C++ 문법

Created	@2025년 5월 17일 오후 6:36
Tags	DefaultCode

Default Code

```
#include<bits/stdc++.h>
#pragma warning(disable:4996)
#pragma comment(linker, "/STACK:336777216")
#pragma GCC optimize("O3,unroll-loops")
#pragma GCC target("avx,avx2,fma")
using namespace std;
using ll = long long;
using pll = pair<ll,ll>;
using ld = long double;
using pld = pair<ld,ld>;
using ull = unsigned long long;
using tlll = tuple<ll,ll,ll>;
using vl = vector<ll>;
using vvl = vector<vl>;
using endl = '\n';

#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
#include <ext/pb_ds/detail/standard_policies.hpp>
using namespace __gnu_pbds;
template<typename T> using ordered_set
    = tree<T, null_type, less<>, rb_tree_tag,
        tree_order_statistics_node_update>;
template<typename T> using ordered_multiset
```

```
    = tree<T, null_type, less_equal<>, rb_tree_tag,
        tree_order_statistics_node_update>;

using LD = __float128;
using LL = __int128;
using ULL = __uint128;
typedef __float128 LD;
typedef __int128_t LL;
typedef __uint128_t ULL;


template<typename T>
ostream& operator<<(ostream& out, vector<T> v) {
    string _;
    out << '(';
    for (T x : v) out << _ << x, _ = " ";
    out << ')';
    return out;
}


#ifdef ONLINE_JUDGE
constexpr bool ndebug = true;
#else
constexpr bool ndebug = false;
#endif

ll gcd(ll a, ll b){return b?gcd(b,a%b):a;}
ll lcm(ll a, ll b){if(a&&b)return a*(b/gcd(a,b)); return a+b;}
ll POW(ll a, ll b, ll rem){ll p=1;a%=rem;for(;b;b>>=1,a=(a*a)% rem)if(b&1)p=
(p*a)%rem;return p;}
pll extended_gcd(ll a, ll b){if(b == 0)return {1, 0};auto t = extended_gcd(b, a
% b);return {t.second, t.first - t.second * (a / b)};}
ll modinverse(ll a, ll m){return (extended_gcd(a, m).first % m + m) % m;}

void setup() {
    if(!ndebug) {
        freopen("input.txt", "r", stdin);
```

```
        freopen("output.txt", "w", stdout);
    }
    else {
        ios_base::sync_with_stdio(0);
        cin.tie(0);
        cout.tie(0);
    }
}

void preprocess() {
    ll i, j, k;

}

void solve(ll testcase){
    ll i, j, k;

}

int main() {
    setup();
    preprocess();
    ll t = 1;
    // cin >> t;
    for (ll testcase = 1; testcase <= t; testcase++){
        solve(testcase);
    }
    return 0;
}
```

실전 압축

```
#include<bits/stdc++.h>
#pragma warning(disable:4996)
#pragma comment(linker, "/STACK:336777216")
#pragma GCC optimize("O3,unroll-loops")
#pragma GCC target("avx,avx2,fma")
```

```
using namespace std;
using ll = long long;
using pll = pair<ll,ll>;
using ld = long double;
using pld = pair<ld,ld>;
using ull = unsigned long long;
using tlll = tuple<ll,ll,ll>;
using vl = vector<ll>;
using vvl = vector<vl>;

#ifdef ONLINE_JUDGE
constexpr bool ndebug = true;
#else
constexpr bool ndebug = false;
#endif

void setup() {
    if(!ndebug) {
        freopen("input.txt", "r", stdin);
        freopen("output.txt", "w", stdout);
    }
    else {
        ios_base::sync_with_stdio(0);
        cin.tie(0);
        cout.tie(0);
    }
}

void preprocess() {
    ll i, j, k;

}

void solve(ll testcase){
    ll i, j, k;

}
```

```
int main() {
    setup();
    preprocess();
    ll t = 1;
    // cin >> t;
    for (ll testcase = 1; testcase <= t; testcase++){
        solve(testcase);
    }
    return 0;
}
```

Math

Created	@2025년 5월 17일 오후 7:13
Tags	전부 완료됨

Basic Arithmetic

```
// calculate floor(log_2(a)), a > 0
ll lg2(ll a) {
```

```
    return 63-__builtin_clzll(a);
}

// calculate the number of 1-bits
ll bitcount(ll a) {
    return __builtin_popcountll(a);
}

// calculate ceil(a/b)
// |a|, |b| <= (2^63)-1
ll ceildiv(ll a, ll b) {
    if (b<0) return ceildiv(-a, -b);
    if (a<0) return (-a)/b;
    return ((ull)a+(ull)b-1ull)/b;
}

// calculate floor(a/b)
// |a|, |b| <= (2^63)-1
ll floordiv(ll a, ll b) {
    if (b<0) return floordiv(-a, -b);
    if (a>=0) return a/b;
    return -(ll)(((ull)(-a)+b-1)/b);
}

// find a pair (s, t) s.t. as + bt = gcd(a, b)
pll extended_gcd(ll a, ll b) {
    if (b==0) return {1, 0};
    auto t=extended_gcd(b, a%b);
    return {t.second, t.first-t.second*(a/b)};
}

// find x in [0, m) s.t. ax === 1 (mod m)
ll modinverse(ll a, ll m) {
    if (gcd(a, m) != 1) return -1;
    return (extended_gcd(a, m).first%m+m)%m;
}

// calculate a*b % m
```

```
// Note: m*m이 ll 범위를 초과할 때 사용
// |m| < 2^62
ll modmul(ll a, ll b, ll m) {
    return ll((__int128)a*(__int128)b%m);
}

// calculate n^k % m
// O(logk)
ll modpow(ll n, ll k, ll m) {
    ll ret=1;
    n%=m;
    while (k) {
        if (k&1) ret=modmul(ret, n, m);
        n=modmul(n, n, m);
        k>>=1;
    }
    return ret;
}

// calculate n^k
// O(logk)
ll powm(ll n, ll k) {
    ll ret=1;
    while (k) {
        if (k&1) ret*=n;
        n*=n;
        k>>=1;
    }
    return ret;
}
```

Fraction 구조체

```
struct F {
    ll ja,mo;
    F(ll _ja=0, ll _mo=1) : ja(_ja), mo(_mo) {
        if (mo<0) ja=-ja,mo=-mo;
```

```
        ll g=gcd(ja, mo);
        ja/=g; mo/=g;
    }
    F operator+(const F o)const {
        return {ja*o.mo+o.ja*mo, mo*o.mo};
    }
    F operator-(const F o)const {
        return {ja*o.mo-o.ja*mo, mo*o.mo};
    }
    F operator*(const F o)const {
        return {ja*o.ja, mo*o.mo};
    }
    F operator/(const F o)const {
        return {ja*o.mo, mo*o.ja};
    }
    bool operator==(const F o)const {
        return ja==o.ja && mo==o.mo;
    }
    bool operator<(const F& o)const {
        return ja*o.mo<o.ja*mo;
    }
};
```

1~n의 모듈러 역원 구하기

조건: 1~n이 모두 모듈러 역원을 가져야 함

(1~mod-1이 모두 모듈러 역원을 가진다 ↔ mod가 소수이다)

시간복잡도: O(n)

```
// Usage: vl modinv = calc_range_modinv(n, mod);
// O(n)
vl calc_range_modinv(ll n, ll mod) {
    vl ret(n+1);
    ret[1]=1;
    for (ll i=2;i<=n;++i) {
        ret[i]=(ll)(mod-mod/i)*ret[mod%i]%mod;
    }
```

```
    return ret;
}
```

mod가 합성수이고, 소인수분해 되어있을 때

```
// Usage:
//   vl inv = calc_range_modinv(n, mods);     // mods = {p1^e1, p2^e2, ...}
//   ll x   = inv[i];                          // 0 => inverse does not exist
// O(n * k * log M)   (k = mods.size(),  M = max mods[i])
vl calc_range_modinv(ll n, vl &mods) {
    ll mod=1;
    for (ll m:mods) mod*=m;
    vl ret(n+1, 0);
    vl a(mods.size()), m=mods;
    for (ll i=1;i<=n;++i) {
        if (gcd(i, mod)!=1) {
            ret[i]=0;
            continue;
        }
        for (ll j=0;j<mods.size();++j)
            a[j]=modinverse(i, mods[j]);
        ret[i]=chinese_remainder(a, m);
    }
    return ret;
}

// 참고 예시: mod = 1000
vl mods {8, 125};
vl modinvs = calc_range_modinv(n, mods);
```

sieve method: prime, divisor, phi

```
// find prime numbers in 1~n
// ret[x] == true -> x is prime
// Usage: vl is_prime = sieve(n);
// O(n*loglogn)
vl sieve(ll n) {
    vl ret(n+1, 1);
```

```
    ret[0]=ret[1]=false;
    for (ll i=2;i*i<=n;++i) {
        if (ret[i])
            for (ll j=i*i;j<=n;j+=i)
                ret[j]=false;
    }
    return ret;
}

// calculate number of divisors for 1~n
// Usage: vl tau = num_of_divisors(n);
// Note: to get sum of divisors, replace ret[j]+=1 to +=i
// O(n*logn)
vl num_of_divisors(ll n) {
    vl ret(n+1);
    for (ll i=1;i<=n;++i) {
        for (ll j=i;j<=n;j+=i)
            ret[j]+=1;
    }
    return ret;
}

// calculate euler totient function for 1~n
// Usage: vl phi = euler_phi(n);
// O(n*loglogn)
vl euler_phi(ll n) {
    vl ret(n+1);
    iota(ret.begin(), ret.end(), 0);
    for (ll i=2;i<=n;++i)
        if (ret[i]==i)
            for (ll j=i;j<=n;j+=i)
                ret[j]-=ret[j]/i;
    return ret;
}
```

Linear sieve

O(n)

```
// Usage: sieve s(n);
// Note: s.sp[x] == x ↔ x is prime
// O(n)
struct sieve {
    vl sp, e, phi, mu, tau, sigma, primes;
    // sp : smallest prime factor, e : exponent of sp, phi : euler phi, mu : mobi
us
    // tau : num of divisors, sigma : sum of divisors
    sieve(ll sz) {
        sp.resize(sz+1), e.resize(sz+1), phi.resize(sz+1), mu.resize(sz+1),
            tau.resize(sz+1), sigma.resize(sz+1);
        phi[1]=mu[1]=tau[1]=sigma[1]=1;
        for (ll i=2;i<=sz;i++) {
            if (!sp[i]) {
                primes.push_back(i), e[i]=1, phi[i]=i-1, mu[i]=-1, tau[i]=2;
                sp[i]=i, sigma[i]=i+1;
            }
            for (auto j : primes) {
                if (i*j>sz) break;
                sp[i*j]=j;
                if (i%j==0) {
                    e[i*j]=e[i]+1, phi[i*j]=phi[i]*j, mu[i*j]=0,
                          tau[i*j]=tau[i]/e[i*j]*(e[i*j]+1),
                          sigma[i*j]=sigma[i]*(j-1)/(powm(j, e[i*j])-1) *
                                        (powm(j, e[i*j]+1)-1)/(j-1);
                    break;
                }
                e[i*j]=1, phi[i*j]=phi[i]*phi[j], mu[i*j]=mu[i]*mu[j],
                     tau[i*j]=tau[i]*tau[j], sigma[i*j]=sigma[i]*sigma[j];
            }
        }
    }
};

// 참고: sieve 이용한 소인수분해, 팀노트엔 안 넣어도 될 듯
// (p, e) 순서쌍 저장 -> p^e
sieve s(n);
vector<pll> factors;
```

```
while (n>1) {
    ll p=s.sp[n];
    ll cnt=0;
    while (s.sp[n]==p) {
        n/=p;
        cnt++;
    }
    factors.push_back({p, cnt});
}
```

밀러-라빈 소수판정법

시간복잡도: $O((log\ n)^2)$

백준, 라이브러리 체커 통과 확인

```
// Usage: bool result = is_prime(n);
// O(logn*logn)
// Note: modpow에서 반드시 modmul로 오버플로우 방지
bool is_prime(ll n) {
    if (n<2 || n%2==0 || n%3==0) return n==2 || n==3;
    ll k=__builtin_ctzll(n-1), d=n-1>>k;
    for (ll a: { 2, 325, 9375, 28178, 450775, 9780504, 1795265022 }) {
        ll p=modpow(a%n, d, n), i=k;
        while (p!=1 && p!=n-1 && a%n && i--) p = modmul(p, p, n);
        if (p!=n-1 && i!=k) return 0;
    }
    return 1;
}
```

폴라드 로

시간복잡도: $O(n^{1/4}\ log\ n)$

백준, 라이브러리 체커 통과 확인 (BOJ 4149 - 큰 수 소인수분해)

입력: n, ret → 빈 벡터

실행 후: ret에 소인수들을 저장 (정렬은 안 되어 있음)

```
// integer factorization, not sorted
// Usage: vl fac; factor(n, fac);
// O(n^0.25 * logn)
ll pollard(ll n) {
    auto f=[n](ll x) { return (modmul(x, x, n)+3)%n; };
    ll x=0, y=0, t=30, p=2, i=1, q;
    while (t++ % 40 || gcd(p, n)==1) {
        if (x==y) x=++i, y=f(x);
        if (q=modmul(p, abs(x-y), n)) p=q;
        x=f(x), y=f(f(y));
    }
    return gcd(p, n);
}
void factor(ll n, vl &ret) {
    if (n==1) return;
    if (is_prime(n)) {
        ret.push_back(n);
        return;
    }
    ll d=pollard(n);
    factor(d, ret);
    factor(n/d, ret);
}

// 이렇게 wrapper 함수 추가하는 건 어떰?
// Usage: vl fac = factor(n);
// O(n^0.25 * logn)
vl factor(ll n) {
    vl ret;
    factor(n, ret);
    sort(ret.begin(), ret.end());
    return ret;
}
```

Chinese Remainder Theorem

n[0], n[1], ... 으로 나눈 나머지가 각각 a[0], a[1], ... 인 x를 반환, (x는 조건을 만족하는 최소 양수)

만약 조건을 만족하는 정수 x가 없으면 → -1 반환

n[i]들은 쌍마다 서로소라고 가정

시간복잡도: $O(k\ log\ M)$ → k = a.size, M = max(n)

실행 이후에도 a, n의 원소는 변하지 않음

BOJ 6064 - 카잉 달력

```
ll chinese_remainder(vl &a, vl &n, ll s=0) {
    ll size=a.size();
    if (s==size-1) return a[s];
    ll tmp=modinverse(n[s], n[s+1]);
    ll tmp2=(tmp*(a[s+1]-a[s])%n[s+1]+n[s+1])%n[s+1];
    ll ora=a[s+1];
    ll tgcd=gcd(n[s],n[s+1]);
    if ((a[s+1]-a[s])%tgcd!=0) return -1;
    a[s+1]=a[s]+n[s]/tgcd*tmp2;
    n[s+1]*=n[s]/tgcd;
    ll ret=chinese_remainder(a, n, s+1);
    n[s+1]/=n[s]/tgcd;
    a[s+1]=ora;
    return ret;
}
```

Modular Equation

x≡a (mod n), x≡b (mod m)을 만족하는 x를 구하는 방법

1. m과 n을 소인수분해
2. 특정 소수에 대하여 모순이 있다 → 해 없음
3. 모든 소수에 대하여 모순이 없다 → CRT로 합치기

$x \equiv x_1\ (mod\ p^{k_1})$ 과 $x \equiv x_2\ (mod\ p^{k_2})$ 가 모순이 생길 조건 → k1 ≤ k2라고 했을 때, $x_1 \not\equiv x_2\ (mod\ p^{k_1})$ 인 경우

모순이 생기지 않았으면 → $x \equiv x_2\ (mod\ p^{k_2})$ 만 남겨주면 된다.

