a) = (a) < (b) ? (b) : (a)
9 #define chmin(a, b) (a) = (a) < (b) ? (a) : (b)
10
11 #define ll long long
12
13 #define FOR(i, a, b) for (int i = a; i <= b; i++)
14
15 int N, W, cur;
16 vector<int> w, v, sz;
17 vector<vector<int>> adj, dp;
18
19 void dfs(int x) {
20     sz[x] = 1;
21     for (int i : adj[x]) dfs(i), sz[x] += sz[i];
22     cur++;
23     // choose x
24     for (int i=w[x] ; i<=W ; i++){
25         dp[cur][i] = dp[cur - 1][i - w[x]] + v[x];
26     }
27     // not choose x
28     for (int i=0 ; i<=W ; i++){
29         chmax(dp[cur][i], dp[cur - sz[x]][i]);
30     }
31 }
32
33 signed main() {
34     cin >> N >> W;
35     adj.resize(N + 1);
36     w.assign(N + 1, 0);
37     v.assign(N + 1, 0);
38     sz.assign(N + 1, 0);
39     dp.assign(N + 2, vector<int>(W + 1, 0));
40     for (int i=1 ; i<=N ; i++){
41         int p; cin >> p;
42         adj[p].push_back(i);
43     }
44
45     for (int i=1 ; i<=N ; i++) cin >> w[i];
46     for (int i=1 ; i<=N ; i++) cin >> v[i];
47     dfs(0);
48     cout << dp[N + 1][W] << '\n';
49 }

```

4.3 SOS DP

```

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

```

4.4 Integer Partition

$dp[i][x]$ = 要將整數 x 拆成 i 堆的「組合數」

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

5 Geometry

5.1 Geometry Struct

```

1 // 判斷數值正負：{1:正數,0:零,-1:負數}
2 int sign(long long x) {return (x >= 0) ? ((bool)x) : -1; }
3 int sign(double x) {
4     return (abs(x) < 1e-9) ? 0 : (x > 0 ? 1 : -1);
5 }
6
7 template<typename T>
8 struct point {
9     T x, y;
10     point() {}
11     point(const T &x, const T &y) : x(x), y(y) {}
12
13     point operator+(point b) {return {x+b.x, y+b.y}; }
14     point operator-(point b) {return {x-b.x, y-b.y}; }
15     point operator*(T b) {return {x*b, y*b}; }
16     point operator/(T b) {return {x/b, y/b}; }
17     bool operator==(point b) {return x==b.x && y==b.y; }
18     // 逆時針極角排序
19     bool operator<(point &b) {return (x*b.y > b.x*y); }
20     friend ostream& operator<<(ostream& os, point p) {
21         os << "(" << p.x << ", " << p.y << ")";
22         return os;
23     }
24     // 判斷 ab 到 ac 的方向：{1:逆時鐘,0:重疊,-1:順時鐘}
25     friend int ori(point a, point b, point c) {
26         return sign((b-a)^(c-a));
27     }
28     friend bool btw(point a, point b, point c) {
29         return ori(a, b, c) == 0 && sign((a-c)*(b-c)) <= 0;
30     }
31     // 判斷線段 ab, cd 是否相交
32     friend bool banana(point a, point b, point c, point d) {
33         int s1 = ori(a, b, c);
34         int s2 = ori(a, b, d);
35         int s3 = ori(c, d, a);
36         int s4 = ori(c, d, b);
37         if (btw(a, b, c) || btw(a, b, d) || btw(c, d, a) ||
38             btw(c, d, b)) return 1;
39         return (s1 * s2 < 0) && (s3 * s4 < 0);
40     }
41
42     T operator*(point b) {return x * b.x + y * b.y; }
43     T operator^(point b) {return x * b.y - y * b.x; }
44     T abs2() {return (*this) * (*this); }
45
46     // 旋轉 Arg(b) 的角度 (小心溢位)
47     point rotate(point b) {return {x*b.x - y*b.y, x*b.y + y*b
48         .x}; }
49
50 template<typename T>
51 struct line {
52     point<T> p1, p2;
53     // ax + by + c = 0
54     T a, b, c; // |a|, |b| ≤ 2C, |c| ≤ 8C²
55     line() {}
56     line(const point<T> &x, const point<T> &y) : p1(x), p2(y){
57         build();
58     }
59
60     void build() {
61         a = p1.y - p2.y;
62         b = p2.x - p1.x;
63         c = (-a*p1.x)-b*p1.y;
64     }
65     // 判斷點和有向直線的關係：{1:左邊,0:在線上,-1:右邊}
66     int ori(point<T> &p) {
67         return sign((p2-p1) ^ (p-p1));
68     }
69     // 判斷直線斜率是否相同
70     bool parallel(line &l) {
71         return ((p1-p2) ^ (l.p1-l.p2)) == 0;
72     }
73     // 兩直線交點
74     point<long double> line_intersection(line &l) {
75         using P = point<long double>;
76         point<T> a = p2-p1, b = l.p2-l.p1, s = l.p1-p1;
77         return P(p1.x,p1.y) + P(a.x,a.y) * (((long double)(s^b))
78             / (a^b));
79     }
80 };
81
82 template<typename T>
83 struct polygon {
84     vector<point<T>> v;
85     polygon() {}
86     polygon(const vector<point<T>> &u) : v(u) {}
87     // simple 為 true 的時候會回傳任意三點不共線的凸包
88     void make_convex_hull(int simple) {
89         auto cmp = [&](point<T> &p, point<T> &q) {
90             return (p.x == q.x) ? (p.y < q.y) : (p.x < q.x);
91         };
92         simple = (bool)simple;
93         sort(v.begin(), v.end(), cmp);
94         v.resize(unique(v.begin(), v.end()) - v.begin());
95         vector<point<T>> hull;
96         for (int t = 0; t < 2; ++t){
97             int sz = hull.size();
98             for (auto &i:v) {
99                 while (hull.size() >= sz+2 && ori(hull[hull.
100                     size()-2], hull.back(), i) < simple) {
101                     hull.pop_back();
102                 }
103                 hull.push_back(i);
104             }
105             hull.pop_back();
106             reverse(v.begin(), v.end());
107         }
108         swap(hull, v);
109     }
110     // 可以在有 n 個點的簡單多邊形內，用 O(n) 判斷一個點：
111     // {1：在多邊形內，0：在多邊形上，-1：在多邊形外}
112     int in_polygon(point<T> a){
113         const T MAX_POS = 1e9 + 5; // [記得修改] 座標的最大值
114         point<T> pre = v.back(), b(MAX_POS, a.y + 1);
115         int cnt = 0;
116
117         for (auto &i:v) {
118             if (btw(pre, i, a)) return 0;
119             if (banana(a, b, pre, i)) cnt++;
120             pre = i;
121         }
122
123         return cnt%2 ? 1 : -1;
124     }
125 };

```

