Contents

1 Setting

1.1 READ THE PROBLEM!!

READ THE PROBLEM CAREFULLY!!!

REMEMBER THE 2018 L.E.D?

1.2 Compile Command

```
Tools -> Build System -> New Build System...
"shell_cmd": "g++ -std=c++17 \"${file}\" -o
\"${file_path}/${file_base_name}\"",
```

1.3 Easy Printing

```
template <class T1, class T2>ostream &operator<<
  (ostream &os, pair<T1,T2> const &x){
  os<<'('<<x.first<<", "<<x.second<<')'; return os;
}
template <class Ch, class Tr, class Container>
  basic_ostream<Ch,Tr> &operator<<
  (basic_ostream<Ch,Tr> &os, Container const &x){
  for(auto &y: x) os<<y<<" "; return os;
}
```

1.4 RBTree

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

int main(){
  ordered_set X;
  for (int i=1; i<10; i+=2) X.insert(i); // 1 3 5 7 9
  cout << *X.find_by_order(2) << endl; // 5
  cout << X.order_of_key(6) << endl; // 3
  cout << X.order_of_key(7) << endl; // 3
  X.erase(3);
}
```

2 Math (Theory)

2.1 Triangles

TODO (kactl)

2.2 Series And Calculus

TODO (kactl), Green

2.3 Theorems

TODO Kirchhoff, Generalized Euler, Konig, Hall, Pick, Burnside

2.4 Combinatorics

TODO Twelvefold (kactl), Stirling, Catalan, Derangement

2.5 Nim Game

TODO Nim games, Grundy number

2.6 Composite and Prime

$n \leq 1,000,000$ 약수 최대 240개 (720,720)
$n \leq 1,000,000,000$ 최대 1,344개 (735,134,400)
up to 10,000: 소수 1,229개 (9,973)
up to 100,000: 소수 9,592개 (99,991)
up to 1,000,000: 소수 78,498개 (999,983)
up to 1,000,000,000: 소수 50,847,534개 (999,999,937)
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
$998244353 = 119 \times 2^{23} + 1$, primitive 3
$985661441 = 235 \times 2^{22} + 1$, primitive 3
$1012924417 = 483 \times 2^{21} + 1$, primitive 5

2.7 Pythagorean Triples

The Pythagorean triples are uniquely generated by $a = k(m^2 - n^2)$, $b = k(2mn)$, $c = k(m^2 + n^2)$ where $m > n > 0, k > 0, gcd(m, n) = 1$, and either m or n is even.

3 Math (Algo)

3.1 Arithmetic stuff

```cpp
//ceil(a/b)
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;
}

//floor(a/b)
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);
}

ll __gcd(ll a, ll b);

//egcd (ac + bd = gcd(a, b))
pair<ll, ll> egcd(ll a, ll b){
  if(b == 0) return {1, 0};
  auto t = egcd(b, a%b);
  return {t.second, t.first-t.second*(a/b)};
}

//modular inverse (ax = gcd(a, m) mod m)
ll modinv(ll a, ll m){return (egcd(a,m).first%m + m)%m;}

//modular power
ll pow(ll a, ll x, ll mod){
  a%= mod; ll r;
  for(r=1; x; x/=2)
    {if(x%2) r=modmul(r,a,m); a=modmul(a,a,m);}
  return r;
}

//modular large multiplication
ll modmul(ll a, ll b, ll m) {
  return ll((__int128)a*(__int128)b%m);
}

int __builtin_clz(int x); // number of leading zero
int __builtin_ctz(int x); // number of trailing zero
int __builtin_popcount(int x); // number of 1-bits in x
// use clzll, ctzll, popcountll for long long x

last_bit == n&-n;
floor(log2(n)) == 31 - __builtin_clz(n|1);
floor(log2(n)) == 63 - __builtin_clzll(n|1);

// 00111 01011 01101 01110 10011 ...
ll next_perm(ll v){
    ll t = v|(v-1);
    int c = __builtin_ctz(v);
    return (t+1) | (((~t & -~t)-1) >> (c+1));
}
```

3.2 Primality

```cpp
typedef unsigned long long ull;
bool witness(ull a, ull n, ull s){
  if(a >= n) a%= n;
  if(a <= 1) return true;
  ull d = n>>s, x = pow(a, d, n);
  if(x == 1 || x == n-1) return true;
  while(s-- > 1){
    x = modmul(x, x, n);
    if(x == 1) return false;
    if(x == n-1) return true;
  }
  return false;
}

bool millerRabin(ull n){
  if(n == 2) return true;
  if(n < 2 || n%2 == 0) return false;
  ull d = n>>1, s = 1;
  for(; (d&1)==0; s++) d>>=1;
#define T(a) witness(a##ull, n, s)
  if(n < 4759123141ull) return T(2) && T(7) && T(61);
  return T(2) && T(325) && T(9375) && T(28178)
    && T(450775) && T(9780504) && T(1795265022);
}

ll rho(ll n){
  random_device rd; mt19937 gen(rd());
  uniform_int_distribution<ll> dis(1, n-1);
  ll x = dis(gen), c = dis(gen), g = 1; ll y = x;
  while(g == 1){
    x = (modmul(x, x, n) + c) % n;
    y = (modmul(y, y, n) + c) % n;
    y = (modmul(y, y, n) + c) % n;
    g = __gcd(abs(x-y), n);
  }
  return g;
}

void factorize(ll n, vector<ll>& fl){
  if(n == 1) return;
  if(n%2 == 0) fl.push_back(2), factorize(n/2, fl);
  else if(millerRabin(n)) fl.push_back(n);
  else{
    ll f = rho(n);
    factorize(f, fl); factorize(n/f, fl);
  }
}
```

3.3 CRT