Catalan, Derangement, Partition, 2nd Stirling

$C_n = \frac{1}{n+1}\binom{2n}{n},\ C_0 = 1,\ C_{n+1} = \sum_{i=0}^{n} C_i\ C_{n-i},\ C_{n+1} = \frac{2(2n+1)}{n+2}C_n$

$D_n = (n-1)(D_{n-1} + D_{n-2}) = n! \sum_{i=1}^{n} \frac{(-1)^{i+1}}{i!}$

$P(n) = \sum_{k \in \mathbb{Z} \setminus 0} (-1)^{k+1} P(n - k(3k-1)/2)$

$P(n) = P(n-1) + P(n-2) - P(n-5) - P(n-7) + P(n-12) + P(n-15) - P(n-22) - ...$

$P(n,\ k) = P(n-1,\ k-1) + P(n-k,\ k)$

$S(n,\ k) = S(n-1,\ k-1) + kS(n-1,\ k)$

Burnside's Lemma

경우의 수를 세는데, 특정 transform operation(회전, 반사, ...)해서 같은 경우들은 하나로 친다. 전체 경우의 수는?

- 각 operation마다 이 operation을 했을 때 변하지 않는 경우의 수를 센다. (단, "아무것도 하지 않는다"라는 operation도 있어야 함!)
- 전체 경우의 수를 더한 후, operation의 수로 나눈다. (답이 맞다면 항상 나누어 떨어져야 한다.)

Kirchoff's Theorem

그래프의 스패닝 트리의 개수를 구하는 정리

무향 그래프의 Laplacian matrix L을 만든다. 이것은 (정점의 차수 대각 행렬) - (인접행렬)이다.

L에서 행과 열을 하나씩 제거한 것을 L'이라 하자. 어느 행/열이든 상관 없다.

그래프의 스패닝 트리의 개수는 $\det(L')$이다.

뤼카 정리

${}_nC_m$ mod p 구하기, p는 소수

fac, invfac을 O(p)에 구해 놓으면 → binomial은 O(1)에 구할 수 있음

binomial을 O(1)이라고 가정하면 → $O(\frac{log\ n}{log\ p})$

BOJ 11402 - 이항 계수 4

```
// binomial은 별도로 구현, binomial(n, m, p) = 0 if n < m
// n, m < p인 경우만 미리 구현
ll binomial(ll n, ll m, ll p);
ll lucas(ll n, ll m, ll p){
    if(m<0 || m>n) return 0;
```

```
    ll res = 1;
    while (n>0 || m>0) {
        ll n_i=n%p;
        ll m_i=m%p;
        if (m_i>n_i) return 0;
        res=res*binomial(n_i, m_i, p)%p;
        n/=p;
        m/=p;
    }
    return res;
}
```

FFT

BOJ 13277 - 큰 수 곱셈

```
// Usage: vl result; mult(a, b, result);
// O(nlogn)
constexpr ld pi = 3.14159265358979323846L;
void fft(ll sign, ll n, vd &real, vd &imag) {
    ld theta=sign*2*pi/n;
    for (ll m=n;m>=2;m>>=1,theta*=2) {
        ld wr=1,wi=0;
        ld c=cos(theta),s=sin(theta);
        ll mh=m>>1;
        for (ll i=0;i<mh;++i) {
            for (ll j=i;j<n;j+=m) {
                ll k=j+mh;
                ld xr=real[j]-real[k];
                ld xi=imag[j]-imag[k];
                real[j]+=real[k];
                imag[j]+=imag[k];
                real[k]=wr*xr-wi*xi;
                imag[k]=wr*xi+wi*xr;
            }
            ld _wr=wr*c-wi*s;
            ld _wi=wr*s+wi*c;
            wr=_wr;wi=_wi;
```

```
        }
    }
    for (ll i=1,j=0;i<n;++i) {
        for (ll k=n>>1;k>(j^=k);k>>=1);
        if (j<i) {
            swap(real[i],real[j]);
            swap(imag[i],imag[j]);
        }
    }
}
void mult(vl &a, vl &b, vl &r) {
    ll n=a.size(),m=b.size();
    ll fn=1;
    while (fn<n+m) fn<<=1;

    vd ra(fn), ia(fn), rb(fn), ib(fn);
    for (ll i=0;i<n;++i) ra[i]=a[i],ia[i]=0;
    for (ll i=n;i<fn;++i) ra[i]=ia[i]=0;
    for (ll i=0;i<m;++i) rb[i]=b[i],ib[i]=0;
    for (ll i=m;i<fn;++i) rb[i]=ib[i]=0;

    fft(1, fn, ra, ia);
    fft(1, fn, rb, ib);
    for (ll i=0;i<fn;++i) {
        ld real_part=ra[i]*rb[i]-ia[i]*ib[i];
        ld imag_part=ra[i]*ib[i]+rb[i]*ia[i];
        ra[i]=real_part;
        ia[i]=imag_part;
    }
    fft(-1, fn, ra, ia);

    r.assign(fn, 0);
    for (ll i=0;i<fn;++i) {
        r[i] = floor(ra[i]/fn+0.5L);
    }
    r.resize(a.size()+b.size()-1);
}
```

Matrix Operations

```
// Usage: vvd A, out; ld det = inverse_and_det(A, out);
// Note: if A is singular, return → 0, out → garbage value
// Note: else, return → det, out → inv(A)
// O(n^3)
inline bool is_zero(ld a) {
    return fabsl(a)<1e-9L;
}
ld inverse_and_det(vvd&A, vvd&out){
    ll n=A.size();
    ld det=1.0L;
    out.assign(n, vd(n, 0));
    for (ll i=0;i<n;++i) out[i][i]=1;
    for (ll i=0;i<n;++i){
        if (is_zero(A[i][i])){
            ld maxv=0;
            ll maxid=-1;
            for (ll j=i+1;j<n;++j) {
                ld cur=fabsl(A[j][i]);
                if(cur>maxv) {
                    maxv=cur;
                    maxid=j;
                }
            }
            if (maxid<0 || is_zero(A[maxid][i])) return 0;
            for (ll k=0;k<n;++k) {
                A[i][k]+=A[maxid][k];
                out[i][k]+=out[maxid][k];
            }
        }
        det*=A[i][i];
        ld coeff=1.0L/A[i][i];
        for (ll j=0;j<n;++j){
            A[i][j]*=coeff;
            out[i][j]*=coeff;
        }
        for (ll j=0;j<n;++j) {
```

```
            if(j==i) continue;
            ld factor=A[j][i];
            for (ll k=0;k<n;++k) {
                A[j][k]-=A[i][k]*factor;
                out[j][k]-=out[i][k]*factor;
            }
        }
    }
    return det;
}
```

Gauss-Jordan Elimination

```
// Gauss-Jordan elimination with full pivoting.
// solve system of linear equations (AX=B)
// Usage: vvd a, b; -> a is n*n, b is n*m
// Usage: bool result = gauss_jordan(a, b);
// Note: after calling, a -> inv(a), b -> X
// O(n^3)
constexpr ld EPS = 1e-10L;
inline bool is_zero(ld a) {
    return fabsl(a)<EPS;
}
bool gauss_jordan(vvd &a,vvd &b){
    ll n=a.size(), m=b[0].size();
    vl irow(n), icol(n), ipiv(n);
    for (ll i=0;i<n;++i){
        ll pj=-1, pk=-1;
        for (ll j=0;j<n;++j) if (!ipiv[j])
            for (ll k=0;k<n;++k) if (!ipiv[k])
                if (pj<0 || fabsl(a[j][k])>fabsl(a[pj][pk])) {
                    pj=j;
                    pk=k;
                }
        if (fabsl(a[pj][pk])<EPS) return false;
        ++ipiv[pk];
        swap(a[pj], a[pk]);
        swap(b[pj], b[pk]);
```

```
        irow[i]=pj;
        icol[i]=pk;
        ld c=1.0L/a[pk][pk];
        a[pk][pk]=1;
        for (ll p=0;p<n;++p) a[pk][p]*=c;
        for (ll p=0;p<m;++p) b[pk][p]*=c;
        for (ll p=0;p<n;++p) if (p!=pk){
            c=a[p][pk];
            a[p][pk]=0;
            for (ll q=0;q<n;++q) a[p][q]-=a[pk][q]*c;
            for (ll q=0;q<m;++q) b[p][q]-=b[pk][q]*c;
        }
    }
    for (ll p=n-1;p>=0;--p) if (irow[p]!=icol[p]){
        for (ll k=0;k<n;++k) swap(a[k][irow[p]], a[k][icol[p]]);
    }
    return true;
}
```

Simplex Algorithm

```
// Two-phase simplex algorithm for solving linear programs of the form
//   maximize c^T x
// subject to Ax <= b
//            x >= 0
// INPUT: A -- an m x n matrix (vvd)
//       b -- an m-dimensional vector (vd)
//       c -- an n-dimensional vector (vd)
//       x -- a vector where the optimal solution will be stored (vd)
// OUTPUT: value of the optimal solution (infinity if unbounded
//        above, nan if infeasible)
// Usage:
// LPSolver lps(A, b, c);
// vd x; → x의 최적값이 저장될 vector<ld>
// ld optimal = lps.solve(x);
typedef vector<ld> vd;
typedef vector<vd> vvd;
const double EPS = 1e-9;
```

```
struct LPSolver {
    ll m, n;
    vl B, N;
    vvd D;
    LPSolver(const vvd &A, const vd &b, const vd &c):
    m(b.size()), n(c.size()), B(m), N(n+1), D(m+2, vd(n+2)) {
        for (ll i=0;i<m;i++)
            for (ll j=0;j<n;j++) D[i][j]=A[i][j];
        for (ll i=0;i<m;i++) {
            B[i]=n+i;
            D[i][n]=-1;
            D[i][n+1]=b[i];
        }
        for (ll j=0;j<n;j++) {
            N[j]=j;
            D[m][j]=-c[j];
        }
        N[n]=-1;
        D[m+1][n]=1;
    }
    void pivot(ll r, ll s) {
        ld inv=1.0L/D[r][s];
        for (ll i=0;i<m+2;i++)
            if (i!=r)
                for (ll j=0;j<n+2;j++)
                    if (j!=s)
                        D[i][j]-=D[r][j]*D[i][s]*inv;
        for (ll j=0;j<n+2;j++)
            if (j!=s) D[r][j]*=inv;
        for (ll i=0;i<m+2;i++)
            if (i!=r) D[i][s]*=-inv;
        D[r][s]=inv;
        swap(B[r], N[s]);
    }
    bool simplex(ll phase) {
        ll x=phase==1 ? m+1 : m;
        while (true) {
            ll s=-1;
```

```
            for (ll j=0;j<=n;j++) {
                if (phase==2 && N[j]==-1) continue;
                if (s==-1 || D[x][j]<D[x][s] || fabsl(D[x][j]-D[x][s])<EPS && N[j]<N
[s])
                    s=j;
            }
            if (D[x][s]>-EPS) return true;
            ll r=-1;
            for (ll i=0;i<m;i++) {
                if (D[i][s]<EPS) continue;
                if (r==-1 || D[i][n+1]/D[i][s]<D[r][n+1]/D[r][s] ||
                    (fabsl(D[i][n+1]/D[i][s]-D[r][n+1]/D[r][s])<EPS) && B[i]<B[r])
                    r=i;
            }
            if (r==-1) return false;
            pivot(r, s);
        }
    }
    ld solve(vd &x) {
        ll r=0;
        for (ll i=1;i<m;i++)
            if (D[i][n+1]<D[r][n+1]) r=i;
        if (D[r][n+1]←EPS) {
            pivot(r, n);
            if (!simplex(1) || D[m+1][n+1]←EPS)
                return numeric_limits<ld>::quiet_NaN();
            for (ll i=0;i<m;i++)
                if (B[i]==-1) {
                    ll s=-1;
                    for (ll j=0;j <= n;j++)
                        if (s==-1 || D[i][j]<D[i][s] || fabsl(D[i][j]-D[i][s])<EPS && N[j]<
N[s]) s=j;
                    pivot(i, s);
                }
        }
        if (!simplex(2))
            return numeric_limits<ld>::infinity();
        x=vd(n);
```

```
        for (ll i=0;i<m;i++)
            if (B[i]<n) x[B[i]]=D[i][n+1];
        return D[m][n+1];
    }
};
```

Nim Game

BOJ 11868 - 님 게임 2

Nim Game의 해법: 모두 XOR했을 때 0이 아니면 선공이, 0이면 후공이 승리

Grundy Number: XOR(MEX(next state grundy))

Subtraction Game: 한 번에 k개 까지의 돌만 가져갈 수 있는 경우 → 각 더미의 돌의 개수를 k+1로 나눈 나머지를 XOR 합하여 판단

Index-k Nim: 한 번에 최대 k개의 더미를 골라 각각의 더미에서 아무렇게나 돌을 제거할 수 있을 때, 각 binary digit에 대하여 합을 k+1로 나눈 나머지를 계산한다. 만약 이 나머지가 모든 digit에 대하여 0이라면 후공이, 하나라도 0이 아니라면 선공이 승리

Permutation and Combination

N과 M 문제집

```
// 둘 다 초기 배열을 오름차순으로 초기화해야 함.

// Permutation
vl arr {1, 2, 3, 4, 5};
do {
    for (auto i: arr) cout << i << ' ';
    cout << '\n';
} while (next_permutation(arr.begin(), arr.end()));
// Also prev_permutation exists

// Combination
// n개 중에 k개 뽑는 방법 -> 0이 k개, 1이 (n-k)개인 배열 사용
vl arr {1, 2, 3, 4, 5};
vl mask {0, 0, 0, 1, 1};
do {
    for (ll i=0;i<mask.size();i++) {
        if (mask[i]==0) cout << arr[i] << ' ';
```

```
    }
    cout << '\n';
} while (next_permutation(mask.begin(), mask.end()));
```