```

120 }
121 /// 警告：以下所有凸包專用的函式都只接受逆時針排序且任三點不
    共線的凸包 ///
122 // 可以在有  $n$  個點的凸包內，用  $O(\log n)$  判斷一個點：
123 // {1：在凸包內，0：在凸包邊上，-1：在凸包外}
124 int in_convex(point<T> p) {
125     int n = v.size();
126     int a = ori(v[0], v[1], p), b = ori(v[0], v[n-1], p);
127     if (a < 0 || b > 0) return -1;
128     if (btw(v[0], v[1], p)) return 0;
129     if (btw(v[0], v[n-1], p)) return 0;
130     int l = 1, r = n - 1, mid;
131     while (l + 1 < r) {
132         mid = (l + r) >> 1;
133         if (ori(v[0], v[mid], p) >= 0) l = mid;
134         else r = mid;
135     }
136     int k = ori(v[l], v[r], p);
137     if (k <= 0) return k;
138     return 1;
139 }
140 // 凸包專用的環狀二分搜，回傳  $\theta$ -based index
141 int cycle_search(auto &f) {
142     int n = v.size(), l = 0, r = n;
143     bool rv = f(1, 0);
144     while (r - l > 1) {
145         int m = (l + r) / 2;
146         if (f(0, m) ? rv : f(m, (m + 1) % n)) r = m;
147         else l = m;
148     }
149     return f(1, r % n) ? l : r % n;
150 }
151 // 可以在有  $n$  個點的凸包內，用  $O(\log n)$  判斷一條直線：
152 // {1：穿過凸包，0：剛好切過凸包，-1：沒碰到凸包}
153 int line_cut_convex(line<T> L) {
154     point<T> p(L.a, L.b); // 記得 L 要 build
155     auto gt = [&](int neg) {
156         auto f = [&](int x, int y) {
157             return sign((v[x] - v[y]) * p) == neg;
158         };
159         return -(v[cycle_search(f)] * p);
160     };
161     T x = gt(1), y = gt(-1);
162     if (L.c < x || y < L.c) return -1;
163     return not (L.c == x || L.c == y);
164 }
165 // 可以在有  $n$  個點的凸包內，用  $O(\log n)$  判斷一個線段：
166 // {1：存在一個凸包上的邊可以把這個線段切成兩半，
167 // 0：有碰到凸包但沒有任何凸包上的邊可以把它切成兩半，
168 // -1：沒碰到凸包}
169 /// 除非線段兩端點都不在凸包邊上，否則此函數回傳 0 的時候不一
    定表示線段沒有通過凸包內部 ///
170 int segment_across_convex(line<T> L) {
171     point<T> p(L.a, L.b); // 記得 L 要 build
172     auto gt = [&](int neg) {
173         auto f = [&](int x, int y) {
174             return sign((v[x] - v[y]) * p) == neg;
175         };
176         return cycle_search(f);
177     };
178     int i = gt(1), j = gt(-1), n = v.size();
179     T x = -(v[i] * p), y = -(v[j] * p);
180     if (L.c < x || y < L.c) return -1;

```

```

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

```

```

246 if (btw(v[(x + n - 1) % n], v[x], p)) edg = (
247     x + n - 1) % n;
248 };
249 auto f = [&](int x, int y) {
250     check(x); check(y);
251     return ori(p, v[x], v[y]) == neg;
252 };
253 return cycle_search(f);
254 };
255 int x = gt(1), y = gt(-1);
256 if (z != -1) {
257     return {(z + n - 1) % n, (z + 1) % n};
258 }
259 else if (edg != -1) {
260     return {edg, (edg + 1) % n};
261 }
262 else {
263     return {x, y};
264 }
265 }
266 friend int halfplane_intersection(vector<line<T>> &s,
    polygon<T> &P) {
267     #define neg(p) ((p.y == 0 ? p.x : p.y) < 0)
268     auto angle_cmp = [&](line<T> &A, line<T> &B) {
269         point<T> a = A.p2-A.p1, b = B.p2-B.p1;
270         return neg(a) < neg(b) || (neg(a) == neg(b) && (a
271             ^ b) > 0);
272     };
273     #undef neg
274     sort(s.begin(), s.end(), angle_cmp); // 線段左側為該
        線段半平面
275     int L, R, n = s.size();
276     vector<point<T>> px(n);
277     vector<line<T>> q(n);
278     q[L = R = 0] = s[0];
279     for (int i = 1; i < n; ++i) {
280         while (L < R && s[i].ori(px[R-1]) <= 0) --R;
281         while (L < R && s[i].ori(px[L]) <= 0) ++L;
282         q[++R] = s[i];
283         if (q[R].parallel(q[R-1])) {
284             --R;
285             if (q[R].ori(s[i].p1) > 0) q[R] = s[i];
286         }
287         if (L < R) px[R-1] = q[R-1].line_intersection(q[R
288             ]);
289     }
290     while (L < R && q[L].ori(px[R-1]) <= 0) --R;
291     P.v.clear();
292     if (R - L <= 1) return 0;
293     px[R] = q[R].line_intersection(q[L]);
294     for (int i = L; i <= R; ++i) P.v.push_back(px[i]);
295     return R - L + 1;
296 }
297 };

```

5.2 Pick's Theorem

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

6 Graph

6.1 2-SAT

```

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

```

```

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

```

6.2 Augment Path

```

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

```

```

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

```

6.3 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 }