```
// a = remainders; n = coprime divisors
ll CRT(vector<ll> &a, vector<ll> &n, int pt = 0){
    if(pt == a.size()-1) return a[pt];
    ll a0 = a[pt], a1 = a[pt+1], n0 = n[pt], n1 = n[pt+1];
    ll tmp = modinv(n0, n1);
    ll tmp2 = (tmp*(a1-a0) % n1 + n1) % n1;
    ll ora = a1, tgcd = __gcd(n0, n1);
    a[pt+1] = a0 + n0/tgcd*tmp2;
    n[pt+1]*= n0/tgcd;
    ll ret = CRT(a, n, pt+1);
    n[pt+1]/= n0/tgcd;
    a[pt+1] = ora;
    return ret;
}
```

TODO when not coprime

3.4 Binomial Number

```
ll nck(ll n, ll k, ll m){
  if(n < k) return 0;
  ll res = 1;
  for(int i=0; i<k; i++){
    res = res*(n-i) % m;
    res = res*pow(i+1, m-2, m) % m;
  }
  return res;
}

ll lucas(ll n, ll k, ll m){
  ll res = 1;
  while(n || k){
    res = res*nck(n%m, k%m, m) % m;
    n/= m, k/= m;
  }
  return res;
}

// or... precalculation and O(1)
// define SZ yourself
ll pow2[SZ+1], fact[SZ+1], ifact[SZ+1];
ll ncr(int n, int k, ll MOD){
  ll a = fact[n]*ifact[k] % MOD;
  return a*ifact[n-k] % MOD;
}
int main(){
  pow2[0] = 1, fact[0] = 1;
  for(int i=1; i<=SZ; i++){
    pow2[i] = pow2[i-1]*2 % MOD;
    fact[i] = fact[i-1]*i % MOD;
  }
  ifact[SZ] = pow(fact[SZ], MOD-2, MOD);
  for(int i=SZ-1; i>=0; i--)
    ifact[i] = ifact[i+1]*(i+1)%MOD;
}
```

3.5 Matrix

```
template <typename T>
struct Matrix{
  int n; T mod; vector<vector<T>> mat;
  Matrix(int n) :n{n}, mat(n, vector<T>(n)) {}
  Matrix operator*(Matrix& B){
    Matrix ans = Matrix(n);
    for(int i=0;i<n;i++) for(int j=0;j<n;j++)
      for(int b=0;b<n;b++)
        ans.mat[i][j]+= mat[i][b]*B.mat[b][j];
    return ans;
  }
  Matrix operator%(T mod){
    Matrix ans = Matrix(n);
    for(int i=0; i<n; i++) for(int j=0; j<n; j++)
      ans.mat[i][j] = mat[i][j]%mod;
    return ans;
  }
};

template <class T>
Matrix<T> matpow(Matrix<T> A, ll x, T mod){
  A = A%mod; Matrix<T> r(A.n);
  for(int i=0; i<A.n; i++) r.mat[i][i] = 1;
  for(; x; x/=2){if(x%2) r = r*A%mod; A = A*A%mod;}
  return r;
}
```

3.6 FFT

```
// f=0 forward; f=1 backward
// 2^17 = 131072; 2^20 = 1048576
int SZ = 19, N = 1<<SZ;
int Rev(int x){
  int r = 0;
  for(int i=0; i<SZ; i++) r = r<<1 | (x&1), x>>= 1;
  return r;
}

const double PI = acos(-1);
typedef complex<double> CD;
typedef vector<CD> VCD;
void FFT(VCD& A, int f){
  CD x, y, z;
  for(int i=0; i<N; i++){
    int j = Rev(i);
    if(i < j) swap(A[i], A[j]);
  }
  for(int i=1; i<N; i<<= 1){
    double w = PI / i;
    if(f) w = -w;
    CD x(cos(w), sin(w));
    for(int j=0; j<N; j+= i<<1){
      CD y(1);
      for(int k=0; k<i; k++){
        CD z = A[i+j+k] * y;
        A[i+j+k] = A[j+k] - z, A[j+k]+= z;
        y*= x;
      }
    }
  }
  if(f) for(int i=0; i<N; i++) A[i]/= N;
}
```

3.7 NTT

TODO

3.8 Gaussian Elimination

