

## Team Note of goraani

djs100201, dong\_gas, wbcho0504

Compiled on November 20, 2025

**Contents****1 Data Structure**

1.1	Segment Tree	2
1.1.1	Segment Tree	2
1.1.2	Lazy Propagation	2
1.1.3	dynamic segment tree	2
1.1.4	persistent segment tree	3
1.1.5	Li-Chao Tree	3
1.2	pbds	4
1.3	Line Container	4

**2 Geometry**

2.1	Basic Template	5
2.2	half plane intersection	6

**3 Graph**

3.1	centroid	7
3.2	Planer Graph	7
3.3	Centroid Tree	8
3.4	Directed MST	8
3.5	Eulerian Circuit	9
3.6	Offline Dynamic Connectivity	9
3.7	Articulation Point	10
3.8	Bridge	10
3.9	HLD	10
3.10	SCC + 2-sat	11
3.11	O(1) LCA with O(NlogN) preprocessing	12

**4 Math**

4.1	convolution(and, or, xor)	12
4.2	CRT	12
4.3	FFT	13
4.4	NTT	13
4.5	Online FFT	14

4.6	Extended Euclidean & MOD Inverse	15
4.7	Miller Rabin	15
4.8	Pollard rho	15
4.9	Gauss mod	16
4.10	Matrix Multiplication	16

**5 String**

5.1	hashing	17
5.2	KMP	18
5.3	manacher	18
5.4	suffix array & lcp	18
5.5	Z	18
5.6	Aho-Corasick	18

**6 Flow**

6.1	dinic	19
6.2	더블어민규당 MCMF	20
6.3	Hopcroft-Karp + Minimum Vertex Cover	21
6.4	Minimum Cost Maximum Flow	21

**7 DP optimization**

7.1	divide and conquer optimization	22
7.2	SOS	22

**8 ETC**

8.1	int128	23
8.2	priority queue my cmp	23
8.3	distance	23
8.4	random	23
8.5	mo's algorithm (hilbert)	23
8.6	degree sequence	24
8.7	bitset	24

# 1 Data Structure

## 1.1 Segment Tree

### 1.1.1 Segment Tree

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

/*
Segment Tree
*/

template<typename T> struct seg {
    ll n; //size
    T id; //identity
    vector<T> t;
    T(*merge)(T, T);
    seg(ll N, T ID, T(*_merge)(T, T)): n(N), id(ID), merge(_merge) { t.resize(N<<1, id); }
    void update(ll p, T val) {
        for (t[p+=n] = val; p > 1; p >>= 1) { //if you want change value, t[p+=n] = newval
            if(p&1) t[p>>1] = merge(t[p^1], t[p]);
            else t[p>>1] = merge(t[p], t[p^1]);
        }
    }
    T query(ll l, ll r) { //query on interval [l, r]
        T lret=id, rret=id;
        for(l += n, r += n; l < r; l >>= 1, r >>= 1) {
            if(l&1) lret = merge(lret, t[l++]);
            if(r&1) rret = merge(t[--r], rret);
        }
        return merge(lret, rret);
    }
};
```

### 1.1.2 Lazy Propagation

```
class lazy_seg {
public:
    vector<ll> tree, lazy, A;

    lazy_seg(int n) {
        // 0~n까지 쓰겠다.
        tree.resize(n * 4);
        lazy.resize(n * 4);
        A.resize(n * 4);
    }

    ll init(ll N, ll s, ll e) {
        if (s == e)
            return tree[N] = A[s];
        ll mid = (s + e) / 2;
        return tree[N] = init(N * 2, s, mid) + init(N * 2 + 1, mid + 1, e);
    }

    void update_lazy(ll N, ll s, ll e) {
```

```
        if (!lazy[N])
            return;
        tree[N] += (e - s + 1) * lazy[N];
        if (s != e) {
            lazy[N * 2] += lazy[N];
            lazy[N * 2 + 1] += lazy[N];
        }
        lazy[N] = 0;
    }

    void update(ll N, ll s, ll e, ll l, ll r, ll val) {
        update_lazy(N, s, e);
        if (l > e || r < s)
            return;
        if (l <= s && e <= r) {
            lazy[N] = val;
            update_lazy(N, s, e);
            return;
        }
        ll mid = (s + e) / 2;
        update(N * 2, s, mid, l, r, val);
        update(N * 2 + 1, mid + 1, e, l, r, val);
        tree[N] = tree[N * 2] + tree[N * 2 + 1];
    }

    ll f(ll N, ll s, ll e, ll l, ll r) {
        update_lazy(N, s, e);
        if (l > e || r < s)
            return 0;
        if (l <= s && e <= r)
            return tree[N];
        ll mid = (s + e) / 2;
        return f(N * 2, s, mid, l, r) + f(N * 2 + 1, mid + 1, e, l, r);
    }
};
```

### 1.1.3 dynamic segment tree

```
struct Node{
    int l, r; //index of leftson, rightson
    ll v; //sum of interval
    Node(){ l = r = -1; v = 0; }
};
Node nd[4040404]; //enough size
//root: nd[0]
int pv = 1; //pv node already used..
void update(int node, int s, int e, int x, int v){
    if(s == e){
        nd[node].v = v; return;
    }
    int m = s + e >> 1;
    if(x <= m){
        if(nd[node].l == -1) nd[node].l = pv++;
        update(nd[node].l, s, m, x, v);
    }else{
        if(nd[node].r == -1) nd[node].r = pv++;
        update(nd[node].r, m+1, e, x, v);
    }
```

```

    }
    ll t1 = nd[node].l != -1 ? nd[nd[node].l].v : 0;
    ll t2 = nd[node].r != -1 ? nd[nd[node].r].v : 0;
    nd[node].v = t1 + t2;
}
ll query(int node, int s, int e, int l, int r){
    if(node == -1) return 0;
    if(r < s || e < l) return 0;
    if(l <= s && e <= r) return nd[node].v;
    int m = s + e >> 1;
    return query(nd[node].l, s, m, l, r) + query(nd[node].r, m+1, e, l, r);
}

```

#### 1.1.4 persistent segment tree

```

namespace pst {
    struct node {
        int v;
        int l, r;
    } nd[45 * MAXN]; // (N+Q)logN,, 문제에 따라 여유있게.
    int root[MAXN], t = 0;

    int new_node(int v) {
        nd[++t] = {v, 0, 0};
        return t;
    }

    int init(int l, int r) {
        int ix = new_node(0);
        if (l == r) return ix;
        int m = (l + r) / 2;
        nd[ix].l = init(l, m);
        nd[ix].r = init(m + 1, r);
        return ix;
    }

    int upd(int prev, int l, int r, int i, int v) {
        int ix = new_node(nd[prev].v + v);
        if (l == r) return ix;
        int m = (l + r) / 2;
        if (i <= m) {
            nd[ix].l = upd(nd[prev].l, l, m, i, v);
            nd[ix].r = nd[prev].r;
        }
        else {
            nd[ix].l = nd[prev].l;
            nd[ix].r = upd(nd[prev].r, m + 1, r, i, v);
        }
        return ix;
    }

    int kth(int i1, int i2, int l, int r, int k) { // k번째 작은 수
        while (l ~ r) {
            int cnt = nd[nd[i2].l].v - nd[nd[i1].l].v;
            int m = (l + r) / 2;
            if (k <= cnt) r = m, i1 = nd[i1].l, i2 = nd[i2].l;
        }
    }
}

```

```

        else l = m + 1, k -= cnt, i1 = nd[i1].r, i2 = nd[i2].r;
    }
    return l;
}

int query(int ix, int nl, int nr, int l, int r) {
    if (nl > r || nr < l) return 0;
    if (nl >= l && nr <= r) return nd[ix].v;
    int m = nl + nr >> 1;
    return query(nd[ix].l, nl, m, l, r) + query(nd[ix].r, m + 1, nr, l, r);
}

/*
 * pst::root[0] = pst::init(1, n); // 첫 세그트리 빈 거로 하나 만들어 두기
 * 다 닫힌 구간임.
 *
 * pst::root[i] = pst::root[i - 1]; // 이전 거 복사
 * pst::root[i] = pst::upd(pst::root[i], 1, n, a[i], 1); // 한 줄기 업데이트
 * update하고 대입 필요함!
 *
 * pst::kth(pst::root[l - 1], pst::root[r], 1, n, k) // [l, r]에 있는 수 중 k번째 작은 수
 *
 * pst::query(pst::root[r], 1, n, l, r); // [l, r]중 답
 */

```

#### 1.1.5 Li-Chao Tree

```

// 최댓값 기준
const ll inf = 1e18;
struct line{
    ll a, b;
    ll get(ll x){return a * x + b;}
};

struct CHT {
    // x 범위 : [S, E]
    const ll S = -1e9;
    const ll E = 1e9;
    struct node{
        ll lc, rc, l, r;
        line ln;
    };
    vector<node> Tree;
    int t; // Tree 노드 개수

    CHT(){Tree.push_back(node(-1, -1, S, E, {0, -inf})); }
    void upd(int ix, line hi){
        ll l = Tree[ix].l, r = Tree[ix].r;
        ll m = (l + r) >> 1;

        line lw = Tree[ix].ln;
        if(lw.get(l) > hi.get(l)) swap(lw, hi);
        if(hi.get(r) >= lw.get(r)) {
            Tree[ix].ln = hi; return;
        }
    }
}

```

```

    if(hi.get(m) >= lw.get(m)){
        Tree[ix].ln = hi;
        if(Tree[ix].rc == -1) {
            Tree[ix].rc = ++t;
            Tree.push_back({-1, -1, m + 1, r, {0, -inf}}); // to modify : 더미 선분
        }
        upd(Tree[ix].rc, lw);
    }
    else{
        Tree[ix].ln = lw;
        if(Tree[ix].lc == -1){
            Tree[ix].lc = ++t;
            Tree.push_back({-1, -1, l, m, {0, -inf}}); // to modify : 더미 선분
        }
        upd(Tree[ix].lc, hi);
    }
}
ll qry(int ix, ll x){
    if(ix == -1) return -inf;
    ll m = (Tree[ix].l + Tree[ix].r) >> 1;
    if(x <= m) return max(Tree[ix].ln.get(x), qry(Tree[ix].lc, x));
    else return max(Tree[ix].ln.get(x), qry(Tree[ix].rc, x));
}
};

```

## 1.2 pbds