Lifting The Exponent

BOJ 7118 - Ones

조건: p는 소수, x와 y는 p의 배수가 아님

1. (x-y)는 p의 배수 → v_p(x^n-y^n) = v_p(x-y) + v_p(n)
2. (x+y)는 p의 배수이고 n이 홀수 → v_p(x^n+y^n) = v_p(x+y) + v_p(n)

Useful Prime Numbers

```
10'007
10'009
10'111
31'567
70'001
1'000'003
1'000'033
4'000'037
99'999'989
999'999'937
1'000'000'007
1'000'000'009
9'999'999'967
99'999'999'977
```

Query of nCr mod M in O(M+QlogM)

BOJ 14854 - 이항 계수 6

```
// Usage: vector<pll> qs(q); vl ans = sol(q, qs, mod);
// O(M + Q*logM)
// Note: qs[i] = {n, r}
auto sol_p_e = [](ll q, vector<pll> &qs, ll p, ll e, ll mod) {
    vl dp(mod, 1);
    for (ll i=0;i<mod;i++) {
```

```
        if (i) dp[i]=dp[i - 1];
        if (i%p==0) continue;
        dp[i]=dp[i]*i%mod;
    }
    auto f=[&](ll n) {
        ll res=0;
        while (n/=p) res+=n;
        return res;
    };
    auto g = [&](ll n) {
        auto rec=[&](auto &self, ll n) -> ll {
            if (n==0) return 1;
            ll q=n/mod, r=n%mod;
            ll ret=self(self, n/p)*dp[r]%mod;
            if (q&1) ret=ret*dp[mod-1]%mod;
            return ret;
        };
        return rec(rec, n);
    };
    auto bino = [&](ll n, ll r) -> ll {
        if (n<r) return 0;
        if (r==0 || r==n) return 1;
        ll a=f(n)-f(r)-f(n-r);
        if (a>=e) return 0;
        ll b=g(n)*modinverse(g(r)*g(n-r)%mod, mod)%mod;
        return modpow(p, a, mod)*b%mod;
    };
    vl res(q, 0);
    for (ll i=0;i<q;i++) {
        auto [n, r]=qs[i];
        res[i]=bino(n, r);
    }
    return res;
};

auto sol = [](ll q, auto &qs, ll mod) {
    vl f;
    factor(mod, f);
```

```
    sort(f.begin(), f.end());
    vector<pll> fac;
    for (auto ff: f) {
        if (fac.empty() || fac.back().first!=ff)
            fac.push_back({ff, 0});
        fac.back().second++;
    }
    vvl r(q, vl(fac.size(), 0));
    vl m(fac.size(), 1);
    for (ll i=0;i<fac.size();i++) {
        auto [p, e]=fac[i];
        for (ll j=0;j<e;j++) m[i]*=p;
        auto res=sol_p_e(q, qs, p, e, m[i]);
        for (ll j=0;j<q;j++) r[j][i]=res[j];
    }
    vl res(q, 0);
    for (ll i=0;i<q;i++) {
        res[i]=chinese_remainder(r[i], m);
    }
    return res;
};
```

NTT

BOJ 13277 - 큰 수 곱셈

라이브러리 체커

```
// Usage: vl conv = multiply(a, b, mod, w);
// O(n*logn)
// Note: a, b는 ref가 아니라 복사
// Note: (mod, w) : (998 244 353, 3) or (985 661 441, 3) or (1 012 924 417, 5)
void ntt(vl &f, ll mod, ll w, bool inv=false) {
    ll n=f.size(), j=0;
    vl root(n>>1);
    for (ll i=1;i<n;i++) {
        ll bit=n>>1;
        while (j>=bit) {
```

```
            j-=bit;
            bit>>=1;
        }
        j+=bit;
        if (i<j) swap(f[i], f[j]);
    }
    ll ang = modpow(w, (mod-1)/n, mod);
    if (inv) ang=modpow(ang, mod-2, mod);
    root[0]=1;
    for (ll i=1;i<(n>>1);i++)
        root[i]=root[i-1]*ang%mod;
    for (ll len=2;len<=n;len<<=1) {
        ll step=n/len;
        for (ll i=0;i<n;i+=len) {
            for (ll k=0;k<(len>>1);k++) {
                ll u=f[i+k];
                ll v=f[i+k+(len>>1)]*root[step*k]%mod;
                f[i+k]=(u+v)%mod;
                f[i+k+(len>>1)]=(u-v)%mod;
                if (f[i+k+(len>>1)]<0)
                    f[i+k+(len>>1)]+=mod;
            }
        }
    }
    if (inv) {
        ll inv_n=modpow(n, mod-2, mod);
        for (ll i=0;i<n;i++)
            f[i]=f[i]*inv_n%mod;
    }
}

vl multiply(vl a, vl b, ll mod, ll w) {
    ll n=2;
    while (n<(ll)a.size()+(ll)b.size()) n<<=1;
    a.resize(n);
    b.resize(n);
    ntt(a, mod, w);
    ntt(b, mod, w);
```

```
    for (ll i=0;i<n;i++)
        a[i]=a[i]*b[i]%mod;
    ntt(a, mod, w, true);
    return a;
}
```

FWHT

BOJ 25563 - AND, OR, XOR

```
// Usage: vl mult = multiply(a, b);
// O(n*logn)
// Note: a, b는 ref가 아니라 복사
vl fwt_or(vl &x, bool inv) {
    vl a=x;
    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];
            }
        }
    }
    return a;
}

vl fwt_and(vl& x, bool inv) {
    vl a=x;
    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];
            }
        }
    }
```

```
    return a;
}

vl fwt_xor(vl& x, bool inv) {
    vl a=x;
    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+i]/=2;
                    a[l+h+i]/=2;
                }
            }
        }
    }
    return a;
}

vl multiply(vl a, vl b) {
    ll n = 1;
    while (n < max(a.size(), b.size())) n<<= 1;
    a.resize(n);
    b.resize(n);
    a = fwt_or(a, false);
    b = fwt_or(b, false);
    vl c(n);
    for (ll i=0;i<n;i++) c[i]=a[i]*b[i];
    return fwt_or(c, true);
}
```

Discrete Log and Discrete Root

solve B^x=N (mod M) and x^e=A (mod M)

BOJ 4357 - 이산 로그

```
// Discrete Log and Root
// Usage: DM dm(M); ll log = dm.discrete_log(B, N); ll root = dm.discrete_ro
ot(A, e);
// 둘 다 O(sqrt(M))
struct DM{
    static constexpr ll X=1e5; // X값은 sqrt(M)보다 크게
    ll mod;
    unordered_map<ll, ll> ht;
    vl aXe, iaXe;
    DM(ll Q) : mod(Q), aXe(X), iaXe(X) {}

    void build(ll B){
        ht.clear();
        ll cur=1;
        for(ll j=0;j<X;j++){
            if(!ht.contains(cur)) ht[cur]=j;
            cur=cur*B%mod;
        }
        ll gx=modpow(B,X,mod);
        aXe[0]=1;
        for(ll i=1;i<X;i++) aXe[i]=aXe[i-1]*gx%mod;
        ll igx=[&]{
            auto [u,v]=extended_gcd(gx,mod);
            ll t=u%mod;
            return (t+mod)%mod;
        }();
        iaXe[0]=1;
        for(ll i=1;i<X;i++) iaXe[i]=iaXe[i-1]*igx%mod;
    }

    // solve B^x=N
    ll discrete_log(ll B, ll N){
        build(B);
        for (ll i=0;i<X;i++) {
            ll need=N*iaXe[i]%mod;
            if (ht.contains(need)) return i*X+ht[need];
        }
        return -1;
```

```
}

// solve x^e=A
ll discrete_root(ll A, ll e) {
    ll m=mod-1;
    ll g=gcd(e,m);
    if (modpow(A, m/g, mod)!=1) return -1;
    auto get_factors=[&](ll x) {
        vl fs;
        for (ll i=2;i*i<=x;i++) {
            if (x%i==0) {
                fs.push_back(i);
                while (x%i==0) x/=i;
            }
        }
        if (x>1) fs.push_back(x);
        return fs;
    };
    vl fac=get_factors(m);
    ll r=2;
    while (true) {
        bool ok=true;
        for (ll f: fac)
            if(modpow(r,m/f,mod)==1) {
                ok=false;
                break;
            }
        if (ok) break;
        r++;
    }
    ll a=discrete_log(r, A);
    if (a<0) return -1;
    ll eg=e/g;
    ll mg=m/g;
    auto [iv, _]=extended_gcd(eg, mg);
    ll inv=iv%mg;
    if (inv<0) inv+=mg;
    ll k0=(__int128)(a/g)*inv%mg;
```

```
        ll step=modpow(r, mg, mod);
        vl cands;
        cands.reserve(g);
        ll cur=modpow(r, k0, mod);
        for (ll t=0;t<g;t++) {
            cands.push_back(cur);
            cur=(__int128)cur*step%mod;
        }
        return *min_element(cands.begin(), cands.end());
    }
};
```

DLAS Heuristic

```
// DLAS Heuristic
// Usage: auto [best_state, best_score] = dlas(init_state, iter);

struct State {
    State() {
        // state 생성
    }
    ll score() {
        // 이 점수를 최소화
    }
    void mutate() {
        // 무작위 변이
    }
};

pair<State, ll> dlas(State &init_state, ll iter) {
    vector s(3, init_state);
    vl buc(5, s[0].score());
    ll cur_score=buc[0];
    ll min_score=cur_score;
    ll cur_pos=0;
    ll min_pos=0;
    ll k=0;
    for (ll i=0;i<iter;i++) {
```

```
        ll prv_score=cur_score;
        ll nxt_pos=(cur_pos+1)%3;
        if (nxt_pos==min_pos)
            nxt_pos=(nxt_pos+1)%3;
        State cur_state=s[cur_pos];
        State &nxt_state=s[nxt_pos];
        nxt_state=cur_state;
        nxt_state.mutate();
        ll nxt_score=nxt_state.score();
        if (nxt_score<min_score) {
            i=0;
            min_pos=nxt_pos;
            min_score=nxt_score;
        }
        if (nxt_score==cur_score || nxt_score<*max_element(buc.begin(), buc.
end())) {
            cur_pos=nxt_pos;
            cur_score=nxt_score;
        }
        ll& fit=buc[k];
        if (cur_score>fit || cur_score<min(fit, prv_score)) {
            fit=cur_score;
        }
        k=(k+1)%5;
    }
    return {s[min_pos], min_score};
}
```

DP

Created	@2025년 5월 18일 오후 7:40
Tags	Bitset CHT D&C DP Knuth LIS SoS DP Tree DP 벌레캠프

Longest Increasing Subsequence

BOJ 12015 - 가장 긴 증가하는 부분 수열 2

```
// Usage: vl result = lis(arr);
// Time Complexity: O(n*logn)
vl lis(vl &arr) {
    ll n=arr.size();
    vl tmp, from;
    for (ll x:arr) {
        ll loc=lower_bound(tmp.begin(), tmp.end(), x)-tmp.begin();
        if (loc==tmp.size()) tmp.push_back(x);
        else tmp[loc]=x;
        from.push_back(loc);
    }
    vl ret=vl(tmp.size());
    ll target=tmp.size()-1;
    for (ll i=n-1;i>=0;i--) {
        if (target==from[i])
```

```
        ret[target--]=arr[i];
    }
    return ret;
}
```

Convex Hull Trick

BOJ 13263 - 나무 자르기

```
// Usage: CHT cht; cht.addLine(b[0], D[0]);
//       for(i = 1; i < n; ++i) { D[i] = cht.query(a[i]); cht.addLine(b[i], D[i]); }
// Memo: O(n^2) → O(n*logn), 아래 조건 중 하나 만족해야 함
// Memo: D[i] = max[j<i](D[j]+b[j]*a[i]), (b[k]<=b[k+1])
// Memo: D[i] = min[j<i](D[j]+b[j]*a[i]), (b[k]>=b[k+1])
struct CHT {
    struct Line {
        ll m, b;   // y = m*x + b
    };
    vector<Line> lines;
    vl xs;
    ll intersect(Line &l1, Line &l2) {
        ll num = l1.b - l2.b;
        ll den = l2.m - l1.m;
        // min인 경우 num과 den에 -1을 곱함
        return num>=0 ? (num+den-1)/den : num/den;
    }
    void addLine(ll m, ll b) {
        if (lines.size() && lines.back().m==m) {
            if (lines.back().b>=b) return; // min인 경우 부등호 반대로
            lines.pop_back();
            xs.pop_back();
        }
        Line L{m,b};
        while (lines.size()) {
            ll x=intersect(lines.back(), L);
            if (x<=xs.back()) {
                lines.pop_back();
                xs.pop_back();
```

```
            }
            else break;
        }
        if (lines.empty()) {
            lines.push_back(L);
            xs.push_back(LLONG_MIN);
        }
        else {
            lines.push_back(L);
            xs.push_back(intersect(lines[lines.size()-2], L));
        }
    }
    ll query(ll x) {
        ll idx=upper_bound(xs.begin(), xs.end(), x)-xs.begin()-1;
        return lines[idx].m*x+lines[idx].b;
    }
};
```

Divide and Conquer Optimization

BOJ 13261 - 탈옥

```
// Usage: f(x, 0, n-1, 0, n-1) → D[x][i]가 모두 채워짐
// Memo: D[t][i] = min[j<i](D[t-1][j] + C[j][i])
// Memo: O(kn^2) → O(knlogn), 아래 조건 만족해야 함
// Memo: C[a][c] + C[b][d] <= C[a][d] + C[b][c], (a<=b<=c<=d)

void f(ll t, ll s, ll e, ll l, ll r){
    if(s>e) return;
    ll m=(s+e)>>1;
    ll opt=l;
    for(ll i=l;i<=r;i++){
        if(D[t-1][opt]+C[opt][m]>D[t-1][i]+C[i][m]) opt=i;
    }
    D[t][m]=D[t-1][opt]+C[opt][m];
    f(t, s, m-1, l, opt);
    f(t, m+1, e, opt, r);
}
```

Tree DP

```
// Tree DP
// Usage: dfs(dfs, root, -1);
// Note: 0-based
vvl adj(v);
vl dp(v);
auto dfs=[&adj, &dp](auto self, ll node, ll pa) {
    for (auto &e: adj[node]) {
        if (e==pa) continue;
        self(self, e, node);
        /* dp expression */
    }
};
```