```
// INPUT:   a[][] = an n*n matrix
//          b[][] = an n*m matrix
// OUTPUT:   X  = an n*m matrix (stored in b[][])
//        A^{-1} = an n*n matrix (stored in a[][])
const double EPS = 1e-10;
typedef vector<vector<double>> VVD;
bool gauss_jordan(VVD& a, VVD& b) {
  const int n = a.size(), m = b[0].size();
  vector<int> irow(n), icol(n), ipiv(n);
  for (int i=0; i<n; i++){
    int pj = -1, pk = -1;
    for(int j=0; j<n; j++) if(!ipiv[j])
      for(int k=0; k<n; k++) if(!ipiv[k])
        if(pj==-1 || fabs(a[j][k]) > fabs(a[pj][pk]))
          pj=j, pk=k;
    if(fabs(a[pj][pk]) < EPS) return false;
    ipiv[pk]++;
    swap(a[pj], a[pk]); swap(b[pj], b[pk]);
    irow[i] = pj; icol[i] = pk;

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

3.9 Black Box Linear Algebra

TODO

4 String

4.1 String Tokenizer

```
vector<string> split(const string &s, char dm){
  // Returns a vector of strings tokenized by dm.
  stringstream ss(s);
  string item; vector<string> tokens;
  while(getline(ss,item,dm)) tokens.push_back(item);
  return tokens;
}
```

4.2 KMP

```
vector<int> fail(string& w){
  vector<int> T(w.size());
  for(size_t i=1, j=0; i < w.size(); i++){
    while(j && w[i] != w[j]) j = T[j-1];
    if(w[i] == w[j]) T[i] = ++j;
  }
  return T;
}

int kmp(string& s, string& w, vector<int>& T){
  int m = 0, i = 0;
  while(m+i < (int)s.size()){
    if(w[i] == s[m+i]){if(i++==(int)w.size()-1) return m;}
    else if(i) m = m+i-T[i], i = T[i];
    else m++, i = 0;
  }
  return -1;
}
```

4.3 Manacher

```
/* Insert space if you need; then
the actual length centered at i is
(M[i]+1)/2*2 if i%2 == 1; M[i]/2*2 + 1 otherwise */
vector<int> manacher(string& s){
  int n = s.size(), R = -1, p = -1;
  vector<int> A(n);
  for(int i=0; i<n; i++){
    if(i <= R) A[i] = min(A[2*p-i], R-i);
    while(i-A[i]-1 >= 0 && i+A[i]+1 < n &&
      s[i-A[i]-1] == s[i+A[i]+1]) A[i]++;
    if(i+A[i] > R) R = i+A[i], p = i;
  }
  return A;
}
```

4.4 Z

```
vector<int> Z(string& s){
  int n = s.size(), L=0, R=0;
  vector<int> z(n); z[0] = n;
  for(int i=1; i<n; i++){
    if(i > R){
      L = R = i;
      while(R < n && s[R-L] == s[R]) R++;
      z[i] = R-L; R--;
      continue;
    }
    int k = i - L;
    if(z[k] < R-i+1) z[i] = z[k];
    else{
      L = i;
      while(R < n && s[R-L] == s[R]) R++;
      z[i] = R-L; R--;
    }
  }
  return z;
}
```

4.5 Aho-Corasick

```
const int ALPHA = 26;
struct Node{
  bool end = false, root;
  Node *fail = nullptr, *out = nullptr;
  Node *nxt[ALPHA];
  Node(bool root = false): root{root}
    {fill(nxt, nxt+ALPHA, nullptr);}
};

struct Trie{
  int tonum(char c){return c - 'a';}
  Node* root = new Node(true);
  bool labeled = false;
  void add(string& s){
    Node* n = root;
    for(char& c: s){
      int x = tonum(c);
      if(!n->nxt[x]) n->nxt[x] = new Node();
      n = n->nxt[x];
    }
    n->end = true;
  }
  void label(){
    queue<Node*> Q; Q.push(root);
    while(!Q.empty()){
      Node *p = Q.front(); Q.pop();
      for(int i=0; i<ALPHA; i++) if(p->nxt[i]){
        Node *pp = p, *q = p->nxt[i];
        while(1){
          if(pp->root){q->fail = pp; break;}
          if(pp->fail->nxt[i])
            {q->fail = pp->fail->nxt[i]; break;}
          pp = pp->fail;
        }
        Q.push(q);
      }
      if(p->root) continue;
      else if(p->end) p->out = p;
      else if(p->fail->out) p->out = p->fail->out;
    }
  }
  int aho(string& s){
    if(!labeled) labeled = true, label();
    Node *p = root; size_t i = 0;
    while(i <= s.size()){
      if(p->out) return 1; // change this as you need
      if(i == s.size()) break;
      int c = tonum(s[i]);
      if(p->nxt[c]) p = p->nxt[c], i++;
      else if(p->root) i++;
      else p = p->fail;
    }
    return 0; // change this as you need
  }
};
```