```

#include <bits/extc++.h>
/*
if error with <bits/extc++.h>...
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
*/
using namespace __gnu_pbds;
template<class T> using PBDS = tree<T, null_type, less<T>, rb_tree_tag,
tree_order_statistics_node_update>;
template<class T> using multiPBDS = tree<T, null_type, less_equal<T>, rb_tree_tag,
tree_order_statistics_node_update>;
PBDS<ll> s;
//s.order_of_key(x): number of less than x
//s.find_by_order(y): return yth element iterator (0-based).
//s={1, 2, 4, 6, 11} -> *s.find_by_order(2) -> 4
//multiPBDS
(1) use m_erase instead of erase
void m_erase(multiPBDS &OS, int val){
    int index = OS.order_of_key(val);
    multiPBDS::iterator it = OS.find_by_order(index);
    if(*it == val) OS.erase(it);
}
(2) find(x)
count! number of less than x with order_of_key := p
check! find_by_order(p) == x

```

## 1.3 Line Container

```

//max query.. if you want min query, use it with m -> -m, k -> -k, query(q) -> -query(q).
struct Line {

```

```

mutable ll k, m, p;
bool operator<((const Line& o) const { return k < o.k; }
bool operator<(ll x) const { return p < x; }
};
struct LineContainer : multiset<Line, less<>> {
    // (for doubles, use inf = 1/.0, div(a,b) = a/b)
    // (for doubles, const double inf = 1/.0;)
    // (for doubles, ll -> double)
    static const ll inf = LLONG_MAX;
    ll div(ll a, ll b) { // floored division
        return a / b - ((a ^ b) < 0 && a % b); }
    bool isect(iterator x, iterator y) {
        if (y == end()) return x->p = inf, 0;
        if (x->k == y->k) x->p = x->m > y->m ? inf : -inf;
        else x->p = div(y->m - x->m, x->k - y->k);
        return x->p >= y->p;
    }
    void add(ll k, ll m) {
        auto z = insert({k, m, 0}), y = z++, x = y;
        while (isect(y, z)) z = erase(z);
        if (x != begin() && isect(--x, y)) isect(x, y = erase(y));
        while ((y = x) != begin() && (--x)->p >= y->p)
            isect(x, erase(y));
    }
    ll query(ll x) {
        assert(!empty());
        auto l = *lower_bound(x);
        return l.k * x + l.m;
    }
};

```

## 2 Geometry

### 2.1 Basic Template

```

struct pt{
    ll x, y;
    pt() {x=0;y=0;}
    pt(ll p, ll q) {x=p, y=q;}
    pt operator + (pt t){return {x + t.x, y + t.y};}
    pt operator - (pt t){return {x - t.x, y - t.y};}
    ll operator * (pt t){return x * t.x + y * t.y;}
    ll operator / (pt t){return x * t.y - y * t.x;}
    bool operator == (const pt t)const{return x == t.x && y == t.y;}
    bool operator <(const pt t)const{return x == t.x ? y < t.y : x < t.x;}
    bool operator >(const pt t)const{return x == t.x ? y > t.y : x > t.x;}
    ll szz(){return x * x + y * y;}
    pt mul(ll m){return {x * m, y * m};}
};

struct pt{ // long double
    ld x, y;
    pt() {x=(ld)0;y=(ld)0;}
    pt(ld p, ld q) {x=p, y=q;}
    pt operator + (pt t){return {x + t.x, y + t.y};}
    pt operator - (pt t){return {x - t.x, y - t.y};}
    ld operator * (pt t){return x * t.x + y * t.y;}
    ld operator / (pt t){return x * t.y - y * t.x;}
    //watch out! <, ==, >
    bool operator == (const pt t)const{return x == t.x && y == t.y;}
    bool operator <(const pt t)const{return x == t.x ? y < t.y : x < t.x;}
    bool operator >(const pt t)const{return x == t.x ? y > t.y : x > t.x;}
    ld szz(){return x * x + y * y;}
    pt mul(ld m){return {x * m, y * m};}
};

ll ccw(pt a, pt b, pt c) {
    b = b - a, c = c - a;
    return b.x * c.y - c.x * b.y;
}

//외심 (실수) djs
pt get_circle_center(pt a,pt b,pt c){
    pt aa=b-a,bb=c-a;
    auto c1 = aa*aa * 0.5, c2 = bb*bb * 0.5;
    auto d = aa / bb;
    auto x = a.x + (c1 * bb.y - c2 * aa.y) / d;
    auto y = a.y + (c2 * aa.x - c1 * bb.x) / d;
    return pt(x, y);
}

// 다각형의 넓이 0(n)
double area(vector<pt>& v){
    double ret = 0;
    for(int i = 0, n = v.size(); i < n; i++){
        ret += v[i] / v[(i + 1) % n];
    }
    return abs(ret) / 2.0;
}

```

```

// 선분 교차 판정
bool intersect(pt p1, pt p2, pt p3, pt p4){
    int a = ccw(p1, p2, p3) * ccw(p1, p2, p4);
    int b = ccw(p3, p4, p1) * ccw(p3, p4, p2);
    if(!a && !b){
        if(p2 < p1) swap(p1, p2);
        if(p4 < p3) swap(p3, p4);
        return !(p2 < p3 || p4 < p1);
    }
    return a <= 0 && b <= 0;
}

// 두 선분의 교점 구하기
bool getpoint(pt p1, pt p2, pt p3, pt p4, pt& p){
    double d = (p4.y - p3.y) * (p2.x - p1.x) - (p4.x - p3.x) * (p2.y - p1.y);
    double t = (p4.x - p3.x) * (p1.y - p3.y) - (p4.y - p3.y) * (p1.x - p3.x);
    double s = (p2.x - p1.x) * (p1.y - p3.y) - (p2.y - p1.y) * (p1.x - p3.x);
    if(!d){
        // t == 0 : 동일한 선

        // t != 0 : 평행
        if(p2 < p1) swap(p1, p2);
        if(p4 < p3) swap(p3, p4);

        // 한 점에서 만나는 경우
        if(p2 == p3) {
            p = p2; return 1;
        }
        if(p4 == p1){
            p = p4; return 1;
        }
        return 0;
    }
    t /= d; s /= d;
    // t >= 0 && t <= 0 : 교점 존재
    p.x = p1.x + (p2.x - p1.x) * t;
    p.y = p1.y + (p2.y - p1.y) * t;
    return 1;
}

// 두 점 사이의 거리 (제공)
ll dist(pt a, pt b){ return (b - a).szz(); }

// 직선(선분)과 점의 거리
double linedist(pt a, pt b, pt c){
    pt t = b - a;
    // 선분일 경우
    if(t * (c - a) <= 0) return sqrt(dist(a, c));
    if(t * (c - b) >= 0) return sqrt(dist(b, c));
    //
    return abs(t / (c - a)) / sqrt(t.szz());
}

```

```
// ConvexHull O(nlogn)
vector<pt> hull(vector<pt> v){
    int ix = min_element(all(v)) - v.begin();
    swap(v[0], v[ix]);
    vector<pt> st;

    sort(v.begin() + 1, v.end(), [&] (pt& a, pt& b){
        pt x = a - v[0], y = b - v[0];
        return x / y ? x / y > 0 : x.sz() < y.sz();
    });
    for(auto& p : v){
        while(st.size() > 1 && ccw(st[st.size() - 2], st.back(), p) <= 0) st.pop_back();
        st.emplace_back(p);
    }

    /*
    # 마지막 점들이 일직선 상에 있는 경우에 예외처리를 해야 하는 경우
    int i = v.size() - 1;
    while(i >= 1 && !ccw(v[0], v[i], v[i - 1])) i--;
    reverse(v.begin() + i, v.end());
    */
    return st;
}

// 삼각형 내부의 점 판별 O(1)
int inTriangle(vector<pt>& t, pt p){
    int sign[3];
    for(int i = 0; i < 3; i++){
        sign[i] = ccw(t[i], t[(i + 1) % 3], p);
        if(sign[0] == sign[1] && sign[1] == sign[2]) return -1;
        for(int i = 0; i < 3; i++) if(sign[i] * sign[(i + 1) % 3] == -1) return 1;
        return 0;
    }
}

// 볼록 다각형 내부의 점 판별 O(n)
int inside(pt p, vector<pt>& v){
    if(v.size() < 3) return 0;
    for(int i = 0, n = v.size(); i < n; i++){
        if(ccw(v[i], v[(i + 1) % n], p) <= 0) return 0;
        return 1;
    }
}

// 볼록 다각형 내부의 점 판별 O(logn)
int inside(pt p, vector<pt>& v){
    int n = v.size();
    if(n < 3 || ccw(v[0], v[1], p) < 0 || ccw(v[0], v[n - 1], p) > 0) return 0;

    int l = 2, r = n - 1, m;
    while(l < r){
        m = (l + r) / 2;
        if(ccw(v[0], v[m], p) < 0) r = m;
        else l = m + 1;
    }
    return ccw(v[l - 1], p, v[l]) < 0;
}
}
```

```
//gumgood's code
ll ccw(pt a, pt b, pt c) {
    b = b - a, c = c - a;
    return b.x * c.y - c.x * b.y;
}

pt o(0,0);
sort(p.begin(), p.end(), [&] (const pt &p, const pt &q){
    if((p<o)^(q<o)) return q < p; //사이 각 pi이내로만 한정. 영역을 o기준 좌/
    우로 나누고 우 영역 점이 먼저 오게 정렬
    if(ll t = ccw(o,p,q)) return t > 0;
    return abs(p.x)<abs(q.x) || abs(p.y)<abs(q.y); //같은 직선이라면 가까운 게 먼저
});
```

## 2.2 half plane intersection

// 1. 도형의 각 변을 임의의 거리 r만큼 내부로 이동했을 때 만들어지는 영역  
 // 2. 모서리와 교차하지 않고 도형 내부의 모든 점까지 그을 수 있는 영역

```
typedef long double ld;
const double eps = 1e-10, inf = 1e9;
struct pd{
    ld x, y;
    pd operator +(pd t){return {x + t.x, y + t.y};}
    pd operator -(pd t){return {x - t.x, y - t.y};}
    ld operator *(pd t){return x * t.x + y * t.y;}
    ld operator /(pd t){return x * t.y - y * t.x;}
    ld sz() {return sqrt(x * x + y * y);}
};

struct line{
    pd p, d;
    ld a;

    line() {}
    //두 점을 지나는 직선
    line(pd p, pd q) : p(p), d(q - p){
        a = atan2l(d.y, d.x);
    }
    // half plane
    // 점이 반평면 밖에 있는 지 확인 (-eps : 교집합으로 점도 가능, eps : 교집합으로 점 불가능)
    bool out(pd t){return d / (t - p) < eps;}
    // 각도 순으로 정렬
    bool operator < (line t){return a < t.a;}
};

// 두 직선의 교점 구하기 (평행x)
pd interPoint(line a, line b){
    ld t = ((b.p - a.p) / b.d) / (a.d / b.d);
    a.p.x += t * a.d.x;
    a.p.y += t * a.d.y;
    return a.p;
}

