Team Notebook

CoU_Miles_To_Go (Comilla University)

Md Rashidul Islam | A F Tamim | Saiful Islam

May 8, 2025

1 Number Theory

1.1 BigMod

```cpp
const int mod = 1e9 + 7;
int power(int x, long long n) { // O(log n)
  int ans = 1 % mod;
  while (n > 0) {
    if (n & 1) {
      ans = 1LL * ans * x % mod;
    }
    x = 1LL * x * x % mod;
    n >>= 1;
  }
  return ans;
}
```

1.2 Computing binomial coefficients modulo nCr mod M

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

const int N = 1e6 + 9;
using ll = long long;

int power(long long n, long long k, const int mod) {
  int ans = 1 % mod; n %= mod; if (n < 0) n += mod;
  while (k) {
    if (k & 1) ans = (long long) ans * n % mod;
    n = (long long) n * n % mod;
    k >>= 1;
  }
  return ans;
}
ll extended_euclid(ll a, ll b, ll &x, ll &y) {
  if (b == 0) {
    x = 1; y = 0;
    return a;
  }
  ll x1, y1;
  ll d = extended_euclid(b, a % b, x1, y1);
  x = y1;
  y = x1 - y1 * (a / b);
  return d;
}
ll inverse(ll a, ll m) {
  ll x, y;
  ll g = extended_euclid(a, m, x, y);
  if (g != 1) return -1;
  return (x % m + m) % m;
}
// returns n! % mod without taking all the multiple
    factors of p into account that appear in the
    factorial
// mod = multiple of p
// O(mod) * log(n)
int factmod(ll n, int p, const int mod) {
  vector<int> f(mod + 1);
  f[0] = 1 % mod;
  for (int i = 1; i <= mod; i++) {
    if (i % p) f[i] = 1LL * f[i - 1] * i % mod;
    else f[i] = f[i - 1];
  }
  int ans = 1 % mod;
  while (n > 1) {
    ans = 1LL * ans * f[n % mod] % mod;
    ans = 1LL * ans * power(f[mod], n / mod, mod) % mod
     ;
    n /= p;
  }
  return ans;
}
ll multiplicity(ll n, int p) {
  ll ans = 0;
  while (n) {
    n /= p;
    ans += n;
  }
  return ans;
}
// C(n, r) modulo p^k
// O(p^k log n)
int ncr(ll n, ll r, int p, int k) {
  if (n < r or r < 0) return 0;
  int mod = 1;
  for (int i = 0; i < k; i++) {
    mod *= p;
  }
  ll t = multiplicity(n, p) - multiplicity(r, p) -
     multiplicity(n - r, p);
  if (t >= k) return 0;
  int ans = 1LL * factmod(n, p, mod) * inverse(factmod(
     r, p, mod), mod) % mod * inverse(factmod(n - r, p,
      mod), mod) % mod;
  ans = 1LL * ans * power(p, t, mod) % mod;
  return ans;
}
// finds x such that x % m1 = a1, x % m2 = a2. m1 and
    m2 may not be coprime
// here, x is unique modulo m = lcm(m1, m2). returns (x
    , m). on failure, m = -1.
pair<ll, ll> CRT(ll a1, ll m1, ll a2, ll m2) {
  ll p, q;
  ll g = extended_euclid(m1, m2, p, q);
  if (a1 % g != a2 % g) return make_pair(0, -1);
  ll m = m1 / g * m2;
  p = (p % m + m) % m;
```

```
  q = (q % m + m) % m;
  return make_pair((p * a2 % m * (m1 / g) % m + q * a1 % m * (m2 / g) % m) %  m, m);
}
int spf[N];
vector<int> primes;
void sieve() {
  for(int i = 2; i < N; i++) {
    if (spf[i] == 0) spf[i] = i, primes.push_back(i);
    int sz = primes.size();
    for (int j = 0; j < sz && i * primes[j] < N && primes[j] <= spf[i]; j++) {
      spf[i * primes[j]] = primes[j];
    }
  }
}
// O(m log(n) log(m))
int ncr(ll n, ll r, int m) {
  if (n < r or r < 0) return 0;
  pair<ll, ll> ans({0, 1});
  while (m > 1) {
    int p = spf[m], k = 0, cur = 1;
    while (m % p == 0) {
      m /= p; cur *= p;
      ++k;
    }
    ans = CRT(ans.first, ans.second, ncr(n, r, p, k), cur);
  }
  return ans.first;
}
int32_t main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  sieve();
  int t; cin >> t;
  while (t--) {
    ll n, k; cin >> n >> k;
    int m; cin >> m;
    ll r = (n + k - 1) / k;
    cout << r << ' ' << ncr((k - n % k) % k + r - 1, r - 1, m) << '\n';
  }
  return 0;
}
```

1.3 Extended Euclidean Algorithm