4.6 Suffix Array, LCP

```
// change string& to something else if you need
vector<int> suffixarray(string& in) {
  int n = (int)in.size(), c = 0;
  vector<int> temp(n), p2b(n), bckt(n), bpos(n), out(n);
  for(int i=0; i<n; i++) out[i] = i;
  sort(out.begin(), out.end(),
       [&](int a, int b){return in[a] < in[b];});
  for(int i=0; i<n; i++) {
    bckt[i] = c;
    if (i + 1 == n || in[out[i]] != in[out[i+1]]) c++;
  }
  for (int h = 1; h < n && c < n; h <<= 1) {
    for (int i=0; i<n; i++) p2b[out[i]] = bckt[i];
    for (int i=n-1; i>=0; i--) bpos[bckt[i]] = i;
    for (int i=0; i<n; i++) if (out[i] >= n-h)
      temp[bpos[bckt[i]]++] = out[i];
    for (int i=0; i<n; i++) if (out[i] >= h)
      temp[bpos[p2b[out[i]-h]]++] = out[i] - h;
    c = 0;
    for (int i = 0; i + 1 < n; i++) {
      int a = (bckt[i] != bckt[i+1]) || (temp[i] >= n-h)
          || (p2b[temp[i+1]+h] != p2b[temp[i]+h]);
      bckt[i] = c; c += a;
    }
    bckt[n-1] = c++;
    temp.swap(out);
  }
  return out;
}

// change string& to something else if you need
vector<int> lcparray(string& in, vector<int>& sa) {
  int n = (int)in.size();
  if (n == 0) return vector<int>();
  vector<int> rank(n), height(n - 1);
  for(int i=0; i<n; i++) rank[sa[i]] = i;
  for(int i=0, h=0; i<n; i++){
    if(rank[i] == 0) continue;
    int j = sa[rank[i]-1];
    while (i+h < n && j+h < n && in[i+h]==in[j+h]) h++;
    height[rank[i]-1] = h;
    if (h > 0) h--;
  }
  return height;
}
```

5 Geometry

5.1 Line Segment Intersection

```
struct pnt{ll x, y;};
ll ccw(pnt a, pnt b, pnt c){
  // 1 ccw, -1 cw, 0 collinear
  ll z = (b.x-a.x)*(c.y-a.y) - (b.y-a.y)*(c.x-a.x);
  return z? (z>0? 1:-1):0;
}
ll cc(pnt a, pnt b, pnt c){
  return min(a.x, b.x) <= c.x && c.x <= max(a.x, b.x) &&
    min(a.y, b.y) <= c.y && c.y <= max(a.y, b.y);
}

bool intersect(pnt a, pnt b, pnt c, pnt d){
  ll s1 = ccw(a,b,c), s2 = ccw(a,b,d);
  if(!s1 && !s2)
    return cc(a,b,c) || cc(a,b,d) ||
           cc(c,d,a) || cc(c,d,b);
  if(s1 && s1 == s2) return false;
  s1 = ccw(c,d,a), s2 = ccw(c,d,b);
  return !s1 || s1 != s2;
}
```

5.2 Convex Hull, Calipers

```
// Remove rotcal if you don't need it
template <typename T>
struct pnt{
  T x, y;
  pnt(): x{0}, y{0} {}
  pnt(T x, T y): x{x}, y{y} {}
  bool operator<(const pnt<T>& b)
    {return x<b.x || (x==b.x && y<b.y);}
};
```

```
template <typename T>
T ccw(const pnt<T>& a, const pnt<T>& b, const pnt<T>& c){
  return (b.x-a.x)*(c.y-a.y) - (b.y-a.y)*(c.x-a.x);
}
template <typename T>
struct Hull{
  vector<pnt<T>> p, U, L;
  void add(T x, T y){p.push_back(pnt<T>(x,y));}
  void make(){
    sort(entire(p));
    for(auto q: p){
      while(U.size() > 1 && ccw(U[U.size()-2],
        U[U.size()-1], q) >= 0) U.pop_back();
      while(L.size() > 1 && ccw(L[L.size()-2],
        L[L.size()-1], q) <= 0) L.pop_back();
      U.push_back(q); L.push_back(q);
    }
  }
  vector<pnt<T>> rotcal(){
    vector<pnt<T>> res;
    size_t i = 0, j = L.size()-1;
    while(i < U.size()-1 || j > 0){
      res.push_back(U[i]);
      res.push_back(L[j]);
      if(i == U.size()-1) j--;
      else if(j == 0) i++;
      else if((U[i+1].y-U[i].y)*(L[j].x-L[j-1].x) >
        (L[j].y-L[j-1].y)*(U[i+1].x-U[i].x)) i++;
      else j--;
    }
    return res;
  }
  int size(){return U.size() + L.size() - 2;}
};
```

What now?