Knuth Optimization

BOJ 11066 - 파일 합치기

```
// Note: O(n^3) → O(n^2), 아래 3가지 조건 만족
// D[i][j] = min{i<k<j}(D[i][k]+D[k][j]) + C[i][j]
// C[a][c]+C[b][d] <= C[a][d]+C[b][c], (a<=b<=c<=d)
// C[b][c] <= C[a][d]
// Usage: vvl dp = knuth(arr);
vvl knuth(vl &arr) {
    constexpr ll INF = 2e18;
    ll i, k;
    ll n=arr.size();
    vl s(n+1); // 누적 합 배열
    for (i=0;i<n;i++) s[i+1]=s[i]+arr[i];

    vvl dp(n+1, vl(n+1)), opt(n+1, vl(n+1));
    for (i=1;i<=n;i++) {
        dp[i-1][i]=0;
        opt[i-1][i]=i;
    }

    for (i=2;i<=n;i++) {
```

```
        for (ll l=0;i+l<=n;l++) {
            ll r=i+l;
            dp[l][r]=INF;
            ll start=opt[l][r-1];
            ll end=opt[l+1][r];
            for (k=start;k<=end;k++) {
                ll cost=dp[l][k]+dp[k][r]+s[r]-s[l];
                if (cost<dp[l][r]) {
                    dp[l][r]=cost;
                    opt[l][r]=k;
                }
            }
        }
    }

    return dp;
}
```

Subset of Sum DP

```
// Usage: vl F = sos_dp(n, A);
// Note: mask의 부분 집합 x에 대하여, F[mask] = sum of A[x]
// Note: A.size == 2^n
// time: O(N*2^N), memory: O(2^N)
vl sos_dp(ll n, vl &A) {
    ll i;
    vl F(1<<n);
    for (i=0;i<(1<<n);i++) F[i]=A[i];
    for (i=0;i<n;i++) for (ll mask=0;mask<(1<<n);mask++) {
        if (mask&(1<<i)) F[mask]+=F[mask^(1<<i)];
    }
    return F;
}
```

Bitset Optimization

```
// ==================== 이 부분은 항상 코드의 맨 윗부분에 있어야 함====
=============
#define private public
#include <bitset>
#undef private
#include <x86intrin.h>
// =======================================================
==========================

template <size_t _Nw>
void _M_do_sub(_Base_bitset<_Nw> &A, const _Base_bitset<_Nw> &B) {
   for (int i=0, c=0;i<_Nw;i++)
      c=_subborrow_u64(c, A._M_w[i], B._M_w[i], (ull*)&A._M_w[i]);
}
template <>
void _M_do_sub(_Base_bitset<1> &A, const _Base_bitset<1> &B) {
   A._M_w -= B._M_w;
}
template <size_t _Nb>
bitset<_Nb> &operator-=(bitset<_Nb> &A, const bitset<_Nb> &B) {
   _M_do_sub(A, B);
   return A;
}
template <size_t _Nb>
inline bitset<_Nb> operator-(const bitset<_Nb> &A, const bitset<_Nb> &B)
{
   bitset<_Nb> C(A);
   return C -= B;
}
template <size_t _Nw>
void _M_do_add(_Base_bitset<_Nw> &A, const _Base_bitset<_Nw> &B) {
   for (int i=0, c=0;i<_Nw;i++)
      c=_addcarry_u64(c, A._M_w[i], B._M_w[i], (ull*)&A._M_w[i]);
}
template <>
void _M_do_add(_Base_bitset<1> &A, const _Base_bitset<1> &B) {
   A._M_w += B._M_w;
}
```

```
template <size_t _Nb>
bitset<_Nb> &operator+=(bitset<_Nb> &A, const bitset<_Nb> &B) {
    _M_do_add(A, B);
    return A;
}
template <size_t _Nb>
inline bitset<_Nb> operator+(const bitset<_Nb> &A, const bitset<_Nb> &B)
{
    bitset<_Nb> C(A);
    return C += B;
}
```

Berlekamp-Massey

BOJ 11726 - 2×n 타일링

BOJ 9095 - 1, 2, 3 더하기

BOJ 9461 - 파도반 수열

BOJ 1492 - 합

```
// Usage: vl init { 2L개 이상의 초기 항 }; ll result = guess_nth_term(init, n);
// Note: 상수 계수 L차 선형 점화식의 n번째 항을 찾을 때 사용. 0-based
// O(L^2*logn)
vl berlekampMassey(vl s) {
    ll n=s.size(), L=0, m=1, d=0, coef=0;
    vl C(n), B(n), T(n);
    C[0]=B[0]=1;
    ll b=1;
    for (ll i=0;i<n;i++) {
        d=0;
        for (ll j=0;j<=L;j++)
            d=(d+C[j]*s[i-j])%MOD;
        if (d==0) {
            ++m;
            continue;
        }
        coef=d*modpow(b, MOD-2, MOD)%MOD;
        T=C;
```

```
        for (ll j=0;j+m<n;j++) {
            C[j+m]=(C[j+m]-coef*B[j])%MOD;
            if (C[j+m]<0) C[j+m]+=MOD;
        }
        if (2*L<=i) {
            L=i+1-L;
            B=T;
            b=d;
            m=1;
        }
        else {
            ++m;
        }
    }
    C.resize(L+1);
    vl tr(L);
    for (ll i=1;i<=L;++i)
        tr[i-1]=(MOD-C[i])%MOD;
    return tr;
}

ll get_nth(vl &S, vl &tr, ll k) {
    ll n=tr.size();
    auto combine=[&](vl &a, vl &b) {
        vl res(2*n+1);
        for (ll i=0;i<=n;i++)
            for (ll j=0;j<=n;j++)
                res[i+j]=(res[i+j]+a[i]*b[j])%MOD;
        for (ll i=2*n;i>n;i--)
            for (ll j=1;j<=n;j++)
                res[i-j]=(res[i-j]+res[i]*tr[j-1])%MOD;
        res.resize(n+1);
        return res;
    };

    vl pol(n+1), e(n+1);
    e[1]=1;
    pol[0]=1;
```

```
    for (++k;k>0;k>>=1) {
        if (k&1) pol=combine(pol, e);
        e=combine(e, e);
    }
    ll res=0;
    for (ll i=0;i<n;i++)
        res=(res+pol[i+1]*S[i])%MOD;
    return res;
}

ll guess_nth_term(vl x, ll n) {
    if (n<x.size()) return x[n];
    vl tr=berlekampMassey(x);
    if (tr.empty()) return x[0];
    return get_nth(x, tr, n);
}
```

Graph

Created	@2025년 5월 17일 오후 6:36
Tags	BM DSU by size LCA OfDC SCC bellman dijk floyd kruskal prim

Floyd-Warshall's Algorithm

```
//Usage: auto [negative cycle, distance]=floyd(V, adj);
//O(V^3)
pair<bool,vvl> floyd(ll n, vector<vector<pll>>& adj){
    bool cycle = 0;
    const ll INF = 1e18;
    vvl dis(n,vector<ll>(n, INF));
    for (ll i=0;i<n;++i) dis[i][i] = 0;

    for (ll u=0;u<n;++u)
        for (auto& [v,w]:adj[u])
```

```
            dis[u][v]=min(dis[u][v],w); //multi-edges?
    for (ll k=0;k<n;++k)
        for (ll i=0;i<n;++i)
            for (ll j=0;j<n;++j)
                dis[i][j]=min(dis[i][j],dis[i][k]+dis[k][j]);
    for (ll k=0;k<n;++k) // Check negative cycle
        for (ll i=0;i<n;++i)
            for (ll j=0;j<n;++j)
                if (dis[i][j]>dis[i][k]+dis[k][j]) cycle=1;
    return {!cycle, dis};
}
```

Dijkstra's Algorithm

```
//Usage: vl distance = dijk(V, start, adj);
//O(ElogV)
vl dijk(ll n, ll s, vector<vector<pll>>& adj){
    const ll INF = 1e18;
    vl dis(n,INF);
    vector<bool> visit(n, false);
    priority_queue<pll, vector<pll>, greater<pll> > q; // pair(dist, v)
    dis[s] = 0;
    q.push({dis[s], s});
    while (!q.empty()){
        while (!q.empty() && visit[q.top().second]) q.pop();
        if (q.empty()) break;
        ll next=q.top().second; q.pop();
        visit[next]=1;
        for (ll i=0;i<adj[next].size();++i)
            if (dis[adj[next][i].first] > dis[next] + adj[next][i].second){
                dis[adj[next][i].first] = dis[next] + adj[next][i].second;
                q.push({dis[adj[next][i].first], adj[next][i].first});}}
    for(ll i=0;i<n;i++) if(dis[i]==INF) dis[i]=-1;
    return dis;
}
```

Bellman-Ford Algorithm

```
//Usage: auto [negative cycle, distance] = bellman(V, start, adj);
//O(VE)
pair<bool,vl> bellman(ll n, ll s, vector<vector<pll>>& adj){
    bool cycle=0;
    const ll INF=1e18;
    vector<ll>dis(n,INF);
    dis[s]=0;
    for (ll i=0;i<n;++i)
        for (ll j=0;j<n;++j)
            for (ll k=0;k<adj[j].size();++k){
                ll next=adj[j][k].first;
                ll cost=adj[j][k].second;
                if (dis[j]!=INF && dis[next]>dis[j]+cost) {
                    dis[next]=dis[j]+cost;
                    if (i==n-1) cycle=1;
                }
            }
    return {!cycle, dis};
}
```

Prim Algorithm

```
//Usage: ll mst = prim(V, adj);
// O(ElogV)
ll prim(ll n,vector<vector<pll>>& adj) {
    vector<bool> visit(n, false);
    priority_queue<pll, vector<pll>, greater<pll> > q;
    ll count=0; ll ret=0;
    q.push(make_pair(0, 0)); // (cost, vertex)
    while (!q.empty()){
        ll x=q.top().second; // also able to get edges
        visit[x]=1; ret+=q.top().first; q.pop(); count++;
        for (ll i=0;i<adj[x].size();++i)
            q.push({adj[x][i].second, adj[x][i].first});
        while (!q.empty() && visit[q.top().second]) q.pop();
    }
    if (count!=n) return -1;
```

```
    else return ret;
}
```

Kruskal's Algorithm

```
//Usage: ll mst = kruskal(V, adj);
// O(ElogE)
ll kruskal(ll n,vector<vector<pll>>& adj){
    DSU dsu(n);
    ll ret = 0;
    vector<pair<ll, pll>> e;
    for(ll i= 0; i < n; i++)
        for(ll j=0; j < adj[i].size(); j++)
            e.push_back({adj[i][j].second, {i, adj[i][j].first}});
    sort(e.begin(), e.end());
    for(ll i=0; i < e.size(); i++){
        ll x = e[i].second.first,y = e[i].second.second;
        if(dsu.find(x) != dsu.find(y)){
            dsu._union(x, y);
            ret += e[i].first;
        }
    }
    ll p=dsu.find(0);
    for(ll i=1;i<n;i++){
        if(dsu.find(i)!=p) return -1;
    }
    return ret;
}
```

DSU by size

```
//Usage: DSU dsu(V); ll root = dsu._find(node); dsu._union(node,node);
// O(alpha(V))
struct DSU {
    vl par, sz;

    DSU(ll n) {
```

```
        par.resize(n+1);
        sz.assign(n+1,1);
        iota(par.begin(),par.end(),0);
    }
    ll _find(ll x) {
        if (par[x]==x) return x;
        return par[x]=_find(par[x]);
        //for RollBack
        //return _find(par[x]);
    }
    pll _union(ll x,ll y){
        x=_find(x);
        y=_find(y);
        if (x==y) return {-1,-1};
        if (sz[x]<sz[y]) swap(x,y);
        par[y]=x;
        sz[x]+=sz[y];
        return {x,y};
    }
    void _delete(ll x, ll y){
        sz[x]-=sz[y];
        par[y]=y;
    }
};
```

LCA

```
// Usage: LCA lca(V,tree); ll anc = lca.solve(u,v);
// O(logV)
// memo : 0-indexed

struct LCA {
    ll MAXLN;
    vl depth; vvl anc;

    LCA(ll n, vvl& tree){
        ll root = 0;
```

```
        depth.assign(n,0);
        MAXLN=1;
        while ((1<<MAXLN)<=n) ++MAXLN;
        anc.assign(MAXLN,vl(n));

        function<void(ll,ll)> dfs4lca = [&](ll node,ll parent) {
            for (ll next: tree[node]) {
                if (next==parent) continue;
                depth[next]=depth[node]+1;
                anc[0][next]=node;
                dfs4lca(next, node);
            }
        };

        dfs4lca(root,-1);
        anc[0][root]=root;
        for (ll i=1;i<MAXLN;++i)
            for (ll j=0;j<n;++j)
                anc[i][j]=anc[i-1][anc[i-1][j]];
    }

    ll solve(ll u, ll v){
        if (depth[u]<depth[v]) swap(u, v);
        if (depth[u]>depth[v]) {
            for (ll i=MAXLN-1;i>=0;--i)
                if (depth[u]-(1<<i) >= depth[v])
                    u=anc[i][u];
        }
        if (u==v) return u;
        for (ll i=MAXLN-1;i>=0;--i) {
            if (anc[i][u]!=anc[i][v]) {
                u=anc[i][u];
                v=anc[i][v];
            }
        }
        return anc[0][u];
    }
};
```

