Contents

1 Misc

1.1 Default Code [24a798]

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

void solve(){

}

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

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

    return 0;
}
```

1.2 Run

```python
from os import *

f = "pA"

while 1:
    i = input("input: ")
    system("clear")
    p = listdir(".")
    if i != "":
        f = i
    print(f"file = {f}")
    if system(f"g++ {f}.cpp -std=c++17 -Wall -Wextra -Wshadow
         -O2 -D LOCAL -g -fsanitize=undefined,address -o {f}
        "):
        print("CE")
        continue

    for x in sorted(p):
        if f in x and ".in" in x:
            print(x)
            if system(f"./{f} < {x}"):
                print("RE")
            print()
```

1.3 Custom Set PQ Sort [d4df55]

```cpp
// 所有自訂的結構體，務必檢查相等的 case，給所有元素一個排序
    的依據
struct my_struct{
    int val;
    my_struct(int _val) : val(_val) {}
};

auto cmp = [](my_struct a, my_struct b) {
    return a.val > b.val;
};

set<my_struct, decltype(cmp)> ss({1, 2, 3}, cmp);
priority_queue<my_struct, vector<my_struct>, decltype(cmp)>
    pq(cmp, {1, 2, 3});
map<my_struct, my_struct, decltype(cmp)> mp({{1, 4}, {2, 5},
    {3, 6}}, cmp);
```

1.4 Dynamic Bitset [c78aa8]

```cpp
const int MAXN = 2e5 + 5;
template <int len = 1>
void solve(int n) {
    if (n > len) {
        solve<min(len*2, MAXN)>(n);
        return;
    }
    bitset<len> a;
}
```

1.5 Enumerate Subset [a13e46]

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

1.6 Fast Input [6f8879]

```cpp
// fast IO
// 6f8879
inline char readchar(){
    static char buffer[BUFSIZ], * now = buffer + BUFSIZ, *
        end = buffer + BUFSIZ;
    if (now == end)
    {
        if (end < buffer + BUFSIZ)
            return EOF;
        end = (buffer + fread(buffer, 1, BUFSIZ, stdin));
        now = buffer;
    }
    return *now++;
}
inline int nextint(){
    int x = 0, c = readchar(), neg = false;
    while(('0' > c || c > '9') && c!='-' && c!=EOF) c =
        readchar();
    if(c == '-') neg = true, c = readchar();
    while('0' <= c && c <= '9') x = (x<<3) + (x<<1) + (c^'0')
        , c = readchar();
    if(neg) x = -x;
    return x; // returns 0 if EOF
}
```

1.7 OEIS [ec45dc]

```cpp
// 若一個線性遞迴有 k 項，給他恰好 2*k 個項可以求出線性遞迴
// f915c2
template <typename T>
vector<T> BerlekampMassey(vector<T> a) {
    auto scalarProduct = [](vector<T> v, T c) {
        for (T &x: v) x *= c;
        return v;
    };
    vector<T> s, best;
    int bestPos = 0;
    for (int i=0 ; i<a.size() ; i++){
        T error = a[i];
        for (int j=0 ; j<s.size() ; j++) error -= s[j] * a[i
            -1-j];
        if (error == 0) continue;
        if (s.empty()) {
            s.resize(i + 1);
            bestPos = i;
            best.push_back(1 / error);
            continue;
        }
        vector<T> fix = scalarProduct(best, error);
        fix.insert(fix.begin(), i - bestPos - 1, 0);
        if (fix.size() >= s.size()) {
            best = scalarProduct(s, - 1 / error);
            best.insert(best.begin(), 1 / error);
            bestPos = i;
            s.resize(fix.size());
        }
        for (int j = 0; j < fix.size(); j++) s[j] += fix[j];
    }
    reverse(s.begin(), s.end());
    return s;
}
```

1.8 Pragma [09d13e]

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

1.9 Xor Basis [840136]

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

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

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

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

```
}
```

1.10 random int [9cc603]

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

1.11 OEIS

```python
from fractions import Fraction

def BerlekampMassey(a: list[Fraction]) -> list[Fraction]:
    def scale(v: list[Fraction], c: Fraction) -> list[Fraction]:
        return [x * c for x in v]

    s: list[Fraction] = []
    best: list[Fraction] = []
    bestPos = 0

    for i in range(len(a)):
        error: Fraction = a[i]
        for j in range(len(s)):
            error -= s[j] * a[i - 1 - j]
        if error == 0:
            continue

        if not s:
            s = [Fraction(0)] * (i + 1)
            bestPos = i
            best = [Fraction(1, error)]
            continue

        fix = scale(best, error)
        fix = [Fraction(0)] * (i - bestPos - 1) + fix

        if len(fix) >= len(s):
            best = scale(s, Fraction(-1, error))
            best.insert(0, Fraction(1, error))
            bestPos = i
            if len(s) < len(fix):
                s += [Fraction(0)] * (len(fix) - len(s))

        for j in range(len(fix)):
            s[j] += fix[j]

    return list(reversed(s))

n = int(input())
l = list(map(Fraction, input().split()))
for i in range(len(l)):
    coeffs = BerlekampMassey(l[:i+1])
    for x in coeffs:
        print(x, end=" ")
    print()
```

1.12 Python

```python
# Decimal
from decimal import *
getcontext().prec = 6

# system setting
sys.setrecursionlimit(100000)
sys.set_int_max_str_digits(10000)

# turtle
from turtle import *

N = 3000000010
setworldcoordinates(-N, -N, N, N)
hideturtle()
speed(100)

def draw_line(a, b, c, d):
    teleport(a, b)
    goto(c, d)

def write_dot(x, y, text, diff=1): # diff = 文字的偏移
    teleport(x, y)
    dot(5, "red")

    teleport(x+N/100*diff, y+N/100*diff)
    write(text, font=("Arial", 5, "bold"))

# usage
draw_line(*a[i], *(a[i-1]))
write_dot(*a[i], str(a[i]))
# 以自己左方 100 單位為圓心向前畫一個 90 度的弧 / 正五邊形 (參數可以是負的)
circle(100, extent = 90)
circle(100, steps = 5)
left(90) # 左轉 90 度
fd(-100) # 倒退 100 單位

# OOP
class Point:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def __add__(self, o): # use dir(int) to know operator name
        return Point(self.x+o.x, self.y+o.y)

    @property
    def distance(self):
        return (self.x**2 + self.y**2)**(0.5)

a = Point(3, 4)
print(a.distance)

# Fraction
from fractions import Fraction
a = Fraction(Decimal(1.1))
a.numerator # 分子
a.denominator # 分母
```

1.13 diff

```
set -e
g++ ac.cpp -o ac
g++ wa.cpp -o wa
for ((i=0;;i++))
do
    echo "$i"
    python3 gen.py > input
    ./ac < input > ac.out
    ./wa < input > wa.out
    diff ac.out wa.out || break
done
```

1.14 disable ASLR

```
# Disable randomization of memory addresses
setarch `uname -m` -R ./yourProgram
setarch $(uname -m) -R
```

1.15 hash command

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

1.16 hash windows

```python
def get_hash(path):
    from subprocess import run, PIPE
    from hashlib import md5

    p = run(
        ["cpp", "-dD", "-P", "-fpreprocessed", path],
        stdout = PIPE,
        stderr = PIPE,
        text = True
    )

    if p.returncode != 0:
        raise RuntimeError(p.stderr)

    s = ''.join(p.stdout.split())
    ret = md5(s.encode()).hexdigest()
    return ret[:6]

print(get_hash("Suffix_Array.cpp"))
```

2 Convolution

2.1 FFT [16591a]

```cpp
typedef complex<double> cd;

// 778272
void FFT(vector<cd> &a) {
    int n = a.size(), L = 31-__builtin_clz(n);
    vector<complex<long double>> R(2, 1);
    vector<cd> rt(2, 1);
    for (int k=2 ; k<n ; k*=2){
        R.resize(n), rt.resize(n);
        auto x = polar(1.0L, acos(-1.0L) / k);
        for (int i=k ; i<2*k ; i++) rt[i] = R[i] = (i&1 ? R[i/2]*x : R[i/2]);
    }

    vector<int> rev(n);
    for (int i=0 ; i<n ; i++) rev[i] = (rev[i/2] | (i&1)<<L) /2;
    for (int i=0 ; i<n ; i++) if (i<rev[i]) swap(a[i], a[rev[i]]);
    for (int k=1 ; k<n ; k*=2){
```

```
        for (int i=0 ; i<n ; i+=2*k){
            for (int j=0 ; j<k ; j++){
                // cd z = rt[j+k] * a[i+j+k];
                auto x = (double *)&rt[j+k], y = (double *)&a[i+j+k];
                cd z(x[0]*y[0] - x[1]*y[1], x[0]*y[1] + x[1]*y[0]);
                a[i+j+k] = a[i+j]-z;
                a[i+j] += z;
            }
        }
    }
}

// 192528
vector<double> PolyMul(const vector<double> a, const vector<double> b){
    if (a.empty() || b.empty()) return {};
    vector<double> res(a.size()+b.size()-1);
    int L = 32 - __builtin_clz(res.size()), n = 1<<L;
    vector<cd> in(n), out(n);
    copy(a.begin(), a.end(), in.begin());
    for (int i=0 ; i<b.size() ; i++) in[i].imag(b[i]);
    FFT(in);
    for (cd& x : in) x *= x;
    for (int i=0 ; i<n ; i++) out[i] = in[-i & (n - 1)] - conj(in[i]);
    FFT(out);
    for (int i=0 ; i<res.size() ; i++) res[i] = imag(out[i]) / (4 * n);
    return res;
}
```

2.2 FFT any mod [412000]

```

// n=524288, ~=400ms
const int MOD = 998244353;
typedef complex<double> cd;

// bb6d94
void FFT(vector<cd> &a) {
    int n = a.size(), L = 31-__builtin_clz(n);
    vector<complex<long double>> R(2, 1);
    vector<cd> rt(2, 1);
    for (int k=2 ; k<n ; k*=2){
        R.resize(n);
        rt.resize(n);
        auto x = polar(1.0L, acos(-1.0L) / k);
        for (int i=k ; i<2*k ; i++){
            rt[i] = R[i] = (i&1 ? R[i/2]*x : R[i/2]);
        }
    }

    vector<int> rev(n);
    for (int i=0 ; i<n ; i++) rev[i] = (rev[i/2] | (i&1)<<L) /2;
    for (int i=0 ; i<n ; i++) if (i<rev[i]) swap(a[i], a[rev[i]]);
    for (int k=1 ; k<n ; k*=2){
        for (int i=0 ; i<n ; i+=2*k){
            for (int j=0 ; j<k ; j++){
                auto x = (double *)&rt[j+k];
                auto y = (double *)&a[i+j+k];
                cd z(x[0]*y[0] - x[1]*y[1], x[0]*y[1] + x[1]*y[0]);
                a[i+j+k] = a[i+j]-z;
                a[i+j] += z;
            }
        }
    }
}

// c3dcf6
vector<int> PolyMul(vector<int> a, vector<int> b){
    if (a.empty() || b.empty()) return {};
    vector<int> res(a.size()+b.size()-1);
    int B = 32-__builtin_clz(res.size()), n = (1<<B), cut = (int)(sqrt(MOD));
    vector<cd> L(n), R(n), outs(n), outl(n);

    for (int i=0 ; i<a.size() ; i++) L[i] = cd((int) a[i]/cut, (int)a[i]%cut);
    for (int i=0 ; i<b.size() ; i++) R[i] = cd((int) b[i]/cut, (int)b[i]%cut);
    FFT(L), FFT(R);
    for (int i=0 ; i<n ; i++){
        int j = -i&(n-1);
        outl[j] = (L[i]+conj(L[j])) * R[i]/(2.0*n);
        outs[j] = (L[i]-conj(L[j])) * R[i]/(2.0*n)/1i;
    }
    FFT(outl), FFT(outs);
    for (int i=0 ; i<res.size() ; i++){
        int av = (int)(real(outl[i])+0.5), cv = (int)(imag(outs[i])+0.5);
        int bv = (int)(imag(outl[i])+0.5) + (int)(real(outs[i])+0.5);
        res[i] = ((av%MOD*cut+bv) % MOD*cut+cv) % MOD;
    }
    return res;
}
```

2.3 FWT [e50788]

```
// 已經把 mint 刪掉，需要增加註解
vector<int> xor_convolution(vector<int> a, vector<int> b, int k) {
    if (k==0) return vector<int>{a[0]*b[0]};
    vector<int> aa(1 << (k - 1)), bb(1 << (k - 1));
    for (int i=0 ; i<(1<<(k-1)) ; i++){
        aa[i] = a[i] + a[i + (1 << (k - 1))];
        bb[i] = b[i] + b[i + (1 << (k - 1))];
    }
    vector<int> X = xor_convolution(aa, bb, k - 1);
    for (int i=0 ; i<(1<<(k-1)) ; i++){
        aa[i] = a[i] - a[i + (1 << (k - 1))];
        bb[i] = b[i] - b[i + (1 << (k - 1))];
    }
    vector<int> Y = xor_convolution(aa, bb, k - 1);
    vector<int> c(1 << k);
    for (int i=0 ; i<(1<<(k-1)) ; i++){
        c[i                ] = (X[i] + Y[i]) / 2;
        c[i + (1 << (k - 1))] = (X[i] - Y[i]) / 2;
    }
    return c;
};
```

2.4 Min Convolution Concave Concave [ffb28d]

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

2.5 NTT mod 998244353 [f9fed4]

```

const int MOD = (119 << 23) + 1, ROOT = 62; // = 998244353
// For p < 2^30 there is also e.g. 5 << 25, 7 << 26, 479 << 21
// and 483 << 21 (same root). The last two are > 10^9.