```

6.4 Cut BCC

```

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

```

6.5 Dinic

```

1 // 一般圖 :  $O(EV^2)$ 
2 // 二分圖 :  $O(EV)$ 
3 struct Flow{
4     using T = int; // 可以換成別的类型
5     struct Edge{
6         int v; T rc; int rid;
7     };
8     vector<vector<Edge>> G;
9     void add(int u, int v, T c){
10         G[u].push_back({v, c, G[v].size()});
11         G[v].push_back({u, 0, G[u].size()-1});
12     }
13     vector<int> dis, it;
14
15     Flow(int n){
16         G.resize(n);
17         dis.resize(n);
18         it.resize(n);
19     }
20
21     // ce56d6
22     T dfs(int u, int t, T f){
23         if (u == t || f == 0) return f;
24         for (int &i=it[u] ; i<G[u].size() ; i++){
25             auto &[v, rc, rid] = G[u][i];
26             if (dis[v]!=dis[u]+1) continue;
27             T df = dfs(v, t, min(f, rc));
28             if (df <= 0) continue;
29             rc -= df;
30             G[v][rid].rc += df;
31             return df;
32         }
33         return 0;
34     }
35
36     // e22e39
37     T flow(int s, int t){
38         T ans = 0;
39         while (true){
40             fill(dis.begin(), dis.end(), INF);
41             queue<int> q;
42             q.push(s);
43             dis[s] = 0;
44
45             while (q.size()){
46                 int u = q.front(); q.pop();
47                 for (auto [v, rc, rid] : G[u]){
48                     if (rc <= 0 || dis[v] < INF) continue;
49                     dis[v] = dis[u] + 1;
50                     q.push(v);
51                 }
52             }
53             if (dis[t]==INF) break;
54
55             fill(it.begin(), it.end(), 0);
56             while (true){
57                 T df = dfs(s, t, INF);
58                 if (df <= 0) break;
59                 ans += df;
60             }
61         }
62         return ans;
63     }
64
65     // the code below constructs minimum cut

```

```

66 void dfs_mincut(int now, vector<bool> &vis){
67     vis[now] = true;
68     for (auto &[v, rc, rid] : G[now]){
69         if (vis[v] == false && rc > 0){
70             dfs_mincut(v, vis);
71         }
72     }
73 }
74
75 vector<pair<int, int>> construct(int n, int s, vector<
76     pair<int, int>> &E){
77     /// E is G without capacity
78     vector<bool> vis(n);
79     dfs_mincut(s, vis);
80     vector<pair<int, int>> ret;
81     for (auto &[u, v] : E){
82         if (vis[u] == true && vis[v] == false){
83             ret.emplace_back(u, v);
84         }
85     }
86     return ret;
87 }

```

6.6 Dominator Tree

```

1 /*
2 全部都是 0-based
3 G 要是有向無權圖
4 一開始要初始化 G(N, root) · 代表有 N 個節點 · 根是 root
5 用完之後要 build
6 G[i] = i 的 idom · 也就是從 root 走到 i 時 · 一定要走到的點且離
   i 最近
7 */
8 struct DominatorTree{
9     int N;
10    vector<vector<int>> G;
11    vector<vector<int>> buckets, rg;
12    /// dfn[x] = the DFS order of x
13    /// rev[x] = the vertex with DFS order x
14    /// par[x] = the parent of x
15    vector<int> dfn, rev, par;
16    vector<int> sdom, dom, idom;
17    vector<int> fa, val;
18    int stamp;
19    int root;
20
21    int operator [] (int x){
22        return idom[x];
23    }
24
25    DominatorTree(int _N, int _root) :
26        N(_N),
27        G(N), buckets(N), rg(N),
28        dfn(N, -1), rev(N, -1), par(N, -1),
29        sdom(N, -1), dom(N, -1), idom(N, -1),
30        fa(N, -1), val(N, -1)
31    {
32        stamp = 0;
33        root = _root;
34    }
35 }

```

```

36 void add_edge(int u, int v){
37     G[u].push_back(v);
38 }
39
40 void dfs(int x){
41     rev[dfn[x] = stamp] = x;
42     fa[stamp] = sdom[stamp] = val[stamp] = stamp;
43     stamp++;
44
45     for (int u : G[x]){
46         if (dfn[u]==-1){
47             dfs(u);
48             par[dfn[u]] = dfn[x];
49         }
50         rg[dfn[u]].push_back(dfn[x]);
51     }
52 }
53
54 int eval(int x, bool first){
55     if (fa[x]==x) return !first ? -1 : x;
56     int p = eval(fa[x], false);
57
58     if (p==-1) return x;
59     if (sdom[val[x]]>sdom[val[fa[x]]]) val[x] = val[fa[x]];
60     fa[x] = p;
61
62     return !first ? p : val[x];
63 }
64
65 void link(int x, int y){
66     fa[x] = y;
67 }
68
69 void build(){
70     dfs(root);
71
72     for (int x=stamp-1 ; x>=0 ; x--){
73         for (int y : rg[x]){
74             sdom[x] = min(sdom[x], sdom[eval(y, true)]);
75         }
76         if (x>0) buckets[sdom[x]].push_back(x);
77         for (int u : buckets[x]){
78             int p = eval(u, true);
79             if (sdom[p]==x) dom[u] = x;
80             else dom[u] = p;
81         }
82         if (x>0) link(x, par[x]);
83     }
84
85     idom[root] = root;
86     for (int x=1 ; x<stamp ; x++){
87         if (sdom[x]!=dom[x]) dom[x] = dom[dom[x]];
88     }
89     for (int i=1 ; i<stamp ; i++) idom[rev[i]] = rev[dom[i]];
90 }
91 };

```

6.7 Enumerate Triangle

1 // $O(m \log m)$ 枚舉無向圖所有三角形 · θ -based

```

2 void Enumerate_Triangle(vector<pair<int, int>> &edge, vector<
3     int> &deg){
4     int n = deg.size();
5     int m = edge.size();
6     vector<vector<int>> G(n);
7
8     for (int i=0 ; i<m ; i++){
9         if (deg[edge[i].first] > deg[edge[i].second]) swap(
10             edge[i].first, edge[i].second);
11         if (deg[edge[i].first] == deg[edge[i].second] && edge
12             [i].first > edge[i].second) swap(edge[i].first,
13             edge[i].second);
14         G[edge[i].first].push_back(edge[i].second);
15     }
16
17     vector<int> vis(n, false);
18     for (int i=0 ; i<n ; i++){
19         for (auto j : G[i]) vis[j] = true;
20         for (auto j : G[i]){
21             for (auto k : G[j]){
22                 if (vis[k]){
23                     // i, j, k is a triangle
24                 }
25             }
26         }
27         for (auto j : G[i]) vis[j] = false;
28     }
29 }

```

6.8 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
25     return;
26 }

```

6.9 HLD