Bipartite Matching

```
// Usage :
// Constructor1 : BipartiteGraph bg(Lsize, Rsize); bg.add_edge(l_node, r_no
de);
// Constructor2 : BipartiteGraph bㄹg(vvl Adj);
// Maximum matching : bg.maximum_matching();
// O(E*sqrt(V))
struct BipartiteGraph {
    ll n; vl color; vvl adj;
    bool is_bipartite = true;
    ll leftSz, rightSz;
    vl leftNodes, rightNodes;   // partition idx → original idx
    vl leftId, rightId;         // original idx → partition idx
    vvl adjL;                   // left index → [right indices...]
    // maximum matching
    ll matching, distNil, INF = 1e10;
    vl pairL, pairR, dist;

    // 엣지를 바로 입력
    BipartiteGraph(ll L, ll R)
        : leftSz(L), rightSz(R), adjL(L) {}

    void add_edge(ll l, ll r) {
        adjL[l].emplace_back(r);
    }

    void erase_edge(ll ln, ll rn) {  // adjL 정렬 후 사용
        ll l = leftId[ln], r = rightId[rn];
        adjL[l].erase(lower_bound(adjL[l].begin(), adjL[l].end(), r));
    }

    // 일반 그래프 → 이분 그래프 변환
    BipartiteGraph(const vvl& G)
      : n(G.size()), adj(G), color(n, 0)
    {
        for (ll i=0; i<n; ++i) {    // 0-based index
```

```
        if (color[i] == 0 && !bfs_color(i)) {
            is_bipartite = false;
            return;
        }
    }
    build_bipartite();    // 좌우 파티션 노드 분리, 매핑, 인접 리스트 생성
}

bool bfs_color(ll s) {
    queue<ll> q;
    color[s] = 1;
    q.push(s);
    while (!q.empty()) {
        ll node = q.front(); q.pop();
        for (auto& nxt : adj[node]) {
            if (color[nxt] == 0) {
                color[nxt] = -color[node];
                q.push(nxt);
            } else if (color[nxt] == color[node])
                return false;
        }
    }
    return true;
}

void build_bipartite() {
    leftId.assign(n, -1);
    rightId.assign(n, -1);
    for (ll i=0; i<n; ++i) {
        if (color[i] == 1) {
            leftId[i] = leftNodes.size();
            leftNodes.push_back(i);
        } else {
            rightId[i] = rightNodes.size();
            rightNodes.push_back(i);
        }
    }
    leftSz = leftNodes.size();
```

```
    rightSz = rightNodes.size();
    adjL.assign(leftSz, {});
    for (ll l=0; l<leftSz; ++l)
        for (auto& r : adj[leftNodes[l]])
            adjL[l].emplace_back(rightId[r]);
}

// maximum matching: Hopcroft-Karp
ll max_matching() {
    matching = 0;
    pairL.assign(leftSz, -1);
    pairR.assign(rightSz, -1);
    dist.assign(leftSz, 0);
    while (bfs_HK())    // augmenting path 존재하는 동안 반복
        for (ll l=0; l<leftSz; ++l)
            if (pairL[l]==-1 && dfs_HK(l))
                matching++;
    return matching;
}

bool bfs_HK() {    // 가장 짧은 augmenting path 찾음
    queue<ll> q;
    for (ll l=0; l<leftSz; l++) {
        if (pairL[l] == -1) {
            dist[l] = 0;
            q.emplace(l);
        } else dist[l] = INF;
    }
    distNil = INF;
    while (!q.empty()) {
        ll l = q.front(); q.pop();
        if (dist[l] < distNil) {
            for (auto& r : adjL[l]) {
                ll pl = pairR[r];
                if (pl != -1) {
                    if (dist[pl] == INF) {
                        dist[pl] = dist[l] + 1;
                        q.emplace(pl);
```

```
                }
            } else distNil = dist[l] + 1;
        }
    }
  }
  return distNil != INF;
}

bool dfs_HK(ll l) {
  for (ll r : adjL[l]) {
    ll pl = pairR[r];
    if (pl==-1 || (dist[pl]==dist[l]+1 && dfs_HK(pl))) {
      pairL[l] = r;
      pairR[r] = l;
      return true;
    }
  }
  dist[l] = INF;
  return false;
}
};
```

```
// Usage: BipartiteMatching bm(leftV,rightV,graph);
//        ll ans = bm.max_matching; vl LtoR=bm.match; vl RtoL=bm.matched;
// O(E*sqrt(V))
// memo: vertex cover: (reached[0][left_node] == 0) || (reached[1][right_no
de] == 1)
struct BM {
  ll n, m, max_matching;
  vvl graph;
  vl matched, match, edgeview, level;
  vl reached[2];
  BM(ll n, ll m, vvl& graph) : n(n), m(m), graph(graph), matched(m,-1), matc
h(n,-1) {
    ll max_matching = 0;
    while (assignLevel()) {
      edgeview.assign(n, 0);
      for (ll i = 0; i < n; i++)
```

```
            if (match[i]==-1)
                max_matching += findpath(i);
    }
}
bool assignLevel(){
    bool reachable = false;
    level.assign(n,-1);
    reached[0].assign(n, 0);
    reached[1].assign(m, 0);
    queue<ll> q;
    for (ll i = 0; i < n; i++) {
        if (match[i] ==-1) {
            level[i] = 0;
            reached[0][i] = 1;
            q.push(i);
        }
    }
    while (!q.empty()) {
        auto cur = q.front(); q.pop();
        for (auto adj : graph[cur]) {
            reached[1][adj] = 1;
            auto next = matched[adj];
            if (next ==-1) {
                reachable = true;
            }
            else if (level[next] ==-1) {
                level[next] = level[cur] + 1;
                reached[0][next] = 1;
                q.push(next);
            }
        }
    }
    return reachable;
}
ll findpath(ll node){
    for (ll &i = edgeview[node]; i < graph[node].size(); i++) {
        ll adj = graph[node][i];
        ll next = matched[adj];
```

```
            if (next >= 0 && level[next] != level[node] + 1) continue;
            if (next ==-1 || findpath(next)) {
                match[node] = adj;
                matched[adj] = node;
                return 1;
            }
        }
        return 0;
    };
};
```

SCC

```
//Usage: SCC scc(V, graph); vl component = scc.scc_idx;
// O(V+E)
// memo: the order of scc_idx constitutes a reverse topological sort
struct SCC {
    ll n,vtime,scc_cnt;
    vvl graph;
    vl up, visit,scc_idx,stk;

    SCC(ll n, vvl& graph):
    n(n),graph(graph),up(n),visit(n,0),scc_idx(n,0),vtime(0),scc_cnt(0) {
        for (ll i=0;i<n;++i)
            if (visit[i]==0) dfs(i);
    }

    void dfs(ll node){
        up[node] = visit[node] = ++vtime;
        stk.push_back(node);
        for (ll next : graph[node]){
            if (visit[next] == 0) {
                dfs(next);
                up[node] = min(up[node], up[next]);
            }
            else if (scc_idx[next] == 0)
                up[node] = min(up[node], visit[next]);
```

```
        }
        if (up[node]==visit[node]){
            ++scc_cnt;
            ll t;
            do{
                t = stk.back();
                stk.pop_back();
                scc_idx[t] = scc_cnt;
            } while (!stk.empty() && t != node);
        }
    }
};
```

OFDC

```
//Usage: vector<tlll> query; OFDC ofdc(V, #query, query);
//O(QlogQ * alpha(V))
struct OFDC{
    vector<tlll>query;
    vector<vector<pll>> tree;
    map<pll,ll>connected_time;
    ll n, q; vl ans;
    DSU dsu;

    OFDC(ll n, ll q,vector<tlll>&query): n(n), q(q), query(query), tree(4*(q+
1)), dsu(n+1) {
        for(ll i=0;i<q;i++){
            auto&[type,u,v]=query[i];
            if(u>v)swap(u,v);
            if(type==1) connected_time[{u,v}]=i; //union
            else if(type==2){ //delete
                update(1,0,q,connected_time[{u,v}],i,{u,v});
                connected_time.erase({u,v});
            }
        }
        for(auto&[edge,time]:connected_time){
            auto&[u,v]=edge;
```

}

```
            update(1,0,q,time,q,{u,v});
        }
        dfs(1,0,q);
    }

    void update(ll node, ll s, ll e, ll l, ll r, pll edge){
        if(r<s||e<l) return;
        if(l<=s&&e<=r){
            tree[node].pb(edge);
            return;
        }
        ll mid=(s+e)>>1;
        update(node<<1,s,mid,l,r,edge);
        update(node<<1|1,mid+1,e,l,r,edge);
    }

    void dfs(ll node, ll s, ll e){
        vector<pll>real_connected;
        for(auto&[u,v]:tree[node]){
            auto [x,y]=dsu._union(u,v);
            if(x!=-1) real_connected.push_back({x,y});
        }
        if(s==e){
            if(get<0>(query[s])==3){ //connect?
                ans.pb((dsu._find(get<1>(query[s]))==dsu._find(get<2>(query
[s]))));
            }
        }
        else{
            ll mid = (s+e)>>1;
            dfs(node<<1, s, mid);
            dfs(node<<1|1, mid+1, e);
        }
        reverse(all(real_connected));
        for(auto&[x,y]:real_connected) dsu._delete(x,y);
    }
};
```

HLD

```
// Usage: auto [sz, dep, par, in, out, top] = get_hld(adj);
// Time Complexity: O(V)
// Memo: 1-indexed
tuple<vl, vl, vl, vl, vl, vl> get_hld(vvl adj) {
    ll n = adj.size() - 1;
    vl sz(n+1, 1), dep(n+1), par(n+1);
    vl in(n+1), out(n+1), top(n+1);
    ll ord = 0;

    auto dfs1 = [&](auto& self, ll cur, ll prv){
        if (prv) adj[cur].erase(ranges::find(adj[cur], prv));
        for (ll &nxt : adj[cur]) {
            dep[nxt] = dep[cur] + 1;
            par[nxt] = cur;
            self(self, nxt, cur);
            sz[cur] += sz[nxt];
            if (sz[adj[cur][0]] < sz[nxt]) swap(adj[cur][0], nxt);
        }
    };

    auto dfs2 = [&](auto& self, ll cur){
        in[cur] = ++ord;
        for (ll nxt : adj[cur]) {
            top[nxt] = (nxt == adj[cur][0] ? top[cur] : nxt);
            self(self, nxt);
        }
        out[cur] = ord;
    };

    dfs1(dfs1, 1, 0);
    dfs2(dfs2, top[1] = 1);
    return {sz, dep, par, in, out, top};
}
```

Tree Isomorphism

```
// Usage: TreeIsomorphism ti(T1, T2) or ti(T1, r1, T2, r2)
//       bool isIso = ti.isIsomorphic
//       ti.isTreeIsomorphicMap ⇒ ti.mapping[i]=j := T1의 i노드는 T2의 j노드와 대응
// O(NlogN)
struct TreeIsomorphism {
    ll id = 1;
    vl root1, root2, mapping;
    vvl tree1, tree2;
    map<vl,ll> isoClass;

    TreeIsomorphism(const vvl& T1, const vvl& T2)
        : tree1(T1), tree2(T2) {
        root1 = findCenter(T1);
        root2 = findCenter(T2);
    }
    TreeIsomorphism(const vvl& T1, ll r1, const vvl& T2, ll r2)
        : tree1(T1), tree2(T2), root1({r1}), root2({r2}) {}

    vector<ll> findCenter(const vvl &tree) {
        ll n = tree.size();
        vl degree(n), leaves;
        for (ll i=0; i<n; i++) {
            degree[i] = tree[i].size();
            if (degree[i] <= 1)
                leaves.emplace_back(i);
        }
        ll removed = leaves.size();
        while (removed < n) {
            vl newLeaves;
            for (ll u : leaves)
                for (ll v : tree[u])
                    if (--degree[v] == 1)
                        newLeaves.push_back(v);
            removed += newLeaves.size();
            leaves = move(newLeaves);
        }
        return leaves;
```

```
}

ll getID(const vvl& tree, ll node, ll pa) {
    vl childID;
    for (auto& ch : tree[node])
        if (ch != pa)
            childID.emplace_back(getID(tree, ch, node));
    sort(childID.begin(), childID.end());
    if (!isoClass.contains(childID))
        isoClass[childID] = id++;
    return isoClass[childID];
}

bool isTreeIsomorphic() {
    if (tree1.size() <= 1 || tree2.size() <= 1)
        return tree1.size() == tree2.size();
    ll id1 = getID(tree1, root1[0], -1);
    for (auto& r : root2)
        if (id1 == getID(tree2, r, -1))
            return true;
    return false;
}

void mapSubtree(ll node1, ll pa1, ll node2, ll pa2) {
    mapping[node1] = node2;
    vector<pll> ch1, ch2;
    for (auto& ch : tree1[node1])
        if (ch != pa1)
            ch1.emplace_back(getID(tree1, ch, node1), ch);
    for (auto& ch : tree2[node2])
        if (ch != pa2)
            ch2.emplace_back(getID(tree2, ch, node2), ch);
    sort(ch1.begin(), ch1.end());
    sort(ch2.begin(), ch2.end());
    for (ll i=0; i<ch1.size(); i++)
        mapSubtree(ch1[i].second, node1, ch2[i].second, node2);
}
```

```
    bool isTreeIsomorphicMap() {
        if (tree1.size() <= 1 || tree2.size() <= 1) {
            if (tree1.size() != tree2.size()) return false;
            mapping.assign(tree1.size(), 0);
            return true;
        }
        ll id1 = getID(tree1, root1[0], -1);
        for (auto& r : root2) {
            if (id1 == getID(tree2, r, -1)) {
                mapping.assign(tree1.size(), -1);
                mapSubtree(root1[0], -1, r, -1);
                return true;
            }
        }
        return false;
    }
};
```

Maximum Flow (Dinic)

```
// Usage : DINIC flow(#node)
//         flow.add_edge(start, end, capacity)
//         ans = flow.solve(source, sink)
// O(V^2*E)
struct DINIC {
    struct Edge { ll nxt, rev, res; };
    ll n; vl level, start;
    vector<vector<Edge>> graph;
    DINIC(ll _n): n(_n), graph(n), level(n), start(n) {}
    void add_edge(ll s, ll e, ll cap, ll rev_cap = 0) {
        graph[s].push_back({e, (ll)graph[e].size(), cap});
        graph[e].push_back({s, (ll)graph[s].size() - 1, rev_cap});
    }
    bool assign_level(ll src, ll sink) {
        fill(level.begin(), level.end(), -1);
        queue<ll> q;
        level[src] = 0; q.emplace(src);
```

```
        while (!q.empty()) {
            ll cur = q.front(); q.pop();
            for (auto& [nxt,rev,res] : graph[cur]) {
                if (level[nxt]==-1 && res>0) {
                    level[nxt] = level[cur] + 1;
                    q.emplace(nxt);
                }
            }
        }
        return level[sink] != -1;
    }
    ll block_flow(ll cur, ll sink, ll flow) {
        if (cur==sink) return flow;
        for (ll& i=start[cur]; i<graph[cur].size(); i++) {
            auto& [nxt,rev,res] = graph[cur][i];
            if (res>0 && level[nxt]==level[cur]+1) {
                ll pushed = block_flow(nxt, sink, min(flow, res));
                if (pushed > 0) {
                    res -= pushed;
                    graph[nxt][rev].res += pushed;
                    return pushed;
                }
            }
        }
        return 0;
    }
    ll solve(ll src, ll sink) {
        ll total=0;
        while (assign_level(src, sink)) {
            fill(start.begin(), start.end(), 0);
            while (ll pushed = block_flow(src, sink, LLONG_MAX)) {
                total += pushed;
            }
        }
        return total;
    }
};
```