// e169db
void NTT(vector<int> &a) {
    int n = a.size();
    int L = 31-__builtin_clz(n);
    vector<int> rt(2, 1);
    for (int k=2, s=2 ; k<n ; k*=2, s++){
        rt.resize(n);
        int z[] = {1, qp(ROOT, MOD>>s)};
        for (int i=k ; i<2*k ; i++) rt[i] = rt[i/2]*z[i&1]%MOD;
    }

    vector<int> rev(n);
    for (int i=0 ; i<n ; i++) rev[i] = (rev[i/2]|(i&1)<<L)/2;
    for (int i=0 ; i<n ; i++){
        if (i<rev[i]) swap(a[i], a[rev[i]]);
    }

    for (int k=1 ; k<n ; k*=2){
        for (int i=0 ; i<n ; i+=2*k){
            for (int j=0 ; j<k ; j++){
                int z = rt[j+k]*a[i+j+k]%MOD, &ai = a[i+j];
                a[i+j+k] = ai-z+(z>ai ? MOD : 0);
                ai += (ai+z>=MOD ? z-MOD : z);
            }
        }
    }
}

// 290957
vector<int> polyMul(vector<int> &a, vector<int> &b){
    if (a.empty() || b.empty()) return {};
    int s = a.size()+b.size()-1, B = 32-__builtin_clz(s), n = 1<<B;
    int inv = qp(n, MOD-2);
    vector<int> L(a), R(b), out(n);
    L.resize(n), R.resize(n);
    NTT(L), NTT(R);
    for (int i=0 ; i<n ; i++) out[-i&(n-1)] = L[i]*R[i]%MOD*inv%MOD;
    NTT(out);
    out.resize(s);
    return out;
}
```

3 Data-Structure

3.1 BIT [7ef3a9]

```cpp
vector<int> BIT(MAX_SIZE);

// const int MAX_N = (1<<20)
int k_th(int k){ // 回傳 BIT 中第 k 小的元素 ( based-1 )
    int res = 0;
    for (int i=MAX_N>>1 ; i>=1 ; i>>=1)
        if (BIT[res+i]<k)
            k -= BIT[res+=i];
    return res+1;
}
```

3.2 Disjoint Set Persistent [447002]

```cpp
struct Persistent_Disjoint_Set{
    Persistent_Segment_Tree arr, sz;

    void init(int n){
        arr.init(n);
        vector<int> v1;
        for (int i=0 ; i<n ; i++){
            v1.push_back(i);
        }
        arr.build(v1, 0);

        sz.init(n);
        vector<int> v2;
        for (int i=0 ; i<n ; i++){
            v2.push_back(1);
        }
        sz.build(v2, 0);
    }

    int find(int a){
        int res = arr.query_version(a, a+1, arr.version.size
            ()-1).val;
        if (res==a) return a;
        return find(res);
    }

    bool unite(int a, int b){
        a = find(a);
        b = find(b);

        if (a!=b){

            int sz1 = sz.query_version(a, a+1, arr.version.
                size()-1).val;
            int sz2 = sz.query_version(b, b+1, arr.version.
                size()-1).val;

            if (sz1<sz2){
                arr.update_version(a, b, arr.version.size()
                    -1);
                sz.update_version(b, sz1+sz2, arr.version.
                    size()-1);
            }else{
                arr.update_version(b, a, arr.version.size()
                    -1);
                sz.update_version(a, sz1+sz2, arr.version.
                    size()-1);
            }
            return true;
        }
        return false;
    }
};
```

3.3 PBDS GP Hash Table [866cf6]

```cpp
#include <ext/pb_ds/assoc_container.hpp>
using namespace __gnu_pbds;
typedef tree<int,null_type,less<int>,rb_tree_tag,
    tree_order_statistics_node_update> order_set;
struct custom_hash {
    static uint64_t splitmix64(uint64_t x) {
        // http://xorshift.di.unimi.it/splitmix64.c
        x += 0x9e3779b97f4a7c15;
        x = (x ^ (x >> 30)) * 0xbf58476d1ce4e5b9;
        x = (x ^ (x >> 27)) * 0x94d049bb133111eb;
        return x ^ (x >> 31);
    }

    size_t operator()(uint64_t x) const {
        static const uint64_t FIXED_RANDOM = chrono::
            steady_clock::now().time_since_epoch().count();
        return splitmix64(x + FIXED_RANDOM);
    }
};

gp_hash_table<int, int, custom_hash> ss;
```

3.4 PBDS Order Set [231774]

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

3.5 Segment Tree Add Set [bb1898]

```cpp
// [ll, rr), based-0
// 使用前記得 init(陣列大小), build(陣列名稱)
// add(ll, rr): 區間修改
// set(ll, rr): 區間賦值
// query(ll, rr): 區間求和 / 求最大值
struct SegmentTree{
    struct node{
        int add_tag = 0;
        int set_tag = 0;
        int sum = 0;
        int ma = 0;
    };

    vector<node> arr;

    SegmentTree(int n){
        arr.resize(n<<2);
    }

    node pull(node A, node B){
        node C;
        C.sum = A.sum+B.sum;
        C.ma = max(A.ma, B.ma);
        return C;
    }

    // cce0c8
    void push(int idx, int ll, int rr){
        if (arr[idx].set_tag!=0){
            arr[idx].sum = (rr-ll)*arr[idx].set_tag;
            arr[idx].ma = arr[idx].set_tag;
            if (rr-ll>1){
                arr[idx*2+1].add_tag = 0;
                arr[idx*2+1].set_tag = arr[idx].set_tag;
                arr[idx*2+2].add_tag = 0;
                arr[idx*2+2].set_tag = arr[idx].set_tag;
            }
            arr[idx].set_tag = 0;
        }
        if (arr[idx].add_tag!=0){
            arr[idx].sum += (rr-ll)*arr[idx].add_tag;
            arr[idx].ma += arr[idx].add_tag;
            if (rr-ll>1){
                arr[idx*2+1].add_tag += arr[idx].add_tag;
                arr[idx*2+2].add_tag += arr[idx].add_tag;
            }
            arr[idx].add_tag = 0;
        }
    }

    void build(vector<int> &v, int idx = 0, int ll = 0, int
         rr = n){
        if (rr-ll==1){
            arr[idx].sum = v[ll];
            arr[idx].ma = v[ll];
        }else{
            int mid = (ll+rr)/2;
            build(v, idx*2+1, ll, mid);
            build(v, idx*2+2, mid, rr);
            arr[idx] = pull(arr[idx*2+1], arr[idx*2+2]);
        }
    }

    void add(int ql, int qr, int val, int idx = 0, int ll =
         0, int rr =n){
        push(idx, ll, rr);
        if (rr<=ql || qr<=ll) return;
        if (ql<=ll && rr<=qr){
            arr[idx].add_tag += val;
            push(idx, ll, rr);
            return;
        }
        int mid = (ll+rr)/2;
        add(ql, qr, val, idx*2+1, ll, mid);
        add(ql, qr, val, idx*2+2, mid, rr);
        arr[idx]=pull(arr[idx*2+1], arr[idx*2+2]);
    }

    void set(int ql, int qr, int val, int idx=0, int ll=0,
         int rr=n){
        push(idx, ll, rr);
        if (rr<=ql || qr<=ll) return;
        if (ql<=ll && rr<=qr){
            arr[idx].add_tag = 0;
            arr[idx].set_tag = val;
            push(idx, ll, rr);
```

```
            return;
        }
        int mid = (ll+rr)/2;
        set(ql, qr, val, idx*2+1, ll, mid);
        set(ql, qr, val, idx*2+2, mid, rr);
        arr[idx] = pull(arr[idx*2+1], arr[idx*2+2]);
    }

    node query(int ql, int qr, int idx = 0, int ll = 0, int
         rr = n){
        push(idx, ll, rr);
        if (rr<=ql || qr<=ll) return node();
        if (ql<=ll && rr<=qr) return arr[idx];

        int mid = (ll+rr)/2;
        return pull(query(ql, qr, idx*2+1, ll, mid), query(ql
             , qr, idx*2+2, mid, rr));
    }
} ST;
```

3.6 Segment Tree Li Chao Line [45b8ba]

```
// 全部都是 0-based
// 宣告：LC_Segment_Tree st(n);
// 函式：
// update({a, b})：插入一條 y=ax+b 的全域直線
// query(x)：查詢所有直線在位置 x 的最小值
const int MAX_V = 1e6+10; // 值域最大值

struct LC_Segment_Tree{
    struct Node{ // y = ax+b
        int a = 0;
        int b = INF;

        int y(int x){
            return a*x+b;
        }
    };
    vector<Node> arr;

    LC_Segment_Tree(int n = 0){
        arr.resize(4*n);
    }

    void update(Node val, int idx = 0, int ll = 0, int rr =
         MAX_V){
        if (rr-ll==0) return;
        if (rr-ll==1){
            if (val.y(ll)<arr[idx].y(ll)){
                arr[idx] = val;
            }
            return;
        }

        int mid = (ll+rr)/2;
        if (arr[idx].a > val.a) swap(arr[idx], val); // 原本
             的線斜率要比較小
        if (arr[idx].y(mid) < val.y(mid)){ // 交點在左邊
            update(val, idx*2+1, ll, mid);
        }else{ // 交點在右邊
            swap(arr[idx], val); // 在左子樹中，新線比舊線還
                 要好
            update(val, idx*2+2, mid, rr);
        }
        return;
    }

    int query(int x, int idx = 0, int ll = 0, int rr = MAX_V)
         {
        if (rr-ll==0) return INF;
        if (rr-ll==1){
            return arr[idx].y(ll);
        }

        int mid = (ll+rr)/2;
        if (x<mid){
            return min(arr[idx].y(x), query(x, idx*2+1, ll,
                 mid));
        }else{
            return min(arr[idx].y(x), query(x, idx*2+2, mid,
                 rr));
        }
    }
};
```

3.7 Segment Tree Li Chao Segment [2cb0a4]

```
// 全部都是 0-based
// 宣告：LC_Segment_Tree st(n);
// 函式：
// update_segment({a, b}, ql, qr)：在 [ql, qr) 插入一條 y=ax+
     b 的線段
// query(x)：查詢所有直線在位置 x 的最小值
const int MAX_V = 1e6+10; // 值域最大值

struct LC_Segment_Tree{
    struct Node{ // y = ax+b
        int a = 0;
        int b = INF;

        int y(int x){
            return a*x+b;
        }
    };
    vector<Node> arr;

    LC_Segment_Tree(int n = 0){
        arr.resize(4*n);
    }

    void update(Node val, int idx = 0, int ll = 0, int rr =
         MAX_V){
        if (rr-ll==0) return;
        if (rr-ll<=1){
            if (val.y(ll)<arr[idx].y(ll)){
                arr[idx] = val;
            }
            return;
        }

        int mid = (ll+rr)/2;
        if (arr[idx].a > val.a) swap(arr[idx], val); // 原本
             的線斜率要比較小
        if (arr[idx].y(mid) < val.y(mid)){ // 交點在左邊
            update(val, idx*2+1, ll, mid);
        }else{ // 交點在右邊
            swap(arr[idx], val); // 在左子樹中，新線比舊線還
                 要好
            update(val, idx*2+2, mid, rr);
        }
        return;
    }

    // 在 [ql, qr) 加上一條 val 的線段
    void update_segment(Node val, int ql, int qr, int idx =
         0, int ll = 0, int rr = MAX_V){
        if (rr-ll==0) return;
        if (rr<=ql || qr<=ll) return;
        if (ql<=ll && rr<=qr){
            update(val, idx, ll, rr);
            return;
        }

        int mid = (ll+rr)/2;
        update_segment(val, ql, qr, idx*2+1, ll, mid);
        update_segment(val, ql, qr, idx*2+2, mid, rr);
        return;
    }

    int query(int x, int idx = 0, int ll = 0, int rr = MAX_V)
         {
        if (rr-ll==0) return INF;
        if (rr-ll==1){
            return arr[idx].y(ll);
        }

        int mid = (ll+rr)/2;
        if (x<mid){
            return min(arr[idx].y(x), query(x, idx*2+1, ll,
                 mid));
        }else{
            return min(arr[idx].y(x), query(x, idx*2+2, mid,
                 rr));
        }
    }
};
```

3.8 Segment Tree Persistent [3b5aa9]

```
// 全部都是 0-based
// Persistent_Segment_Tree st(n+q);
// st.build(v, 0);
// 函式：
// update_version(pos, val, ver)：對版本 ver 的 pos 位置改成
     val
// query_version(ql, qr, ver)：對版本 ver 查詢 [ql, qr) 的區
     間和
// clone_version(ver)：複製版本 ver 到最新的版本
struct Persistent_Segment_Tree{
    int node_cnt = 0;
    struct Node{
        int lc = -1;
        int rc = -1;
        int val = 0;
    };
    vector<Node> arr;
    vector<int> version;

    Persistent_Segment_Tree(int sz){
        arr.resize(32*sz);
        version.push_back(node_cnt++);
        return;
```

```
    }

    void pull(Node &c, Node a, Node b){
        c.val = a.val+b.val;
        return;
    }

    void build(vector<int> &v, int idx, int ll = 0, int rr = n){
        auto &now = arr[idx];

        if (rr-ll==1){
            now.val = v[ll];
            return;
        }

        int mid = (ll+rr)/2;
        now.lc = node_cnt++;
        now.rc = node_cnt++;
        build(v, now.lc, ll, mid);
        build(v, now.rc, mid, rr);
        pull(now, arr[now.lc], arr[now.rc]);
        return;
    }

    void update(int pos, int val, int idx, int ll = 0, int rr = n){
        auto &now = arr[idx];

        if (rr-ll==1){
            now.val = val;
            return;
        }

        int mid = (ll+rr)/2;
        if (pos<mid){
            arr[node_cnt] = arr[now.lc];
            now.lc = node_cnt;
            node_cnt++;
            update(pos, val, now.lc, ll, mid);
        }else{
            arr[node_cnt] = arr[now.rc];
            now.rc = node_cnt;
            node_cnt++;
            update(pos, val, now.rc, mid, rr);
        }
        pull(now, arr[now.lc], arr[now.rc]);
        return;
    }

    void update_version(int pos, int val, int ver){
        update(pos, val, version[ver]);
    }

    Node query(int ql, int qr, int idx, int ll = 0, int rr = n){
        auto &now = arr[idx];

        if (ql<=ll && rr<=qr) return now;
        if (rr<=ql || qr<=ll) return Node();

        int mid = (ll+rr)/2;

        Node ret;
        pull(ret, query(ql, qr, now.lc, ll, mid), query(ql, qr, now.rc, mid, rr));
        return ret;
    }

    Node query_version(int ql, int qr, int ver){
        return query(ql, qr, version[ver]);
    }

    void clone_version(int ver){
        version.push_back(node_cnt);
        arr[node_cnt] = arr[version[ver]];
        node_cnt++;
    }
};
```

3.9 Sparse Table [31f22a]