6 Graph: Trees

6.1 Segment Tree (0-indexed)

```
struct Segtree{
  typedef ll T; // Change this to what you need
  T id = 0;
  T combine(T a, T b){"a <-- b";}

  int n; vector<T> arr;
  Segtree(int sz): n{1} {
    while(n < sz) n<<=1; arr.resize(n*2, id);
  }
  void update(int i, T v){
    for(arr[i+=n]=v; i>>=1;)
      arr[i] = combine(arr[i<<1], arr[i<<1|1]);
  }
  T query(int l, int r){
    T resl=id, resr=id;
    for(l+= n, r+= n+1; l < r; l/= 2, r/= 2){
      if(l&1) resl = combine(resl, arr[l++]);
      if(r&1) resr = combine(arr[--r], resr);
    }
    return combine(resl, resr);
  }
};
```

6.2 Lazy propagation (0-indexed)

```
struct Seglazy{
  typedef int T1; // Change this to what you need
  typedef int T2; // Change this to what you need
  T1 id = 0, initval = 0;
  T2 unused = 0;
  T1 combine(T1 a, T1 b){"Real a <-- b";}
  T2 combineL(T2 a, T2 b){"Lazy a <-- b";}
  void unlazy(int i, int nl, int nr){}

  int n; vector<T1> arr; vector<T2> lazy;
  Seglazy(int sz): n{1} {
    while(n < sz) n<<=1;
    arr.resize(n*2, initval); lazy.resize(n*2, unused);
  }
  void propagate(int x, int nl, int nr){
    if(lazy[x] == unused) return;
    if(x < n){
      lazy[x*2] = combineL(lazy[x*2], lazy[x]);
      lazy[x*2+1] = combineL(lazy[x*2+1], lazy[x]);
    }
    unlazy(x, nl, nr);
    lazy[x] = unused;
  }
  void update(int l,int r,T2 val)
    {update(l,r,val,1,0,n-1);}
  void update(int l,int r,T2 val,int x,int nl,int nr){
    propagate(x, nl, nr);
    if(r < nl || nr < l) return;
    if(l <= nl && nr <= r){
      lazy[x] = combineL(lazy[x], val);
      propagate(x, nl, nr);
      return;
    }
    int mid = (nl + nr) / 2;
    update(l,r,val,x*2,nl,mid);
    update(l,r,val,x*2+1,mid+1,nr);
    arr[x] = combine(arr[x*2], arr[x*2+1]);
  }
  T1 query(int l,int r){return query(l,r,1,0,n-1);}
  T1 query(int l,int r,int x,int nl,int nr){
    propagate(x, nl, nr);
    if(r < nl || nr < l) return id;
    if(l <= nl && nr <= r) return arr[x];
    int mid = (nl + nr) / 2;
    return combine(query(l,r,x*2,nl,mid),
                   query(l,r,x*2+1,mid+1,nr));
  }
};
```

6.3 Fenwick (0-indexed)

```
template <typename T>
struct Fenwick{
  int n; vector<T> arr;
  Fenwick(int n): n{n}, arr(n) {}
  void add(int i, T val){
    while(i < n) arr[i]+= val, i |= i+1;
  }
  T getsum(int i){
    T res = 0;
    while(i >= 0) res+= arr[i], i = (i&(i+1))-1;
    return res;
  }
  int kth(int k){
    int l = 0, r = arr.size();
    for(int i=0; i<=lsz; i++){
      int mid = (l+r)>>1;
      ll val = arr[mid-1];
      if(val >= k) r = mid;
      else l = mid, k-= val;
    }
    return l;
  }
};
```

6.4 Range Update Range Query Fenwick

```
struct RangeFenwick{
  vector<ll> arrmul, arradd;
  RangeFenwick(int n): arrmul(n), arradd(n){}
  void update(int l, int r, ll val){
    zeroupdate(l, val, -val*(l-1));
    zeroupdate(r, -val, val*r);
  }
  ll intersum(int i, int j){
    return getsum(j) - getsum(i-1);
  }

  void zeroupdate(int i, ll mul, ll add){
    for(; i<arrmul.size(); i|=i+1)
      arrmul[i]+= mul, arradd[i]+= add;
  }
  ll getsum(int i){
    ll mul = 0, add = 0, start = i;
    for(; i>=0; i=(i&(i+1))-1)
      mul+= arrmul[i], add+= arradd[i];
    return mul*start + add;
  }
};
```

6.5 Persistent Segtree

TODO

6.6 Euler Tour Tree

```
// i subtree -> [start[i], end[i]]
void dfs(int v){
    start[v] = i, seq[i++] = v;
    for(int u: adj[v]) if(start[u] == -1) dfs(u);
    end[v] = i-1;
}
```

6.7 Heavy-Light Decomposition

6.8 HLD

```
// YOU MUST CALL init !!!
// in = in-number of the node, usable for the segment tree
// top = top node of the group containing this node

struct HLD{
  int n, t;
  vector<vector<int>> adj, tadj;
  vector<int> sz, par, dep, in, top;

  HLD(int n): n{n}, t{0}, adj(n+1), tadj(n+1),
    sz(n+1), par(n+1), dep(n+1), in(n+1), top(n+1) {}
  void connect(int a, int b)
    {adj[a].push_back(b); adj[b].push_back(a);}

  void dfs_tree(int v, int prev=-1){
    for(int u: adj[v]){
      if(u == prev) continue;
      tadj[v].push_back(u); par[u] = v; dep[u] = dep[v]+1;
      dfs_tree(u, v);
    }
  }
  void dfs_sz(int v){
    sz[v] = 1;
    for(int &u: tadj[v]){
      dfs_sz(u); sz[v]+= sz[u];
      if(sz[u] > sz[tadj[v][0]]) swap(u, tadj[v][0]);
    }
  }
  void dfs_hld(int v){
    in[v] = t++;
    for(int u: tadj[v])
      top[u] = (u==tadj[v][0]? top[v]:u), dfs_hld(u);
  }
  void init(int i=1){
    dfs_tree(i); adj.clear(); dfs_sz(i); dfs_hld(i);
  }
};
```