```

1 #include <bits/stdc++.h>
2 #define int long long
3 using namespace std;
4
5 const int N = 100005;
6 vector<int> G[N];
7 struct HLD {
8     vector<int> pa, sz, depth, mxson, topf, id;
9     int n, idcnt = 0;
10     HLD(int _n) : n(_n), pa(_n + 1), sz(_n + 1), depth(_n +
11         1), mxson(_n + 1), topf(_n + 1), id(_n + 1) {}
12     void dfs1(int v = 1, int p = -1) {
13         pa[v] = p; sz[v] = 1; mxson[v] = 0;
14         depth[v] = (p == -1 ? 0 : depth[p] + 1);
15         for (int u : G[v]) {
16             if (u == p) continue;
17             dfs1(u, v);
18             sz[v] += sz[u];
19             if (sz[u] > sz[mxson[v]]) mxson[v] = u;
20         }
21     }
22     void dfs2(int v = 1, int top = 1) {
23         id[v] = ++idcnt;
24         topf[v] = top;
25         if (mxson[v]) dfs2(mxson[v], top);
26         for (int u : G[v]) {
27             if (u == mxson[v] || u == pa[v]) continue;
28             dfs2(u, u);
29         }
30     }
31     // query 為區間資料結構
32     int path_query(int a, int b) {
33         int res = 0;
34         while (topf[a] != topf[b]) { /// 若不在同一條鍊上
35             if (depth[topf[a]] < depth[topf[b]]) swap(a, b);
36             res = max(res, 0ll); // query : l = id[topf[a]],
37                                     r = id[a]
38             a = pa[topf[a]];
39         }
40         /// 此時已在同一條鍊上
41         if (depth[a] < depth[b]) swap(a, b);
42         res = max(res, 0ll); // query : l = id[b], r = id[a]
43         return res;
44     }
45 };

```

6.10 Kosaraju

```

1 /*
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 */

```

```

13 struct SCC_compress{
14     int N, M, sz;
15     vector<vector<int>> G, inv_G, result;
16     vector<pair<int, int>> edges;
17     vector<bool> vis;
18     vector<int> order;
19
20     vector<vector<int>> SCC;
21     vector<int> SCC_id;
22
23     SCC_compress(int N) :
24         N(_N), M(0), sz(0),
25         G(N), inv_G(N),
26         vis(N), SCC_id(N)
27     {}
28
29     vector<int> operator [] (int x){
30         return result[x];
31     }
32
33     void add_edge(int u, int v){
34         G[u].push_back(v);
35         inv_G[v].push_back(u);
36         edges.push_back({u, v});
37         M++;
38     }
39
40     void dfs1(vector<vector<int>> &G, int now){
41         vis[now] = 1;
42         for (auto x : G[now]) if (!vis[x]) dfs1(G, x);
43         order.push_back(now);
44     }
45
46     void dfs2(vector<vector<int>> &G, int now){
47         SCC_id[now] = SCC.size()-1;
48         SCC.back().push_back(now);
49         vis[now] = 1;
50         for (auto x : G[now]) if (!vis[x]) dfs2(G, x);
51     }
52
53     void compress(){
54         fill(vis.begin(), vis.end(), 0);
55         for (int i=0; i<N; i++) if (!vis[i]) dfs1(G, i);
56
57         fill(vis.begin(), vis.end(), 0);
58         reverse(order.begin(), order.end());
59         for (int i=0; i<N; i++){
60             if (!vis[order[i]]){
61                 SCC.push_back(vector<int>());
62                 dfs2(inv_G, order[i]);
63             }
64         }
65
66         result.resize(SCC.size());
67         sz = SCC.size();
68         for (auto [u, v] : edges){
69             if (SCC_id[u]!=SCC_id[v]) result[SCC_id[u]].
                push_back(SCC_id[v]);
70         }
71         for (int i=0; i<SCC.size(); i++){
72             sort(result[i].begin(), result[i].end());
73             result[i].resize(unique(result[i].begin(), result
                [i].end())-result[i].begin());
74         }
75     }
76 };

```

6.11 Kuhn Munkres

```

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

```

```

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

```

6.12 LCA

```

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

```



```

28 }
29
30 void build_LCA(int root = 0){
31     dfs(root, root);
32     for (int i=0 ; i<N ; i++) LCA[0][i] = parent[i];
33     for (int i=1 ; i<H ; i++){
34         for (int j=0 ; j<N ; j++){
35             LCA[i][j] = LCA[i-1][LCA[i-1][j]];
36         }
37     }
38 }
39
40 int jump(int u, int step){
41     for (int i=0 ; i<H ; i++){
42         if (step&(1<<i)) u = LCA[i][u];
43     }
44     return u;
45 }
46
47 int get_LCA(int u, int v){
48     if (dep[u]<dep[v]) swap(u, v);
49     u = jump(u, dep[u]-dep[v]);
50     if (u==v) return u;
51     for (int i=H-1 ; i>=0 ; i--){
52         if (LCA[i][u]!=LCA[i][v]){
53             u = LCA[i][u];
54             v = LCA[i][v];
55         }
56     }
57     return parent[u];
58 }
59 };

```

6.13 MCMF

```

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

```

```

29         in_q[v] = false;
30
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     return dis[t];
43 }
44
45 // return <max flow, min cost>, 150093
46 pair<int, int> flow(int s, int t){
47     int fl = 0, cost = 0, d;
48     while ((d = spfa(s, t))<INF){
49         int cur = INF;
50         for (int v=t ; v!=s ; v=par[v])
51             cur = min(cur, G[par[v]][par_eid[v]].rc);
52         fl += cur;
53         cost += d*cur;
54         for (int v=t ; v!=s ; v=par[v]){
55             G[par[v]][par_eid[v]].rc -= cur;
56             G[v][G[par[v]][par_eid[v]].rv].rc += cur;
57         }
58     }
59     return {fl, cost};
60 }
61
62 vector<pair<int, int>> construct(){
63     vector<pair<int, int>> ret;
64     for (int i=0 ; i<n ; i++){
65         for (auto x : G[i]){
66             if (x.rc==0){
67                 ret.push_back({i+1, x.u-n+1});
68                 break;
69             }
70         }
71     }
72     return ret;
73 }
74
75 };

```

6.14 Tarjan

```

1 struct tarjan_SCC {
2     int now_T, now_SCCs;
3     vector<int> dfn, low, SCC;
4     stack<int> S;
5     vector<vector<int>> E;
6     vector<bool> vis, in_stack;
7
8     tarjan_SCC(int n) {
9         init(n);
10    }
11
12    void init(int n) {
13        now_T = now_SCCs = 0;
14        dfn = low = SCC = vector<int>(n);

```