```
struct SparseTable{
    vector<vector<int>> st;
    void build(vector<int> v){
        int h = __lg(v.size());
        st.resize(h+1);
        st[0] = v;

        for (int i=1 ; i<=h ; i++){
            int gap = (1<<(i-1));
            for (int j=0 ; j+gap<st[i-1].size() ; j++){
                st[i].push_back(min(st[i-1][j], st[i-1][j+gap]));
            }
        }
    }

    // 回傳 [ll, rr) 的最小值
    int query(int ll, int rr){
        int h = __lg(rr-ll);
        return min(st[h][ll], st[h][rr-(1<<h)]);
    }
};
```

3.10 Treap2 [3b0cca]

```
// 1-based，請注意 MAX_N 是否足夠大
int root = 0;
int lc[MAX_N], rc[MAX_N];
int pri[MAX_N], val[MAX_N];
int sz[MAX_N], tag[MAX_N], fa[MAX_N], total[MAX_N];
// tag 為不包含自己（僅要給子樹）的資訊
int nodeCnt = 0;
int& new_node(int v){
    nodeCnt++;
    val[nodeCnt] = v;
    total[nodeCnt] = v;
    sz[nodeCnt] = 1;
    pri[nodeCnt] = rand();
    return nodeCnt;
}

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

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

int merge(int a, int b){
    if (!a or !b) return a|b;
    push(a), push(b);

    if (pri[a]>pri[b]){
        rc[a] = merge(rc[a], b);
        return pull(a);
    }else{
        lc[b] = merge(a, lc[b]);
        return pull(b);
    }
}

// [1, k] [k+1, n]
void split(int x, int k, int &a, int &b) {
    if (!x) return a = b = 0, void();
    push(x);
    if (sz[lc[x]] >= k) {
        split(lc[x], k, a, lc[x]);
        b = x;
        pull(a); pull(b);
    }else{
        split(rc[x], k - sz[lc[x]] - 1, rc[x], b);
        a = x;
        pull(a); pull(b);
    }
}

// functions
// 回傳 x 在 Treap 中的位置
int get_pos(int x){
    vector<int> sta;
    while (fa[x]){
        sta.push_back(x);
        x = fa[x];
    }
    while (sta.size()){
        push(x);
        x = sta.back();
        sta.pop_back();
    }
    push(x);

    int res = sz[x] - sz[rc[x]];
    while (fa[x]){
        if (rc[fa[x]]==x){
            res += sz[fa[x]]-sz[x];
        }
```

```
        x = fa[x];
    }
    return res;
}

// 1-based <前 [1, l-1] 個元素, [l, r] 個元素, [r+1, n] 個元素>
array<int, 3> cut(int x, int l, int r){
    array<int, 3> ret;
    split(x, r, ret[1], ret[2]);
    split(ret[1], l-1, ret[0], ret[1]);
    return ret;
}

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

3.11 Trie [b6475c]

```
struct Trie{
    struct Data{
        int nxt[2]={0, 0};
    };

    int sz=0;
    vector<Data> arr;

    void init(int n){
        arr.resize(n);
    }

    void insert(int n){
        int now=0;
        for (int i=N ; i>=0 ; i--){
            int v=(n>>i)&1;
            if (!arr[now].nxt[v]){
                arr[now].nxt[v]=++sz;
            }
            now=arr[now].nxt[v];
        }
    }

    int query(int n){
        int now=0, ret=0;
        for (int i=N ; i>=0 ; i--){
            int v=(n>>i)&1;
            if (arr[now].nxt[1-v]){
                ret+=(1<<i);
                now=arr[now].nxt[1-v];
            }else if (arr[now].nxt[v]){
                now=arr[now].nxt[v];
            }else{
                return ret;
            }
        }
        return ret;
    }

} tr;
```

4 Dynamic-Programming

4.1 Digit DP [133f00]

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

long long l, r;
long long dp[20][10][2][2]; // dp[pos][pre][limit] = 後 pos 位，pos 前一位是 pre，（是/否）有上界，（是/否）有前綴零的答案數量

long long memorize_search(string &s, int pos, int pre, bool limit, bool lead){

    // 已經被找過了，直接回傳值
    if (dp[pos][pre][limit][lead]!=-1) return dp[pos][pre][limit][lead];

    // 已經搜尋完畢，紀錄答案並回傳
    if (pos==(int)s.size()){
        return dp[pos][pre][limit][lead] = 1;
    }

    // 枚舉目前的位數數字是多少
    long long ans = 0;
    for (int now=0 ; now<=(limit ? s[pos]-'0' : 9) ; now++){
        if (now==pre){

            // 1~9 絕對不能連續出現
            if (pre!=0) continue;

            // 如果已經不在前綴零的範圍內，0 不能連續出現
            if (lead==false) continue;
        }

        ans += memorize_search(s, pos+1, now, limit&(now==(s[pos]-'0')), lead&(now==0));
    }

    // 已經搜尋完畢，紀錄答案並回傳
    return dp[pos][pre][limit][lead] = ans;
}

// 回傳 [0, n] 有多少數字符合條件
long long find_answer(long long n){
    memset(dp, -1, sizeof(dp));
    string tmp = to_string(n);

    return memorize_search(tmp, 0, 0, true, true);
}

int main(){

    // input
    cin >> l >> r;

    // output - 計算 [l, r] 有多少數字任意兩個位數都不相同
    cout << find_answer(r)-find_answer(l-1) << "\n";

    return 0;
}
```

4.2 Knaspack On Tree [df69b1]

```
// 需要重構、需要增加註解
#include <bits/stdc++.h>
#define F first
#define S second
#define all(x) begin(x), end(x)
using namespace std;

#define chmax(a, b) (a) = (a) < (b) ? (b) : (a)
#define chmin(a, b) (a) = (a) < (b) ? (a) : (b)

#define ll long long

#define FOR(i, a, b) for (int i = a; i <= b; i++)

int N, W, cur;
vector<int> w, v, sz;
vector<vector<int>> adj, dp;

void dfs(int x) {
    sz[x] = 1;
    for (int i : adj[x]) dfs(i), sz[x] += sz[i];
    cur++;
    // choose x
    for (int i=w[x] ; i<=W ; i++){
        dp[cur][i] = dp[cur - 1][i - w[x]] + v[x];
    }
    // not choose x
    for (int i=0 ; i<=W ; i++){
        chmax(dp[cur][i], dp[cur - sz[x]][i]);
    }
}

signed main() {
    cin >> N >> W;
    adj.resize(N + 1);
    w.assign(N + 1, 0);
    v.assign(N + 1, 0);
    sz.assign(N + 1, 0);
    dp.assign(N + 2, vector<int>(W + 1, 0));
    for (int i=1 ; i<=N ; i++){
        int p; cin >> p;
        adj[p].push_back(i);
    }

    for (int i=1 ; i<=N ; i++) cin >> w[i];
    for (int i=1 ; i<=N ; i++) cin >> v[i];
    dfs(0);
    cout << dp[N + 1][W] << '\n';
}
```

4.3 SOS DP [8dfa8b]

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

4.4 Integer Partition

$dp[i][x]$ = 要將整數 x 拆成 i 堆的「組合數」

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

5 Geometry

5.1 Geometry Struct [d9966f]

```cpp
using ld = double;

// 判斷數值正負：{1:正數,0:零,-1:負數}
int sign(long long x) {return (x >= 0) ? ((bool)x) : -1; }
int sign(ld x) {return (abs(x) < 1e-9) ? 0 : (x>0 ? 1 : -1);}

template<typename T>
struct point {
    T x, y;
    point() {}
    point(const T &x, const T &y) : x(x), y(y) {}
    explicit operator point<ld>() {return point<ld>(x, y); }
    // A [6357c4], Line 9 ~ 13
    point operator+(point b) {return {x+b.x, y+b.y}; }
    point operator-(point b) {return {x-b.x, y-b.y}; }
    point operator*(T b) {return {x*b, y*b}; }
    point operator/(T b) {return {x/b, y/b}; }
    bool operator==(point b) {return x==b.x && y==b.y; }

    T operator*(point b) {return x * b.x + y * b.y; }
    T operator^(point b) {return x * b.y - y * b.x; }
    // B [c415da], Line 14 ~ 22
    // 逆時針極角排序
    bool side() const{return (y == 0) ? (x > 0) : (y < 0); }
    bool operator<(const point &b) const {
        return side() == b.side() ?
            (x*b.y > b.x*y) : side() < b.side();
    }
    friend ostream& operator<<(ostream& os, point p) {
        return os << "(" << p.x << ", " << p.y << ")";
    }
    // 判斷 ab 到 ac 的方向：{1:逆時鐘,0:重疊,-1:順時鐘}
    friend int ori(point a, point b, point c) {
        return sign((b-a)^(c-a));
    }
    friend bool btw(point a, point b, point c) {
        return ori(a, b, c) == 0 && sign((a-c)*(b-c)) <= 0;
    }
    // 判斷線段 ab, cd 是否相交
    friend bool banana(point a, point b, point c, point d) {
        if (btw(a, b, c) || btw(a, b, d)
            || btw(c, d, a) || btw(c, d, b)) return true;
        int u = ori(a, b, c) * ori(a, b, d);
        int v = ori(c, d, a) * ori(c, d, b);
        return u < 0 && v < 0;
    } // C [09fd7c], only this function
    // 判斷 "射線 ab" 與 "線段 cd" 是否相交
    friend bool rayHitSeg(point a,point b,point c,point d) {
        if (a == b) return btw(c, d, a); // Special case
        if (((a - b) ^  (c - d)) == 0) {
            return btw(a, c, b) || btw(a, d, b) || banana(a,
                b, c, d);
        }
        point u = b - a, v = d - c, s = c - a;
        return sign(s ^ v) * sign(u ^ v) >= 0 && sign(s ^ u)
            * sign(u ^ v) >= 0 && abs(s ^ u) <= abs(u ^ v);
    } // D [db541a], only this function
    // 旋轉 Arg(b) 的角度 ( 小心溢位 )
    point rotate(point b){return {x*b.x-y*b.y, x*b.y+y*b.x};}
    // 回傳極座標角度，值域：[-π, +π]
    friend ld Arg(point b) {
        return (b.x != 0 || b.y != 0) ? atan2(b.y, b.x) : 0;
    }
    friend T abs2(point b) {return b * b; }
};

template<typename T>
struct line {
    point<T> p1, p2;
    // ax + by + c = 0
    T a, b, c; // |a|, |b| ≤ 2C, |c| ≤ 8C²
    line() {}
    line(const point<T> &x,const point<T> &y) : p1(x), p2(y){
        build();
    }
    void build() {
        a = p1.y - p2.y;
        b = p2.x - p1.x;
        c = (-a*p1.x)-b*p1.y;
    } // E [683239], Line 68 ~ 79
    // 判斷點和有向直線的關係：{1:左邊,0:在線上,-1:右邊}
    int ori(point<T> &p) {
        return sign((p2-p1) ^ (p-p1));
    }
    // 判斷直線斜率是否相同
    bool parallel(line &l) {
        return ((p1-p2) ^ (l.p1-l.p2)) == 0;
    }
    // 兩直線交點
    point<ld> line_intersection(line &l) {
        using P = point<ld>;
        point<T> u = p2-p1, v = l.p2-l.p1, s = l.p1-p1;
        return P(p1) + P(u) * ((ld(s^v)) / (u^v));
    }
};

template<typename T>
struct polygon {
    vector<point<T>> v;
    polygon() {}
    polygon(const vector<point<T>> &u) : v(u) {}
    // simple 為 true 的時候會回傳任意三點不共線的凸包
    void make_convex_hull(int simple) {
        auto cmp = [&](point<T> &p, point<T> &q) {
            return (p.x == q.x) ? (p.y < q.y) : (p.x < q.x);
        };
        simple = (bool)simple;
        sort(v.begin(), v.end(), cmp);
        v.resize(unique(v.begin(), v.end()) - v.begin());
        if (v.size() <= 1) return;
        vector<point<T>> hull;
        for (int t = 0; t < 2; ++t){
            int sz = hull.size();
            for (auto &i:v) {
                while (hull.size() >= sz+2 && ori(hull[hull.
                    size()-2], hull.back(), i) < simple) {
                    hull.pop_back();
                }
                hull.push_back(i);
            }
            hull.pop_back();
            reverse(v.begin(), v.end());
        }
        swap(hull, v);
    } // F [2bb3ef], only this function
//  可以在有 n 個點的簡單多邊形內，用 O(n) 判斷一個點：
//  {1 : 在多邊形內, 0 : 在多邊形上, -1 : 在多邊形外}
    int in_polygon(point<T> a){
        const T MAX_POS = 1e9 + 5; // [記得修改] 座標的最大值
        point<T> pre = v.back(), b(MAX_POS, a.y + 1);
        int cnt = 0;

        for (auto &i:v) {
            if (btw(pre, i, a)) return 0;
            if (banana(a, b, pre, i)) cnt++;
            pre = i;
        }

        return cnt%2 ? 1 : -1;
    } // G [f11340], only this function
/// 警告：以下所有凸包專用的函式都只接受逆時針排序且任三點不
    共線的凸包 ///
//  可以在有 n 個點的凸包內，用 O(log n) 判斷一個點：
//  {1 : 在凸包內, 0 : 在凸包邊上, -1 : 在凸包外}
    int in_convex(point<T> p) {
        int n = v.size();
        int a = ori(v[0], v[1], p), b = ori(v[0], v[n-1], p);
        if (a < 0 || b > 0) return -1;
        if (btw(v[0], v[1], p)) return 0;
        if (btw(v[0], v[n - 1], p)) return 0;
        int l = 1, r = n - 1, mid;
        while (l + 1 < r) {
            mid = (l + r) >> 1;
            if (ori(v[0], v[mid], p) >= 0) l = mid;
            else r = mid;
        }
        int k = ori(v[l], v[r], p);
        if (k <= 0) return k;
        return 1;
    } // H [e64f1e], only this function
//  凸包專用的環狀二分搜，回傳 0-based index
    int cycle_search(auto &f) {
        int n = v.size(), l = 0, r = n;
        if (n == 1) return 0;
        bool rv = f(1, 0);
        while (r - l > 1) {
            int m = (l + r) / 2;
            if (f(0, m) ? rv: f(m, (m + 1) % n)) r = m;
            else l = m;
        }
        return f(l, r % n) ? l : r % n;
    } // I [fe2f51], only this function
//  可以在有 n 個點的凸包內，用 O(log n) 判斷一條直線：
//  {1 : 穿過凸包, 0 : 剛好切過凸包, -1 : 沒碰到凸包}
    int line_cut_convex(line<T> L) {
        L.build();
        point<T> p(L.a, L.b);
        auto gt = [&](int neg) {
            auto f = [&](int x, int y) {
                return sign((v[x] - v[y]) * p) == neg;
            };
            return -(v[cycle_search(f)] * p);
        };
        T x = gt(1), y = gt(-1);
        if (L.c < x || y < L.c) return -1;
        return not (L.c == x || L.c == y);
    } // J [b6a4c8], only this function
```