7 Graph: Flow and Matching

7.1 Hopcroft-Karp (1-indexed)

```
// 1-indexed
const int INF = 0x3f3f3f3f;
struct Hopk{
  int n, m; vector<vector<int>> adj;
  vector<int> pu, pv, dist;
  Hopk(int n, int m): n(n), m(m), adj(n+1),
    pu(n+1), pv(m+1), dist(n+1, -1) {}
  void connect(int a, int b){adj[a].push_back(b);}
  bool bfs(){
    queue<int> Q;
    for(int u=1; u<=n; u++){
      if(!pu[u]) dist[u] = 0, Q.push(u);
      else dist[u] = INF;
    }
    dist[0] = INF;
    while(!Q.empty()){
      int u = Q.front(); Q.pop();
      if(dist[u] >= dist[0]) continue;
      for(int v: adj[u]) if(dist[pv[v]] == INF){
        dist[pv[v]] = dist[u] + 1;
        Q.push(pv[v]);
      }
    }
    return dist[0] != INF;
  }
  bool dfs(int u){
    if(!u) return true;
    for(int v: adj[u])
      if(dist[pv[v]] == dist[u] + 1 && dfs(pv[v])){
        pv[v] = u, pu[u] = v; return true;
      }
    dist[u] = INF;
    return false;
  }
  int send(){
    int match = 0;
    while(bfs()) for(int u=1; u<=n; u++)
```

```
      if(!pu[u]) match+= dfs(u);
    return match;
  }
};
```

7.2 Dinic

My code is flawed, please add the correct one. TODO

7.3 MCMF

My code is flawed, please add the correct one. TODO

7.4 LR-Flow

방법 1: a번 정점에서 b번 정점으로 가는 하한 l, 상한 r인 간선이 있을 때, a번 정점에서 b번 정점으로 가는 유량 l, 비용 -1인 간선, 유량 r-l, 비용 0인 간선으로 만든 뒤 MCMF

방법 2: a번 정점에서 b번 정점으로 가는 하한 l, 상한 r인 간선이 있을 때, 새로운 source에서 b번 정점으로 가는 유량 l인 간선 추가, a번 정점에서 새로운 sink로 가는 유량 l인 간선 추가, a번 정점에서 b번 정점으로 가는 유량 r-l인 간선으로 만들고, 기존 sink에서 기존 source로 가는 유량 무한인 간선 추가 이후 최대 유량이 l의 합이 되는지 확인

7.5 Blossom

```
// Thanks to koosaga for code
struct Blossom{
  int n, t;
  vector<vector<int>> adj;
  vector<int> orig, par, vis, match, aux;
  queue<int> Q;
  Blossom(int n): n{n}, t{0}, adj(n+1), orig(n+1),
    par(n+1), vis(n+1), match(n+1), aux(n+1) {}
  void connect(int a, int b){
    adj[a].push_back(b);
    adj[b].push_back(a);
  }

  void augment(int u, int v){
    int pv = v, nv;
    do{
      pv = par[v], nv = match[pv];
      match[v] = pv, match[pv] = v;
      v = nv;
    } while(u != pv);
  }
  int lca(int v, int w){
    ++t;
    while(1){
      if(v){
        if(aux[v] == t) return v;
        aux[v] = t, v = orig[par[match[v]]];
      }
      swap(v, w);
    }
  }
  void blossom(int v, int w, int a){
    while(orig[v] != a){
      par[v] = w, w = match[v];
      if(vis[w] == 1) Q.push(w), vis[w] = 0;
      orig[v] = orig[w] = a;
      v = par[w];
    }
  }
  bool bfs(int u){
    fill(entire(vis), -1), iota(entire(orig), 0);
    Q = queue<int>(); Q.push(u), vis[u] = 0;
    while(!Q.empty()){
      int v = Q.front(); Q.pop();
      for(int x: adj[v]){
        if(vis[x] == -1){
          par[x] = v, vis[x] = 1;
          if(!match[x]) return augment(u, x), true;
          Q.push(match[x]); vis[match[x]] = 0;
        }
        else if(vis[x] == 0 && orig[v] != orig[x]){
          int a = lca(orig[v], orig[x]);
          blossom(x, v, a), blossom(v, x, a);
        }
      }
    }
    return false;
  }

  int solve(){
    int ans = 0;
    for(int x=1; x<=n; x++) if(!match[x]){
      for(int y: adj[x]) if(!match[y]){
        match[x] = y, match[y] = x;
        ++ans; break;
      }
    }
    for(int i=1; i<=n; i++) if(!match[i] && bfs(i)) ++ans;
    return ans;
  }
};
```