```

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

```

6.15 Tarjan Find AP

```

1 vector<int> dep(MAX_N), low(MAX_N), AP;
2 bitset<MAX_N> vis;
3
4 void dfs(int now, int pre){
5     int cnt = 0;
6     bool ap = 0;
7     vis[now] = 1;
8     low[now] = dep[now] = (now==1 ? 0 : dep[pre]+1);
9
10    for (auto x : G[now]){
11        if (x==pre){
12            continue;
13        }
14        else if (vis[x]==0){
15            cnt++;
16            dfs(x, now);
17            low[now] = min(low[now], low[x]);
18            if (low[x]>=dep[now]) ap=1;
19        }
20        else{
21            low[now] = min(low[now], dep[x]);
22        }
23    }
24 }

```



```

22 |
23 |     if ((now==pre && cnt>=2) || (now!=pre && ap)){
24 |         AP.push_back(now);
25 |     }
26 | }

```

6.16 Tree Isomorphism

```

1 | #include <bits/stdc++.h>
2 | #pragma GCC optimize("O3,unroll-loops")
3 | #define fastio ios::sync_with_stdio(0), cin.tie(0), cout.tie
   | (0)
4 | #define dbg(x) cerr << #x << " = " << x << endl
5 | #define int long long
6 | using namespace std;
7 |
8 | // declare
9 | const int MAX_SIZE = 2e5+5;
10 | const int INF = 9e18;
11 | const int MOD = 1e9+7;
12 | const double EPS = 1e-6;
13 | typedef vector<vector<int>> Graph;
14 | typedef map<vector<int>, int> Hash;
15 |
16 | int n, a, b;
17 | int id1, id2;
18 | pair<int, int> c1, c2;
19 | vector<int> sz1(MAX_SIZE), sz2(MAX_SIZE);
20 | vector<int> we1(MAX_SIZE), we2(MAX_SIZE);
21 | Graph g1(MAX_SIZE), g2(MAX_SIZE);
22 | Hash m1, m2;
23 | int testcase=0;
24 |
25 | void centroid(Graph &g, vector<int> &s, vector<int> &w, pair<
   | int, int> &rec, int now, int pre){
26 |     s[now]=1;
27 |     w[now]=0;
28 |     for (auto x : g[now]){
29 |         if (x!=pre){
30 |             centroid(g, s, w, rec, x, now);
31 |             s[now]+=s[x];
32 |             w[now]=max(w[now], s[x]);
33 |         }
34 |     }
35 |
36 |     w[now]=max(w[now], n-s[now]);
37 |     if (w[now]<=n/2){
38 |         if (rec.first==0) rec.first=now;
39 |         else rec.second=now;
40 |     }
41 | }
42 |
43 | int dfs(Graph &g, Hash &m, int &id, int now, int pre){
44 |     vector<int> v;
45 |     for (auto x : g[now]){
46 |         if (x!=pre){
47 |             int add=dfs(g, m, id, x, now);
48 |             v.push_back(add);
49 |         }
50 |     }
51 |     sort(v.begin(), v.end());
52 |
53 |     if (m.find(v)!=m.end()){

```

```

54 |         return m[v];
55 |     }else{
56 |         m[v]++;id;
57 |         return id;
58 |     }
59 | }
60 |
61 |
62 | void solve1(){
63 |
64 |     // init
65 |     id1=0;
66 |     id2=0;
67 |     c1={0, 0};
68 |     c2={0, 0};
69 |     fill(sz1.begin(), sz1.begin()+n+1, 0);
70 |     fill(sz2.begin(), sz2.begin()+n+1, 0);
71 |     fill(we1.begin(), we1.begin()+n+1, 0);
72 |     fill(we2.begin(), we2.begin()+n+1, 0);
73 |     for (int i=1 ; i<=n ; i++){
74 |         g1[i].clear();
75 |         g2[i].clear();
76 |     }
77 |     m1.clear();
78 |     m2.clear();
79 |
80 |     // input
81 |     cin >> n;
82 |     for (int i=0 ; i<n-1 ; i++){
83 |         cin >> a >> b;
84 |         g1[a].push_back(b);
85 |         g1[b].push_back(a);
86 |     }
87 |     for (int i=0 ; i<n-1 ; i++){
88 |         cin >> a >> b;
89 |         g2[a].push_back(b);
90 |         g2[b].push_back(a);
91 |     }
92 |
93 |     // get tree centroid
94 |     centroid(g1, sz1, we1, c1, 1, 0);
95 |     centroid(g2, sz2, we2, c2, 1, 0);
96 |
97 |     // process
98 |     int res1=0, res2=0, res3=0;
99 |     if (c2.second!=0){
100 |         res1=dfs(g1, m1, id1, c1.first, 0);
101 |         m2=m1;
102 |         id2=id1;
103 |         res2=dfs(g2, m1, id1, c2.first, 0);
104 |         res3=dfs(g2, m2, id2, c2.second, 0);
105 |     }else if (c1.second!=0){
106 |         res1=dfs(g2, m1, id1, c2.first, 0);
107 |         m2=m1;
108 |         id2=id1;
109 |         res2=dfs(g1, m1, id1, c1.first, 0);
110 |         res3=dfs(g1, m2, id2, c1.second, 0);
111 |     }else{
112 |         res1=dfs(g1, m1, id1, c1.first, 0);
113 |         res2=dfs(g2, m1, id1, c2.first, 0);
114 |     }
115 |
116 |     // output
117 |     cout << (res1==res2 || res1==res3 ? "YES" : "NO") << endl
   | ;
118 |

```

```

119 |     return;
120 | }
121 |
122 | signed main(void){
123 |     fastio;
124 |
125 |     int t=1;
126 |     cin >> t;
127 |     while (t--){
128 |         solve1();
129 |     }
130 |     return 0;
131 | }

```

6.17 圓方樹