```cpp
//  可以在有 n 個點的凸包內，用 O(log n) 判斷一個線段：
//  {1 : 存在一個凸包上的邊可以把這個線段切成兩半，
//   0 : 有碰到凸包但沒有任何凸包上的邊可以把它切成兩半，
//  -1 : 沒碰到凸包}
/// 除非線段兩端點都不在凸包邊上，否則此函數回傳 0 的時候不一定表示線段沒有通過凸包內部 ///
int segment_across_convex(line<T> L) {
    L.build();
    point<T> p(L.a, L.b);
    auto gt = [&](int neg) {
        auto f = [&](int x, int y) {
            return sign((v[x] - v[y]) * p) == neg;
        };
        return cycle_search(f);
    };
    int i = gt(1), j = gt(-1), n = v.size();
    T x = -(v[i] * p), y = -(v[j] * p);
    if (L.c < x || y < L.c) return -1;
    if (L.c == x || L.c == y) return 0;

    if (i > j) swap(i, j);
    auto g = [&](int x, int lim) {
        int now = 0, nxt;
        for (int i = 1 << __lg(lim); i > 0; i /= 2) {
            if (now + i > lim) continue;
            nxt = (x + i) % n;
            if (L.ori(v[x]) * L.ori(v[nxt]) >= 0) {
                x = nxt;
                now += i;
            }
        } //   ↓ BE CAREFUL
        return -(ori(v[x], v[(x + 1) % n], L.p1) * ori(v[x], v[(x + 1) % n], L.p2));
    };
    return max(g(i, j - i), g(j, n - (j - i)));
} // K [b4f073], only this function
//  可以在有 n 個點的凸包內，用 O(log n) 判斷一個線段：
//  {1 : 線段上存在某一點位於凸包內部（邊上不算），
//   0 : 線段上存在某一點碰到凸包的邊但線段上任一點均不在凸包內部，
//  -1 : 線段完全在凸包外面}
int segment_pass_convex_interior(line<T> L) {
    if (in_convex(L.p1) == 1 || in_convex(L.p2) == 1) return 1;
    L.build();
    point<T> p(L.a, L.b);
    auto gt = [&](int neg) {
        auto f = [&](int x, int y) {
            return sign((v[x] - v[y]) * p) == neg;
        };
        return cycle_search(f);
    };
    int i = gt(1), j = gt(-1), n = v.size();
    T x = -(v[i] * p), y = -(v[j] * p);
    if (L.c < x || y < L.c) return -1;
    if (L.c == x || L.c == y) return 0;

    if (i > j) swap(i, j);
    auto g = [&](int x, int lim) {
        int now = 0, nxt;
        for (int i = 1 << __lg(lim); i > 0; i /= 2) {
            if (now + i > lim) continue;
            nxt = (x + i) % n;
            if (L.ori(v[x]) * L.ori(v[nxt]) > 0) {
                x = nxt;
                now += i;
            }
        } //   ↓ BE CAREFUL
        return -(ori(v[x], v[(x + 1) % n], L.p1) * ori(v[x], v[(x + 1) % n], L.p2));
    };
    int ret = max(g(i, j - i), g(j, n - (j - i)));
    return (ret == 0) ? (in_convex(L.p1) == 0 && in_convex(L.p2) == 0) : ret;
} // L [5f45ca], only this function
//  回傳點過凸包的兩條切線的切點的 0-based index（不保證兩條切線的順逆時針關係）
pair<int,int> convex_tangent_point(point<T> p) {
    int n = v.size(), z = -1, edg = -1;
    auto gt = [&](int neg) {
        auto check = [&](int x) {
            if (v[x] == p) z = x;
            if (btw(v[x], v[(x + 1) % n], p)) edg = x;
            if (btw(v[(x + n - 1) % n], v[x], p)) edg = (x + n - 1) % n;
        };
        auto f = [&](int x, int y) {
            check(x); check(y);
            return ori(p, v[x], v[y]) == neg;
        };
        return cycle_search(f);
    };
    int x = gt(1), y = gt(-1);
    if (z != -1) {
        return {(z + n - 1) % n, (z + 1) % n};
    }
    else if (edg != -1) {
        return {edg, (edg + 1) % n};
    }
    else {
        return {x, y};
    }
} // M [a6f66b], only this function
friend int halfplane_intersection(vector<line<T>> &s, polygon<T> &P) {
    auto angle_cmp = [&](line<T> &A, line<T> &B) {
        point<T> a = A.p2-A.p1, b = B.p2-B.p1;
        return (a < b);
    };
    sort(s.begin(), s.end(), angle_cmp); // 線段左側為該線段半平面
    int L, R, n = s.size();
    vector<point<T>> px(n);
    vector<line<T>> q(n);
    q[L = R = 0] = s[0];
    for(int i = 1; i < n; ++i) {
        while(L < R && s[i].ori(px[R-1]) <= 0) --R;
        while(L < R && s[i].ori(px[L])   <= 0) ++L;
        q[++R] = s[i];
        if(q[R].parallel(q[R-1])) {
            --R;
            if(q[R].ori(s[i].p1) > 0) q[R] = s[i];
        }
        if(L<R) px[R-1] = q[R-1].line_intersection(q[R]);
    }
    while(L < R && q[L].ori(px[R-1]) <= 0) --R;
    P.v.clear();
    if(R - L <= 1) return 0;
    px[R] = q[R].line_intersection(q[L]);
    for(int i = L; i <= R; ++i) P.v.push_back(px[i]);
    return R - L + 1;
} // N [102d48], only this function
};

struct Cir {
    point<ld> o; ld r;
    friend ostream& operator<<(ostream& os, Cir c) {
        return os << "(x" << "+-"[c.o.x >= 0] << abs(c.o.x) << ")^2 + (y" << "+-"[c.o.y >= 0] << abs(c.o.y) << ")^2 = " << c.r * c.r;
    }
    bool covers(Cir b) {
        return sqrt((ld)abs2(o - b.o)) + b.r <= r;
    }
    vector<point<ld>> Cir_intersect(Cir c) {
        ld d2 = abs2(o - c.o), d = sqrt(d2);
        if (d < max(r, c.r) - min(r, c.r) || d > r + c.r) return {};
        auto sqdf = [&](ld x, ld y) { return x*x - y*y; };
        point<ld> u = (o + c.o) / 2 + (o - c.o) * (sqdf(c.r, r) / (2 * d2));
        ld A = sqrt(sqdf(r + d, c.r) * sqdf(c.r, d - r));
        point<ld> v = (c.o - o).rotate({0,1}) * A / (2 * d2);
        if (sign(v.x) == 0 && sign(v.y) == 0) return {u};
        return {u - v, u + v};
    } // O [330a1c], only this function
    auto point_tangent(point<ld> p) {
        vector<point<ld>> res;
        ld d_sq = abs2(p - o);
        if (sign(d_sq - r * r) <= 0) {
            res.pb(p + (p - o).rotate({0, 1}));
        } else if (d_sq > r * r) {
            ld s = d_sq - r * r;
            point<ld> v = p + (o - p) * s / d_sq;
            point<ld> u = (o - p).rotate({0, 1}) * sqrt(s) * r / d_sq;
            res.pb(v + u);
            res.pb(v - u);
        }
        return res;
    } // P [0067e6], only this function
};
```

5.2 Geometry Struct 3D [4a50c9]

```cpp
using ld = long double;

// 判斷數值正負：{1:正數,0:零,-1:負數}
int sign(long long x) {return (x >= 0) ? ((bool)x) : -1; }
int sign(ld x) {return (abs(x) < 1e-9) ? 0 : (x>0 ? 1 : -1);}

template<typename T>
struct pt3 {
    T x, y, z;
    pt3(){}
    pt3(const T &x, const T &y, const T &z):x(x),y(y),z(z){}
    explicit operator pt3<ld>() {return pt3<ld>(x, y, z); }

    pt3 operator+(pt3 b) {return {x+b.x, y+b.y, z+b.z}; }
    pt3 operator-(pt3 b) {return {x-b.x, y-b.y, z-b.z}; }
    pt3 operator*(T b) {return {x * b, y * b, z * b}; }
    pt3 operator/(T b) {return {x / b, y / b, z / b}; }
    bool operator==(pt3 b){return x==b.x&&y==b.y&&z==b.z;}

```

```
    T operator*(pt3 b) {return x*b.x+y*b.y+z*b.z; }
    pt3 operator^(pt3 b) {
        return pt3(y * b.z - z * b.y,
                   z * b.x - x * b.z,
                   x * b.y - y * b.x);
    }
    friend T abs2(pt3 b) {return b * b; }
    friend T len (pt3 b) {return sqrt(abs2(b)); }
    friend ostream& operator<<(ostream& os, pt3 p) {
        return os << "(" << p.x << ", " <<
                             p.y << ", " << p.z << ")";
    }
};

template<typename T>
struct face {
    int a, b, c; // 三角形在 vector 裡面的 index
    pt3<T> q;    // 面積向量（朝外）
};

/// 警告；v 在過程中可能被修改，回傳的 face 以修改後的為準
// O(n²)，最多只有 2n-4 個面
// 當凸包退化時會回傳空的凸包，否則回傳凸包上的每個面
template<typename T>
vector<face<T>> hull3(vector<pt3<T>> &v) {
    int n = v.size();
    if (n < 3) return {};
    // don't use "==" when you use ld
    sort(all(v), [&](pt3<T> &p, pt3<T> &q) {
        return sign(p.x - q.x) ? (p.x < q.x) :
         (sign(p.y - q.y) ? p.y < q.y : p.z < q.z);
    });
    v.resize(unique(v.begin(), v.end()) - v.begin());
    for (int i = 2; i <= n; ++i) {
        if (i == n) return {};
        if (sign(len(((v[1] - v[0]) ^ (v[i] - v[0])))) {
            swap(v[2], v[i]);
            break;
        }
    }
    pt3<T> tmp_q = (v[1] - v[0]) ^ (v[2] - v[0]);
    for (int i = 3; i <= n; ++i) {
        if (i == n) return {};
        if (sign((v[i] - v[0]) * tmp_q)) {
            swap(v[3], v[i]);
            break;
        }
    }

    vector<face<T>> f;
    vector<vector<int>> dead(n, vector<int>(n, true));
    auto add_face = [&](int a, int b, int c) {
        f.emplace_back(a, b, c, (v[b] - v[a]) ^ (v[c] - v[a])
            );
        dead[a][b] = dead[b][c] = dead[c][a] = false;
    };
    add_face(0, 1, 2);
    add_face(0, 2, 1);

    for (int i = 3; i < n; ++i) {
        vector<face<T>> f2;
        for (auto &[a, b, c, q] : f) {
            if (sign((v[i] - v[a]) * q) > 0)
                dead[a][b] = dead[b][c] = dead[c][a] = true;
            else f2.emplace_back(a, b, c, q);
        }
        f.clear();
        for (face<T> &F : f2) {
            int arr[3] = {F.a, F.b, F.c};
            for (int j = 0; j < 3; ++j) {
                int a = arr[j], b = arr[(j + 1) % 3];
                if (dead[b][a]) add_face(b, a, i);
            }
        }
        f.insert(f.end(), all(f2));
    }
    return f;
} // 15ef50
```

5.3 Pick's Theorem

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

6 Graph

6.1 2-SAT [5a6317]

```
struct TWO_SAT {
    int n, N;
    vector<vector<int>> G, rev_G;
    deque<bool> used;
    vector<int> order, comp;
    deque<bool> assignment;
    void init(int _n) {
        n = _n;
        N = _n * 2;
        G.resize(N + 5);
        rev_G.resize(N + 5);
    }
    void dfs1(int v) {
        used[v] = true;
        for (int u : G[v]) {
            if (!used[u])
                dfs1(u);
        }
        order.push_back(v);
    }
    void dfs2(int v, int cl) {
        comp[v] = cl;
        for (int u : rev_G[v]) {
            if (comp[u] == -1)
                dfs2(u, cl);
        }
    }
    bool solve() {
        order.clear();
        used.assign(N, false);
        for (int i = 0; i < N; ++i) {
            if (!used[i])
                dfs1(i);
        }
        comp.assign(N, -1);
        for (int i = 0, j = 0; i < N; ++i) {
            int v = order[N - i - 1];
            if (comp[v] == -1)
                dfs2(v, j++);
        }
        assignment.assign(n, false);
        for (int i = 0; i < N; i += 2) {
            if (comp[i] == comp[i + 1])
                return false;
            assignment[i / 2] = (comp[i] > comp[i + 1]);
        }
        return true;
    }
    // A or B 都是 0-based
    void add_disjunction(int a, bool na, int b, bool nb) {
        // na is true => ~a, na is false => a
        // nb is true => ~b, nb is false => b
        a = 2 * a ^ na;
        b = 2 * b ^ nb;
        int neg_a = a ^ 1;
        int neg_b = b ^ 1;
        G[neg_a].push_back(b);
        G[neg_b].push_back(a);
        rev_G[b].push_back(neg_a);
        rev_G[a].push_back(neg_b);
        return;
    }
    void get_result(vector<int>& res) {
        res.clear();
        for (int i = 0; i < n; i++)
            res.push_back(assignment[i]);
    }
};
```

6.2 Augment Path [f8a5dd]

```
struct AugmentPath{
    int n, m;
    vector<vector<int>> G;
    vector<int> mx, my;
    vector<int> visx, visy;
    int stamp;

    AugmentPath(int _n, int _m) : n(_n), m(_m), G(n), mx(n,
         -1), my(m, -1), visx(n), visy(n){
        stamp = 0;
    }

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

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

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

    vector<pair<int, int>> find_max_matching(){
```

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

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

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

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

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

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

        vector<pair<int, int>> ret;
        for (int i=0 ; i<n ; i++){
            if (mx[i]==-1) dfs2(i);
        }

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

        return ret;
    }
};
```