// 반평면 교집합 구하기 O(NlogN)
vector<pd> half_plane(vector<line> v){
    // unbounded case 일 경우 boundary를 만들어주기
}
```

```

/*
pd box[4] = {
    pd{inf, inf},
    pd{-inf, inf},
    pd{-inf, -inf},
    pd{inf, -inf},
};
for(int i = 0; i < 4; i++) v.push_back(line(box[i], box[(i + 1) % 4]));
*/

vector<pd> ret;
deque<line> dq;
sort(all(v));
for(auto&l : v){
    while(dq.size() > 1 && l.out(interPoint(dq[dq.size() - 2], dq.back())))
        dq.pop_back();
    while(dq.size() > 1 && l.out(interPoint(dq[0], dq[1]))) dq.pop_front();

    // 평행한 두 직선 예외 처리하기
    if(dq.size() && fabs1(dq.back().d / l.d) < eps){
        if(dq.back().d * l.d < 0.0) return ret;

        if(l.out(dq.back().p)) dq.pop_back();
        else continue;
    }
    dq.emplace_back(l);

    while(dq.size() > 2 && dq[0].out(interPoint(dq[dq.size() - 2], dq.back())))
        dq.pop_back();
    while(dq.size() > 2 && dq.back().out(interPoint(dq[0], dq[1]))) dq.pop_front();
    if(dq.size() < 3) return ret;
    for(int i = 0, n = dq.size(); i < n; i++)
        ret.emplace_back(interPoint(dq[i], dq[(i + 1) % n]));
    return ret;
}

```

### 3 Graph

#### 3.1 centroid

```

int sz[NMAX], use[NMAX], cent_papa[NMAX];
int get_size(int u, int p=0) {
    sz[u]=1;
    for(int v:adj[u]) {
        if(use[v] || p==v) continue;
        sz[u]+=get_size(v,u);
    }
    return sz[u];
}
int get_cent(int u, int p, int cnt) {
    for(int v:adj[u]) {
        if(use[v] || v==p) continue;
        if(sz[v]>cnt/2) return get_cent(v,u,cnt);
    }
    return u;
}
void dnc(int u, int p=0) { //p: before cent
    int cent=get_cent(u,p,get_size(u,p));
    cent_papa[cent]=p;
    use[cent]=1;
    for(int v:adj[cent]) if(!use[v]) dnc(v,cent);
}

/*
node update, node query -> you need to update all of your centroid ancestor
property: u-v route must pass u's centroid ancestor
*/
void update(int u) {
    color[u]^=1;
    int now=u;
    do {
        int dist=get_dist(now,u);
        if(color[u]) s[now].insert(dist);
        else s[now].erase(s[now].find(dist));
        now=cent_papa[now];
    } while(now!=0);
}
int query(int u) {
    int now=u, ret=1e9;
    do {
        if(!s[now].empty()) ret=min(ret, get_dist(u,now) + *s[now].begin());
        now=cent_papa[now];
    } while(now!=0);
    return (ret<1e9) ? ret : -1;
}

```

#### 3.2 Planer Graph

- /\*
- 평면그래프  $\Leftrightarrow K_5, K_{3,3}$  이 없음
- 오일러 공식  
 $v - e + f = 2$
- 평면 그래프 ( $v \geq 3$ ) 의 특징 (충분조건은 X)

1.  $e \leq 3v - 6$
2. 이분 그래프  $\rightarrow e \leq 2v - 4$
3.  $f \leq 2v - 4$

- degree  $\leq 5$ 인 점이 적어도 하나 존재  
 if 모든 점이 degree  $\geq 6$ 이라고 가정하면  $e \geq 3v$ 여야하는데 1번 조건인  $e \leq 3v-6$ 과 모순

- 모든 정점의 degree  $\geq 3$ 이면 길이 5 이하의 사이클이 존재  
 모든 사이클의 길이가 6 이상이라고 하면  
 $2e \geq 6f$  이므로  $e \geq 3f = 3(e - v + 2)$  즉,  $2e \leq 3v - 6$   
 모든 정점의 degree가  $\geq 3$ 이면  $2e \geq 3v$ 여야 하는데 모순

- 사이클 찾는법 : 매번 degree가 가장 작은 점을 뽑고 지워감  
 $\min(\text{degree}) \leq 5$ 임이 보장됨.  
 연결된 간선을 out edge로 고정해줌  
 bfs를 통해 길이 3, 4 사이클 찾기  $P(5^3 * N + 5 * 2 * N \log N)$   
 \*/

### 3.3 Centroid Tree

// level of tree :  $O(\log(N))$

```
int n, m, op, u, v, color[NMAX];
int W[NMAX], vis[NMAX];
// in Centroid tree
vector<pair<int, int>> Up[NMAX]; // <anc, dist>
multiset<int> S[NMAX]; // for subtree of vertex x
```

```
int DFS(int x, int p) {
    W[x] = 1;
    for (int &nx : adj[x])
        if (nx != p && !vis[nx]) W[x] += DFS(nx, x);
    return W[x];
}
```

```
// find centroid
int Cent(int x, int p, int k) {
    for (int &nx : adj[x])
        if (nx != p && !vis[nx] && W[nx] > k) return Cent(nx, x, k);
    return x;
}
```

```
// Centroid Tree에서 ct의 서브트리 값 갱신
void DFS2(int x, int p, int d, int ct) {
    Up[x].emplace_back(ct, d);
    for (int &nx : adj[x])
        if (nx != p && !vis[nx]) DFS2(nx, x, d + 1, ct);
    return;
}
```

```
// Centroid Tree 만들기
void make_tree(int x) {
    int k = DFS(x, -1) / 2;
    int ct = Cent(x, -1, k); // ct : 현재 centroid 정점
    DFS2(ct, -1, 0, ct); // ct의 서브트리 update
    vis[ct] = 1;
    for (int &nx : adj[ct])
```

```
        if (!vis[nx]) make_tree(nx);
    return;
}
```

### 3.4 Directed MST

```
// Chu[Liu Edmonds' algorithm
// Directed MST (ElogE)
typedef long long ll;
const int NMAX = 2e5 + 5;
struct edge{
    int u, v;
    ll c;
    bool operator <(const edge& t) const{return c > t.c;}
};
int U[NMAX * 2], P[NMAX * 2], par[NMAX * 2];
int find(int x){return par[x] == -1 ? x : par[x] = find(par[x]);}
ll sum[NMAX * 2];
```

// 0-based, 각 정점의 부모를 배열로 반환  
 // 모든 정점이 루트에서 도달가능한 경우

```
vector<int> DMST(int n, int r, const vector<edge> & E){
    memset(U, -1, sizeof(U));
    memset(P, -1, sizeof(P));
    memset(par, -1, sizeof(par));
    edge par_edge[2 * n];
    vector<int> grp[2 * n];
```

```
    priority_queue<edge> Q[2 * n];
    for(auto& e : E) Q[e.v].emplace(e);
    for(int i = 0; i < n; i++) Q[(i + 1) % n].push({i, (i + 1) % n, (ll)1e18});
```

```
    int cur = 0, t = n;
    while(Q[cur].size()){
        auto e = Q[cur].top(); e.c += sum[cur];
        Q[cur].pop();
        int bef = find(e.u);
        if(cur == bef) continue;
        U[cur] = bef;
        par_edge[cur] = e;

        if(U[bef] == -1) cur = bef;
        else{
            // make new node t
            int x = bef;
            do{
                sum[x] -= par_edge[x].c;
                // merge edges with small to large
                if(Q[x].size() > Q[t].size()) swap(Q[x], Q[t]), swap(sum[x], sum[t]);
                while(Q[x].size()){
                    auto tmp = Q[x].top(); tmp.c += sum[x];
                    Q[x].pop();
                    tmp.c -= sum[t];
                    Q[t].emplace(tmp);
                }
                grp[t].emplace_back(x);
                x = find(U[x]);
```

```

    } while(x != bef);
    for(auto x : grp[t]) par[x] = t, P[x] = t;
    cur = t++;
}
}

vector<int> res(n, -1);
queue<int> q; q.emplace(r);
while(q.size()){
    int x = q.front(); q.pop();
    while(P[x] != -1){
        for(auto&y : grp[P[x]]){
            if(y == x) continue;
            if(par_edge[y].v < n)
                res[par_edge[y].v] = par_edge[y].u;
            // v를 루트로하는 서브트리에서 다시 탐색
            q.emplace(par_edge[y].v);
            P[y] = -1;
        }
        x = P[x];
    }
}
res[r] = r;
return res;
}

```

### 3.5 Eulerian Circuit

```

vector<P>v[n_];
vector<ll>checked;
void dfs(ll x){
    while(1){
        while(v[x].size() && checked[v[x].back().second])v[x].pop_back();
        //쓴거 다 지우기
        if(v[x].empty())break;
        auto [a,b]=v[x].back();
        v[x].pop_back();
        checked[b]=true;
        dfs(a);
    }
    cout<<x<<' ';
}

void solve(){
    cin>>n;
    for(int i=1;i<=n;i++){
        for(int j=1;j<=n;j++){
            cin>>a;
            if(i<j)continue;
            for(int k=0;k<a;k++){
                v[i].push_back({j,base});
                v[j].push_back({i,base});
                base++;
            }
        }
    }
    checked.resize(base+1);
    for(int i=1;i<=n;i++){
        if(v[i].size()%2){

```

```

        cout<<"-1";
        return;
    }
}
dfs(1);
}

```

### 3.6 Offline Dynamic Connectivity

```

const int B = 1 << 17;
pair<int, int> E[B]; // query 저장
vector<pair<int, int>> seg[B * 2]; // 각 쿼리 구간 [l, r]을 포함하는 간선
int par[B], sz[B], ans[B];
vector<int> st;

int find(int x){return par[x] == -1 ? x : find(par[x]);}

/*
pair<int, int> find(int x) { // return : < root, color>
    if (par[x] == -1) return {x, 0};
    auto [p, c] = find(par[x]);
    return pair(p, c ^ C[x]);
}
*/

void init(int n){
    for (int i = 1; i <= n; i++) par[i] = -1, sz[i] = 1;
}

void Union(int a, int b) { // rank compression
    a = find(a); b = find(b);
    if (sz[a] < sz[b]) swap(a, b);
    par[b] = a; sz[a] += sz[b];
    st.emplace_back(b);
}

void Undo() {
    int y = st.back(); st.pop_back();
    sz[par[y]] -= sz[y];
    par[y] = -1;
}

void upd(int ix, int nl, int nr, int l, int r, pair<int, int> v) {
    if (nl > r || nr < l) return;
    if (nl >= l && nr <= r) {
        seg[ix].emplace_back(v);
        return;
    }
    int m = (nl + nr) / 2;
    upd(ix * 2, nl, m, l, r, v);
    upd(ix * 2 + 1, m + 1, nr, l, r, v);
}

void dfs(int ix, int l, int r) {
    int cnt = 0;

```

```

for (auto& [a, b] : seg[ix])
    if (find(a) != find(b)) Union(a, b), cnt++;
if (l == r) { // leaf node : answer query
    auto&[a, b] = E[l];
    if (a) ans[l] = find(a) == find(b); // 하나의 component에 속하는지 확인
}
else {
    int m = (l + r) / 2;
    dfs(ix * 2, l, m);
    dfs(ix * 2 + 1, m + 1, r);
}
while (cnt--) Undo();
}

map<pair<int, int>, int> S; // a < b
int n, m, op, a, b;