```

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

```

```

46 for(auto &ne:E[v]){
47     int i = EV[ne].to;
48     if(i == par) continue;
49     if(!dfn[i]){
50         S.push(ne);
51         tarjan(i, v, S);
52         child_s += 1;
53         low[v] = min(low[v], low[i]);
54
55         if(par >= 0 && low[i] >= dfn[v]){
56             vector<int> bcc;
57             int tmp;
58             do{
59                 tmp = S.top(); S.pop();
60                 if(!to_add[EV[tmp].fr]){
61                     to_add[EV[tmp].fr] = true;
62                     bcc.pb(EV[tmp].fr);
63                 }
64                 if(!to_add[EV[tmp].to]){
65                     to_add[EV[tmp].to] = true;
66                     bcc.pb(EV[tmp].to);
67                 }
68             }while(tmp != ne);
69             for(auto &j:bcc){
70                 to_add[j] = false;
71                 F[last_special_node].pb(j);
72                 F[j].pb(last_special_node);
73             }
74             last_special_node += 1;
75         }
76     }
77     else{
78         low[v] = min(low[v], dfn[i]);
79         if(dfn[i] < dfn[v]){ // edge i--v will be visited
80             // twice at here, but we only need one.
81             S.push(ne);
82         }
83     }
84 }
85
86 int dep[mxn], jmp[mxn][mxlg];
87 void dfs_lca(int v, int par, int depth){
88     dep[v] = depth;
89     for(auto &i:F[v]){
90         if(i == par) continue;
91         jmp[i][0] = v;
92         dfs_lca(i, v, depth + 1);
93     }
94 }
95
96 inline void build_lca(){
97     jmp[1][0] = 1;
98     dfs_lca(1, -1, 1);
99     lp(j, 1, mxlg){
100         lp(i, 1, mxn){
101             jmp[i][j] = jmp[jmp[i][j-1]][j-1];
102         }
103     }
104 }
105
106 inline int lca(int x, int y){
107     if(dep[x] < dep[y]){ swap(x, y); }
108
109     int diff = dep[x] - dep[y];
110     lp(j, 0, mxlg){

```

```

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

```

6.18 最大權閉合圖

```

1 /*
2 Problem:

```

```

3 Given w = [w_0, w_1, ..., 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 -> 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(mVn))
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 > Σw_i /
30         G.add(u, v, INF);
31     }
32     int cut = G.flow(S, T);
33     return sum - cut;

```

6.19 Theorem

- 任意圖
 - 不能有孤點 · 最大匹配 + 最小邊覆蓋 = n - 點覆蓋的補集是獨立集。最小點覆蓋 + 最大獨立集 = n
- 二分圖
 - 最小點覆蓋 = 最大匹配 = n - 最大獨立集
- 只有邊帶權的二分圖
 - w-vertex-cover (帶權點覆蓋): 每條邊的兩個連接點被選中的次數總和至少要是 w_e 。
 - w-weight matching (帶權匹配)
 - minimum vertex count of w-vertex-cover = maximum weight count of w-weight matching (一個點可以被選很多次 · 但邊不行)
- 點、邊都帶權的二分圖的定理
 - b-matching: 假設 v 的點權是 b_v · 那所有 v 的匹配邊 e 的權重都要滿足 $\sum w_e \leq b_v$ 。
 - The maximum w-weight of a b-matching equals the minimum b-weight of vertices in a w-vertex-cover.

7 Math

7.1 CRT m Coprime

```

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 }

```

7.2 CRT m Not Coprime

```

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;

```

```

23     m1=m1*m2/g;
24     a1=(a1*m1+m1)%m1;
25     return {1, {a1, m1}};
26 }

```

7.3 Josephus Problem

```

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

```

7.4 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 };

```

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

```

```

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

```

7.6 Lucas's Theorem

```

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

```

7.7 Matrix

```

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

```

```

26 }
27
28 Matrix pow(int p){
29     Matrix ret(n, n), mul = *this;
30     for (int i=0 ; i<n ; i++){
31         ret.arr[i][i] = 1;
32     }
33
34     for ( ; p ; p>>=1){
35         if (p&1) ret = ret*mul;
36         mul = mul*mul;
37     }
38
39     return ret;
40 }
41
42 int det(){
43     vector<vector<int>> arr = this->arr;
44     bool flag = false;
45     for (int i=0 ; i<n ; i++){
46         int target = -1;
47         for (int j=i ; j<n ; j++){
48             if (arr[j][i]){
49                 target = j;
50                 break;
51             }
52         }
53         if (target==-1) return 0;
54         if (i!=target){
55             swap(arr[i], arr[target]);
56             flag = !flag;
57         }
58
59         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
69     int ret = !flag ? 1 : MOD-1;
70     for (int i=0 ; i<n ; i++){
71         ret *= arr[i][i];
72         ret %= MOD;
73     }
74     return ret;
75 }
76 }
77 };

```

7.8 Matrix 01

```

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

```

```

8         m = _m;
9         arr.resize(n);
10     }
11
12     Matrix operator * (Matrix b){
13         Matrix b_t(b.m, b.n);
14         for (int i=0 ; i<b.n ; i++){
15             for (int j=0 ; j<b.m ; j++){
16                 b_t.arr[j][i] = b.arr[i][j];
17             }
18         }
19
20         Matrix ret(n, b.m);
21         for (int i=0 ; i<n ; i++){
22             for (int j=0 ; j<b.m ; j++){
23                 ret.arr[i][j] = ((arr[i]&b_t.arr[j]).count()
24                                     &1);
25             }
26         }
27         return ret;
28     }
29 }

```

7.9 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;

```

```

38     }
39
40     if (x==n-1) continue;
41     return 0;
42 }
43
44 return 1;
45 }

```

7.10 Pollard Rho

```

1 mt19937 seed(chrono::steady_clock::now().time_since_epoch().
  count());
2 int rnd(int l, int r){
3     return uniform_int_distribution<int>(l, r)(seed);
4 }
5
6 //  $O(n^{1/4})$  回傳 1 或自己的因數、記得先判斷  $n$  是不是質數
  (用 Miller-Rabin)
7 // c1670c
8 int Pollard_Rho(int n){
9     int s = 0, t = 0;
10    int c = rnd(1, n-1);
11
12    int step = 0, goal = 1;
13    int val = 1;
14
15    for (goal=1 ; ; goal<=1, s=t, val=1){
16        for (step=1 ; step<=goal ; step++){
17
18            t = ((__int128)t*t+c)%n;
19            val = ((__int128)val*abs(t-s)%n;
20
21            if ((step % 127) == 0){
22                int d = __gcd(val, n);
23                if (d>1) return d;
24            }
25        }
26
27        int d = __gcd(val, n);
28        if (d>1) return d;
29    }
30 }

```

7.11 Polynomial

```

1 struct Poly {
2     int len, deg;
3     int *a;
4     //  $len = 2^k \geq \text{the original length}$ 
5     Poly(): len(0), deg(0), a(nullptr) {}
6     Poly(int _n) {
7         len = 1;
8         deg = _n - 1;
9         while (len < _n) len <= 1;
10        a = (ll*) calloc(len, sizeof(ll));
11    }
12    Poly(int l, int d, int *b) {
13        len = 1;