6.3 C3C4 [d00465]

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

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

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

    vector<int> vis(N);

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

6.4 Cut BCC [2af809]

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

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

void dfs(int v, int p) {
    stk.push(v);
    vis[v] = true;
    low[v] = depth[v] = (p == -1 ? 1 : depth[p] + 1);
    for (int u : G[v]) {
        if (u == p) continue;
        if (!vis[u]) {
            /// (v, u) 是樹邊
            dfs(u, v);
            low[v] = min(low[v], low[u]);
            /// u 無法在不經過父邊的情況走到 v 的祖先
            if (low[u] >= depth[v]) {
                bcc.emplace_back();
                while (stk.top() != u) {
                    bcc.back().push_back(stk.top());
                    stk.pop();
                }
                bcc.back().push_back(stk.top());
                stk.pop();
                bcc.back().push_back(v);
            }
        } else {
            /// (v, u) 是回邊
            low[v] = min(low[v], depth[u]);
        }
    }
}
```

6.5 Dinic [961b34]

```
// 一般圖：O(EV²)
// 二分圖：O(EVV)
struct Flow{
    using T = int; // 可以換成別的型別
    struct Edge{
        int v; T rc; int rid;
    };
    vector<vector<Edge>> G;
    void add(int u, int v, T c){
        G[u].push_back({v, c, G[v].size()});
        G[v].push_back({u, 0, G[u].size()-1});
    }
    vector<int> dis, it;

    Flow(int n){
        G.resize(n);
        dis.resize(n);
        it.resize(n);
    }

    // ce56d6
    T dfs(int u, int t, T f){
        if (u == t || f == 0) return f;
        for (int &i=it[u] ; i<G[u].size() ; i++){
            auto &[v, rc, rid] = G[u][i];
            if (dis[v]!=dis[u]+1) continue;
            T df = dfs(v, t, min(f, rc));
            if (df <= 0) continue;
            rc -= df;
            G[v][rid].rc += df;
            return df;
        }
        return 0;
    }

    // e22e39
    T flow(int s, int t){
        T ans = 0;
        while (true){
            fill(dis.begin(), dis.end(), INF);
            queue<int> q;
            q.push(s);
            dis[s] = 0;

            while (q.size()){
                int u = q.front(); q.pop();
                for (auto [v, rc, rid] : G[u]){
                    if (rc <= 0 || dis[v] < INF) continue;
                    dis[v] = dis[u] + 1;
                    q.push(v);
                }
            }
            if (dis[t]==INF) break;

            fill(it.begin(), it.end(), 0);
            while (true){
                T df = dfs(s, t, INF);
                if (df <= 0) break;
                ans += df;
            }
        }
}
```

```
        return ans;
    }

    //  the code below constructs minimum cut
    void dfs_mincut(int now, vector<bool> &vis){
        vis[now] = true;
        for (auto &[v, rc, rid] : G[now]){
            if (vis[v] == false && rc > 0){
                dfs_mincut(v, vis);
            }
        }
    }

    vector<pair<int, int>> construct(int n, int s, vector<pair<int,int>> &E){
        // E is G without capacity
        vector<bool> vis(n);
        dfs_mincut(s, vis);
        vector<pair<int, int>> ret;
        for (auto &[u, v] : E){
            if (vis[u] == true && vis[v] == false){
                ret.emplace_back(u, v);
            }
        }
        return ret;
    }
};
```

6.6 Dominator Tree [52b249]

```
// 全部都是 0-based
// G 要是有向無權圖
// 一開始要初始化 G(N, root)
// 用完之後要 build
// G[i] = 從 root 走到 i 時，一定要走到的點且離 i 最近
struct DominatorTree{
    int N;
    vector<vector<int>> G;
    vector<vector<int>> buckets, rg;
    // dfn[x] = the DFS otder of x
    // rev[x] = the vertex with DFS order x
    // par[x] = the parent of x
    vector<int> dfn, rev, par;
    vector<int> sdom, dom, idom;
    vector<int> fa, val;
    int stamp;
    int root;

    int operator [] (int x){
        return idom[x];
    }

    DominatorTree(int _N, int _root) :
        N(_N),
        G(N), buckets(N), rg(N),
        dfn(N, -1), rev(N, -1), par(N, -1),
        sdom(N, -1), dom(N, -1), idom(N, -1),
        fa(N, -1), val(N, -1)
    {
        stamp = 0;
        root = _root;
    }

    void add_edge(int u, int v){
        G[u].push_back(v);
    }

    void dfs(int x){
        rev[dfn[x] = stamp] = x;
        fa[stamp] = sdom[stamp] = val[stamp] = stamp;
        stamp++;

        for (int u : G[x]){
            if (dfn[u]==-1){
                dfs(u);
                par[dfn[u]] = dfn[x];
            }
            rg[dfn[u]].push_back(dfn[x]);
        }
    }

    int eval(int x, bool first){
        if (fa[x]==x) return !first ? -1 : x;
        int p = eval(fa[x], false);

        if (p==-1) return x;
        if (sdom[val[x]]>sdom[val[fa[x]]]) val[x] = val[fa[x]];
        fa[x] = p;

        return !first ? p : val[x];
    }

    void link(int x, int y){
        fa[x] = y;
    }

    void build(){
        dfs(root);

        for (int x=stamp-1 ; x>=0 ; x--){
            for (int y : rg[x]){
                sdom[x] = min(sdom[x], sdom[eval(y, true)]);
            }
            if (x>0) buckets[sdom[x]].push_back(x);
            for (int u : buckets[x]){
                int p = eval(u, true);
                if (sdom[p]==x) dom[u] = x;
                else dom[u] = p;
            }
            if (x>0) link(x, par[x]);
        }

        idom[root] = root;
        for (int x=1 ; x<stamp ; x++){
            if (sdom[x]!=dom[x]) dom[x] = dom[dom[x]];
        }
        for (int i=1 ; i<stamp ; i++) idom[rev[i]] = rev[dom[i]];
    }
};
```

6.7 EdgeBCC [d09eb1]

```
// d09eb1
// 0-based，支援重邊
struct EdgeBCC{
    int n, m, dep, sz;
    vector<vector<pair<int, int>>> G;
    vector<vector<int>> bcc;
    vector<int> dfn, low, stk, isBridge, bccId;
    vector<pair<int, int>> edge, bridge;

    EdgeBCC(int _n) : n(_n), m(0), sz(0), dfn(n), low(n), G(n), bcc(n), bccId(n) {}

    void add_edge(int u, int v) {
        edge.push_back({u, v});
        G[u].push_back({v, m});
        G[v].push_back({u, m++});
    }

    void dfs(int now, int pre) {
        dfn[now] = low[now] = ++dep;
        stk.push_back(now);

        for (auto [x, id] : G[now]){
            if (!dfn[x]){
                dfs(x, id);
                low[now] = min(low[now], low[x]);
            }else if (id!=pre){
                low[now] = min(low[now], dfn[x]);
            }
        }

        if (low[now]==dfn[now]){
            if (pre!=-1) isBridge[pre] = true;
            int u;
            do{
                u = stk.back();
                stk.pop_back();
                bcc[sz].push_back(u);
                bccId[u] = sz;
            } while (u!=now);
            sz++;
        }
    }

    void get_bcc() {
        isBridge.assign(m, 0);
        dep = 0;
        for (int i=0 ; i<n ; i++){
            if (!dfn[i]) dfs(i, -1);
        }

        for (int i=0 ; i<m ; i++){
            if (isBridge[i]){
                bridge.push_back({edge[i].first , edge[i].second});
            }
        }
    }
};
```

6.8 EnumeratePlanarFace [e70ee1]

```
// 0-based
struct PlanarGraph{
    int n, m, id;
    vector<point<int>> v;
    vector<vector<pair<int, int>>> G;
    vector<int> conv, nxt, vis;

    PlanarGraph(int n, int m, vector<point<int>> _v) :
    n(n), m(m), id(0),
```

```
    v(_v), G(n),
    conv(2*m), nxt(2*m), vis(2*m) {}

    void add_edge(int x, int y){
        G[x].push_back({y, 2*id});
        G[y].push_back({x, 2*id+1});
        conv[2*id] = x;
        conv[2*id+1] = y;
        id++;
    }

    vector<int> enumerate_face(){
        for (int i=0 ; i<n ; i++){
            sort(G[i].begin(), G[i].end(), [&](pair<int, int>
                 a, pair<int, int> b){
                return (v[a.first]-v[i])<(v[b.first]-v[i]);
            });

            int sz = G[i].size(), pre = sz-1;
            for (int j=0 ; j<sz ; j++){
                nxt[G[i][pre].second] = G[i][j].second^1;
                pre = j;
            }
        }

        vector<int> ret;
        for (int i=0 ; i<2*m ; i++){
            if (vis[i]==false){
                int area = 0, now = i;
                vector<int> pt;

                while (!vis[now]){
                    vis[now] = true;
                    pt.push_back(conv[now]);
                    now = nxt[now];
                }

                pt.push_back(pt.front());
                for (int i=0 ; i+1<pt.size() ; i++){
                    area -= (v[pt[i]]^v[pt[i+1]]);
                }

                // pt = face boundary
                if (area>0){
                    ret.push_back(area);
                }else{
                    // pt is outer face
                }
            }
        }
        return ret;
    }
};
```

6.9 HLD [f57ec6]

```
#include <bits/stdc++.h>
#define int long long
using namespace std;

const int N = 100005;
vector <int> G[N];
struct HLD {
    vector<int> pa, sz, depth, mxson, topf, id;
    int n, idcnt = 0;
    HLD(int _n) : n(_n), pa(_n + 1), sz(_n + 1), depth(_n +
         1), mxson(_n + 1), topf(_n + 1), id(_n + 1) {}
    void dfs1(int v = 1, int p = -1) {
        pa[v] = p; sz[v] = 1; mxson[v] = 0;
        depth[v] = (p == -1 ? 0 : depth[p] + 1);
        for (int u : G[v]) {
            if (u == p) continue;
            dfs1(u, v);
            sz[v] += sz[u];
            if (sz[u] > sz[mxson[v]]) mxson[v] = u;
        }
    }
    void dfs2(int v = 1, int top = 1) {
        id[v] = ++idcnt;
        topf[v] = top;
        if (mxson[v]) dfs2(mxson[v], top);
        for (int u : G[v]) {
            if (u == mxson[v] || u == pa[v]) continue;
            dfs2(u, u);
        }
    }
    // query 為區間資料結構
    int path_query(int a, int b) {
        int res = 0;
        while (topf[a] != topf[b]) { /// 若不在同一條鍊上
            if (depth[topf[a]] < depth[topf[b]]) swap(a, b);
            res = max(res, 0ll); // query : l = id[topf[a]],
                 r = id[a]
            a = pa[topf[a]];
        }
        /// 此時已在同一條鍊上
        if (depth[a] < depth[b]) swap(a, b);
        res = max(res, 0ll); // query : l = id[b], r = id[a]
        return res;
    }
};
```

6.10 Kosaraju [c7d5aa]

```
// 給定一個有向圖，迴回傳縮點後的圖、SCC 的資訊
// 所有點都以 based-0 編號
// 函式：
// .compress: O(n log n) 計算 G3、SCC、SCC_id 的資訊，並把縮
     點後的結果存在 result 裡
// SCC[i] = 某個 SCC 中的所有點
// SCC_id[i] = 第 i 個點在第幾個 SCC
struct SCC_compress{
    int N, M, sz;
    vector<vector<int>> G, inv_G, result;
    vector<pair<int, int>> edges;
    vector<bool> vis;
    vector<int> order;

    vector<vector<int>> SCC;
    vector<int> SCC_id;

    SCC_compress(int _N) :
    N(_N), M(0), sz(0),
    G(N), inv_G(N),
    vis(N), SCC_id(N)
    {}

    vector<int> operator [] (int x){
        return result[x];
    }

    void add_edge(int u, int v){
        G[u].push_back(v);
        inv_G[v].push_back(u);
        edges.push_back({u, v});
        M++;
    }

    void dfs1(vector<vector<int>> &G, int now){
        vis[now] = 1;
        for (auto x : G[now]) if (!vis[x]) dfs1(G, x);
        order.push_back(now);
    }

    void dfs2(vector<vector<int>> &G, int now){
        SCC_id[now] = SCC.size()-1;
        SCC.back().push_back(now);
        vis[now] = 1;
        for (auto x : G[now]) if (!vis[x]) dfs2(G, x);
    }

    void compress(){
        fill(vis.begin(), vis.end(), 0);
        for (int i=0 ; i<N ; i++) if (!vis[i]) dfs1(G, i);

        fill(vis.begin(), vis.end(), 0);
        reverse(order.begin(), order.end());
        for (int i=0 ; i<N ; i++){
            if (!vis[order[i]]){
                SCC.push_back(vector<int>());
                dfs2(inv_G, order[i]);
            }
        }

        result.resize(SCC.size());
        sz = SCC.size();
        for (auto [u, v] : edges){
            if (SCC_id[u]!=SCC_id[v]) result[SCC_id[u]].
                 push_back(SCC_id[v]);
        }
        for (int i=0 ; i<SCC.size() ; i++){
            sort(result[i].begin(), result[i].end());
            result[i].resize(unique(result[i].begin(), result
                 [i].end())-result[i].begin());
        }
    }
};
```

6.11 Kuhn Munkres [e66c35]

```
// O(n^3) 找到最大權匹配
struct KuhnMunkres{
    int n; // max(n, m)
    vector<vector<int>> G;
    vector<int> match, lx, ly, visx, visy;
    vector<int> slack;
    int stamp = 0;