// <main>
// init(n);
// 마지막까지 있는 간선 update

```

### 3.7 Articulation Point

```

//단절점
vector<ll>v[n_],res;
ll checked[n_],dep[n_],low[n_];
void dfs(ll x,ll par){
    dep[x]=low[x]=base++;
    bool flag=0;
    int cnt=0;
    for(auto nxt:v[x]){
        if(nxt==par)continue;
        if(checked[nxt]){
            low[x]=min(low[x],dep[nxt]);
            continue;
        }
        checked[nxt]=true;
        dfs(nxt,x);
        if(par && dep[x]<=low[nxt])flag=1;
        low[x]=min(low[x],low[nxt]);
        cnt++;
    }
    if(par==0 && cnt>=2)flag=true;
    if(flag)res.push_back(x);
}

void solve(){
    cin>>n>>m;
    while(m--){
        cin>>a>>b;
        v[a].push_back(b);
        v[b].push_back(a);
    }
    for(int i=1;i<=n;i++){
        if(checked[i])continue;
        checked[i]=true;

```

```

        dfs(i,0);
    }
    cout<<res.size()<<endl;
    sort(all(res));
    for(auto nxt:res)cout<<nxt<<' ';
}

```

### 3.8 Bridge

```

//단절선
vector<pair<ll, ll>>ans;
ll dfs(ll x, ll par) {
    A[x] = ++d;
    ll ret = A[x];
    for (ll nxt : v[x]) {
        if (nxt == par)continue;
        if (!A[nxt]) {
            a = dfs(nxt, x);
            if (a > A[x])
                ans.push_back({ min(x,nxt),max(x,nxt) });
            ret = min(ret, a);
        }
        else ret = min(ret, A[nxt]);
    }
    return ret;
}

int main() {
    ios_base::sync_with_stdio(0);
    cin.tie(0), cout.tie(0);
    cin >> V >> E;
    for (int i = 0; i < E; i++) {
        cin >> a >> b;
        v[a].push_back(b);
        v[b].push_back(a);
    }
    dfs(1, 0);
    sort(ans.begin(), ans.end());
    cout << ans.size() << '\n';
    for (int i = 0; i < ans.size(); i++) {
        cout << ans[i].first << ' ' << ans[i].second << '\n';
    }
}

```

### 3.9 HLD

```

// lazy seg 사용 코드. 기준이 노드인지 간선인지 확인 후 적절히 변형 필요.
struct Heavy_Light_Decomposition {
    ll sz[MAXN], dep[MAXN], papa[MAXN], top[MAXN], in[MAXN], out[MAXN], pv;
    vector<ll> inp[MAXN], g[MAXN];
    //inp: input graph(bidirectional), g: convert inp to one-way graph
    void init() {
        top[1] = 1;
        dfs(1, 0), dfs1(1), dfs2(1);

```

```

}

void dfs(int u, int p) {
    for (auto& v: inp[u])
        if (v != p) {
            dfs(v, u);
            g[u].emplace_back(v);
        }
}

void dfs1(int u) {
    sz[u] = 1;
    for (auto& v: g[u]) {
        dep[v] = dep[u] + 1, papa[v] = u;
        dfs1(v), sz[u] += sz[v];
        if (sz[v] > sz[g[u][0]]) swap(v, g[u][0]);
    }
}

void dfs2(int u) {
    in[u] = ++pv;
    for (auto v: g[u]) {
        top[v] = (v == g[u][0]) ? top[u] : v;
        dfs2(v);
    }
    out[u] = pv;
}

void update(int u, int v, int val) {
    while (top[u] ^ top[v]) {
        if (dep[top[u]] < dep[top[v]]) swap(u, v);
        int st = top[u];
        seg.update(1, 1, n, in[st], in[u], val);
        u = papa[st];
    }
    // if (u == v) return; // 간선 기준일 때는 여기서 return 필요
    if (dep[u] > dep[v]) swap(u, v);
    seg.update(1, 1, n, in[u] + 1, in[v], val); // 노드 기준이면 in[u]로 해야 함.
}

ll query(int u, int v) {
    ll ret = 1e18;
    while (top[u] ^ top[v]) {
        if (dep[top[u]] < dep[top[v]]) swap(u, v);
        int st = top[u];
        ret = min(ret, seg.query(1, 1, n, in[st], in[u]));
        u = papa[st];
    }
    // if (u == v) return ret; // 간선 기준일땐 u == v면 반영 안 해야 함.
    if (dep[u] > dep[v]) swap(u, v);
    ret = min(ret, seg.query(1, 1, n, in[u] + 1, in[v])); // 노드 기준이면 in[u]로 해야 함.
    return ret;
}
} hld;

```

### 3.10 SCC + 2-sat

```

//in 2-sat
//(a & b) -> (a or a) & (b or b)
//(a ^ b) -> (a or b) & (!a or !b)
ll n, m, id, SN = 1;
ll d[200020], sn[200020], ans[200020];
bool finished[200020];
vector<ll> adj[200020];
vector<vector<ll>> SCC;
stack<ll> s;
//reverse SCC -> topology sort
ll dfs(ll x) {
    d[x] = ++id;
    s.push(x);
    ll parent = d[x];
    for (ll i = 0; i < adj[x].size(); i++) {
        ll y = adj[x][i];
        if (d[y] == 0) parent = min(parent, dfs(y));
        else if (!finished[y]) parent = min(parent, d[y]);
    }
    if (parent == d[x]) {
        vector<ll> scc;
        while (1) {
            ll t = s.top();
            s.pop();
            scc.push_back(t);
            finished[t] = true;
            sn[t] = SN;
            if (t == x) break;
        }
        SCC.push_back(scc);
        SN++;
    }
    return parent;
}

ll rev(ll x) { //get not in 2-sat
    if (x <= n) return x + n;
    return x - n;
}

void solve() {
    cin >> n >> m;
    while (m--) {
        ll u, v; cin >> u >> v;
        adj[u].push_back(v);
    }
    for (ll i = 1; i <= 2 * n; i++) if (!d[i]) dfs(i);

    //2-sat start
    //(a or b) -> !a->b, !b->a
    for (ll i = 1; i <= n; i++) {
        if (sn[i] == sn[i + n]) { //same SCC
            cout << 0 << endl;
            return;
        }
    }
    cout << 1 << endl;
}

```

```

reverse(SCC.begin(), SCC.end()); //topology sort
for (auto vec : SCC) {
    for (auto now : vec) { //Starting from the front, fill in 0 first.
        if (ans[now]) continue;
        ans[now] = 0;
        ans[rev(now)] = 1;
    }
}
}

3.11 O(1) LCA with O(NlogN) preprocessing
const int NMAX = 100201;
int n, q, ord;
int depth[NMAX], idx[NMAX], log_2[2*NMAX], euler[2*NMAX];
int dp[22][2*NMAX];
vector<int> adj[NMAX];
void dfs(int now, int papa=0) {
    depth[now]=depth[papa]+1;
    idx[now]=++ord;
    euler[ord]=now;
    for(int nxt:adj[now]) if(nxt != papa) {
        dfs(nxt, now);
        euler[++ord]=now;
    }
}
void init() {
    int j=-1;
    for(int i=1;i<=ord;i++) {
        if(1<<(j+1)==i) j++;
        log_2[i]=j;
    }
    for(int i=1;i<=ord;i++) dp[0][i]={depth[euler[i]],euler[i]};
    for(int j=1;j<22;j++) for(int i=1;i+(1<<(j-1))<=ord;i++) dp[j][i]=min(dp[j-1][i],
    dp[j-1][i+(1<<(j-1))]);
}
int get_lca(int u, int v) {
    int l=idx[u], r=idx[v];
    if(l>r) swap(l,r);
    int len=log_2[r-l+1];
    return min(dp[len][l], dp[len][r-(1<<len)+1]).second;
}
void solve() {
    cin>>n;
    for(int i=0;i<n-1;i++) {
        int u, v; cin>>u>>v;
        adj[u].emplace_back(v), adj[v].emplace_back(u);
    }
    dfs(1), init();
    cin>>q;
    while(q --> 0) {
        int u, v; cin>>u>>v;
        cout<<get_lca(u,v)<<'\\n';
    }
}

```

## 4 Math

### 4.1 convolution(and, or, xor)

```

void fwht_and(vector<ll> &a, bool inv) {
    ll n = a.size();
    ll dir = inv ? -1 : 1;
    for(ll s = 2, h = 1; s <= n; s <= 1, h <= 1)
        for(ll l = 0; l < n; l += s)
            for(ll i = 0; i < h; i++)
                a[l + i] += dir * a[l + h + i];
}
void fwht_or(vector<ll> &a, bool inv) {
    ll n = a.size();
    ll dir = inv ? -1 : 1;
    for(ll s = 2, h = 1; s <= n; s <= 1, h <= 1)
        for(ll l = 0; l < n; l += s)
            for(ll i = 0; i < h; i++)
                a[l + h + i] += dir * a[l + i];
}
void fwht_xor(vector<ll> &a, bool inv) {
    ll n = a.size();
    for(ll s = 2, h = 1; s <= n; s <= 1, h <= 1) {
        for(ll l = 0; l < n; l += s) {
            for(ll i = 0; i < h; i++) {
                ll t = a[l + h + i];
                a[l + h + i] = a[l + i] - t;
                a[l + i] += t;
                if(inv) a[l + h + i] /= 2, a[l + i] /= 2;
            }
        }
    }
}
vector<ll> convolution(vector<ll> a, vector<ll> b) {
    fwht_xor(a, false); //and or xor
    fwht_xor(b, false); //and or xor
    for(ll i=0;i<(1ll<n);i++) a[i]*=b[i];
    fwht_xor(a, true); //and or xor
    return a;
}

4.2 CRT
//ax+by=1의 값을 x,y에 저장해준다. gcd(a,b)를 return 해준다.
ll ex_gcd(ll a, ll b, ll& x, ll& y) {
    if (!b) {
        x = 1, y = 0;
        return a;
    }
    ll ret = ex_gcd(b, a % b, x, y);
    ll temp = y;
    y = x - (a / b) * y;
    x = temp;
    if (x <= 0) {
        x += b;
        y -= a;
    }
    return ret;
}

```