Min-Cost Maximum Flow (SSAP)

```
// Usage : MCMF flow(#node)
//        flow.add_edge(start, end, capacity, cost)
//        [maxFlow, minCost] = flow.solve(source, sink, [한 번에 흘릴 수 있는 최대 유량])
// O(VE + F·ElogV)  **F:=증강 횟수
struct MCMF {
    struct Edge { ll nxt, rev, res, cost; };
    ll n;
    vector<vector<Edge>> g;
    MCMF(ll n): n(n), g(n) {}

    void add_edge(ll s, ll e, ll cap, ll cost, ll rev_cap = 0){
        g[s].emplace_back(e, (ll)g[e].size(), cap, cost);
        g[e].emplace_back(s, (ll)g[s].size()-1, rev_cap, -cost);
    }

    // s->t로 최대 maxf만큼 보냄(기본: 무한). (flow, cost) 반환
    pll solve(ll src, ll sink, ll maxf = LLONG_MAX){
        const ll INF = LLONG_MAX;
        ll flow = 0, minCost = 0;

        vl pi(n, 0), dist(n), avail(n);
        vl pv(n), pe(n);    // prev vertex, prev edge idx

        // 초기 포텐셜: 음수비용 간선이 있을 수 있으면 SPFA/BF로 한 번 계산
        auto spfa_init = [&](){
            deque<ll> dq; vector<bool> inq(n,false);
            fill(pi.begin(), pi.end(), INF);
            pi[src] = 0; dq.push_back(src); inq[src] = true;
            while (!dq.empty()){
                ll cur = dq.front(); dq.pop_front(); inq[cur] = false;
                for (auto& [nxt,rev,res,cost] : g[cur])
                    if (res>0 && pi[nxt]>pi[cur]+cost) {
                        pi[nxt] = pi[cur] + cost;
                        if (!inq[nxt]) {
                            inq[nxt] = true;
```

```
                    if (!dq.empty() && pi[nxt]<pi[dq.front()]) dq.push_front(nx
t);
                    else dq.push_back(nxt);
                }
            }
        }
        for (ll i=0; i<n; i++)
            if (pi[i] == INF) pi[i] = 0; // 도달불가는 0으로
    };
    spfa_init();

    while (flow < maxf){
        fill(dist.begin(), dist.end(), INF);
        fill(avail.begin(), avail.end(), 0);
        dist[src] = 0; avail[src] = INF;
        priority_queue<pll,vector<pll>,greater<pll>> pq;
        pq.emplace(0, src);
        while (!pq.empty()){
            auto [d,cur] = pq.top(); pq.pop();
            if (d != dist[cur]) continue;
            for (ll i=0; i<g[cur].size(); i++) {
                auto& [nxt,rev,res,cost] = g[cur][i];
                if (res <= 0) continue;
                ll w = cost + pi[cur] - pi[nxt]; // 감소비용
                if (dist[nxt] > dist[cur]+w){
                    dist[nxt] = dist[cur] + w;
                    pv[nxt] = cur; pe[nxt] = i;
                    avail[nxt] = min(avail[cur], res);
                    pq.push({dist[nxt], nxt});
                }
            }
        }
        if(dist[sink] == INF) break; // 더 이상 증가경로 없음
        for(ll i=0; i<n; i++)
            if(dist[i] < INF) pi[i] += dist[i]; // 포텐셜 업데이트
        ll add = min(avail[sink], maxf-flow);
        flow += add;
        for (ll i=sink; i!=src; i=pv[i]) {
```

```
                auto& [nxt,rev,res,cost] = g[pv[i]][pe[i]];
                res -= add;
                g[i][rev].res += add;
                minCost += add*cost;
            }
        }
        return {flow, minCost};
    }
};
```

Data Structure

Created	@2025년 5월 17일 오후 6:36
Tags	2-SAT FW FW2D LAZY SEG SEG2D gmSEG iterLAZY iterSEG

SEG

```
// usage: SEG(n, arr); update(tar, val); ll sum = query(qs, qe);
// memo : 0-indexed
struct SEG {
    ll n;
    vl seg, arr;

    SEG(ll n, vl& arr): n(n),arr(arr),seg(n<<2){
        init(1,0,n-1);
    }

    void init(ll node,ll l,ll r) {
        if (l==r) {
            seg[node]=arr[l];
            return;
```

```
        }
        ll m=(l+r)>>1;
        init(node<<1,l,m);
        init(node<<1|1,m+1,r);
        seg[node]=seg[node<<1]+seg[node<<1|1];
    }

    ll range_sum(ll node, ll l, ll r, ll s, ll e) {
        if (r<s||e<l) return 0;
        if (s<=l&&r<=e) return seg[node];
        ll m=(l+r)>>1;
        return range_sum(node<<1,l,m,s,e) + range_sum(node<<1|1,m+1,r,s,e);
    }

    void point_update(ll node, ll l, ll r, ll tar, ll val) {
        if (r<tar||tar<l) return;
        if (l==r) {
            seg[node]=val;
            return;
        }
        ll m=(l+r)>>1;
        point_update(node<<1,l,m,tar,val);
        point_update(node<<1|1,m+1,r,tar,val);
        seg[node]=seg[node<<1]+seg[node<<1|1];
    }
    void update(ll tar,ll val){point_update(1,0,n-1,tar,val);}
    ll query(ll s,ll e){return range_sum(1,0,n-1,s,e);}
};
```

LAZY

```
// usage: LAZY(n, arr); update(qs, qe, val); ll sum = query(qs, qe);
// memo : 0-indexed
struct LAZY {
    ll n;
```

```
vl seg, lazy, arr;

LAZY(ll n, vl& arr): n(n),arr(arr),seg(n<<2),lazy(n<<2){
    init(1,0,n-1);
}

void init(ll node,ll l,ll r) {
    if (l==r) {
        seg[node]=arr[l];
        return;
    }
    ll m=(l+r)>>1;
    init(node<<1,l,m);
    init(node<<1|1,m+1,r);
    seg[node]=seg[node<<1]+seg[node<<1|1];
}

void relax(ll node, ll l, ll r) {
    if (lazy[node]==0) return;
    if (l<r) {
        ll m = (l+r)>>1;
        seg[node<<1]+=(m-l+1)*lazy[node];
        lazy[node<<1]+=lazy[node];
        seg[node<<1|1]+=(r-m)*lazy[node];
        lazy[node<<1|1]+=lazy[node];
    }
    lazy[node]=0;
}

ll range_sum(ll node, ll l, ll r, ll s, ll e) {
    if (r<s||e<l) return 0;
    if (s<=l&&r<=e) return seg[node];
    relax(node,l,r);
    ll m=(l+r)>>1;
    return range_sum(node<<1,l,m,s,e) + range_sum(node<<1|1,m+1,r,s,e);
}

void range_update(ll node, ll l, ll r, ll s, ll e, ll val) {
```

```
        if (r<s||e<l) return;
        if (s<=l&&r<=e) {
            seg[node]+=(r-l+1)*val;
            lazy[node]+=val;
            return;
        }
        relax(node,l,r);
        ll m=(l+r)>>1;1
        range_update(node<<1,l,m,s,e,val);
        range_update(node<<1|1,m+1,r,s,e,val);
        seg[node]=seg[node<<1]+seg[node<<1|1];
    }
    void update(ll s,ll e,ll val){range_update(1,0,n-1,s,e,val);}
    ll query(ll s,ll e){return range_sum(1,0,n-1,s,e);}
};
```

FW

```
//FW fw(n); fw.add(idx,val); ll sum=fw.range_query(l,r);
//memo : 0-indexed
struct FW {
    ll n;
    vl fw;
    FW(ll n):n(n),fw(n+1) {}

    void add(ll i,ll val) {
        for (++i;i<=n;i+=i&-i) fw[i]+=val;
    }

    ll query(ll i) {
        ll ret=0;
        for (++i;i>0;i-=i&-i) ret+=fw[i];
        return ret;
    }
```

```
    ll range_query(ll l, ll r){return query(r)-query(l-1);}
};
```

FW2D

```
//FW2D fw2d(n); fw2d.add(i,j,val); ll sum=fw2d.range_query(x1,y1,x2,y2);
//memo : 0-indexed
struct FW2D {
    ll n,m;
    vvl fw;
    FW2D(ll n, ll m):n(n),m(m),fw(n+1,vl(m+1)) {}

    void add(ll i,ll j,ll val) {
        for (++i;i<=n;i+=i&-i)
            for (ll k=j+1;k<=m;k+=k&-k)
                fw[i][k]+=val;
    }

    ll query(ll i,ll j) {
        ll ret=0;
        for (++i;i>0;i-=i&-i)
            for (ll k=j+1;k>0;k-=k&-k)
                ret+=fw[i][k];
        return ret;
    }

    ll range_query(ll x1, ll y1, ll x2, ll y2) {
        return query(x2,y2)+query(x1-1,y1-1)-query(x1-1,y2)-query(x2,y1-1);
    }
};
```

SEG2D

```
//usage: SEG2D seg2d(n,m,arr); seg2d.update(i,j,val); ll sum=seg2d.query
(x1,y1,x2,y2);
```

```
//memo: 0-indexed
struct SEG2D {
    ll n,m;
    vvl seg,arr;

    SEG2D(ll n,ll m,vvl& arr): n(n),m(m),arr(arr) {
        seg.assign(n<<2,vl(m<<2));
        build_x(1,0,n-1);
    }

    void build_y(ll vx,ll vy,ll lx,ll ly,ll rx,ll ry) {
        if (ly==ry) {
            if (lx==rx) seg[vx][vy]=arr[lx][ly];
            else seg[vx][vy]=seg[vx<<1][vy]+seg[vx<<1|1][vy];
            return;
        }
        ll mid=(ly+ry)>>1;
        build_y(vx,vy<<1,lx,ly,rx,mid);
        build_y(vx,vy<<1|1,lx,mid+1,rx,ry);
        seg[vx][vy]=seg[vx][vy<<1]+seg[vx][vy<<1|1];
    }
    void build_x(ll vx,ll lx,ll rx) {
        if (lx!=rx) {
            ll mid=(lx+rx)>>1;
            build_x(vx<<1,lx,mid);
            build_x(vx<<1|1,mid+1,rx);
        }
        build_y(vx,1,lx,0,rx,m-1);
    }
    void update_y(ll vx,ll vy,ll lx,ll ly,ll rx,ll ry,ll x,ll y,ll val) {
        if (ly==ry) {
            if (lx==rx) seg[vx][vy]=val;
            else seg[vx][vy]=seg[vx<<1][vy]+seg[vx<<1|1][vy];
            return;
        }
        ll mid=(ly+ry)>>1;
        if (y<=mid) update_y(vx,vy<<1,lx,ly,rx,mid,x,y,val);
        else update_y(vx,vy<<1|1,lx,mid+1,rx,ry,x,y,val);
```

```
        seg[vx][vy]=seg[vx][vy<<1]+seg[vx][vy<<1|1];
    }
    void update_x(ll vx,ll lx,ll rx,ll x,ll y,ll val) {
        if (lx!=rx) {
            ll mid=(lx+rx)>>1;
            if (x<=mid) update_x(vx<<1,lx,mid,x,y,val);
            else update_x(vx<<1|1,mid+1,rx,x,y,val);
        }
        update_y(vx,1,lx,0,rx,m-1,x,y,val);
    }

    ll query_y(ll vx,ll vy,ll ly,ll ry,ll y1,ll y2) {
        if (ry<y1||y2<ly) return 0;
        if (y1<=ly&&ry<=y2) return seg[vx][vy];
        ll mid=(ly+ry)>>1;
        return query_y(vx,vy<<1,ly,mid,y1,y2)+query_y(vx,vy<<1|1,mid+1,ry,y1,y
2);
    }

    ll query_x(ll vx,ll lx,ll rx,ll x1,ll x2,ll y1,ll y2){
        if (rx<x1||x2<lx) return 0;
        if (x1<=lx&&rx<=x2) return query_y(vx,1,0,m-1,y1,y2);
        ll mid=(lx+rx)>>1;
        return query_x(vx<<1,lx,mid,x1,x2,y1,y2)+query_x(vx<<1|1,mid+1,rx,x1,
x2,y1,y2);
    }
    void update(ll x,ll y,ll val){update_x(1,0,n-1,x,y,val);}
    ll query(ll x1,ll y1,ll x2,ll y2){return query_x(1,0,n-1,x1,x2,y1,y2);}
};
```

iterSEG

```
// usage: iterSEG(n, arr); update(tar, val); ll sum = query(qs, qe);
// memo : 0-indexed
#define OP(a,b) (max((a),(b)))
#define E LLONG_MIN
```

```
struct iterSEG {
    ll n;
    vl seg;

    iterSEG(ll size, const vl& arr){
        n=1; while (n<size) n<<=1;
        seg.assign(2*n, E);
        for (ll i=0;i<size;++i) seg[n+i]=arr[i];
        for (ll i=n-1;i>0;--i)
            seg[i] = OP(seg[i<<1], seg[i<<1|1]);
    }

    void update(ll idx, ll val) {
        ll p = idx + n;
        seg[p] = val;
        for (p>>=1;p;p>>=1)
            seg[p] = OP(seg[p<<1], seg[p<<1|1]);
    }

    ll query(int l, int r) {
        ll resL=E, resR=E;
        for (l+=n,r+=n; l<=r; l>>=1,r>>=1) {
            if (l & 1) resL = OP(resL, seg[l++]);
            if (!(r & 1)) resR = OP(seg[r--], resR);
        }
        return OP(resL, resR);
    }
};
```

iterLAZY

```
// usage: iterLAZY(n, arr); update(qs, qe, val); ll sum = query(qs, qe);
// memo : 0-indexed
#define OP(a,b)             ((a)+(b))
#define E                   0LL
#define MAPPING(x,f,len)    ((x) + (f)*(len))
#define COMPOSE(f_new,f_old)  ((f_new) + (f_old))
```

```
#define lzE                    0LL

struct iterLAZY {
    ll n, sz, h;
    vl seg, lz;

    iterLAZY(ll size, vl& arr){
        n=1; while (n<size) n<<=1;
        h=0; for (ll i=n;i>1;i>>=1) ++h;
        seg.assign(2*n, E);
        lz .assign(  n, lzE);
        copy(arr.begin(),arr.end(),seg.begin()+n);
        for (ll i=n-1;i;--i) seg[i]=seg[i<<1]+seg[i<<1|1];
    }

   void apply(ll p, ll f, ll len_unused){
       seg[p] = MAPPING(seg[p], f, len_unused);
       if (p < n) lz[p] = COMPOSE(f, lz[p]);
   }

    void pull(ll p) {
       for (ll i = 1; p > 1; i <<= 1, p >>= 1) {
           ll parent = p>>1;
           ll combined = OP(seg[p], seg[p^1]);
           ll added = MAPPING(E, lz[parent], i*2);
           seg[parent] = OP(combined, added);
       }
    }

    void push(ll p) {
        for (ll s=h, len=1<<(h-1); s; --s, len>>=1) {
            ll i=p>>s;
            if (!lz[i]) continue;
            apply(i<<1,   lz[i], len);
            apply(i<<1|1, lz[i], len);
            lz[i]=lzE;
        }
    }
```

```
    void update(ll l,ll r,ll v) {
        ll s=(l+=n), e=(r+=n), len=1;
        push(s); push(e);
        for (; l<=r; l>>=1, r>>=1, len<<=1) {
            if (l&1) apply(l++, v, len);
            if (!(r&1)) apply(r--, v, len);
        }
        pull(s); pull(e);
    }

    ll query(ll l,ll r) {
        l+=n; r+=n;
        push(l); push(r);
        ll Lsum=E, Rsum=E;
        while (l<=r) {
            if (l&1) Lsum = OP(Lsum, seg[l++]);
            if (!(r&1)) Rsum = OP(seg[r--], Rsum);
            l>>=1; r>>=1;
        }
        return OP(Lsum, Rsum);
    }
};
```