```
int extended_euclid(int a, int b, int &x, int &y) {
    if (b == 0) {
        x = 1;
        y = 0;
        return a;
    }
    int x1, y1;
    int d = extended_euclid(b, a % b, x1, y1);
    x = y1;
    y = x1 - y1 * (a / b);
    return d;
}
```

1.4 First n digit and last n digit of $a^{\hat{b}}$

```
ll binpow(ll n, ll k) {
    ll res = 1;
    while (k > 0) {
        if (k & 1) res = (res * n) % 1000;  // Mod 1000 to get last 3 digits
        n = (n * n) % 1000;
        k >>= 1;
    }
    return res;
}

void akam() {
    ll n, k;
    cin >> n >> k;

    string last = to_string(binpow(n, k));
    while (last.size() < 3) last = '0' + last;  // Pad with leading zeros if needed

    long double d = k * log10(n) + 1;
    ll d_er_floor = floor(d);
    d = d - (d_er_floor + 1 - 3);  // Adjust to extract first 3 digits

    ll first = pow(10.0, d);

    cout << first << " ... " << last << endl;
}
```

1.5 Modular Inverse

```
// Eikhane extended_euclidian hbe
ll mod_inverse(ll a, ll m) {
    ll x, y;
    ll gc = extended_euclid(a, m, x, y);
    if (gc != 1) return -1;
    return (x % m + m) % m;
}
```

1.6 nCr (Bionomial Co-efficient)

```
long long ncr(long long n, long long r) {
    if (n < r) return 0;

    ll ans = 1;
    ans *= fact[n];

    ll d = fact[r];
    d *= fact[n - r];
    d %= MOD;

    ans *= binpow(d, MOD - 2, MOD);
    ans %= MOD;

    return ans;
}
```

1.7 Number of Divisor

```
// com->O(n * log n)
long long NumberOfDivisor(long long n) {
    long long ans = 1;
    for (long long i = 0; prime[i] * prime[i] <= n; i++) {
        long long counter = 0;
        while (n % prime[i] == 0) {
            n /= prime[i];
            counter++;
        }
        ans *= (counter + 1);
    }
    if (n > 1) ans *= 2;
    return ans;
}
```

1.8 Phi of N

```
// com->sqrt(n);
// number of coprimes with n
long long phi(long long n) {
    long long result = n;
    for (long long p = 2; p * p <= n; ++p) {
        if (n % p == 0) {
            while (n % p == 0) {
                n /= p;
            }
            result -= result / p;
        }
    }
    if (n > 1) result -= result / n;
    return result;
}
```

1.9 Phi 1 to N

```
// complexity-> nloglogn
// phi[i]->number of coprimes with i
const int N = 1e5 + 9;
int phi[N];
void totient() {
  for (int i = 1; i < N; i++) phi[i] = i;
  for (int i = 2; i < N; i++) {
    if (phi[i] == i) {
      for (int j = i; j < N; j += i) phi[j] -= phi[j] / i;
    }
  }
}
```

1.10 Product of divisors

```
void akam() {
    ll prf[n + 1], suf[n + 1];
    suf[n] = 1;
    prf[n] = 1;

    prf[0] = (v[0].S + 1) % (mod - 1);
    suf[n - 1] = (v[n - 1].S + 1) % (mod - 1);

    j = n - 2;
    for (i = 1; i < n; i++) {
        prf[i] = ((v[i].S + 1) * prf[i - 1]) % (mod - 1);
        suf[j] = ((v[j].S + 1) * suf[j + 1]) % (mod - 1);
        j--;
    }

    ans = 1;
    for (i = 0; i < n; i++) {
        /// This part is for product of divisors
        x = ((v[i].S + 1) * v[i].S) / 2;
        x %= (mod - 1);