```

11 crt(vector<ll>&A, vector<ll>&B) {
    //b로 나뉘었을 때, 나머지가 a
    ll a1 = A[0], b1 = B[0];
    for (int i = 1; i < A.size(); i++) {
        ll x, y, a2 = A[i], b2 = B[i], G = gcd(b1, b2);
        if ((abs(a2 - a1)) % G) return -1;
        ex_gcd(b1 / G, b2 / G, x, y);
        x *= (a2 - a1) / G;
        x %= (b2 / G);
        x = (x + b2 / G) % (b2 / G);
        ll t = b1 * b2 / G;
        a1 = b1 * x + a1;
        a1 %= t;
        b1 = t;
    }
    return a1;
}

void solve() {
    n = 3;
    vector<ll>A(n), B(n);
    for (int i = 0; i < n; i++) cin >> B[i];
    for (int i = 0; i < n; i++) cin >> A[i];
    cout << crt(A, B) << '\n';
}

```

### 4.3 FFT

```

// if TLE, long double -> double (less precise...)
//생성함수 유명한 등식
// —— Formal power series 기본(수렴 가정 불필요; 해석적 의미로는  $|x| < 1$ ) ——
// [기하급수 Geometric series]
/*
 $1/(1 - x) = \sum_{n \geq 0} x^n$ 
 $1/(1 - ax) = \sum_{n \geq 0} a^n x^n$ 
 $(1 - x)^{-m} = \sum_{n \geq 0} C(n+m-1, m-1) x^n$ 
 $[x^n] (1 - ax)^{-m} = a^n * C(n+m-1, m-1)$ 
 $(1 - x^L)/(1 - x) = 1 + x + \dots + x^{L-1}$ 
*/

// [로그/지수 Log/Exp]
/*
 $-\log(1 - x) = \sum_{n \geq 1} x^n / n$ 
 $\log(1/(1 - x)) = \sum_{n \geq 1} x^n / n$  // 같은 식
 $-\log(1 - x^k) = \sum_{r \geq 1} x^{kr} / r$  //  $t = x^k$  치환
 $\exp(x) = \sum_{n \geq 0} x^n / n!$ 
 $\exp(ax) = \sum_{n \geq 0} a^n x^n / n!$ 
 $\log(1 + U) = \sum_{m \geq 1} (-1)^{m+1} U^m / m$  // 상수항(U(0)) = 0일 때 형식적 정의
*/

```

```

#define double long double
typedef complex<double> base;
void fft(vector<base>& a, bool invert)
{
    int n = sz(a);
    vector<base> roots(n / 2);
    for (int i = 1, j = 0; i < n; i++) {
        int bit = n >> 1;

```

```

        for (; j >= bit; bit >>= 1) j -= bit;
        j += bit;
        if (i < j) swap(a[i], a[j]);
    }
    for (int i = 2; i <= n; i <= 1) {
        vector<base> w(i / 2);
        for (int j = 0; j < i / 2; ++j) {
            double th = 2 * acos(-1.L) * j / i * (invert ? -1 : 1);
            w[j] = base(cos(th), sin(th));
        }
        for (int j = 0; j < n; j += i) {
            for (int k = 0; k < i / 2; ++k) {
                base u = a[j + k], v = a[j + k + i / 2] * w[k];
                a[j + k] = u + v;
                a[j + k + i / 2] = u - v;
            }
        }
    }
    if (invert) for (int i = 0; i < n; i++) a[i] /= n;
}

void multiply(const vector<ll>& a, const vector<ll>& b, vector<ll>& res)
{
    vector<base> fa(all(a)), fb(all(b));
    ll n = 1;
    while (n < max(sz(a), sz(b))) n <= 1;
    n <= 1;
    fa.resize(n); fb.resize(n);
    fft(fa, false); fft(fb, false);
    for (int i = 0; i < n; i++) fa[i] *= fb[i];
    fft(fa, true);
    res.resize(n);
    for (int i = 0; i < n; i++) res[i] = ll(fa[i].real() + (fa[i].real() > 0 ? 0.5 : -0.5));
}

```

### 4.4 NTT

```

ll pw(ll a, ll b, ll mod){
    ll ret = 1;
    while(b){
        if(b & 1) ret = ret * a % mod;
        b >>= 1; a = a * a % mod;
    }
    return ret;
}

```

```

template<ll mod, ll w>
class NTT{
public:
    void ntt(vector<ll> &f, bool inv = 0){
        int n = f.size(), j = 0;
        vector<ll> root(n >> 1);
        for(int i=1; i<n; i++){
            int bit = (n >> 1);
            while(j >= bit){
                j -= bit; bit >>= 1;
            }
            j += bit;

```

```

        if(i < j) swap(f[i], f[j]);
    }
    ll ang = pw(w, (mod - 1) / n, mod); if(inv) ang = pw(ang, mod - 2, mod);
    root[0] = 1; for(int i=1; i<(n >> 1); i++) root[i] = root[i-1] * ang % mod;
    for(int i=2; i<=n; i<=1){
        int step = n / i;
        for(int j=0; j<n; j+=i){
            for(int k=0; k<(i >> 1); k++){
                ll u = f[j | k], v = f[j | k | i >> 1] * root[step * k] % mod;
                f[j | k] = (u + v) % mod;
                f[j | k | i >> 1] = (u - v) % mod;
                if(f[j | k | i >> 1] < 0) f[j | k | i >> 1] += mod;
            }
        }
    }
    ll t = pw(n, mod - 2, mod);
    if(inv) for(int i=0; i<n; i++) f[i] = f[i] * t % mod;
}
vector<ll> multiply(vector<ll> &a, vector<ll> &b){
    vector<ll> a(all(_a)), b(all(_b));
    int n = 2;
    while(n < a.size() + b.size()) n <= 1;
    a.resize(n); b.resize(n);
    ntt(a); ntt(b);
    for(int i=0; i<n; i++) a[i] = a[i] * b[i] % mod;
    ntt(a, 1);
    return a;
}
};

```

NTT<mod, w> ntt; //mod값, 원시근 넣고 사용

```

/*
mod / w
998'244'353 3
985'661'441 3
1'012'924'417 5
2'281'701'377 3
2'483'027'969 3
2'113'929'217 5
104'857'601 3
1'092'616'193 3
2013265921 31
*/

```

## 4.5 Online FFT

```

int n;
vector<ll> A,F,G;
ll Psum = 0; // prefix sum of G: Psum = sum_{k=0}^{l-1} G[k]

```

NTT<MOD, 3> ntt;

```

// CDQ on [l, r)
void DnC(int l, int r) {
    if (l + 1 == r) {
        ll add = F[l] * F[0] % MOD;
        if (l) add = (add + F[0] * F[l]) % MOD;
        G[l] = (G[l] + add) % MOD;
        Psum = (Psum + G[l]) % MOD;
        F[l + 1] = A[l + 1] * Psum % MOD;
        return;
    }

    int m = (l + r) >> 1;

    DnC(l, m);
    {
        int fLen = m - 1;
        int gLen = min(m, r - l);
        vector<ll> f(fLen), g(gLen);
        for (int i = 0; i < fLen; i++) f[i] = F[l + i];
        for (int i = 0; i < gLen; i++) g[i] = F[i];
        vector<ll> h = ntt.multiply(f, g);
        int startT = m - 1;
        int endT = r - l - 1;
        startT = max(startT, 0);
        endT = min(endT, (int)h.size() - 1);
        for (int t = startT; t <= endT; t++) {
            int k = l + t;
            G[k] = (G[k] + h[t]) % MOD;
        }
    }

    {
        int fLen = m - 1;
        int gLen = min(l, r - l);
        vector<ll> f(fLen), g(gLen);
        for (int i = 0; i < fLen; i++) f[i] = F[l + i];
        for (int i = 0; i < gLen; i++) g[i] = F[i];
        vector<ll> h = ntt.multiply(f, g);
        int startT = m - 1;
        int endT = r - l - 1;
        startT = max(startT, 0);
        endT = min(endT, (int)h.size() - 1);
        for (int t = startT; t <= endT; t++) {
            int k = l + t;
            G[k] = (G[k] + h[t]) % MOD;
        }
    }

    DnC(m, r);
}

void solve() {
    cin >> n;
    A.assign(n + 1, 0);
    for (int i = 1; i <= n; i++) cin >> A[i];

    F.assign(n + 1, 0);

```

```

    G.assign(n + 1, 0);
    F[0] = 1;
    Psum = 0;

    DnC(0, n);

    for (int i = 1; i <= n; i++) cout<<F[i]<<' ';
}

```

#### 4.6 Extended Euclidean & MOD Inverse

```

#define MOD 998244353
#define ll long long
#define tll array<ll, 3>
// ax + by = g
// eeu(a, b) return : [g, y, x] (y, x 순서 반대인거 주의)

```

```

// (x, y) 값 음수 나올 수 있음 주의
// x 양수화 : (x + b) % b, y 양수화 : (y + a) % a

```

```

tll eeu(ll a, ll b) {
    if (a == 0) return { b, 1, 0 };
    else {
        auto [g, x, y] = eeu(b % a, a);
        return { g, y, x - b / a * y };
    }
}

```

```

// MOD inverse
auto [_, __, a_inv] = eeu(a, MOD);
a_inv = (a_inv + MOD) % MOD;

```

#### 4.7 Miller Rabin

```

//O(k(logN)^3)
//k: number of test
using ull = unsigned long long;
ull mul(ull x, ull y, ull mod){ return (ull)((__int128) x * y % mod); }
ull ipow(ull x, ull y, ull p){
    ull ret = 1, piv = x % p;
    while(y){
        if(y&1) ret = mul(ret, piv, p);
        piv = mul(piv, piv, p);
        y >>= 1;
    }
    return ret;
}
bool miller_rabin(ull x, ull a){
    if(x % a == 0) return 0;
    ull d = x - 1;
    while(1){
        ull tmp = ipow(a, d, x);
        if(d&1) return (tmp != 1 && tmp != x-1);
        else if(tmp == x-1) return 0;
        d >>= 1;
    }
}

```

```

}
}
bool isprime(ll x){ //long long range
    for(auto &i : {2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37}){
        if(x == i) return 1;
        if(x > 40 && miller_rabin(x, i)) return 0;
    }
    if(x <= 40) return 0;
    return 1;
}
bool isprime(int x){ //int range
    for(auto &i : {2, 7, 61}){
        if (x == i) return 1;
        if (miller_rabin(x, i)) return 0;
    }
    return 1;
}

```

#### 4.8 Pollard rho

// \*\*밀러 라빈에 있는 몇몇 코드 가져와야 할 수도 있음\*\* //

//integer factorization  $O(N^{1/4})$