8 Graph: Misc

8.1 Topological Sort (1-indexed)

```
struct Graph{
  int n; vector<vector<int>> adj;
  Graph(int n): n{n}, adj(n+1) {}
  void connect(int a, int b){adj[a].push_back(b);}
  vector<int> toposort(){
    vector<int> indg(n+1), L;
    queue<int> Q;
    for(int i=1; i<=n; i++) for(int j: adj[i]) indg[j]++;
    for(int i=1; i<=n; i++) if(!indg[i]) Q.push(i);
    for(int i=0; i<n; i++){
      if(Q.empty()) assert(0); // Change this if you need
      int p = Q.front(); Q.pop();
      L.push_back(p);
      for(int j: adj[p]) if (!--indg[j]) Q.push(j);
    }
    return L;
  }
};
```

8.2 SCC (1-indexed)

```
struct Graph{
  int n, cnt, sccnt; vector<vector<int>> adj;
  vector<int> up, visit, scx, stk;
  Graph(int n): n(n), adj(n+1), up(n+1), visit(n+1), scx(n+1) {}
  void connect(int a, int b){adj[a].push_back(b);}
  void dfs(int v){
    up[v] = visit[v] = ++cnt;
    stk.push_back(v);
    for(int nxt: adj[v]){
      if(!visit[nxt]) dfs(nxt), up[v] = min(up[v], up[nxt]);
      else if(!scx[nxt]) up[v] = min(up[v], visit[nxt]);
    }
    if(up[v] == visit[v]){
      ++sccnt; int t = -1;
      while(!stk.empty() && t != v){
        t = stk.back(); stk.pop_back();
        scx[t] = sccnt;
      }
    }
  }
  void getscc(){
    cnt = sccnt = 0;
    for(int i=1; i<=n; i++) if(!visit[i]) dfs(i);
  }
  vector<vector<int>> scc(){
    getscc();
    vector<vector<int>> res(sccnt);
    for(int i=1; i<=n; i++) res[scx[i]-1].push_back(i);
    return res;
  }
  vector<vector<int>> dag(){
    getscc();
    vector<vector<int>> res(*max_element(entire(scx))+1);
    for(int v=1; v<=n; v++){
      for(int u: adj[v]) if(scx[u] != scx[v])
        res[scx[v]].push_back(scx[u]);
    }
    return res;
  }
  vector<vector<int>> rdag(){
    getscc();
    vector<vector<int>> res(*max_element(entire(scx))+1);
    for(int v=1; v<=n; v++){
      for(int u: adj[v]) if(scx[u] != scx[v])
        res[scx[u]].push_back(scx[v]);
    }
    return res;
  }
};
```

8.3 2-SAT

```
struct TwoSAT{
  int n; Graph G;
  TwoSAT(int var): n{var}, G{Graph(2*var+1)} {}
  int v(int x){return n+x+1;}
  void cnf(int x, int y)
    {G.connect(v(-x),v(y)); G.connect(v(-y),v(x));}
  bool solve(){
    G.getscc();
    for(int x=1; x<=n; x++) if(G.scx[v(x)] == G.scx[v(-x)])
      return false;
    return true;
  }
};

struct TwoSATsol{
  vector<int> solve(){
    vector<vector<int>> scc = G.scc();
    vector<int> sol(n+1, -1);
    for(int x=1; x<=n; x++)
      if(G.scx[v(x)] == G.scx[v(-x)]) return {0};
    for(int i = scc.size()-1; i>=0; i--)
      for(int var: scc[i]) if(sol[abs(var-n-1)] == -1)
        sol[abs(var-n-1)] = var<n+1;
    return sol;
  }
};
```

8.4 Stoer-Wagner (0-indexed)

```
struct Mincut{
  int n; vector<vector<int>> graph;
  Mincut(int n): n{n}, graph(n, vector<int>(n)) {}
  void connect(int a, int b, int w)
    {if(a != b) graph[a][b]+= w, graph[b][a]+= w;}

  pair<int, pair<int, int>> stmin(vector<int> &active){
    vector<int> key(n), v(n);
    int s = -1, t = -1;
    for(size_t i=0; i<active.size(); i++){
      int maxv = -1, cur = -1;
      for(auto j: active) if(!v[j] && maxv < key[j])
        maxv = key[j], cur = j;
      t = s, s = cur; v[cur] = 1;
      for(auto j: active) key[j]+= graph[cur][j];
    }
    return make_pair(key[s], make_pair(s, t));
  }

  vector<int> cut;
  int solve(){
    int res = numeric_limits<int>::max();
    vector<vector<int>> grps; vector<int> active;
    cut.resize(n);
    for(int i=0; i<n; i++) grps.emplace_back(1, i);
    for(int i=0; i<n; i++) active.push_back(i);
    while(active.size() >= 2){
      auto stcut = stmin(active);
      if(stcut.first < res){
        res = stcut.first;
        fill(entire(cut), 0);
        for(auto v: grps[stcut.second.first]) cut[v] = 1;
      }
      int s, t; tie(s, t) = stcut.second;
      if(grps[s].size() < grps[t].size()) swap(s, t);
      active.erase(find(entire(active), t));
      grps[s].insert(grps[s].end(), entire(grps[t]));
      for(int i=0; i<n; i++)
        graph[i][s]+= graph[i][t], graph[i][t] = 0;
      for(int i=0; i<n; i++)
        graph[s][i]+= graph[t][i], graph[t][i] = 0;
      graph[s][s] = 0;
    }
    return res;
  }
};
```