```

```

14        deg = d;
15        a = b;
16    }
17    void resize(int _n) {
18        int len1 = 1;
19        while (len1 < _n) len1 <= 1;
20        int *res = (ll*) calloc(len1, sizeof(ll));
21        for (int i = 0; i < min(len, _n); i++) {
22            res[i] = a[i];
23        }
24        len = len1;
25        deg = _n - 1;
26        free(a);
27        a = res;
28    }
29    Poly& operator=(const Poly rhs) {
30        this->len = rhs.len;
31        this->deg = rhs.deg;
32        this->a = (ll*)realloc(this->a, sizeof(ll) * len);
33        copy(rhs.a, rhs.a + len, this->a);
34        return *this;
35    }
36    Poly operator*(Poly rhs) {
37        int l1 = this->len, l2 = rhs.len;
38        int d1 = this->deg, d2 = rhs.deg;
39        while (l1 > 0 and this->a[l1 - 1] == 0) l1--;
40        while (l2 > 0 and rhs.a[l2 - 1] == 0) l2--;
41        int l = 1;
42        while (l < max(l1 + l2 - 1, d1 + d2 + 1)) l <= 1;
43        int *x, *y, *res;
44        x = (ll*) calloc(l, sizeof(ll));
45        y = (ll*) calloc(l, sizeof(ll));
46        res = (ll*) calloc(l, sizeof(ll));
47        copy(this->a, this->a + l1, x);
48        copy(rhs.a, rhs.a + l2, y);
49        ntt.tran(l, x); ntt.tran(l, y);
50        FOR (i, 0, l - 1)
51            res[i] = x[i] * y[i] % mod;
52        ntt.tran(l, res, true);
53        free(x); free(y);
54        return Poly(l, d1 + d2, res);
55    }
56    Poly operator+(Poly rhs) {
57        int l1 = this->len, l2 = rhs.len;
58        int l = max(l1, l2);
59        Poly res;
60        res.len = l;
61        res.deg = max(this->deg, rhs.deg);
62        res.a = (ll*) calloc(l, sizeof(ll));
63        FOR (i, 0, l1 - 1) {
64            res.a[i] += this->a[i];
65            if (res.a[i] >= mod) res.a[i] -= mod;
66        }
67        FOR (i, 0, l2 - 1) {
68            res.a[i] += rhs.a[i];
69            if (res.a[i] >= mod) res.a[i] -= mod;
70        }
71        return res;
72    }
73    Poly operator-(Poly rhs) {
74        int l1 = this->len, l2 = rhs.len;
75        int l = max(l1, l2);
76        Poly res;
77        res.len = l;
78        res.deg = max(this->deg, rhs.deg);
79        res.a = (ll*) calloc(l, sizeof(ll));

```

```

80        FOR (i, 0, l1 - 1) {
81            res.a[i] += this->a[i];
82            if (res.a[i] >= mod) res.a[i] -= mod;
83        }
84        FOR (i, 0, l2 - 1) {
85            res.a[i] -= rhs.a[i];
86            if (res.a[i] < 0) res.a[i] += mod;
87        }
88        return res;
89    }
90    Poly operator*(const int rhs) {
91        Poly res;
92        res = *this;
93        FOR (i, 0, res.len - 1) {
94            res.a[i] = res.a[i] * rhs % mod;
95            if (res.a[i] < 0) res.a[i] += mod;
96        }
97        return res;
98    }
99    Poly(vector<int> f) {
100        int _n = f.size();
101        len = 1;
102        deg = _n - 1;
103        while (len < _n) len <= 1;
104        a = (ll*) calloc(len, sizeof(ll));
105        FOR (i, 0, deg) a[i] = f[i];
106    }
107    Poly derivative() {
108        Poly g(this->deg);
109        FOR (i, 1, this->deg) {
110            g.a[i - 1] = this->a[i] * i % mod;
111        }
112        return g;
113    }
114    Poly integral() {
115        Poly g(this->deg + 2);
116        FOR (i, 0, this->deg) {
117            g.a[i + 1] = this->a[i] * ::inv(i + 1) % mod;
118        }
119        return g;
120    }
121    Poly inv(int len1 = -1) {
122        if (len1 == -1) len1 = this->len;
123        Poly g(1); g.a[0] = ::inv(a[0]);
124        for (int l = 1; l < len1; l <= 1) {
125            Poly t; t = *this;
126            t.resize(l < 1);
127            t = g * t;
128            t.resize(l < 1);
129            Poly g1 = g * 2 - t;
130            swap(g, g1);
131        }
132        return g;
133    }
134    Poly ln(int len1 = -1) {
135        if (len1 == -1) len1 = this->len;
136        auto g = *this;
137        auto x = g.derivative() * g.inv(len1);
138        x.resize(len1);
139        x = x.integral();
140        x.resize(len1);
141        return x;
142    }
143    Poly exp() {
144        Poly g(1);
145        g.a[0] = 1;

```

```

146     for (int l = 1; l < len; l <= 1) {
147         Poly t, g1; t = *this;
148         t.resize(l < 1); t.a[0]++;
149         g1 = (t - g.ln(1 < 1)) * g;
150         g1.resize(l < 1);
151         swap(g, g1);
152     }
153     return g;
154 }
155 Poly pow(ll n) {
156     Poly &a = *this;
157     int i = 0;
158     while (i <= a.deg and a.a[i] == 0) i++;
159     if (i and (n > a.deg or n * i > a.deg)) return Poly(a
160         .deg + 1);
161     if (i == a.deg + 1) {
162         Poly res(a.deg + 1);
163         res.a[0] = 1;
164         return res;
165     }
166     Poly b(a.deg - i + 1);
167     int inv1 = ::inv(a.a[i]);
168     FOR (j, 0, b.deg)
169         b.a[j] = a.a[j + i] * inv1 % mod;
170     Poly res1 = (b.ln() * (n % mod)).exp() * (::power(a.a
171         [i], n));
172     Poly res2(a.deg + 1);
173     FOR (j, 0, min((ll)(res1.deg), (ll)(a.deg - n * i)))
174         res2.a[j + n * i] = res1.a[j];
175     return res2;
176 }
177 };

```

7.12 數論分塊

```

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

```

7.13 最大質因數

```

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

```

7.14 歐拉公式

```

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

```

7.15 Burnside's Lemma

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

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

7.16 Catalan Number

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

7.17 Matrix Tree Theorem

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

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

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。

7.18 Stirling's formula

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

7.19 Theorem

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

7.20 二元一次方程式

$\begin{cases} ax + by = c \\ dx + ey = f \end{cases} = \begin{cases} x = \frac{ed-bf}{ad-bc} \\ y = \frac{af-ec}{ad-bc} \end{cases}$
若 $x = \frac{0}{0}$ 且 $y = \frac{0}{0}$ · 則代表無限多組解。若 $x = \frac{*}{0}$ 且 $y = \frac{*}{0}$ · 則代表無解。