GoldenMine SEG

```
// usage: gmSEG(n, arr); update(tar, val); ll sum = query(qs, qe);
// memo : 0-indexed
struct node {
    ll l,r,res,total;
};
struct gmSEG {
    ll n;
    vl arr;
    vector<node> seg;

    gmSEG(ll n, vl& arr): n(n),arr(arr),seg(n<<2){
        init(1,0,n-1);
```

```
}

node merge(node& a,node& b) {
    if (a.total==LLONG_MIN) return b;
    if (b.total==LLONG_MIN) return a;
    node res;
    res.total = a.total + b.total;
    res.l = max(a.l, a.total + b.l);
    res.r = max(b.r, b.total + a.r);
    res.res = max({a.res, b.res, a.r + b.l});
    return res;
}

void init(ll node,ll l,ll r) {
    if (l==r) {
        seg[node]={arr[l],arr[l],arr[l],arr[l]};
        return;
    }
    ll m=(l+r)>>1;
    init(node<<1,l,m);
    init(node<<1|1,m+1,r);
    seg[node]=merge(seg[node<<1],seg[node<<1|1]);
}

node range_sum(ll node, ll l, ll r, ll s, ll e) {
    if (r<s||e<l) return {0,0,LLONG_MIN,LLONG_MIN};
    if (s<=l&&r<=e) return seg[node];
    ll m=(l+r)>>1;
    struct node left = range_sum(node<<1,l,m,s,e);
    struct node right = range_sum(node<<1|1,m+1,r,s,e);
    return merge(left,right);
}

void point_update(ll node, ll l, ll r, ll tar, ll val) {
    if (r<tar||tar<l) return;
    if (l==r) {
        seg[node]={val,val,val,val};
        return;
```

```
        }
        ll m=(l+r)>>1;
        point_update(node<<1,l,m,tar,val);
        point_update(node<<1|1,m+1,r,tar,val);
        seg[node]=merge(seg[node<<1],seg[node<<1|1]);
    }
    void update(ll tar,ll val){point_update(1,0,n-1,tar,val);}
    node query(ll s,ll e){return range_sum(1,0,n-1,s,e);}
};
```

2-SAT

```
// Usage: operation(lit(x), lit(y)); bool satisfiable = TwoSAT.solve()
// memo : 1-based variable
// O(N + M)  N:#variable M:#relation
inline ll neg(ll x) { return x%2 ? x+1 : x-1; }
inline ll lit(ll x) { return x>0 ? (x<<1)-1 : -x<<1; }
struct TwoSAT {
    ll N;                    // # boolean variables
    vvl g;                   // 1-based, size = 2*N+1
    vector<bool> assignment; // final truth values: +1=true, -1=false
    vl comp;

    TwoSAT(ll vars): N(vars), g((N << 1) + 1), assignment(N + 1, 0) {}

    void addImp(ll u, ll v) {
        g[u].push_back(v);
    }
    void addOr(ll x, ll y) {
        addImp(neg(x), y);
        addImp(neg(y), x);
    }
    void addTrue(ll x) {
        addImp(neg(x), x);
    }
    bool satisfiable() {
        SCC scc(g.size(), g);    // 0-index SCC
```

```
        comp = scc.scc_idx;
        for (ll i=1; i<=N; i++)
            if (comp[(i<<1)-1] == comp[i<<1])
                return false;
        return true;
    }
    bool solve(ll x) {
        return comp[(x<<1)-1] < comp[x<<1];
    }
    void solveAll() {
        for (ll i=1; i<=N; i++)
            assignment[i] = comp[(i<<1)-1]<comp[i<<1];
    }
};
```

Geometry

Created	@2025년 5월 18일 오후 7:40
Tags	BasicOps ConvexHull Pick'sTheorem PointInPolygon PolygonCut RotatingCalipers SortingPoints

Basic Operations

```
const ld eps = 1e-9;
inline ll diff(ld lhs, ld rhs) {
    if (lhs-eps<rhs && rhs<lhs+eps) return 0;
    return lhs<rhs ? -1 : 1;
}
struct Point {
    ld x, y;
    bool operator==(const Point& o) const {
        return diff(x,o.x)==0 && diff(y, o.y)==0;
    }
    Point operator+(const Point& o) const {
        return Point{ x+o.x, y+o.y };
    }
    Point operator-(const Point& o) const {
        return Point{ x-o.x, y-o.y };
    }
    Point operator*(ld t) const {
        return Point{ x*t, y*t };
```

```
    }
    friend istream& operator>>(istream& is, Point& p) {
        return is >> p.x >> p.y;
    }
};
struct Line {
    Point pos, dir;
    Line()=default;
    Line(const Point& a, const Point& b): pos(a), dir(b - a) {}
    friend istream& operator>>(istream& is, Line& l) {
        Point p1, p2;
        is >> p1 >> p2;
        l.pos = p1;
        l.dir = Point{ p2.x-p1.x, p2.y-p1.y };
        return is;
    }
};
struct Circle {
    Point center;
    ld r;
};
inline ld inner(const Point& a, const Point& b) {
    return a.x*b.x+a.y*b.y;
}
inline ld outer(const Point& a, const Point& b) {
    return a.x*b.y-a.y*b.x;
}
// point-line positional relationship, (+) l, (-) r, (0) on line
inline ll ccw_line(const Line& line, const Point& point) {
    return diff(outer(line.dir, point-line.pos), 0);
}
// a→b→c rotation orientation, (+) CCW, (-) CW, (0) collinear
inline ll ccw(const Point& a, const Point& b, const Point& c) {
    return diff(outer(b-a, c-a), 0);
}
inline bool is_between(ld check, ld a, ld b) {
    if (a<b) return a-eps<check && check<b+eps;
    return b-eps<check && check<a+eps;
```

```
}
// Euclidean distance
inline ld dist(const Point& a, const Point& b) {
    return sqrt(inner(a-b, a-b));
}
// Euclidean distance squared
inline ld dist2(const Point &a, const Point &b) {
    return inner(a-b, a-b);
}
// point-line distance, point-segment distance
inline ld dist(const Line& line, const Point& point, bool segment = false) {
    ld c1 = inner(point-line.pos, line.dir);
    if (segment && diff(c1, 0)<=0) return dist(line.pos, point);
    ld c2 = inner(line.dir, line.dir);
    if (segment && diff(c2, c1)<=0) return dist(line.pos+line.dir, point);
    return dist(line.pos+line.dir*(c1/c2), point);
}
// line intersection test, ret: intersection point
bool line_line(const Line& a, const Line& b, Point& ret) {
    ld mdet = outer(b.dir, a.dir);
    if (diff(mdet, 0) == 0) return false;
    ld t2 = outer(a.dir, b.pos-a.pos)/mdet;
    ret = b.pos+b.dir*t2;
    return true;
}
// segment intersection test, ret: intersection point
// 0: no intersection, 1: single point, 2: multiple points
ll seg_seg(const Line& a, const Line& b, Point& ret) {
    ld mdet = outer(b.dir, a.dir);
    if (diff(mdet, 0) == 0) {
        if (ccw_line(a,b.pos)!=0) return 0;
        ld t0 = inner(b.pos-a.pos, a.dir)/inner(a.dir,a.dir);
        ld t1 = inner(b.pos+b.dir-a.pos,a.dir) / inner(a.dir,a.dir);
        if (t0>t1) swap(t0,t1);
        ld l=max((ld)0,t0),h=min((ld)1,t1);
        if (diff(h,l)<0) return 0;
        if (diff(h,l)>0) return 2;
        ret = a.pos+a.dir*l;
```

```
        return 1;
    }
    ld t1 = -outer(b.pos-a.pos, b.dir)/mdet;
    ld t2 = outer(a.dir, b.pos-a.pos)/mdet;
    if (!is_between(t1, 0, 1) || !is_between(t2, 0, 1)) return 0;
    ret = b.pos+b.dir*t2;
    return 1;
}
// line-point intersection test
inline bool line_point(const Line& line,const Point& point) {
    return ccw_line(line,point)==0;
}
// seg-point intersection test
inline bool seg_point(const Line& seg, const Point& point) {
    if (ccw_line(seg,point) != 0) return false;
    ld t = inner(point - seg.pos, seg.dir) / inner(seg.dir,seg.dir);
    return is_between(t,0,1);
}
// line-seg intersection test, ret: intersection point
// 0: no intersection, 1: single point, 2: multiple points
ll line_seg(const Line& line, const Line& seg, Point& ret) {
    ld mdet = outer(line.dir,seg.dir);
    if (diff(mdet,0)==0) {
        if (ccw_line(line,seg.pos)!=0) return 0;
        return 2;
    }
    ld t1 = -outer(line.pos-seg.pos,line.dir) / mdet;
    if (!is_between(t1,0,1)) return 0;
    ret = seg.pos + seg.dir * t1;
    return 1;
}
// circle-line intersection
vector<Point> circle_line(const Circle& circle, const Line& line) {
    vector<Point> result;
    ld a = 2*inner(line.dir, line.dir);
    ld b = 2*(line.dir.x*
        (line.pos.x-circle.center.x+line.dir.y*(line.pos.y-circle.center.y)));
    ld c = inner(line.pos-circle.center, line.pos-circle.center)-circle.r*circle.r;
```

```
    ld det = b*b-2*a*c;
    ll pred = diff(det, 0);
    if (pred==0)
        result.push_back(line.pos+line.dir*(-b/a));
    else if (pred>0) {
        det = sqrt(det);
        result.push_back(line.pos+line.dir*((-b+det)/a));
        result.push_back(line.pos+line.dir*((-b-det)/a));
    }
    return result;
}
// circle-circle intersection
vector<Point> circle_circle(const Circle& a, const Circle& b) {
    vector<Point> result;
    ll pred = diff(dist(a.center, b.center), a.r+b.r);
    if (pred>0) return result;
    if (pred==0) {
        result.push_back((a.center*b.r+b.center*a.r)*(1/(a.r+b.r)));
        return result;
    }
    ld aa = a.center.x*a.center.x+a.center.y*a.center.y-a.r*a.r;
    ld bb = b.center.x*b.center.x+b.center.y*b.center.y-b.r*b.r;
    ld tmp = (bb-aa)/2.0;
    Point cdiff = b.center-a.center;
    if (diff(cdiff.x, 0)==0) {
        if (diff(cdiff.y, 0)==0)
        return result; // if (diff(a.r, b.r) == 0): same circle
        return circle_line(a, Line{Point{0, tmp/cdiff.y}, Point{1, 0}});
    }
    return circle_line(a,Line{Point{tmp/cdiff.x, 0}, Point{-cdiff.y, cdiff.x}});
}
// circumcircle with three points
Circle circle_from_3pts(const Point& a, const Point& b, const Point& c) {
    Point ba = b-a, cb = c-b;
    Line p{(a+b)*0.5, Point{ba.y, -ba.x}};
    Line q{(b+c)*0.5, Point{cb.y, -cb.x}};
    Circle circle;
    if (!line_line(p, q, circle.center)) circle.r = -1;
```

```
    else  circle.r = dist(circle.center, a);
    return circle;
}
// circumcircle with two points and radius
Circle circle_from_2pts_rad(const Point& a, const Point& b, ld r) {
    ld det = r*r/dist2(a, b)-0.25;
    Circle circle;
    if (det<0)
        circle.r = -1;
    else {
        ld h = sqrt(det);
        // center is to the left of a->b
        circle.center = (a+b)*0.5+Point{a.y-b.y, b.x-a.x}*h;
        circle.r = r;
    }
    return circle;
}
Point inner_center(const Point &a, const Point &b, const Point &c) {
    ld wa = dist(b, c), wb = dist(c, a), wc = dist(a, b);
    ld w = wa+wb+wc;
    return { (wa*a.x+wb*b.x+wc*c.x)/w, (wa*a.y+wb*b.y+wc*c.y)/w };
}
Point outer_center(const Point &a, const Point &b, const Point &c) {
    Point d1 = b-a, d2 = c-a;
    ld area = outer(d1, d2);
    ld dx = d1.x*d1.x*d2.y-d2.x*d2.x*d1.y+d1.y*d2.y*(d1.y-d2.y);
    ld dy = d1.y*d1.y*d2.x-d2.y*d2.y*d1.x+d1.x*d2.x*(d1.x-d2.y);
    return { a.x+dx/area/2.0, a.y-dy/area/2.0 };
}
```

Sorting Points

주어진 점들을 반시계방향으로 정렬

```
struct Point {
    ll x, y;
    ll dist2() const {
            return x*x + y*y;
```

```
    }
    ll quadrant() const {
        if (x>=0 && y>=0) return 1;
        if (x<=0 && y>=0) return 2;
        if (x<=0 && y<=0) return 3;
        if (x>=0 && y<=0) return 4;
    }
    bool operator<(const Point& o) const {
        if (quadrant() != o.quadrant()) return quadrant() < o.quadrant();
        ll ccw = x*o.y - y*o.x;
        if (!ccw) return dist2()<o.dist2();
        return 0<ccw;
    }
};
```

Convex Hull & Rotating Calipers (Monotone Chain)

주어진 점들의 외곽을 감싸는 가장 작은 볼록 다각형 (점들에 고무줄 씌운 모양)

```
// Usage: Find the smallest convex polygon that encloses all points
// O(NlogN)
void getConvexHull(vector<Point>& pt, vector<Point>& convex_hull) {
    sort(pt.begin(), pt.end(), [](const Point& a, const Point& b) {
        return a.x==b.x ? a.y<b.y : a.x<b.x;
    });
    vector<Point> up, lo;
    for (const auto& p : pt) {
        while (up.size() >= 2 && ccw(*++up.rbegin(), *up.rbegin(), p) >= 0) u
p.pop_back();
        while (lo.size() >= 2 && ccw(*++lo.rbegin(), *lo.rbegin(), p) <= 0) lo.po
p_back();
        up.push_back(p);
        lo.push_back(p);
    }
    // rotating calipers
    // Usage: Get all antipodal pairs
    // O(N)
    for (ll i=0, j=(ll)lo.size()-1; i+1<up.size() || j > 0;) {
```

```
        get_pair(up[i], lo[j]); // DO WHAT YOU WANT
        if (i + 1 == up.size()) --j;
        else if (j == 0) ++i;
        else if ((up[i+1].y - up[i].y) * (lo[j].x - lo[j-1].x) >
        (up[i+1].x - up[i].x) * (lo[j].y - lo[j-1].y))++i;
        else --j;
    }
    up.insert(up.end(), ++lo.rbegin(), --lo.rend());
    swap(up, convex_hull);
}
```