        if (i == 0) {
            y = suf[i + 1] % (mod - 1);
        } else if (i == n - 1) {
            y = prf[i - 1] % (mod - 1);
        } else {
            y = (prf[i - 1] * suf[i + 1]) % (mod - 1);
        }

        x = (x * y) % (mod - 1);
        y = big(v[i].F, x, mod);
        ans = (ans * y) % mod;
    }

    cout << ans << endl;
}
```

1.11 Sieve

```
const int N = 1e6 + 9;
vector<int> primes;
bool is_prime[N];
// use bitset<N> is_prime; to have O(N/64) memory complexity
// using bitset you can solve upto around N = 10^8 in 1s
void sieve_v0() {
  for (int i = 2; i < N; i++) {
    is_prime[i] = true;
  }
  for (int i = 2; i * i < N; i++) {
    if (is_prime[i]) {
      for (int j = i * i; j < N; j += i) {
        is_prime[j] = false;
      }
    }
  }
  for (int i = 2; i < N; i++) {
    if (is_prime[i]) {
      primes.push_back(i);
    }
  }
}
// sieve with smallest prime factors (spf)
int spf[N];
void sieve() {
  for (int i = 2; i < N; i++) {
    spf[i] = i;
  }
  for (int i = 2; i * i < N; i++) {
    if (spf[i] == i) {
      for (int j = i * i; j < N; j += i) {
        spf[j] = min(spf[j], i);
      }
    }
  }
  for (int i = 2; i < N; i++) {
    if (spf[i] == i) {
      primes.push_back(i);
    }
  }
}
```

1.12 Smallest Number Having Exactly K Divisors

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

const int N = 1e6 + 9, mod = 1e9 + 7;

int power(long long n, long long k) {
  int ans = 1 % mod; n %= mod; if (n < 0) n += mod;
  while (k) {
    if (k & 1) ans = (long long) ans * n % mod;
    n = (long long) n * n % mod;
    k >>= 1;
  }
  return ans;
}
int spf[N];
vector<int> primes;
void sieve() {
  for(int i = 2; i < N; i++) {
    if (spf[i] == 0) spf[i] = i, primes.push_back(i);
    int sz = primes.size();
    for (int j = 0; j < sz && i * primes[j] < N && primes[j] <= spf[i]; j++) {
      spf[i * primes[j]] = primes[j];
    }
  }
}
double lgp[N];
vector<long long> v;
unordered_map<long long, pair<double, int>> dp[100];
pair<double, int> yo(int i, long long n) { // it solves for odd divisors
  if (n == 1) {
    return {0, 1};
  }
  if (dp[i].find(n) != dp[i].end()) {
    return dp[i][n];
  }
  pair<double, int> ans = {1e50, 0};
  for (auto x: v) {
    if (x > n) break;
    if (n % x != 0) continue;
    auto z = lgp[i + 1] * (x - 1); // i for all divisors
    if (z > ans.first) {
      break;
    }
    auto cur = yo(i + 1, n / x);
    cur.first += z;
    cur.second = 1LL * cur.second * power(primes[i + 1], x - 1) % mod; // i for all divisors
    ans = min(ans, cur);
  }
  return dp[i][n] = ans;
}
int32_t main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  sieve();
  for (int i = 0; i < 100; i++) {
    lgp[i] = log(primes[i]);
  }
  int t, cs = 0; cin >> t;
  while (t--) {
    long long n; cin >> n;
    ++n;
    if (n == 1) {
      cout << "Case " << ++cs << ": " << 1 << '\n';
      continue;
    }
    v.clear();
    for (int i = 1; 1LL * i * i <= n; i++) {
      if (n % i == 0) {
        if (i > 1) v.push_back(i);
        if (i != n / i) {
          v.push_back(n / i);
        }
      }
    }
    sort(v.begin(), v.end());
    cout << "Case " << ++cs << ": " << yo(0, n).second << '\n';
  }
  return 0;
}
```

```
// https://lightoj.com/problem/politeness
```

1.13 Sum of divisors 1 to N

```
ll inv2;

long long fun(long long start, long long end) {
    return ((((end - start + 1) % mod) * ((start + end) % mod) % mod) * inv2 % mod);
}

void akam() {
    ll n, i, j, c = 0, x, y, k, ans = 0, sum = 0;
    cin >> n;
    inv2 = 500000004;

    ll first_same = 1, last_same;

    while (first_same <= n) {
        j = n / first_same;
        last_same = n / j;
        sum = (sum + j * fun(first_same, last_same)) % mod;
        first_same = last_same + 1;
    }

    cout << sum << endl;
}
```