    KuhnMunkres(int n) : n(n), G(n, vector<int>(n)), lx(n),
         ly(n), slack(n), match(n), visx(n), visy(n) {}

    void add(int x, int y, int w){
        G[x][y] = max(G[x][y], w);
    }
```

```

    bool dfs(int i, bool aug){ // aug = true 表示要更新 match
        if (visx[i]==stamp) return false;
        visx[i] = stamp;

        for (int j=0 ; j<n ; j++){
            if (visy[j]==stamp) continue;
            int d = lx[i]+ly[j]-G[i][j];

            if (d==0){
                visy[j] = stamp;
                if (match[j]==-1 || dfs(match[j], aug)){
                    if (aug){
                        match[j] = i;
                    }
                    return true;
                }
            }else{
                slack[j] = min(slack[j], d);
            }
        }
        return false;
    }

    bool augment(){
        for (int j=0 ; j<n ; j++){
            if (visy[j]!=stamp && slack[j]==0){
                visy[j] = stamp;
                if (match[j]==-1 || dfs(match[j], false)){
                    return true;
                }
            }
        }
        return false;
    }

    void relabel(){
        int delta = INF;
        for (int j=0 ; j<n ; j++){
            if (visy[j]!=stamp) delta = min(delta, slack[j]);
        }
        for (int i=0 ; i<n ; i++){
            if (visx[i]==stamp) lx[i] -= delta;
        }
        for (int j=0 ; j<n ; j++){
            if (visy[j]==stamp) ly[j] += delta;
            else slack[j] -= delta;
        }
    }

    int solve(){

        for (int i=0 ; i<n ; i++){
            lx[i] = 0;
            for (int j=0 ; j<n ; j++){
                lx[i] = max(lx[i], G[i][j]);
            }
        }

        fill(ly.begin(), ly.end(), 0);
        fill(match.begin(), match.end(), -1);

        for(int i = 0; i < n; i++) {
            fill(slack.begin(), slack.end(), INF);
            stamp++;
            if(dfs(i, true)) continue;

            while(augment()==false) relabel();
            stamp++;
            dfs(i, true);
        }

        int ans = 0;
        for (int j=0 ; j<n ; j++){
            if (match[j]!=-1){
                ans += G[match[j]][j];
            }
        }
        return ans;
    }
};
```

6.12 LCA [5b6a5b]

```
// 1-based，可以支援森林，0 是超級源點，所有樹都要跟他建邊
struct Tree{
    int N, M = 0, H;
    vector<int> parent, dep;
    vector<vector<int>> G, LCA;

    Tree(int _N) : N(_N+1), H(__lg(N)+1), parent(N, -1), dep(N), G(N){
        LCA.resize(H, vector<int>(N, 0));
    }

    void add_edge(int u, int v){
        M++;
        G[u].push_back(v);
        G[v].push_back(u);
    }

    void dfs(int now = 0, int pre = 0){
        dep[now] = dep[pre]+1;
        parent[now] = pre;
        for (auto x : G[now]){
            if (x==pre) continue;
            dfs(x, now);
        }
    }

    void build_LCA(int root = 0){
        dfs();
        for (int i=0 ; i<N ; i++) LCA[0][i] = parent[i];
        for (int i=1 ; i<H ; i++){
            for (int j=0 ; j<N ; j++){
                LCA[i][j] = LCA[i-1][LCA[i-1][j]];
            }
        }
    }

    int jump(int u, int step){
        for (int i=0 ; i<H ; i++){
            if (step&(1<<i)) u = LCA[i][u];
        }
        return u;
    }

    int get_LCA(int u, int v){
        if (dep[u]<dep[v]) swap(u, v);
        u = jump(u, dep[u]-dep[v]);
        if (u==v) return u;
        for (int i=H-1 ; i>=0 ; i--){
            if (LCA[i][u]!=LCA[i][v]){
                u = LCA[i][u];
                v = LCA[i][v];
            }
        }
        return parent[u];
    }
};
```

6.13 MCMF [0d5244]

```
struct Flow {
  struct Edge {
    int u, rc, k, rv;
  };

  vector<vector<Edge>> G;
  vector<int> par, par_eid;
  Flow(int n) : G(n+1), par(n+1), par_eid(n+1) {}

  // v->u, capcity: c, cost: k
  void add(int v, int u, int c, int k){
    G[v].push_back({u, c, k, G[u].size()});
    G[u].push_back({v, 0, -k, G[v].size()-1});
  }

  // 6d1140
  int spfa(int s, int t){
    fill(par.begin(), par.end(), -1);
    vector<int> dis(par.size(), INF);
    vector<bool> in_q(par.size(), false);
    queue<int> Q;
    dis[s] = 0;
    in_q[s] = true;
    Q.push(s);

    while (!Q.empty()){
      int v = Q.front();
      Q.pop();
      in_q[v] = false;

      for (int i=0 ; i<G[v].size() ; i++){
        auto [u, rc, k, rv] = G[v][i];
        if (rc>0 && dis[v]+k<dis[u]){
          dis[u] = dis[v]+k;
          par[u] = v;
          par_eid[u] = i;
          if (!in_q[u]) Q.push(u);
          in_q[u] = true;
        }
      }
    }

    return dis[t];
  }

  // return <max flow, min cost>, d7e7ad
  pair<int, int> flow(int s, int t){
    int fl = 0, cost = 0, d;
    while ((d = spfa(s, t))<INF){
      int cur = INF;
      for (int v=t ; v!=s ; v=par[v])
        cur = min(cur, G[par[v]][par_eid[v]].rc);
```

```
      fl += cur;
      cost += d*cur;
      for (int v=t ; v!=s ; v=par[v]){
        G[par[v]][par_eid[v]].rc -= cur;
        G[v][G[par[v]][par_eid[v]].rv].rc += cur;
      }
    }
    return {fl, cost};
  }
};
```

6.14 Tarjan [8b2350]

```
struct tarjan_SCC {
    int now_T, now_SCCs;
    vector<int> dfn, low, SCC;
    stack<int> S;
    vector<vector<int>> E;
    vector<bool> vis, in_stack;

    tarjan_SCC(int n) {
        init(n);
    }
    void init(int n) {
        now_T = now_SCCs = 0;
        dfn = low = SCC = vector<int>(n);
        E = vector<vector<int>>(n);
        S = stack<int>();
        vis = in_stack = vector<bool>(n);
    }
    void add(int u, int v) {
        E[u].push_back(v);
    }
    void build() {
        for (int i = 0; i < dfn.size(); ++i) {
            if (!dfn[i]) dfs(i);
        }
    }
    void dfs(int v) {
        now_T++;
        vis[v] = in_stack[v] = true;
        dfn[v] = low[v] = now_T;
        S.push(v);
        for (auto &i:E[v]) {
            if (!vis[i]) {
                vis[i] = true;
                dfs(i);
                low[v] = min(low[v], low[i]);
            }
            else if (in_stack[i]) {
                low[v] = min(low[v], dfn[i]);
            }
        }
        if (low[v] == dfn[v]) {
            int tmp;
            do {
                tmp = S.top();
                S.pop();
                SCC[tmp] = now_SCCs;
                in_stack[tmp] = false;
            } while (tmp != v);
            now_SCCs += 1;
        }
    }
};
```

6.15 Tarjan Find AP [1daed6]

```
vector<int> dep(MAX_N), low(MAX_N), AP;
bitset<MAX_N> vis;

void dfs(int now, int pre){
    int cnt = 0;
    bool ap = 0;
    vis[now] = 1;
    low[now] = dep[now] = (now==1 ? 0 : dep[pre]+1);

    for (auto x : G[now]){
        if (x==pre){
            continue;
        }else if (vis[x]==0){
            cnt++;
            dfs(x, now);
            low[now] = min(low[now], low[x]);
            if (low[x]>=dep[now]) ap=1;
        }else{
            low[now] = min(low[now], dep[x]);
        }
    }

    if ((now==pre && cnt>=2) || (now!=pre && ap)){
        AP.push_back(now);
    }
}
```

6.16 Tree Isomorphism [cd2bbc]

```
#include <bits/stdc++.h>
#pragma GCC optimize("O3,unroll-loops")
#define fastio ios::sync_with_stdio(0), cin.tie(0), cout.tie(0)
#define dbg(x) cerr << #x << " = " << x << endl
#define int long long
using namespace std;

// declare
const int MAX_SIZE = 2e5+5;
const int INF = 9e18;
const int MOD = 1e9+7;
const double EPS = 1e-6;
typedef vector<vector<int>> Graph;
typedef map<vector<int>, int> Hash;

int n, a, b;
int id1, id2;
pair<int, int> c1, c2;
vector<int> sz1(MAX_SIZE), sz2(MAX_SIZE);
vector<int> we1(MAX_SIZE), we2(MAX_SIZE);
Graph g1(MAX_SIZE), g2(MAX_SIZE);
Hash m1, m2;
int testcase=0;

void centroid(Graph &g, vector<int> &s, vector<int> &w, pair<int, int> &rec, int now, int pre){
    s[now]=1;
    w[now]=0;
    for (auto x : g[now]){
        if (x!=pre){
            centroid(g, s, w, rec, x, now);
            s[now]+=s[x];
            w[now]=max(w[now], s[x]);
        }
    }

    w[now]=max(w[now], n-s[now]);
    if (w[now]<=n/2){
        if (rec.first==0) rec.first=now;
        else rec.second=now;
    }
}

int dfs(Graph &g, Hash &m, int &id, int now, int pre){
    vector<int> v;
    for (auto x : g[now]){
        if (x!=pre){
            int add=dfs(g, m, id, x, now);
            v.push_back(add);
        }
    }
    sort(v.begin(), v.end());

    if (m.find(v)!=m.end()){
        return m[v];
    }else{
        m[v]=++id;
        return id;
    }
}


void solve1(){

    // init
    id1=0;
    id2=0;
    c1={0, 0};
    c2={0, 0};
    fill(sz1.begin(), sz1.begin()+n+1, 0);
    fill(sz2.begin(), sz2.begin()+n+1, 0);
    fill(we1.begin(), we1.begin()+n+1, 0);
    fill(we2.begin(), we2.begin()+n+1, 0);
    for (int i=1 ; i<=n ; i++){
        g1[i].clear();
        g2[i].clear();
    }
    m1.clear();
    m2.clear();

    // input
    cin >> n;
    for (int i=0 ; i<n-1 ; i++){
        cin >> a >> b;
        g1[a].push_back(b);
        g1[b].push_back(a);
    }
    for (int i=0 ; i<n-1 ; i++){
        cin >> a >> b;
        g2[a].push_back(b);
        g2[b].push_back(a);
    }

    // get tree centroid
    centroid(g1, sz1, we1, c1, 1, 0);
    centroid(g2, sz2, we2, c2, 1, 0);

    // process
    int res1=0, res2=0, res3=0;
```

```
    if (c2.second!=0){
        res1=dfs(g1, m1, id1, c1.first, 0);
        m2=m1;
        id2=id1;
        res2=dfs(g2, m1, id1, c2.first, 0);
        res3=dfs(g2, m2, id2, c2.second, 0);
    }else if (c1.second!=0){
        res1=dfs(g2, m1, id1, c2.first, 0);
        m2=m1;
        id2=id1;
        res2=dfs(g1, m1, id1, c1.first, 0);
        res3=dfs(g1, m2, id2, c1.second, 0);
    }else{
        res1=dfs(g1, m1, id1, c1.first, 0);
        res2=dfs(g2, m1, id1, c2.first, 0);
    }

    // output
    cout << (res1==res2 || res1==res3 ? "YES" : "NO") << endl
         ;

    return;
}

signed main(void){
    fastio;

    int t=1;
    cin >> t;
    while (t--){
        solve1();
    }
    return 0;
}
```

6.17 圓方樹 [675aec]

```
#include <bits/stdc++.h>
#define lp(i,a,b) for(int i=(a);i<(b);i++)
#define pii pair<int,int>
#define pb push_back
#define ins insert
#define ff first
#define ss second
#define opa(x) cerr << #x << " = " << x << ", ";
#define op(x) cerr << #x << " = " << x << endl;
#define ops(x) cerr << x;
#define etr cerr << endl;
#define spc cerr << ' ';
#define BAE(x) (x).begin(), (x).end()
#define STL(x) cerr << #x << " : "; for(auto &qwe:x) cerr <<
    qwe << ' '; cerr << endl;
#define deb1 cerr << "deb1" << endl;
#define deb2 cerr << "deb2" << endl;
#define deb3 cerr << "deb3" << endl;
#define deb4 cerr << "deb4" << endl;
#define deb5 cerr << "deb5" << endl;
#define bye exit(0);
using namespace std;

const int mxn = (int)(2e5) + 10;
const int mxlg = 17;
int last_special_node = (int)(1e5) + 1;
vector<int> E[mxn], F[mxn];

struct edg{
    int fr, to;
    edg(int _fr, int _to){
        fr = _fr;
        to = _to;
    }
};
ostream& operator<<(ostream& os, edg x){os << x.fr << "--" <<
    x.to;}
vector<edg> EV;

void tarjan(int v, int par, stack<int>& S){
    static vector<int> dfn(mxn), low(mxn);
    static vector<bool> to_add(mxn);
    static int nowT = 0;

    int childs = 0;
    nowT += 1;
    dfn[v] = low[v] = nowT;
    for(auto &ne:E[v]){
        int i = EV[ne].to;
        if(i == par) continue;
        if(!dfn[i]){
            S.push(ne);
            tarjan(i, v, S);
            childs += 1;
            low[v] = min(low[v], low[i]);

            if(par >= 0 && low[i] >= dfn[v]){
                vector<int> bcc;
                int tmp;
                do{
                    tmp = S.top(); S.pop();
                    if(!to_add[EV[tmp].fr]){
                        to_add[EV[tmp].fr] = true;
                        bcc.pb(EV[tmp].fr);
                    }
                    if(!to_add[EV[tmp].to]){
                        to_add[EV[tmp].to] = true;
                        bcc.pb(EV[tmp].to);
                    }
                }while(tmp != ne);
                for(auto &j:bcc){
                    to_add[j] = false;
                    F[last_special_node].pb(j);
                    F[j].pb(last_special_node);
                }
                last_special_node += 1;
            }
        }
        else{
            low[v] = min(low[v], dfn[i]);
            if(dfn[i] < dfn[v]){ // edge i--v will be visited
                  twice at here, but we only need one.
                S.push(ne);
            }
        }
    }
}

int dep[mxn], jmp[mxn][mxlg];
void dfs_lca(int v, int par, int depth){
    dep[v] = depth;
    for(auto &i:F[v]){
        if(i == par) continue;
        jmp[i][0] = v;
        dfs_lca(i, v, depth + 1);
    }
}

inline void build_lca(){
    jmp[1][0] = 1;
    dfs_lca(1, -1, 1);
    lp(j,1,mxlg){
        lp(i,1,mxn){
            jmp[i][j] = jmp[jmp[i][j-1]][j-1];
        }
    }
}

inline int lca(int x, int y){
    if(dep[x] < dep[y]){ swap(x, y); }

    int diff = dep[x] - dep[y];
    lp(j,0,mxlg){
        if((diff >> j) & 1){
            x = jmp[x][j];
        }
    }
    if(x == y) return x;

    for(int j = mxlg - 1; j >= 0; j--){
        if(jmp[x][j] != jmp[y][j]){
            x = jmp[x][j];
            y = jmp[y][j];
        }
    }
    return jmp[x][0];
}

inline bool can_reach(int fr, int to){
    if(dep[to] > dep[fr]) return false;

    int diff = dep[fr] - dep[to];
    lp(j,0,mxlg){
        if((diff >> j) & 1){
            fr = jmp[fr][j];
        }
    }
    return fr == to;
}

int main(){
    ios::sync_with_stdio(false); cin.tie(0);
//    freopen("test_input.txt", "r", stdin);
    int n, m, q; cin >> n >> m >> q;
    lp(i,0,m){
        int u, v; cin >> u >> v;
        E[u].pb(EV.size());
        EV.pb(edg(u, v));
        E[v].pb(EV.size());
        EV.pb(edg(v, u));
    }
    E[0].pb(EV.size());
    EV.pb(edg(0, 1));
    stack<int> S;
    tarjan(0, -1, S);
    build_lca();

    lp(queries,0,q){
        int fr, to, relay; cin >> fr >> to >> relay;
        if(fr == relay || to == relay){
```