```

ll gcd(ll a, ll b) {
    if(!b) return a;
    return gcd(b, a%b);
}
void rec(ll n, vector<ll>& v) {
    if (n == 1) return;
    if (n % 2 == 0) {
        v.push_back(2);
        rec(n / 2, v);
        return;
    }
    if (isprime(n)) {
        v.push_back(n);
        return;
    }
    ll a, b, c, g = n;
    auto f = [&](ll x) {
        return (c + mul(x, x, n)) % n;
    };
    do {
        if (g == n) {
            a = b = rand() % (n - 2) + 2;
            c = rand() % 20 + 1;
        }
        a = f(a);
        b = f(f(b));
        g = gcd(abs(a - b), n);
    } while (g == 1);
    rec(g, v);
    rec(n / g, v);
}
vector<ll> factorize(ll n) {
    vector<ll> ret;
    rec(n, ret);
    sort(ret.begin(), ret.end());
}

```

```
    return ret;
}
```

## 4.9 Gauss mod

```
//O(n^3)
vector<int> gauss_mod(vector<vector<int>> &a,int mod){
    vector<int> inv(mod); // modulo inverse
    inv[1] = 1;
    for(int i = 2; i < mod; ++i)
        inv[i] = mod - (mod/i) * inv[mod%i] % mod;

    int n = a.size();
    int m = a[0].size();
    //cout<<n<<' '<<m<<endl;

    vector<int> w(m, -1); // i번째 열에 있는 pivot이 몇 번째 행에 있는지 저장
    for(int c = 0, r = 0; c < m && r < n; ++c){
        int p = r; // pivot row
        for(int i = r; i < n; ++i)
            if(a[p][c] < a[i][c])
                p = i;
        if(a[p][c] == 0) continue; // free variable

        for(int j = 0; j < m; ++j)
            swap(a[p][j], a[r][j]);
        w[c] = r;

        int t = a[r][c];
        for(int j = 0; j < m; ++j)
            a[r][j] = a[r][j] * inv[t] % mod;

        for(int i = 0; i < n; ++i) if(i != r){
            int t = a[i][c];
            for(int j = c; j < m; ++j)
                a[i][j] = (a[i][j] - a[r][j] * t % mod + mod) % mod;
        }
        ++r;
    }

    // existence of solution
    for(int i = 0; i < n; ++i)
        if(count(a[i].begin(), --a[i].end(), 0) == m-1 && a[i][m-1])
            return vector<int>(); // no solution

    vector<int> ans(m);
    for(int i = 0; i < m; ++i)
        if(~w[i]) ans[i] = a[w[i]][m-1];
    return ans; // solution exist
}
```

```
void solve(){
    cin>>n;
    vector<vector<int>>>G;
    for(int i=1;i<=n;i++){
        vector<int>T(n+1);
        for(int j=0;j<=n;j++)cin>>T[j];
        G.push_back(T);
    }
    vector<int>res=gauss_mod(G,101);
    for(int i=0;i<n;i++)cout<<res[i]<<' ';
    cout<<'\n';
}
```

## 4.10 Matrix Multiplication

```
matrix operator*(const matrix &a, const matrix &b) {
    ll size = a.size(), size2 = b[0].size(), size3 = b.size();
    matrix res(size, vector<ll>(size2));
    for (int i = 0; i < size; i++)
        for (int j = 0; j < size2; j++) {
            for (int k = 0; k < size3; k++) {
                res[i][j] += a[i][k] * b[k][j];
                res[i][j] %= MOD;
            }
            res[i][j] %= MOD;
        }
    return res;
}

matrix power(matrix a, ll n) { //a행렬을 n제곱 하겠다!
    ll size = a.size();
    matrix res(size, vector<ll>(size));
    for (int i = 0; i < size; i++)res[i][i] = 1; //단위 행렬 생성
    while (n) {
        if (n % 2)res = res * a;
        n /= 2;
        a = a * a;
    }
    return res;
}

using poly = vector<ll>;
poly mul(const poly& a, const poly& b) {
    poly ret(a.size() + b.size() + 1);
    for (int i = 0; i < a.size(); i++)
        for (int j = 0; j < b.size(); j++)ret[i + j] = (ret[i + j] + a[i] * b[j]) % mod;
    return ret;
}

poly div(const poly& a, const poly& b) {
    poly ret = a;
    for (int i = ret.size() - 1; i >= b.size() - 1; i--)
        for (int j = 0; j < b.size(); j++) {
            ret[i + j - b.size() + 1] = ((ret[i + j - b.size() + 1] - ret[i] * b[j]) % mod + mod) % mod;
        }
}
```

```

    ret.resize(b.size() - 1);
    return ret;
}
//키타마사법  $O(k^2n^3)$ 
ll kitamasa(poly c, poly a, ll n) {
    //초기항 a[i]와 상수 c[i]
    poly result = { 1 };
    poly xn = { 0, 1 }; //xn=  $x^1, x^2, x^n$ 
    poly f(c.size() + 1);
    f.back() = 1;
    for (int i = 0; i < c.size(); i++) f[i] = ((-c[i]) % mod + mod) % mod;
    while (n) {
        if (n % 2) result = div(mul(result, xn), f);
        n /= 2;
        xn = div(mul(xn, xn), f);
    }
    ll ret = 0;
    for (int i = 0; i < a.size(); i++) {
        ret += a[i] * result[i];
        ret %= mod;
        ret += mod;
        ret %= mod;
    }
    return ret;
}

```

## 5 String

### 5.1 hashing

```

mt19937 rng(chrono::steady_clock::now().time_since_epoch().count());
struct Hashing {
    const ll M=998244353;
    ll P;
    vector<ll> H, B;
    void Build(const string& S, ll cnt) { //cnt: number of character
        //if cnt+1 < enough gap < sqrt(M), use f(cnt+1, sqrt(M))
        static uniform_int_distribution<int> f(cnt+1, M - 1);
        P=f(rng);
        H.resize(S.size() + 1);
        B.resize(S.size() + 1);
        B[0] = 1;
        for (ll i = 1; i <= S.size(); i++) H[i] = (H[i - 1] * P + S[i - 1]) % M;
        for (ll i = 1; i <= S.size(); i++) B[i] = B[i-1] * P % M;
    }
    ll sub(ll s, ll e) { //call with 0-based
        s++; e++;
        ll ret = (H[e] - H[s - 1] * B[e - s + 1]) % M;
        return ret < 0 ? ret + M : ret;
    }
};

```

```

//double hashing
mt19937 rng(chrono::steady_clock::now().time_since_epoch().count());
struct Hashing {
    const ll M0=998244353, M1=1'000'000'007;
    ll P0, P1;
    vector<vector<ll>> H, B;
    void Build(const string& S, ll cnt) { //cnt: number of character
        //if cnt+1 < enough gap < sqrt(M), use f(cnt+1, sqrt(M))
        static uniform_int_distribution<int> f(cnt+1, M0 - 1);
        static uniform_int_distribution<int> g(cnt+1, M1 - 1);
        P0=f(rng), P1=g(rng);
        H.resize(S.size() + 1, vector<ll>(2));
        B.resize(S.size() + 1, vector<ll>(2));
        B[0][0] = B[0][1]= 1;
        for (ll i = 1; i <= S.size(); i++) {
            H[i][0] = (H[i - 1][0] * P0 + S[i - 1]) % M0;
            H[i][1] = (H[i - 1][1] * P1 + S[i - 1]) % M1;
        }
        for (ll i = 1; i <= S.size(); i++) {
            B[i][0] = B[i-1][0] * P0 % M0;
            B[i][1] = B[i-1][1] * P1 % M1;
        }
    }
    pll sub(ll s, ll e) { //call with 0-based
        s++; e++;
        ll ret0 = (H[e][0] - H[s - 1][0] * B[e - s + 1][0]) % M0;
        ll ret1 = (H[e][1] - H[s - 1][1] * B[e - s + 1][1]) % M1;
        if (ret0 < 0) ret0 += M0;
        if (ret1 < 0) ret1 += M1;
        return {ret0, ret1};
    }
};

```

## 5.2 KMP

```
string s, t;
ll fail[1000010];
vector<ll> ans;
void kmp() {
    getline(cin, s);
    getline(cin, t);
    for (ll i = 1, j = 0; i < t.size(); i++) {
        while (j > 0 && t[i] != t[j]) j = fail[j - 1];
        if (t[i] == t[j]) fail[i] = ++j;
    }
    for (ll i = 0, j = 0; i < s.size(); i++) {
        while (j > 0 && s[i] != t[j]) j = fail[j - 1];
        if (s[i] == t[j]) {
            if (j == t.size() - 1) {
                ans.push_back(i - t.size() + 2);
                j = fail[j];
            }
            else j++;
        }
    }
    cout << ans.size() << endl;
    for (ll i = 0; i < ans.size(); i++) cout << ans[i] << endl;
}
```

## 5.3 manacher

```
ll n, ans;
ll a[N]; //a[i]: Maximum length of palindrome centered on i
string s, t;
void solve() {
    cin >> s;
    n = s.size();
    for (ll i = 0; i < n; i++) t += '#', t += s[i];
    t += '#';
    n = t.size();
    ll r = 0, p = 0;
    for (ll i = 0; i < n; i++) {
        if (i <= r) a[i] = min(a[2 * p - i], r - i);
        else a[i] = 0;
        while (i - a[i] - 1 >= 0 && i + a[i] + 1 < n && t[i - a[i] - 1] == t[i + a[i] + 1])
            a[i]++;
        if (r < i + a[i]) r = i + a[i], p = i;
    }
}
```

## 5.4 suffix array & lcp

```
vector<ll> SA(string& s) { //O(nlogn)
    ll n = s.size(), m = max(256LL, n) + 1;
    vector<ll> sa(n), r(n + n), nr(n + n), idx(n), cnt(m);
    for (ll i = 0; i < n; i++) sa[i] = i, r[i] = s[i];
    for (ll d = 1; d < n; d <= 1) {
        auto cmp = [&](ll i, ll j) {
            return r[i] < r[j] || (r[i] == r[j] && r[i + d] < r[j + d]);
        };
    }
```