8.5 Offline Dynamic Connectivity

```
struct DisjointHist{
  vector<int> par, sz;
  vector<pair<int, int>> history;
  DisjointHist(int n): par(n+1), sz(n+1, 1)
    {iota(entire(par), 0);}

  void un(int x, int y){
    x = fd(x), y = fd(y);
    if(x == y) return;
    if(sz[x] < sz[y]) swap(x, y);
    history.push_back({x, y});
    par[y] = x, sz[x]+= sz[y];
  }
  int fd(int x){return par[x]!=x? fd(par[x]) : x;}
  void rollback(){
    auto [x,y] = history.back(); history.pop_back();
    par[x] = x, par[y] = y, sz[x]-= sz[y];
  }
};

struct ODC{
  int n, Q, time = 0;
  DisjointHist DS;
  vector<map<int,int>> adj; // [u][v] -> addtime
  vector<int> end; // ending time of edge added at i
  vector<int> qtype; // 1 add 2 remove 3 question
  vector<pair<int,int>> quest; // pair of v at query i
  vector<bool> answer; // answer to the questions
  ODC(int n, int Q): n{n}, Q{Q}, DS(n), adj(n+1),
    end(Q), qtype(Q), quest(Q), answer(Q) {}

  void add(int x, int y){
    if(x > y) swap(x, y);
    quest[time] = {x,y}, qtype[time] = 1, end[time] = Q-1;
    adj[x][y] = time++;
  }
  void rem(int x, int y){
    if(x > y) swap(x, y);
    quest[time] = {x,y}, qtype[time] = 2;
    end[adj[x][y]] = time++;
  }
  void query(int x, int y){
    if(x > y) swap(x, y);
    quest[time] = {x, y}, qtype[time++] = 3;
  }

  void solve(){
    vector<int> E;
    for(int i=0; i<Q; i++) if(qtype[i] == 1)
      E.push_back(i);
    solve(0, Q-1, E);
  };
```

```
  void solve(int s, int e, vector<int> &E){
    if(s > e) return;
    int startnum = DS.history.size();

    int mid = (s+e)/2;
    vector<int> E1, E2;
    for(int i: E){
      auto [x,y] = quest[i];
      if(i <= s && e <= end[i]) DS.un(x, y);
      else{
        if(!(end[i] < s || i > mid)) E1.push_back(i);
        if(!(end[i] < mid+1 || i > e)) E2.push_back(i);
      }
    }

    if(s == e){
      auto [x,y] = quest[s];
      if(qtype[s] == 3) answer[s] = (DS.fd(x)==DS.fd(y));
    }
    else solve(s, mid, E1), solve(mid+1, e, E2);
    while(DS.history.size() != startnum) DS.rollback();
  }
};
```

9 Misc.

9.1 Hungarian

```
int hungarian(vector<vector<int>>& mat){
  int n = mat.size(), m = mat[0].size();
  vector<int> u(n+1), v(m+1), p(m+1), way(m+1), minv(m+1);
  vector<char> used(m+1);
  for(int i=1; i<=n; ++i) {
    int j0 = 0; p[0] = i;
    fill(entire(minv), INF);
    fill(entire(used), false);
    while(1){
      used[j0] = true;
      int i0 = p[j0], delta = INF, j1;
      for (int j=1; j<=m; ++j) if(!used[j]) {
        int cur = mat[i0-1][j-1] - u[i0] - v[j];
        if (cur < minv[j]) minv[j] = cur, way[j] = j0;
        if (minv[j] < delta) delta = minv[j], j1 = j;
      }
      for (int j=0; j<=m; ++j) {
        if (used[j]) u[p[j]] += delta, v[j] -= delta;
        else minv[j] -= delta;
      }
      j0 = j1;
      if(p[j0] == 0) break;
    }
    while(1){
      int j1 = way[j0];
      p[j0] = p[j1]; j0 = j1;
      if(!j0) break;
    }
  }
  //for (int j = 1; j <= m; ++j) matched[p[j]] = j;
  return -v[0];
}
```

9.2 O(1) RMQ

TODO

9.3 Convex Hull Trick

TODO

9.4 DNC Optimization

TODO

9.5 Knuth Optimization

TODO

9.6 Regex

TODO

9.7 References

```
/*
https://github.com/koosaga/olympiad/tree/master/Library
idk (TODO)
*/
```