1.14 Power tower of Three Number

```
// (x^y)^k%mod
ll po(ll n, ll m, ll md) {
    ll res = 1;
    while (m > 0) {
        if (m & 1) res = (res * n) % md;
        n = (n * n) % md;
        m >>= 1;
    }
    return res;
}

void akam() {
    ll n, i, j, c = 0, x, y, k, ans = 0, sum = 0;
    // cout << "Case " << tst << ": ";
    cin >> x >> y >> k;

    y = po(y, k, mod - 1);  // Using Euler's totient of mod
    cout << po(x, y, mod) << endl;
}
```

1.5 Miller Rabin

```
// implementation MillerRabin::init();
// MillerRabin::miller_rabin(n)
using ll = long long;
namespace MillerRabin {
  mt19937 rnd(chrono::steady_clock::now().time_since_epoch().count());
  const int P = 1e6 + 9;
  int primes[P], spf[P];
  inline ll mul_mod(ll x, ll y, ll m) {
    ll res = __int128(x) * y % m;
    return res;
  }
  inline ll pow_mod(ll x, ll n, ll m) {
    ll res = 1 % m;
    for (; n; n >>= 1) {
      if (n & 1) res = mul_mod(res, x, m);
      x = mul_mod(x, x, m);
    }
    return res;
  }
  inline bool miller_rabin(ll n) {
    if (n <= 2 || (n & 1 ^ 1)) return (n == 2);
    if (n < P) return spf[n] == n;
    ll c, d, s = 0, r = n - 1;
    for (; !(r & 1); r >>= 1, s++) {}
    // each iteration is a round
    for (int i = 0; primes[i] < n && primes[i] < 32; i++) {
      c = pow_mod(primes[i], r, n);
      for (int j = 0; j < s; j++) {
        d = mul_mod(c, c, n);
        if (d == 1 && c != 1 && c != (n - 1)) return false;
        c = d;
      }
      if (c != 1) return false;
    }
    return true;
  }
  void init() {
    int cnt = 0;
    for (int i = 2; i < P; i++) {
      if (!spf[i]) primes[cnt++] = spf[i] = i;
      for (int j = 0, k; (k = i * primes[j]) < P; j++) {
        spf[k] = primes[j];
        if (spf[i] == spf[k]) break;
      }
    }
  }
}
```

2 Data Structures

2.1 DSU

```
const int N = 3e5 + 9;
struct DSU {
  vector<int> par, rnk, sz;
  int c;
  DSU(int n) : par(n + 1), rnk(n + 1, 0), sz(n + 1, 1), c(n) {
    for (int i = 1; i <= n; ++i) par[i] = i;
  }
  int find(int i) {
    return (par[i] == i ? i : (par[i] = find(par[i])));
  }
  bool same(int i, int j) {
    return find(i) == find(j);
  }
  int get_size(int i) {
    return sz[find(i)];
  }
  int count() {
    return c;    //no of connected components
  }
  int merge(int i, int j) {
    if ((i = find(i)) == (j = find(j))) return -1;
    else --c;
    if (rnk[i] > rnk[j]) swap(i, j);
    par[i] = j;
    sz[j] += sz[i];
    if (rnk[i] == rnk[j]) rnk[j]++;
    return j;
  }
};
//init -> DSU dsu(n);
```