```
        for (ll i = 0; i < m; i++) cnt[i] = 0;
        for (ll i = 0; i < n; i++) cnt[r[i + d]]++;
        for (ll i = 1; i < m; i++) cnt[i] += cnt[i - 1];
        for (ll i = n - 1; i >= 0; i--) idx[--cnt[r[i + d]]] = i;
        for (ll i = 0; i < m; i++) cnt[i] = 0;
        for (ll i = 0; i < n; i++) cnt[r[i]]++;
        for (ll i = 1; i < m; i++) cnt[i] += cnt[i - 1];
        for (ll i = n - 1; i >= 0; i--) sa[--cnt[r[idx[i]]]] = idx[i];
        nr[sa[0]] = 1;
        for (ll i = 1; i < n; ++i) nr[sa[i]] = nr[sa[i - 1]] + cmp(sa[i - 1], sa[i]);
        r = nr;
        if (r[sa[n - 1]] == n) break;
    }
    return sa;
}
vector<ll> LCP(vector<ll>& sa, string& s) {
    ll n = s.size();
    vector<ll> lcp(n, 0), isa(n);
    for (ll i = 0; i < n; i++) isa[sa[i]] = i;
    for (ll k = 0, i = 0; i < n; i++) {
        if (isa[i]) {
            for (ll j = sa[isa[i] - 1]; s[i + k] == s[j + k]; k++);
            lcp[isa[i]] = (k ? k - 1 : 0);
        }
    }
    return lcp;
}
```

## 5.5 Z

// z[i] -> suffix[i] 와 전체 문자열의 최장 공통 접두사

```
vector<int> make_z(string& s) {
    ll n = s.size();
    vector<int> z(n);
    for (int i = 1, l = 0, r = 0; i < n; i++) {
        if (i <= r) z[i] = min(r - i + 1, z[i - l]);
        while (i + z[i] < n && s[z[i]] == s[i + z[i]]) z[i]++;
        if (i + z[i] - 1 > r) l = i, r = i + z[i] - 1;
    }
    z[0] = n;
    return z;
}
```

## 5.6 Aho-Corasick

```
const int nmax = 1e4 + 1;
class aho {
public:
    // 최대 단어개수?
    int fail[nmax], fin[nmax], num = 1;
    char last[nmax];
    vector<pair<char, int>> edge[nmax]; // edge[x]={c,y}->c를 만나면 y로 이동
    int in(int node, char go) {
        for (auto nxt : edge[node]) {
            if (nxt.first == go) return nxt.second;
        }
        return -1;
    }
}
```

```

void add(string S) {
    ll now = 0;
    for (auto nxt : S) {
        ll go = in(now, nxt);
        if (go == -1) {
            last[num] = nxt;
            edge[now].push_back({nxt, num});
            now = num++;
        }
        else now = go;
    }
    fin[now] = 1;
}

aho(vector<string> S) {
    for (int i = 0; i < nmax; i++) fail[i] = fin[i] = last[i] = 0;
    for (auto nxt : S) add(nxt);
    queue<int> q;
    q.push(0);
    while (q.size()) {
        auto cur = q.front();
        q.pop();
        for (auto [a, nxt] : edge[cur]) {
            ll f = fail[cur];
            while (f && in(f, a) == -1) f = fail[f];
            if (in(f, a) != -1) f = in(f, a);
            if (!cur) f = 0;
            fin[nxt] |= fin[f];
            fail[nxt] = f;
            q.push(nxt);
        }
    }
}

int match(string s) {
    int now = 0;
    for (auto nxt : s) {
        while (now > 0 && in(now, nxt) == -1) now = fail[now];
        if (in(now, nxt) != -1) now = in(now, nxt);
        if (fin[now]) return 1;
    }
    return 0;
}

};

void solve() {
    ll n; cin >> n;
    vector<string> S(n);
    for (int i = 0; i < n; i++) cin >> S[i];
    aho A(S);
    cin >> n;
    for (int i = 0; i < n; i++) {
        string s; cin >> s;
        if (A.match(s)) cout << "YES";
        else cout << "NO";
        cout << '\n';
    }
}

```

## 6 Flow

### 6.1 dinic

//  $O(\min(fE, V^2E))$ . But all edge's capacity are 0 or 1, then  $O(\min(V^{2/3}, E^{1/2}))$

```

const int MAXN = 555;
struct edge {
    int to, cap, rev;
};
int level[MAXN];
int work[MAXN];
vector<edge> adj[MAXN];
void add_edge(int from, int to, int c) {
    adj[from].push_back({ to, c, (int)adj[to].size() });
    adj[to].push_back({ from, 0, (int)adj[from].size() - 1 });
}

bool bfs(int src, int sink) {
    fill(level, level + MAXN, -1);
    fill(work, work + MAXN, 0);
    level[src] = 0;
    queue<int> q;
    q.push(src);
    while (!q.empty()) {
        int now = q.front(); q.pop();
        for (auto& e : adj[now]) {
            if (e.cap > 0 && level[e.to] == -1) {
                level[e.to] = level[now] + 1;
                q.push(e.to);
            }
        }
    }
    return level[sink] != -1;
}

int dfs(int now, int sink, int amount) {
    if (now == sink) return amount;
    for (int& i = work[now]; i < adj[now].size(); i++) {
        auto e = adj[now][i];
        if (e.cap > 0 && level[e.to] == level[now] + 1) {
            int df = dfs(e.to, sink, min(amount, e.cap));
            if (df > 0) {
                adj[now][i].cap -= df;
                adj[e.to][e.rev].cap += df;
                return df;
            }
        }
    }
    return 0;
}

int dinic(int src, int sink) {
    int max_flow = 0;
    while (bfs(src, sink)) {
        while (1) {
            int df = dfs(src, sink, INF);
            if (!df) break;
            max_flow += df;
        }
    }
    return max_flow;
}

```

```

/*
LR flow 구현 시
1. edge (u, v) 용량이 [l, r] 이면, cap(u->T') = l, cap(S'->v) = l, cap(u->v) = r-l
2. cap(T->S) 무한 용량 간선 추가
3. S'->T' flow가 sum(1)이면 flow가 존재.
4. (T->S) 간선(역간선 포함!) 지우고 S->T flow를 흘림
5. 3과 4에서 흘린 flow의 합이 max-flow
max flow는
*/

```

## 6.2 더블어민규당 MCMF

<https://github.com/koosaga/DeobureoMinkyuParty/blob/master/teamnote.tex>

```

struct mincostflow {
    struct edg {
        int pos, cap, rev, cost;
    };

    vector<edg> gph[MAXN];

    void clear() {
        for (int i = 0; i < MAXN; i++) gph[i].clear();
    }

    void add_edge(int s, int e, int x, int c) {
        gph[s].push_back({e, x, (int) gph[e].size(), c});
        gph[e].push_back({s, 0, (int) gph[s].size() - 1, -c});
    }

    int phi[MAXN], inque[MAXN], dist[MAXN];

    void prep(int src, int sink) {
        memset(phi, 0x3f, sizeof(phi));
        memset(dist, 0x3f, sizeof(dist));
        queue<int> que;
        que.push(src);
        inque[src] = 1;
        while (!que.empty()) {
            int x = que.front();
            que.pop();
            inque[x] = 0;
            for (auto& i: gph[x]) {
                if (i.cap > 0 && phi[i.pos] > phi[x] + i.cost) {
                    phi[i.pos] = phi[x] + i.cost;
                    if (!inque[i.pos]) {
                        inque[i.pos] = 1;
                        que.push(i.pos);
                    }
                }
            }
        }
    }

    for (int i = 0; i < MAXN; i++) {
        for (auto& j: gph[i]) {
            if (j.cap > 0) j.cost += phi[i] - phi[j.pos];
        }
    }
}

```

```

    }
    priority_queue<pair<int, int>, vector<pair<int, int>>, greater<pair<int, int>>> > pq;
    pq.push({0, src});
    dist[src] = 0;
    while (!pq.empty()) {
        auto l = pq.top();
        pq.pop();
        if (dist[l.second] != l.first) continue;
        for (auto& i: gph[l.second]) {
            if (i.cap > 0 && dist[i.pos] > l.first + i.cost) {
                dist[i.pos] = l.first + i.cost;
                pq.push({dist[i.pos], i.pos});
            }
        }
    }
}

bool vis[MAXN];
int ptr[MAXN];

int dfs(int pos, int sink, int flow) {
    vis[pos] = 1;
    if (pos == sink) return flow;
    for (; ptr[pos] < gph[pos].size(); ptr[pos]++) {
        auto& i = gph[pos][ptr[pos]];
        if (!vis[i.pos] && dist[i.pos] == i.cost + dist[pos] && i.cap > 0) {
            int ret = dfs(i.pos, sink, min(i.cap, flow));
            if (ret != 0) {
                i.cap -= ret;
                gph[i.pos][i.rev].cap += ret;
                return ret;
            }
        }
    }
    return 0;
}

int match(int src, int sink, int sz) { // sz: 노드 값 중 가장 큰 거 + 1. 즉 [0, sz)를 사용
    prep(src, sink);
    for (int i = 0; i < sz; i++) dist[i] += phi[sink] - phi[src];
    int ret = 0;
    while (dist[sink] < 0) {
        memset(ptr, 0, sizeof(ptr));
        memset(vis, 0, sizeof(vis));
        int tmp = 0;
        while ((tmp = dfs(src, sink, 1e9))) {
            ret += dist[sink] * tmp;
            memset(vis, 0, sizeof(vis));
        }
        tmp = 1e9;
        for (int i = 0; i < sz; i++) {
            if (!vis[i]) continue;
            for (auto& j: gph[i]) {
                if (j.cap > 0 && !vis[j.pos]) {
                    tmp = min(tmp, (dist[i] + j.cost) - dist[j.pos]);
                }
            }
        }
    }
}

```

```

    }
}
if (tmp > 1e9 - 200) break;
for (int i = 0; i < sz; i++) {
    if (!vis[i]) dist[i] += tmp;
}
}
return ret;
}
} mcmf;

```

### 6.3 Hopcroft-Karp + Minimum Vertex Cover

```

const int MAXN = 1010, MAXM = 1010;
int dis[MAXN], l[MAXN], r[MAXN], vis[MAXN];
vector<int> adj[MAXN];
void add_edge(int l, int r) {adj[l].push_back(r);} //don't use add_edge(i, n+j), you have to
add_edge(i, j)!
bool bfs(int n) {
    queue<int> q;
    bool ok=0;
    memset(dis, 0, sizeof(dis));
    for (int i=1; i<=n; i++) {
        if (l[i]==-1 && !dis[i]) q.push(i), dis[i]=1;
    }
    while (!q.empty()) {
        int x=q.front(); q.pop();
        for (auto &i: adj[x]) {
            if (r[i]==-1) ok=1;
            else if (!dis[r[i]]) {
                dis[r[i]]=dis[x]+1;
                q.push(r[i]);
            }
        }
    }
    return ok;
}
bool dfs(int x) {
    if (vis[x]) return 0;
    vis[x]=1;
    for (auto &i: adj[x]) {
        if (r[i]==-1 || (!vis[r[i]] && dis[r[i]] == dis[x]+1 && dfs(r[i]))) {
            l[x]=i, r[i]=x;
            return 1;
        }
    }
    return 0;
}
int match(int n) {
    memset(l, -1, sizeof(l));
    memset(r, -1, sizeof(r));
    int ret=0;
    while (bfs(n)) {
        memset(vis, 0, sizeof(vis));
        for (int i=1; i<=n; i++) if (l[i]==-1 && dfs(i)) ret++;
    }
    return ret;
}

```