```
            cout << "NO\n";
            continue;
        }
        if((can_reach(fr, relay) || can_reach(to, relay)) &&
             dep[relay] >= dep[lca(fr, to)]){
            cout << "NO\n";
            continue;
        }
        cout << "YES\n";
    }
}
```

6.18 最大權閉合圖 [6ca663]

```
// 邊 u → v 表示選 u 就要選 v (0-based)
// 保證回傳值非負
// 構造：從 S 開始 dfs，不走最小割的邊，
// 所有經過的點就是要選的那些點。
// 一般圖：O(n²m) / 二分圖：O(mVn)
template<typename U>
U maximum_closure(vector<U> w, vector<pair<int,int>> EV) {
    int n = w.size(), S = n + 1, T = n + 2;
    Flow G(T + 5); // Graph/Dinic.cpp
    U sum = 0;
    for (int i = 0; i < n; ++i) {
        if (w[i] > 0) {
            G.add(S, i, w[i]);
            sum += w[i];
        }
        else if (w[i] < 0) {
            G.add(i, T, abs(w[i]));
        }
    }
    for (auto &[u, v] : EV) { // 請務必確保 INF > Σ|w_i|
        G.add(u, v, INF);
    }
    U cut = G.flow(S, T);
    return sum - cut;
}
```

6.19 Theorem

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

7 Math

7.1 CRT [682ac6]

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

// CRT maybe need use int128
int CRT_m_coprime(vector<int> &a, vector<int> &m) {
    int n = a.size(), p = 1, ans = 0;
    vector<int> M(n), invM(n);

    for (int i=0 ; i<n ; i++) p *= m[i];
    for (int i=0 ; i<n ; i++){
        M[i] = p/m[i];
        auto [d, x, y] = extgcd(M[i], m[i]);
        invM[i] = x;
        ans += a[i]*invM[i]*M[i];
        ans %= p;
    }
    return (ans%p+p)%p;
}

// CRT maybe need use int128
int CRT_m_not_coprime(vector<int> &a, vector<int> &m) {
    int n = a.size();

    for (int i=1 ; i<n ; i++){
        int g = __gcd(m[0], m[i]);
        if ((a[i]-a[0])%g!=0) return -1;

        auto [d, x, y] = extgcd(m[0], m[i]);
        x = (a[i]-a[0])*x/g;

        a[0] = x*m[0]+a[0];
        m[0] = m[0]*m[i]/g;
        a[0] = (a[0]%m[0]+m[0])%m[0];
    }

    if (a[0]<0) return a[0]+m[0];
    return a[0];
}

// ans = a / b (mod m)
// ans = ret.F + k * ret.S, k is integer
pair<int, int> div(int a, int b, int m) {
    int flag = 1;
    if (a < 0) { a = -a; flag *= -1; }
    if (b < 0) { b = -b; flag *= -1; }
    int t = -1, k = -1;
    int res = extgcd_abc(b, m, a, t, k);
    if (res == INF) return {INF, INF};
    m = abs(m / res);
    t = t * flag;
    t = (t % m + m) % m;
    return {t, m};
}
```

7.2 Josephus Problem [e0ed50]

```
// 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
int solve(int n, int k){
    if (n==1) return 1;
    if (k<=(n+1)/2){
        if (2*k>n) return 2*k%n;
        else return 2*k;
    }else{
        int res=solve(n/2, k-(n+1)/2);
        if (n&1) return 2*res+1;
        else return 2*res-1;
    }
}
```

7.3 Lagrange any x [1f2c26]

```
// init: (x1, y1), (x2, y2) in a vector
struct Lagrange{
    int n;
    vector<pair<int, int>> v;

    Lagrange(vector<pair<int, int>> &_v){
        n = _v.size();
        v = _v;
    }

    // O(n^2 log MAX_A)
    int solve(int x){
        int ret = 0;
        for (int i=0 ; i<n ; i++){
            int now = v[i].second;
            for (int j=0 ; j<n ; j++){
                if (i==j) continue;
                now *= ((x-v[j].first)+MOD)%MOD;
                now %= MOD;
                now *= (qp((v[i].first-v[j].first+MOD)%MOD,
                     MOD-2)+MOD)%MOD;
                now %= MOD;
            }

            ret = (ret+now)%MOD;
        }
        return ret;
    }
};
```

7.4 Lagrange continuous x [57536a]

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

const int MAX_N = 5e5 + 10;
const int mod = 1e9 + 7;

long long inv_fac[MAX_N];

inline int fp(long long x, int y) {
    int ret = 1;
    for (; y; y >>= 1) {
        ret = (y & 1) ? (ret * x % mod) : ret;
        x = x * x % mod;
    }
    return ret;
}
```

```cpp

// TO USE THIS TEMPLATE, YOU MUST MAKE SURE THAT THE MOD NUMBER IS A PRIME.
struct Lagrange {
/*
    Initialize a polynomial with f(x_0), f(x_0 + 1), ..., f(x_0 + n).
    This determines a polynomial f(x) whose degree is at most n.
    Then you can call sample(x) and you get the value of f(x).
    Complexity of init() and sample() are both O(n).
*/
    int m, shift; // m = n + 1
    vector<int> v, mul;
// You can use this function if you don't have inv_fac array already.
    void construct_inv_fac() {
        long long fac = 1;
        for (int i = 2; i < MAX_N; ++i) {
            fac = fac * i % mod;
        }
        inv_fac[MAX_N - 1] = fp(fac, mod - 2);
        for (int i = MAX_N - 1; i >= 1; --i) {
            inv_fac[i - 1] = inv_fac[i] * i % mod;
        }
    }
// You call init() many times without having a second instance of this struct.
    void init(int X_0, vector<int> &u) {
        v = u;
        shift = ((1 - X_0) % mod + mod) % mod;
        if (v.size() == 1) v.push_back(v[0]);
        m = v.size();
        mul.resize(m);
    }
// You can use sample(x) instead of sample(x % mod).
    int sample(int x) {
        x = ((long long)x + shift) % mod;
        x = (x < 0) ? (x + mod) : x;
        long long now = 1;
        for (int i = m; i >= 1; --i) {
            mul[i - 1] = now;
            now = now * (x - i) % mod;
        }
        int ret = 0;
        bool neg = (m - 1) & 1;
        now = 1;
        for (int i = 1; i <= m; ++i) {
            int up = now * mul[i - 1] % mod;
            int down = inv_fac[m - i] * inv_fac[i - 1] % mod;
            int tmp = ((long long)v[i - 1] * up % mod) * down % mod;
            ret += (neg && tmp) ? (mod - tmp) : (tmp);
            ret = (ret >= mod) ? (ret - mod) : ret;
            now = now * (x - i) % mod;
            neg ^= 1;
        }
        return ret;
    }
};

int main() {
    int n; cin >> n;
    vector<int> v(n);
    for (int i = 0; i < n; ++i) {
        cin >> v[i];
    }
    Lagrange L;
    L.construct_inv_fac();
    L.init(0, v);
    int x; cin >> x;
    cout << L.sample(x);
}
```

7.5 Linear Mod Inverse [ecf71e]

```cpp
// 線性求 1-based a[i] 對 p 的乘法反元素
vector<int> s(n+1, 1), invS(n+1), invA(n+1);
for (int i=1 ; i<=n ; i++) s[i] = s[i-1]*a[i]%p;
invS[n] = qp(s[n], p-2, p);
for (int i=n ; i>=1 ; i--) invS[i-1] = invS[i]*a[i]%p;
for (int i=1 ; i<=n ; i++) invA[i] = invS[i]*s[i-1]%p;
```

7.6 Lucas's Theorem [b37dcf]

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

7.7 Matrix [8d1a23]

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

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

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

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

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

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

        return ret;
    }


    int det(){
        vector<vector<int>> arr = this->arr;
        bool flag = false;
        for (int i=0 ; i<n ; i++){
            int target = -1;
            for (int j=i ; j<n ; j++){
                if (arr[j][i]){
                    target = j;
                    break;
                }
            }
            if (target==-1) return 0;
            if (i!=target){
                swap(arr[i], arr[target]);
                flag = !flag;
            }

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

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

7.8 Matrix 01 [8d542a]

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

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

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

```

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

7.9 Miller Rabin [24bd0d]

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

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

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

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

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

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

    return true;
}
```

7.10 Pollard Rho [a5daef]

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

// O(n^{1/4}) 回傳 1 或自己的因數、記得先判斷 n 是不是質數
    (用 Miller-Rabin)
// c1670c
int Pollard_Rho(int n){
    int s = 0, t = 0;
    int c = rnd(1, n-1);

    int step = 0, goal = 1;
    int val = 1;

    for (goal=1 ; ; goal<<=1, s=t, val=1){
        for (step=1 ; step<=goal ; step++){

            t = ((__int128)t*t+c)%n;
            val = (__int128)val*abs(t-s)%n;

            if ((step % 127) == 0){
                int d = __gcd(val, n);
                if (d>1) return d;
            }
        }

        int d = __gcd(val, n);
        if (d>1) return d;
    }
}
```

7.11 Polynomial [51ca3b]

```
struct Poly {
    int len, deg;
    int *a;
    // len = 2^k >= the original length
    Poly(): len(0), deg(0), a(nullptr) {}
    Poly(int _n) {
        len = 1;
        deg = _n - 1;
        while (len < _n) len <<= 1;
        a = (ll*) calloc(len, sizeof(ll));
    }
    Poly(int l, int d, int *b) {
        len = l;
        deg = d;
        a = b;
    }
    void resize(int _n) {
        int len1 = 1;
        while (len1 < _n) len1 <<= 1;
        int *res = (ll*) calloc(len1, sizeof(ll));
        for (int i = 0; i < min(len, _n); i++) {
            res[i] = a[i];
        }
        len = len1;
        deg = _n - 1;
        free(a);
        a = res;
    }
    Poly& operator=(const Poly rhs) {
        this->len = rhs.len;
        this->deg = rhs.deg;
        this->a = (ll*)realloc(this->a, sizeof(ll) * len);
        copy(rhs.a, rhs.a + len, this->a);
        return *this;
    }
    Poly operator*(Poly rhs) {
        int l1 = this->len, l2 = rhs.len;
        int d1 = this->deg, d2 = rhs.deg;
        while (l1 > 0 and this->a[l1 - 1] == 0) l1--;
        while (l2 > 0 and rhs.a[l2 - 1] == 0) l2--;
        int l = 1;
        while (l < max(l1 + l2 - 1, d1 + d2 + 1)) l <<= 1;
        int *x, *y, *res;
        x = (ll*) calloc(l, sizeof(ll));
        y = (ll*) calloc(l, sizeof(ll));
        res = (ll*) calloc(l, sizeof(ll));
        copy(this->a, this->a + l1, x);
        copy(rhs.a, rhs.a + l2, y);
        ntt.tran(l, x); ntt.tran(l, y);
        FOR (i, 0, l - 1)
            res[i] = x[i] * y[i] % mod;
        ntt.tran(l, res, true);
        free(x); free(y);
        return Poly(l, d1 + d2, res);
    }
    Poly operator+(Poly rhs) {
        int l1 = this->len, l2 = rhs.len;
        int l = max(l1, l2);
        Poly res;
        res.len = l;
        res.deg = max(this->deg, rhs.deg);
        res.a = (ll*) calloc(l, sizeof(ll));
        FOR (i, 0, l1 - 1) {
            res.a[i] += this->a[i];
            if (res.a[i] >= mod) res.a[i] -= mod;
        }
        FOR (i, 0, l2 - 1) {
            res.a[i] += rhs.a[i];
            if (res.a[i] >= mod) res.a[i] -= mod;
        }
        return res;
    }
    Poly operator-(Poly rhs) {
        int l1 = this->len, l2 = rhs.len;
        int l = max(l1, l2);
        Poly res;
        res.len = l;
        res.deg = max(this->deg, rhs.deg);
        res.a = (ll*) calloc(l, sizeof(ll));
        FOR (i, 0, l1 - 1) {
            res.a[i] += this->a[i];
            if (res.a[i] >= mod) res.a[i] -= mod;
        }
        FOR (i, 0, l2 - 1) {
            res.a[i] -= rhs.a[i];
            if (res.a[i] < 0) res.a[i] += mod;
        }
        return res;
    }
    Poly operator*(const int rhs) {
        Poly res;
        res = *this;
        FOR (i, 0, res.len - 1) {
            res.a[i] = res.a[i] * rhs % mod;
            if (res.a[i] < 0) res.a[i] += mod;
        }
        return res;
    }
    Poly(vector<int> f) {
        int _n = f.size();
        len = 1;
```

```
        deg = _n - 1;
        while (len < _n) len <<= 1;
        a = (ll*) calloc(len, sizeof(ll));
        FOR (i, 0, deg) a[i] = f[i];
    }
    Poly derivative() {
        Poly g(this->deg);
        FOR (i, 1, this->deg) {
            g.a[i - 1] = this->a[i] * i % mod;
        }
        return g;
    }
    Poly integral() {
        Poly g(this->deg + 2);
        FOR (i, 0, this->deg) {
            g.a[i + 1] = this->a[i] * ::inv(i + 1) % mod;
        }
        return g;
    }
    Poly inv(int len1 = -1) {
        if (len1 == -1) len1 = this->len;
        Poly g(1); g.a[0] = ::inv(a[0]);
        for (int l = 1; l < len1; l <<= 1) {
            Poly t; t = *this;
            t.resize(l << 1);
            t = g * g * t;
            t.resize(l << 1);
            Poly g1 = g * 2 - t;
            swap(g, g1);
        }
        return g;
    }
    Poly ln(int len1 = -1) {
        if (len1 == -1) len1 = this->len;
        auto g = *this;
        auto x = g.derivative() * g.inv(len1);
        x.resize(len1);
        x = x.integral();
        x.resize(len1);
        return x;
    }
    Poly exp() {
        Poly g(1);
        g.a[0] = 1;
        for (int l = 1; l < len; l <<= 1) {
            Poly t, g1; t = *this;
            t.resize(l << 1); t.a[0]++;
            g1 = (t - g.ln(l << 1)) * g;
            g1.resize(l << 1);
            swap(g, g1);
        }
        return g;
    }
    Poly pow(ll n) {
        Poly &a = *this;
        int i = 0;
        while (i <= a.deg and a.a[i] == 0) i++;
        if (i and (n > a.deg or n * i > a.deg)) return Poly(a
            .deg + 1);
        if (i == a.deg + 1) {
            Poly res(a.deg + 1);
            res.a[0] = 1;
            return res;
        }
        Poly b(a.deg - i + 1);
        int inv1 = ::inv(a.a[i]);
        FOR (j, 0, b.deg)
            b.a[j] = a.a[j + i] * inv1 % mod;
        Poly res1 = (b.ln() * (n % mod)).exp() * (::power(a.a
            [i], n));
        Poly res2(a.deg + 1);
        FOR (j, 0, min((ll)(res1.deg), (ll)(a.deg - n * i)))
            res2.a[j + n * i] = res1.a[j];
        return res2;
    }
};
```

