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};
}
```