```

}
//find minimum vertex cover (=bipartite matching)
//before call getcover function, you have to call match function first
bool chk[MAXN + MAXM];
void rdfs(int x, int n) {
    if (chk[x]) return;
    chk[x] = 1;
    for (auto &i: gph[x]) {
        chk[i + n] = 1;
        rdfs(r[i], n);
    }
}
vector<int> getcover(int n, int m) {
    match(n);
    memset(chk, 0, sizeof(chk));
    for (int i=1; i<=n; i++) if (l[i] == -1) rdfs(i, n);
    vector<int> v;
    for (int i=1; i<=n; i++) if (!chk[i]) v.push_back(i); //A
    for (int i=n+1; i<=n+m; i++) if (chk[i]) v.push_back(i); //B
    return v;
}

```

### 6.4 Minimum Cost Maximum Flow

```

const ll N = 222, INF = 1e9;
struct edge { ll to, cap, rev, cost; };
vector<edge> adj[N];
ll dist[N], p[N], pe[N];
bool inQ[N];
void add_edge(ll u, ll v, ll c, ll cost) {
    adj[u].push_back({ v, c, (ll)adj[v].size(), cost });
    adj[v].push_back({ u, 0, (ll)adj[u].size() - 1, -cost });
}
bool spfa(ll s, ll t) {
    fill(dist, dist + N, INF);
    memset(inQ, 0, sizeof(inQ));
    queue<ll> q;
    q.emplace(s); dist[s] = 0; inQ[s] = 1;
    bool ok=0;
    while (!q.empty()) {
        ll x = q.front(); q.pop();
        if (x==t) ok=1;
        inQ[x] = 0;
        for (ll i = 0; i < adj[x].size(); i++) {
            auto e = adj[x][i];
            if (e.cap > 0 && dist[x] + e.cost < dist[e.to]) {
                dist[e.to] = dist[x] + e.cost;
                p[e.to] = x; pe[e.to] = i;
                if (!inQ[e.to]) {
                    inQ[e.to] = 1;
                    q.emplace(e.to);
                }
            }
        }
    }
    return ok;
}

```

```

}
pll mcmf(ll s, ll t) {
    ll min_cost = 0, max_flow=0, flow, rev;
    while (spfa(s, t)) {
        flow = INF;
        for (ll i = t; i != s; i = p[i]) flow = min(flow, adj[p[i]][pe[i]].cap);
        min_cost += flow * dist[t];
        for (ll i = t; i != s; i = p[i]) {
            rev = adj[p[i]][pe[i]].rev;
            adj[p[i]][pe[i]].cap -= flow;
            adj[i][rev].cap += flow;
        }
        max_flow+=flow;
    }
    return { min_cost, max_flow };
}

```

## 7 DP optimization

### 7.1 divide and conquer optimization

```

//if cost is monge  $C(a,c)+C(b,d) \leq C(a,d)+C(b,c)$ ,  $a \leq b \leq c \leq d$ 
//or if monotonicity  $opt(i,j) \leq opt(i,j+1)$ 
void f(int i, int s, int e, int optl, int optr) {
    //want to get dp[i][s~e]
    if(s>e) return;
    int m=s+e>>1;
    ll ret=1e18, opt=optl;
    for(ll j=optl; j<=optr; j++) {
        ll now=dp[i-1][j]+(a[m]-a[j])*(m-j);
        if(now<ret) {
            ret=now;
            opt=j;
        }
    }
    dp[i][m]=ret;
    f(i, s, m-1, optl, opt), f(i, m+1, e, opt, optr);
}

```

### 7.2 SOS

```

for (int i = 0; i < (1<<n); i++)
    F[i] = A[i];

for (int i = 0; i < n; i++) { // 0...n-1 번째 축으로 훑기
    for (int x = 0; x < (1<<n); x++) {
        if (x & (1<<i)) // i번째 축 좌표가 1이므로 누적합 계산
            F[x] += F[x^(1<<i)];
    }
}

```

## 8 ETC

### 8.1 int128

```
// -2^127 ~ 2^127 - 1
// use g++20(64bit) in codeforces
// use bits/stdc++.h (extc++.h cause error)
// do cin & cout with ull Or use below function
__int128 read() {
    __int128 x = 0, f = 1;
    char ch = getchar();
    while (ch < '0' || ch > '9') {
        if (ch == '-') f = -1;
        ch = getchar();
    }
    while (ch >= '0' && ch <= '9') {
        x = x * 10 + ch - '0';
        ch = getchar();
    }
    return x * f;
}

void print(__int128 x) {
    if (x < 0) {
        putchar('-');
        x = -x;
    }
    if (x > 9) print(x / 10);
    putchar(x % 10 + '0');
}

bool cmp(__int128 x, __int128 y) { return x > y; }
__int128 a = read();
print(a);
cout << '\n';
```

### 8.2 priority queue my cmp

```
// ex) min heap
struct cmp {
    bool operator()(int x, int y) {
        return a > b;
    }
};

std::priority_queue<int, vector<int>, cmp> pq;
```

### 8.3 distance

$\max(|x|, |y|) = |x+y|/2 + |x-y|/2$   
 $\min(|x|, |y|) = |x|+|y| - \max(|x|, |y|) = |x|+|y| - (|x+y|/2 + |x-y|/2)$

manhattan(taxi) -> 45 rotate -> chebyshev  
 $|x_1 - x_2| + |y_1 - y_2| \rightarrow \max(|x_1 - x_2|, |y_1 - y_2|)$   
 $(x, y) \rightarrow (x+y, x-y)$

### 8.4 random

```
mt19937 rng(chrono::steady_clock::now().time_since_epoch().count());
shuffle(all(v), rng); // shuffle randomly!
cout << rng(); // random [0, 2^32-1]
int x = rng() % 100; // [0, 99]
```

### 8.5 mo's algorithm (hilbert)

```
// 정렬 기준을 HilOrd로..
// 단, HilOrd로 하면 구간 넓히고 줄일 때, add/del 연산 순서가 중요하다.
// add -> del 순서를 지켜주자.
ll HilOrd(int x, int y, int k) {
    // long long 써야 함.
    // k는 n < 2^k인 첫 k로
    // HilOrd 오름차순 기준으로 쿼리 처리
    // 0(QB + NN/B)라서 B = N/sqrt(Q)로 잡으면
    // 0(Nsqrt(Q)) 이고, Mo's의 (N+Q)sqrt(N)보다 항상 빠르다.
    ll c, d = 0, p = 0, r = 0, v;
    for (; k >= 0; k--) {
        v = 0;
        if (y - 1 >> k & 1) v = 3;
        if (x - 1 >> k & 1) v ^= 1;
        c = (v + r ^ d) % 4;
        p += c << 2 * k;
        if (c == d) r += 3;
        if (c + d == 3) r++;
        if (c == 0 || c == 3) d ^= 3;
    }
    return p + 1;
}

#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
int sqrtN;
struct Query {
    int idx, s, e;
    bool operator < (Query &x) {
        if (s/sqrtN != x.s/sqrtN) return s/sqrtN < x.s/sqrtN;
        return e < x.e;
    }
};

vector<Query> query;
vector<int> v;
ll res = 0;
ll ans[101010];
int main() {
    ios_base::sync_with_stdio(0); cin.tie(0);
    int n, q; cin >> n >> q; sqrtN = sqrt(n);
    v.resize(n+1);
    for (int i = 1; i <= n; i++) {
        cin >> v[i];
    }
    for (int i = 0; i < q; i++) {
        int s, e; cin >> s >> e;
        query.push_back({i, s, e});
    }
    sort(query.begin(), query.end());
    int s = query[0].s, e = query[0].e;
    for (int i = s; i <= e; i++) {
        res += v[i];
    }
    ans[query[0].idx] = res;
```

```

for(int i=1; i<q; i++){
    // 삽입, 삭제 순서 중요한 문제는 주의 필요. (런타임에러)
    while(s < query[i].s) res -= v[s++];
    while(s > query[i].s) res += v[--s];
    while(e < query[i].e) res += v[++e];
    while(e > query[i].e) res -= v[e--];
    ans[query[i].idx] = res;
}
for(int i=0; i<q; i++) cout << ans[i] << "\n";
}

```

## 8.6 degree sequence

if satisfy 1 & 2 -> can make graph  
 let  $d_1 \geq \dots \geq d_n$   
 1.  $d_1 + \dots + d_n$  is even number  
 2. satisfy below inequality for all  $k$   
 $d_1 + \dots + d_k \leq k(k-1) + \min(d_{k+1}, k) + \dots + \min(d_n, k)$

how to construct?

1)  
 ex) 6 4 4 3 3 2 2  
 repeat  
     match largest number  
     6 4 4 3 3 2 2  
     0 3 3 2 2 1 1  
     0 0 2 1 1 1 1  
     ....  
     0 0 0 0 0 0 0

2. special case: tree

if sum of  $d_i$  is  $2n-2$  : can  
 if sum of  $d_i$  isn't  $2n-2$  : cant  
 repeat  
     connect leaf with nonleaf

## 8.7 bitset

```

/*
아래 pragma 추가하면 /64 -> /256
근데, 로컬에선 잘 안 됨.. boj나 cf에선 잘 되는 듯.
단, c++20이상에서는 avx2를 헤더파일을 밑에 두면 돈다.
ex)
#pragma GCC optimize("O3,unroll-loops")
#include <bits/stdc++.h>
#pragma GCC target("avx2")
*/

bitset<1010> b;
//default: 0

/*
reset(2) : 2번을 0으로 변경 생략 시 전체 비트 0
set(2, 0 or 1) : 2번을 0 또는 1로 변경 생략 시 전체 비트 1

```

```

flip(2) : 2번을 반전(0이면 1로, 1이면 0으로) 생략 시 전체 비트 반전
all() : 비트셋에서 모든 비트가 1일 경우 true, 그렇지 않으면 false
any() : 비트셋에 비트가 1인 것이 하나라도 존재한다면 true 그렇지 않으면 false
none() : 비트셋에 비트가 1인 것이 하나라도 존재하지 않는다면 true 그렇지 않으면 false
to_string() : 비트셋을 문자열로 변환한다.
to_ulong() : 비트셋을 unsigned long integer 타입으로 변환한다.
to_ullong() : 비트셋을 unsigned long long 타입으로 변환한다.
size() : bitset의 size를 반환한다.
count() : 현재 bitset의 있는 비트 1의 개수를 반환한다.
*/

```