7.21 歐拉定理

若 a, m 互質 · 則：

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

若 a, m 不互質 · 則：

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

7.22 錯排公式

錯排公式：(n 個人中 · 每個人皆不再原來位置的組合數)

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

8 String

8.1 Hash

```

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

```

8.2 KMP

```

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

```

8.3 Manacher

```

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

```

8.4 Min Rotation

```

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

```

8.5 Suffix Array

```

1 // 注意，當 |s|=1 時，lcp 不會有值，務必測試 |s|=1 的 case
2 struct SuffixArray {
3     string s;
4     vector<int> sa, lcp;
5
6     // 69ced9
7     SuffixArray(string _s, int lim = 256) {
8         s = _s;
9         int n = s.size()+1, k = 0, a, b;
10        vector<int> x(s.begin(), s.end()), y(n), ws(max(n,
11        lim)), rank(n);
12        x.push_back(0);
13        sa = lcp = y;
14        iota(sa.begin(), sa.end(), 0);
15        for (int j=0, p=0 ; p<n ; j=max(1LL, j*2), lim=p) {
16            p = j;
17            iota(y.begin(), y.end(), n-j);
18            for (int i=0 ; i<n ; i++) if (sa[i] >= j) y[p++]
19            = sa[i] - j;
20            fill(ws.begin(), ws.end(), 0);
21            for (int i=0 ; i<n ; i++) ws[x[i]]++;

```

```

22        for (int i=1 ; i<lim ; i++) ws[i] += ws[i - 1];
23        for (int i = n; i--;) sa[--ws[x[i]]] = y[i];
24        swap(x, y), p = 1, x[sa[0]] = 0;
25        for (int i=1 ; i<n ; i++){
26            a = sa[i - 1];
27            b = sa[i];
28            x[b] = (y[a] == y[b] && y[a + j] == y[b + j])
29                ? p - 1 : p++;
30        }
31
32        for (int i=1 ; i<n ; i++) rank[sa[i]] = i;
33        for (int i=0, j ; i<n-1 ; lcp[rank[i++]]=k)
34            for (k && k--, j=sa[rank[i]-1] ; i+k<s.size() &&
35                j+k<s.size() && s[i+k]==s[j+k] ; k++);
36        sa.erase(sa.begin());
37        lcp.erase(lcp.begin(), lcp.begin()+2);
38    }
39
40    // f49583
41    vector<int> pos; // pos[i] = i 這個值在 pos 的哪個地方
42    SparseTable st;
43    void init_lcp(){
44        pos.resize(sa.size());
45        for (int i=0 ; i<sa.size() ; i++){
46            pos[sa[i]] = i;
47        }
48        if (lcp.size()){
49            st.build(lcp);
50        }
51
52        // 用之前記得 init
53        // 回傳 [l1, r1] 跟 [l2, r2] 的 lcp，0-based
54        int get_lcp(int l1, int r1, int l2, int r2){
55            int pos_1 = pos[l1], len_1 = r1-l1+1;
56            int pos_2 = pos[l2], len_2 = r2-l2+1;
57            if (pos_1>pos_2){
58                swap(pos_1, pos_2);
59                swap(len_1, len_2);
60            }
61
62            if (l1==l2){
63                return min(len_1, len_2);
64            }else{
65                return min({st.query(pos_1, pos_2), len_1, len_2
66                });
67            }
68        }
69
70        // 檢查 [l1, r1] 跟 [l2, r2] 的大小關係，0-based
71        // 如果前者小於後者，就回傳 <0，相等就回傳 =0，否則回傳
72        >0
73        // 5b8db0
74        int substring_cmp(int l1, int r1, int l2, int r2){
75            int len_1 = r1-l1+1;
76            int len_2 = r2-l2+1;
77            int res = get_lcp(l1, r1, l2, r2);
78
79            if (res<len_1 && res<len_2){
80                return s[l1+res]-s[l2+res];
81            }else if (len_1==res && len_2==res){
82                // 如果不需要以 index 作為次要排序參數，這裡要回
83                傳 0
84                return l1-l2;

```



```

80     }else{
81         return len_1==res ? -1 : 1;
82     }
83 }
84
85 // 對於位置在 <=p 的後綴，找離他左邊/右邊最接近位置 >p 的
    後綴的 lcp · 0-based
86 // pre[i] = s[i] 離他左邊最接近位置 >p 的後綴的 lcp · 0-
    based
87 // suf[i] = s[i] 離他右邊最接近位置 >p 的後綴的 lcp · 0-
    based
88 // da12fa
89 pair<vector<int>, vector<int>> get_left_and_right_lcp(int
    p){
90     vector<int> pre(p+1);
91     vector<int> suf(p+1);
92
93     { // build pre
94         int now = 0;
95         for (int i=0 ; i<s.size() ; i++){
96             if (sa[i]<=p){
97                 pre[sa[i]] = now;
98                 if (i<lcp.size()) now = min(now, lcp[i]);
99             }else{
100                 if (i<lcp.size()) now = lcp[i];
101             }
102         }
103     }
104     { // build suf
105         int now = 0;
106         for (int i=s.size()-1 ; i>=0 ; i--){
107             if (sa[i]<=p){
108                 suf[sa[i]] = now;
109                 if (i-1>=0) now = min(now, lcp[i-1]);
110             }else{
111                 if (i-1>=0) now = lcp[i-1];
112             }
113         }
114     }
115
116     return {pre, suf};
117 }
118 };

```

```

16         rr = i+j;
17     }
18 }
19
20 ret[0] = s.size();
21 return ret;
22 }

```

8.7 k-th Substring1

```

1 // 回傳 s 所有子字串 (完全不同) 中 · 第 k 大的
2 string k_th_substring(string &s, int k){
3     int n = s.size();
4     SuffixArray sa(s);
5     sa.init_lcp();
6
7     int prePrefix = 0, nowRank = 0;
8     for (int i=0 ; i<n ; i++){
9         int len = n-sa[i];
10        int add = len-prePrefix;
11
12        if (nowRank+add>=k){
13            return s.substr(sa[i], prePrefix+k-nowRank);
14        }
15
16        prePrefix = sa.lcp[i];
17        nowRank += add;
18    }
19 }

```

8.6 Z Algorithm

```

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

```