7.12 josephus [0be067]

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

7.13 數論分塊 [8ccab5]

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

7.14 最大質因數 [ca5e52]

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

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

7.15 歐拉公式 [85f3b1]

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

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

    return ret;
}

// O(n log n) ，回傳 1~n 的 phi 值
vector<int> phi_1_to_n(int n){
    vector<int> phi(n+1);
    phi[0]=0;
    phi[1]=1;

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

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

    return phi;
}
```

7.16 Burnside's Lemma

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

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

7.17 Catalan Number

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

7.18 Matrix Tree Theorem

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

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

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

方法：先把所有自環刪掉，定義 Q 為以下矩陣

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

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

7.19 Stirling's formula

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

7.20 Theorem

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

7.21 二元一次方程式

$$\begin{cases} ax+by=e \\ cx+dy=f \end{cases} = \begin{cases} x=\frac{ed-bf}{ad-bc} \\ y=\frac{af-ec}{ad-bc} \end{cases}$$

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

7.22 歐拉定理

若 a, m 互質，則：

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

若 a, m 不互質，則：

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

7.23 錯排公式

錯排公式：(n 個人中，每個人皆不再原來位置的組合數)

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

8 String

8.1 AC automation [018290]

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

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

    void add(string s){
        int now = 0;
        for (char c : s){
            if (!go[now][c-'a']) go[now][c-'a'] = ++sz;
            now = go[now][c-'a'];
        }
        pos.push_back(now);
    }

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

    // counting pattern
    void buildMatch(string &s){
        int now = 0;
        for (char c : s){
            now = go[now][c-'a'];
            match[now]++;
        }

        vector<int> in(sz+1), que;
        for (int i=1 ; i<=sz ; i++) in[fail[i]]++;
        for (int i=1 ; i<=sz ; i++) if (in[i]==0) que.push_back(i);
        for (int i=0 ; i<que.size() ; i++){
            int now = que[i];
            match[fail[now]] += match[now];
            if (--in[fail[now]]==0) que.push_back(fail[now]);
        }
    }
};
```

8.2 Enumerate Runs [94ca46]

```cpp
vector<array<int, 3>> enumerate_run(string s){

    int n = s.size();
    SuffixArray sa(s), saBar(string(s.rbegin(), s.rend()));
    sa.init_lcp(), saBar.init_lcp();

    set<pair<int, int>> ss;
    vector<array<int, 3>> runs;

    for (int len=1 ; len<=n ; len++){
        vector<int> lcp;
        for (int i=0 ; i+len<n ; i+=len){
            int pos1 = sa.pos[i];
            int pos2 = sa.pos[i+len];
            lcp.push_back(sa.get_lcp(pos1, pos2));
        }

        for (int ll=0, rr=0 ; ll<lcp.size() ; rr++, ll=rr){
            while (rr<lcp.size() && lcp[rr]>=len) rr++;

            int preLen = 0;
            if (ll!=0){
                int p = n-1;
                int pos1 = saBar.pos[p-(ll*len-1)];
                int pos2 = saBar.pos[p-((ll+1)*len-1)];
                preLen = saBar.get_lcp(pos1, pos2);
            }
            int sufLen = rr<lcp.size() ? lcp[rr] : 0;

            int ansL = ll*len-preLen, ansR = (rr+1)*len-1+sufLen;
            if (ansL!=ansR && ansR-ansL+1>=2*len && ss.find({ansL, ansR+1})==ss.end()){
                ss.insert({ansL, ansR+1});
                runs.push_back({len, ansL, ansR+1});
            }
        }
    }

    return runs;
}
```

8.3 Hash [942f42]

```cpp
mt19937 seed(chrono::steady_clock::now().time_since_epoch().count());
int rng(int l, int r){
    return uniform_int_distribution<int>(l, r)(seed);
}
int A = rng(1e5, 8e8);
const int B = 1e9+7;

// 2f6192
struct RollingHash{
    vector<int> Pow, Pre;
    RollingHash(string s = ""){
        Pow.resize(s.size());
        Pre.resize(s.size());

        for (int i=0 ; i<s.size() ; i++){
            if (i==0){
                Pow[i] = 1;
                Pre[i] = s[i];
            }else{
                Pow[i] = Pow[i-1]*A%B;
                Pre[i] = (Pre[i-1]*A+s[i])%B;
            }
        }

        return;
    }

    int get(int l, int r){ // 取得 [l, r] 的數值
        if (l==0) return Pre[r];
        int res = (Pre[r]-Pre[l-1]*Pow[r-l+1])%B;
        if (res<0) res += B;
        return res;
    }
};
```

8.4 KMP [7b95d6]

```cpp
// KMP[i] = s[0...i] 的最長共同前後綴長度，KMP[KMP[i]-1] 可以跳 fail link
// e5b7ce
vector<int> KMP(string &s){
    vector<int> ret(n);
    for (int i=1 ; i<s.size() ; i++){
        int j = ret[i-1];
```

```
        while (j && s[i]!=s[j]) j = ret[j-1];
        ret[i] = j + (s[i]==s[j]);
    }
    return ret;
}
```

8.5 Manacher [9a4b4d]

```
string Manacher(string str) {
    string tmp = "$#";
    for(char i : str) {
        tmp += i;
        tmp += '#';
    }

    vector<int> p(tmp.size(), 0);
    int mx = 0, id = 0, len = 0, center = 0;
    for(int i=1 ; i<(int)tmp.size() ; i++) {
        p[i] = mx > i ? min(p[id*2-i], mx-i) : 1;

        while(tmp[i+p[i]] == tmp[i-p[i]]) p[i]++;
        if(mx<i+p[i]) mx = i+p[i], id = i;
        if(len<p[i]) len = p[i], center = i;
    }
    return str.substr((center-len)/2, len-1);
}
```

8.6 Min Rotation [b24786]

```
int minRotation(string s) {
    int a = 0, n = s.size();
    s += s;

    for (int b=0 ; b<n ; b++){
        for (int k=0 ; k<n ; k++){
            if (a+k==b || s[a+k]<s[b+k]){
                b += max(0LL, k-1);
                break;
            }
            if (s[a+k]>s[b+k]){
                a = b;
                break;
            }
        }
    }

    return a;
}
```

8.7 Suffix Array [f66629]

```
// 注意，當 |s|=1 時，lcp 不會有值，務必測試 |s|=1 的 case
struct SuffixArray {
    string s;
    vector<int> sa, lcp;

    // 69ced9
    // lim 要調整成字元集大小，_s 不可以有 0
    SuffixArray(string _s, int lim = 256) {
        s = _s;
        int n = s.size()+1, k = 0, a, b;
        vector<int> x(s.begin(), s.end()), y(n), ws(max(n,
            lim)), rank(n);
        x.push_back(0);
        sa = lcp = y;
        iota(sa.begin(), sa.end(), 0);
        for (int j=0, p=0 ; p<n ; j=max(1LL, j*2), lim=p) {
            p = j;
            iota(y.begin(), y.end(), n-j);
            for (int i=0 ; i<n ; i++) if (sa[i] >= j) y[p++]
                = sa[i] - j;
            fill(ws.begin(), ws.end(), 0);
            for (int i=0 ; i<n ; i++) ws[x[i]]++;
            for (int i=1 ; i<lim ; i++) ws[i] += ws[i - 1];
            for (int i = n; i--;) sa[--ws[x[y[i]]]] = y[i];
            swap(x, y), p = 1, x[sa[0]] = 0;
            for (int i=1 ; i<n ; i++){
                a = sa[i - 1];
                b = sa[i];
                x[b] = (y[a] == y[b] && y[a + j] == y[b + j])
                    ? p - 1 : p++;
            }
        }

        for (int i=1 ; i<n ; i++) rank[sa[i]] = i;
        for (int i=0, j ; i<n-1 ; lcp[rank[i++]]=k)
            for (k && k--, j=sa[rank[i]-1] ; i+k<s.size() &&
                j+k<s.size() && s[i+k]==s[j+k] ; k++);
        sa.erase(sa.begin());
        lcp.erase(lcp.begin(), lcp.begin()+2);
    }

    // f49583
    vector<int> pos; // pos[i] = i 這個值在 pos 的哪個地方
    SparseTable st;
    void init_lcp(){
        pos.resize(sa.size());
        for (int i=0 ; i<sa.size() ; i++){
            pos[sa[i]] = i;
        }
        if (lcp.size()){
            st.build(lcp);
        }
    }

    // 用之前記得 init
    // 查詢「sa 上的位置」的 x 跟 y 的 lcp
    int get_lcp(int x, int y){
        if (x==y) return s.size()-x;
        if (x>y) swap(x, y);
        return st.query(x, y);
    }

    // 回傳 [l1, r1] 跟 [l2, r2] 的 lcp，0-based
    int get_lcp(int l1, int r1, int l2, int r2){
        int pos_1 = pos[l1], len_1 = r1-l1+1;
        int pos_2 = pos[l2], len_2 = r2-l2+1;
        if (pos_1>pos_2){
            swap(pos_1, pos_2);
            swap(len_1, len_2);
        }

        if (l1==l2){
            return min(len_1, len_2);
        }else{
            return min({st.query(pos_1, pos_2), len_1, len_2
                });
        }
    }

    // 檢查 [l1, r1] 跟 [l2, r2] 的大小關係，0-based
    // 如果前者小於後者，就回傳 <0，相等就回傳 =0，否則回傳
        >0
    // 5b8db0
    int substring_cmp(int l1, int r1, int l2, int r2){
        int len_1 = r1-l1+1;
        int len_2 = r2-l2+1;
        int res = get_lcp(l1, r1, l2, r2);

        if (res<len_1 && res<len_2){
            return s[l1+res]-s[l2+res];
        }else if (len_1==res && len_2==res){
            return 0;
        }else{
            return len_1==res ? -1 : 1;
        }
    }

    // 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
    // pre[i] = s[i] 離他左邊最接近位置 >p 的後綴的 lcp，0-
        based
    // suf[i] = s[i] 離他右邊最接近位置 >p 的後綴的 lcp，0-
        based
    // da12fa
    pair<vector<int>, vector<int>> get_left_and_right_lcp(int
         p){
        vector<int> pre(p+1);
        vector<int> suf(p+1);

        { // build pre
            int now = 0;
            for (int i=0 ; i<s.size() ; i++){
                if (sa[i]<=p){
                    pre[sa[i]] = now;
                    if (i<lcp.size()) now = min(now, lcp[i]);
                }else{
                    if (i<lcp.size()) now = lcp[i];
                }
            }
        }
        { // build suf
            int now = 0;
            for (int i=s.size()-1 ; i>=0 ; i--){
                if (sa[i]<=p){
                    suf[sa[i]] = now;
                    if (i-1>=0) now = min(now, lcp[i-1]);
                }else{
                    if (i-1>=0) now = lcp[i-1];
                }
            }
        }

        return {pre, suf};
    }
};
```

8.8 Wildcard Matching [e93074]

```

const int B = 27;
string p;
for (int i=0 ; i<B ; i++) p += ('a'+i);
```

```
p += '*';
vector<vector<double>> res(B);
for (int i=0 ; i<B ; i++){
    vector<double> ss, tt;
    for (auto x : s) ss.push_back(x==p[i] || x=='*');
    for (auto x : t) tt.push_back(x==p[i] || x=='*');
    reverse(tt.begin(), tt.end());
    res[i] = PolyMul(ss, tt);
}
for (int i=t.size()-1 ; i+t.size()-1<res[0].size() ; i++){
    int total = 0;
    for (int j=0 ; j<B-1 ; j++) total += (int)abs(round(res[j
        ][i]));
    total -= 25*(int)abs(round(res[26][i]));
    cout << (total==t.size());
}
```

8.9 Z Algorithm [9d559a]

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

8.10 k-th Substring1 [61f66b]

```
// 回傳 s 所有子字串（完全不同）中，第 k 大的
string k_th_substring(string &s, int k){
    int n = s.size();
    SuffixArray sa(s);
    sa.init_lcp();

    int prePrefix = 0, nowRank = 0;
    for (int i=0 ; i<n ; i++){
        int len = n-sa[i];
        int add = len-prePrefix;

        if (nowRank+add>=k){
            return s.substr(sa[i], prePrefix+k-nowRank);
        }

        prePrefix = sa.lcp[i];
        nowRank += add;
    }
}
```