Point in Polygon Test

점이 다각형 안에 있는 지 판별

```
// Usage: Is pointer in polygon?
// O(N)
bool is_in_polygon(Point p, vector<Point>& poly) {
    ll wn = 0;
    for (ll i=0; i<poly.size(); ++i) {
        ll ni = i+1==poly.size() ? 0 : i+1;
        if (poly[i].y<=p.y) {
            if (poly[ni].y>p.y && ccw(poly[i], poly[ni], p)>0) ++wn;
        }
        else if (poly[ni].y<=p.y && ccw(poly[i], poly[ni], p)<0) --wn;
    }
    return wn != 0;
}
```

Polygon Cut

직선의 왼쪽 기준으로 다각형 자르기

```
// Usage: Get polygon vector left of line
// O(N)
void cut_polygon(Line line, const vector<Point>& polygon, vector<Point>&
cut) {
    if (!polygon.size()) {
```

```
        cut = polygon;
        return;
    }
    using piter = vector<Point>::const_iterator;
    piter la, lan, fi, fip, i, j;
    la = lan = fi = fip = polygon.end();
    i = polygon.end()-1;
    bool lastin = diff(ccw_line(line, polygon[polygon.size()-1]), 0) > 0;
    for (j=polygon.begin(); j!=polygon.end(); j++) {
        bool thisin = diff(ccw_line(line, *j), 0) > 0;
        if (lastin && !thisin)
            la = i, lan = j;
        if (!lastin && thisin)
            fi = j, fip = i;
        i = j;
        lastin = thisin;
    }
    if (fi==polygon.end()) {
        cut = lastin ? polygon : vector<Point>();
        return;
    }
    for (i=fi; i!=lan; i++) {
        if (i == polygon.end()) {
            i = polygon.begin();
            if (i == lan) break;
        }
        cut.push_back(*i);
    }
    Point lc, fc;
    get_cross(Line{ *la, *lan-*la }, line, lc);
    get_cross(Line{ *fip, *fi-*fip }, line, fc);
    cut.push_back(lc);
    if (diff(dist2(lc, fc), 0) != 0) cut.push_back(fc);
}
```

Pick’s Theorem

격자점으로 구성된 simple polygon에 대해 i 는 polygon 내부의 격자수, b 는 polygon 선분 위 격자수, A 는 polygon 넓이라고 할 때, $A = i + b/2 - 1$

String

Created	@2025년 5월 19일 오후 10:02
Tags	Aho-Corasick KMP LCP Z

KMP
Z
LCP with Suffix Array
아호코라식
Trie

KMP

π[i] = j := str[0 ~ j] == str[i-j ~ i]

i.e. **i 번째 문자까지 봤을 때, "가장 긴 접두사이자 접미사"가 패턴 앞의 (j+1) 글자만큼 일치한다.**

→ 패턴과 문자열을 비교하는 중, 불일치가 발생하였을 때, **j = pi[j]** 로 돌아가면서 불필요한 비교연산 줄임

** j = π[j]인 이유? :

새로운 s[i]가 오기 전까지는 **[0 1 … j]** *[…]* **[(i-1-j) (i-j) … (i-1)]** 꼴이었음. (굵은 글씨 동일한 문자열)

s[i] ≠ s[j+1] 이기 때문에 기존의 j 길이 만큼의 접미사는 매칭 실패.

따라서 j 까지 보았을 때의 최대 매칭 길이인 π[j]로 j를 대체하여

[0 1 … j] *[…]* **[(i-1-j) (i-j) … (i-1)]** [i]

→ **[0 1 … π[j] … j]** *[…]* **[(i-1-j) (i-j) … (i-1-π[j]) … (i-1)]** [i]

→ **[0 1 … π[j]]** ***[(π[j]+1) … j]*** *[…]* ***[(i-1-j) … (i-2-π[j])]*** **[(i-1-π[j]) … (i-1)]** [i]

→ **[0 1 … π[j]]** *[…]* **[(i-1-π[j]) … (i-1)]** [i]

→ **[0 1 … j]** *[…]* **[(i-1-j) … (i-1)]** [i]

이 과정을 s[i] ≠ s[j+1] 인 동안 반복

만약 매칭을 유지하면서 s[i] == s[j+1] 에 도달하는 것이 성공했다면, 매칭 길이에 1 추가하여 저장.

여전히 매칭을 실패했다면 -1 (매칭 실패) 로 저장

pi를 구하는 것과 동일하게, KMP 에서도 비교 중 실패가 발생하면, 패턴의 접두사와 (현재까지 본) 문자열의 접미사+새롭게 추가된 문자 가 같아질 때까지 j=π[j] 과정 반복

BOJ 1786 - 찾기

BOJ 4354 - 문자열 제곱

BOJ 1305 - 광고

```
// Usage: vl pos = kmp(str, pattern);
// O(|s|+|p|)
// Note: return matched position (0-based)
// pi[i]=j := str[0...j]==str[(i-j)...i]

void calculate_pi(vl &pi, string &s) {
    pi[0]=-1;
    for (ll i=1,j=-1;i<s.size();i++) {
        while (j>=0 && s[i]!=s[j+1]) j=pi[j];
        if (s[i]==s[j+1]) pi[i]=++j;
        else pi[i]=-1;
    }
}

vl kmp(string& str, string& pattern) {
    vl pi(pattern.size()), ans;
    if (pattern.empty()) return ans;
    calculate_pi(pi, pattern);
    for (ll i=0,j=-1;i<str.size();i++) {
        while (j>=0 && str[i]!=pattern[j+1]) j=pi[j];
        if (str[i]==pattern[j+1]) {
            j++;
            if (j+1==pattern.size()) ans.push_back(i-j), j=pi[j];
        }
    }
    return ans;
}
```

Z

Z[i] = max{ j | 0≤j≤|str|−i && str[0 … (j−1)] = str[i … (i+j−1)]}

i.e. **i 에서 시작하는 부분문자열과 원래 문자열의 공통 접두사의 최대 길이**

[0 1 … (r-l-1)] […] **[l … (r-1)]** […]

현재 위와 같은 상태일 때, [l, r) 구간을 Z-box라 부르며, Z-box는 prefix와 일치한다.

i가 Z-box 바깥에 있으면, 처음부터 비교를 시작한다. Z-box 안에 있으면 불필요한 비교를 줄일 수 있다.

[0 1 … (r-l-1)] *[…]* **[l … i … (r-1)]** *[…]*

→ **[0 … (i-l) … (r-l-1)]** *[…]* **[l … i … (r-1)]** *[…]*

→ **[0 … i′ … (r′-1)]** *[…]* **[l … i … (r-1)]** *[…]*

→ **[0 … (Z[i′]-1)] […] [i′ … (i′+Z[i′]-1)] [… (r′-1)]** *[…]* **[l … (l+Z[i′]-1)] […] [i … (i+Z[i′]-1)]**

[… (r-1)] *[…]*

→ **[0 … (Z[i′]-1)]** *[…]* **[i … (i+Z[i′]-1)]** *[…]*

Z[i′]가 r-i 보다 크다면, 일단 이미 알려진 최대 매칭 길이(r-i) 만큼 부여하고, 그 이후에 대해서는 직접 비교.

Z[i′]가 기존 Z-box 안에 위치한다면 값 그대로 사용.

BOJ 13506 - 카멜레온 부분 문자열

```
// Z[i] : maximum common prefix length of &s[0] and &s[i]
// O(|s|)
void get_z(string& s, vl& Z) {
    ll n = s.size();
    Z.resize(n);
    Z[0] = n;
    for (ll i=1,l=0,r=0; i<n; i++) {
        if (i<r) Z[i] = min(r-i, Z[i-l]);
        while (i+Z[i]<n && s[Z[i]]==s[i+Z[i]]) Z[i]++;
        if (i+Z[i]>r) l=i, r=i+Z[i];
    }
}
```

LCP with Suffix Array

SA[i] = j := i번째로 작은 접미사의 시작 인덱스가 j

i.e. 모든 접미사를 정렬했을 때, i번째 접미사는 문자열 j번째 에서 시작하는 접미사이다.

rank[i]=j := SA[j]=i

i.e. 문자열 i에서 시작하는 접미사는 j번째 접미사이다.

LCP[i]=j := SA[i] 접미사와 SA[i-1] 접미사가 앞에서부터 j개 만큼 동일하다.

i.e. 모든 접미사를 정렬했을 때, i번째 접미사와 그 이전(i-1번째) 접미사의 최장 공통 접두사 길이가 j이다.

** LCP 구하는 게 O(N)인 이유?

if(cnt) cnt--; 코드 한 줄 덕분에 선형 시간 유지 가능.

예를 들어 `banana` 에서 `anana`와 우선순위 전단계 접미사 `ana`의 비교를 통해 cnt==3을 얻어냈다고 하자.

순서 상, `anana`의 다음 접미사인 `nana`와 그 우선순위 전단계 접미사를 비교해야 하는데,

우리는 이미 `ana`에서 앞 글자 하나를 뺀 `na`가 접미사에 포함되어 있음을 알고 있음.

따라서 처음부터 비교할 필요 없이 cnt==2 부터 비교하면 됨.

(`nana`의 우선순위 전 단계 접미사가 `na`임을 의미하지는 않음)

BOJ 1701 - Cubeditor

```
// O(Nlog^2N)
void getSuffixArray(string& str, vl& SA, vl& rank) {
    ll n = str.size();
    SA.resize(n), rank.resize(n);
    for (ll i=0; i<n; i++)
        SA[i] = i, rank[i] = str[i];
    for (ll L=1; L<n; L<<=1) {
        auto cmp = [&](ll a, ll b) {
            if (rank[a] != rank[b])     // rank는 L만큼의 비교 정보 담고 있음
                return rank[a] < rank[b];
            ll ra = a+L<n ? rank[a+L] : -1;
            ll rb = b+L<n ? rank[b+L] : -1;
            return ra < rb;
        };
        sort(SA.begin(), SA.end(), cmp);
        vl tmp(n, 0);   // tmp[SA[0]] = 0;
```

```
        for (ll i=1; i<n; i++)
            tmp[SA[i]] = tmp[SA[i-1]] + (cmp(SA[i-1], SA[i]) ? 1 : 0);
        rank = tmp;
        if (rank[SA[n-1]] == n-1) break;
    }
}
// O(n)
void getLCP (string& str, vl& LCP) {
    ll n = str.size(); LCP.resize(n);
    vl SA, rank; getSuffixArray(str, SA, rank);
    for (ll cnt=0, i=0; i<n; i++) {
        if (!rank[i]) { cnt = 0; continue; }
        ll j = SA[rank[i] - 1];
        while (i+cnt<n && j+cnt<n && str[i+cnt]==str[j+cnt]) cnt++;
        LCP[rank[i]] = cnt;
        if (cnt) cnt--;
    }
}
```

아호코라식

```
// Usage: aho_corasick ac; ac.init(patterns); bool matched = ac.query(text);
// init: O(sum of |p|)
// query: O(|s|)

struct aho_corasick {
    static constexpr ll MAXN=100005, MAXC=26;
    vector<array<ll, MAXC>> trie;
    vl fail,term;
    ll piv=0;
    ll offset='A'; // 소문자라면 'a'로 바꿔야 함
    void init(vector<string> &v) {
        trie.assign(MAXN, array<ll,MAXC>{});
        fail.assign(MAXN, 0);
        term.assign(MAXN, 0);
        piv=0;
        for (auto &i: v) {
```

```
        ll p=0;
        for (auto &j: i) {
            ll k=j-offset;
            if (!trie[p][k]) trie[p][k]=++piv;
            p=trie[p][k];
        }
        term[p]=1;
    }
    queue<ll> que;
    for (ll i=0;i<MAXC;i++) if (trie[0][i]) que.push(trie[0][i]);
    while (!que.empty()) {
        ll x=que.front(); que.pop();
        for (ll i=0;i<MAXC;i++) if (trie[x][i]) {
            ll p=fail[x];
            while (p && !trie[p][i]) p=fail[p];
            p=trie[p][i];
            fail[trie[x][i]]=p;
            if (term[p]) term[trie[x][i]]=1;
            que.push(trie[x][i]);
        }
    }
}
bool query(string &s) {
    ll p=0;
    for (auto &i: s) {
        ll k=i-offset;
        while (p && !trie[p][k]) p=fail[p];
        p=trie[p][k]; if (term[p]) return 1;
    }
    return 0;
}
// 문자열 s에서 모든 매칭된 (패턴ID, 시작위치)를 반환
vector<pll> query_pos(const string& s, const vector<string>& patterns) {
    vector<pll> matches;
    ll p = 0;
    for (ll i = 0; i < (ll)s.size(); i++) {
        ll k = s[i] - offset;
        while (p && !trie[p][k]) p = fail[p];
```

```
            p = trie[p][k];
            // 이 노드에 매칭된 패턴들이 있으면
            for (ll id : out[p]) {
                ll start_idx = i - (ll)patterns[id].size() + 1;
                matches.emplace_back(id, start_idx);
            }
        }
        return matches;
    }
};
```

Trie

```
struct Trie {
    struct Node {
        map<char,ll> nxt;
        bool isEnd = false;
    };

    vector<Node> tree;

    Trie() { tree.emplace_back(); }

    void insert(const string& word) {
        ll cur = 0; // 루트
        for (auto& c : word) {
            if (!tree[cur].nxt.contains(c)) {
                tree[cur].nxt[c] = tree.size();
                tree.emplace_back();
            }
            cur = tree[cur].nxt[c];
        }
        tree[cur].isEnd = true;
    }

    bool search(const string& word) {
        ll cur = 0;
```

```
        for (auto& c : word) {
            if (!tree[cur].nxt.contains(c)) return false;
            cur = tree[cur].nxt[c];
        }
        return tree[cur].isEnd;
    }

    // 해당 prefix로 시작하는 단어 존재 여부
    bool startsWith(const string& prefix) {
        ll cur = 0;
        for (auto& c : prefix) {
            if (!tree[cur].nxt.contains(c)) return false;
            cur = tree[cur].nxt[c];
        }
        return true;
    }
};
```

Others

Created	@2025년 7월 1일 오후 9:56
Tags	

좌표 압축

```
vl coord(N);
for (auto& e : coord) cin >> e;
vl comp = coord;
sort(comp.begin(), comp.end());
comp.erase(unique(comp.begin(), comp.end()), comp.end());
for (auto& e : coord)
    e = lower_bound(comp.begin(), comp.end(), e) - comp.begin();
```