2.2 Euler Tour + Query

```
ll in_time[200001];
ll seg_tree[800001];
ll lazy[800001];
vector<ll> darr; // dfs array or euler tour array

ll dfs(vector<vector<ll>>& tree, ll node, ll parent, ll & timer) {
    subtree[node] = 1;
    in_time[node] = timer;
    timer++;
    darr.pb(node);

    for (ll child : tree[node]) {
        if (child != parent) {
            subtree[node] += dfs(tree, child, node, timer);
        }
    }

    return subtree[node];
}

void lazy_update(ll node, ll b, ll e) {
    seg_tree[node] = ((e - b + 1) * lazy[node]);
    if (b != e) {
        ll left = node << 1;
        ll right = left + 1;
```

```
        lazy[left] += lazy[node];
        lazy[right] += lazy[node];
    }
    lazy[node] = 0;
}

void initi(ll node, ll b, ll e, ll* arr) {
    if (b == e) {
        seg_tree[node] = arr[darr[b]];
        return;
    }
    ll left = node * 2;
    ll right = left + 1;
    ll mid = (b + e) / 2;

    initi(left, b, mid, arr);
    initi(right, mid + 1, e, arr);
    seg_tree[node] = seg_tree[left] + seg_tree[right];
}

void stu(ll node, ll b, ll e, ll i, ll j, ll newval) {
    if (lazy[node] != 0) lazy_update(node, b, e);
    if (i > e || j < b) return;

    if (b >= i && e <= j) {
        lazy[node] = newval;
        lazy_update(node, b, e);
        return;
    }

    ll left = node * 2;
    ll right = left + 1;
    ll mid = (b + e) / 2;

    stu(left, b, mid, i, j, newval);
    stu(right, mid + 1, e, i, j, newval);

    seg_tree[node] = seg_tree[left] + seg_tree[right];
}

ll stq(ll node, ll b, ll e, ll i, ll j) {
    if (lazy[node] != 0) lazy_update(node, b, e);
    if (i > e || j < b) return 0;
    if (b >= i && e <= j) return seg_tree[node];

    ll left = node * 2;
    ll right = left + 1;
    ll mid = (b + e) / 2;

    ll p1 = stq(left, b, mid, i, j);
    ll p2 = stq(right, mid + 1, e, i, j);

    return p1 + p2;
}

void akam() {
    ll n, i, x, y, k, ans = 0;
    cin >> n >> k;

    ll arr[n + 1];
    for (i = 1; i <= n; i++) {
        cin >> arr[i];
    }

    vector<vector<ll>> tree(n + 1);
    for (i = 0; i < n - 1; i++) {
        cin >> x >> y;
        tree[x].pb(y);
        tree[y].pb(x);
    }

    ll timer = 0;
    darr.pb(0);
    dfs(tree, 1, -1, timer);

    ll ind[n + 1];
    for (i = 1; i < darr.size(); i++) {
        ind[darr[i]] = i;
    }

    initi(1, 1, n, arr);

    for (i = 0; i < k; i++) {
        cin >> x;
        if (x == 1) {
            cin >> x >> y;
            stu(1, 1, n, ind[x], ind[x], y);
        } else if (x == 2) {
            cin >> x;
            y = ind[x];
            ans = stq(1, 1, n, y, y + subtree[x] - 1);
            cout << ans << '\n';
        }
    }
}

int main() {
    ios_base::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
    akam();
    return 0;
}
```

2.3 TRIE OF INTEGER

```
struct Trie {
    const int B = 64;
    struct node {
      node* nxt[2];
      int sz;
      node() {
        nxt[0] = nxt[1] = NULL;
        sz = 0;
      }
    }*root;

    Trie() {
      root = new node();
    }
    void insert(int val) {
      node* cur = root;
      cur->sz++;
      for (int i = B - 1; i >= 0; i--) {
        int b = val >> i & 1;
        if (cur->nxt[b] == NULL) cur->nxt[b] = new node
     ();
        cur = cur->nxt[b];
        cur->sz++;
      }
    }

    int query(int x, int k) { // number of values s.t.
     val ^ x < k
      node* cur = root;
      int ans = 0;
      for (int i = B - 1; i >= 0; i--) {
        if (cur == NULL) break;
        int b1 = x >> i & 1, b2 = k >> i & 1;
        if (b2 == 1) {
          if (cur->nxt[b1])
              ans += cur->nxt[b1]->sz;
          cur = cur->nxt[!b1];
        } else cur = cur->nxt[b1];
      }
      return ans;
    }

    void erase(int x) {
      node* cur = root;
      cur->sz--;
      for (int i = B - 1; i >= 0; i--) {
        int b = (x >> i) & 1;
        if (cur->nxt[b] == NULL) return;
        cur = cur->nxt[b];
        cur->sz--;
      }
    }

    int get_max(int x) { // returns maximum of val ^ x
      node* cur = root;
      int ans = 0;
      for (int i = B - 1; i >= 0; i--) {
        int k = x >> i & 1;
        if (cur->nxt[!k] && cur->nxt[!k]->sz > 0)
          cur = cur->nxt[!k], ans <<= 1, ans++;
        else if (cur->nxt[k]->sz > 0)
          cur = cur->nxt[k], ans <<= 1;
        else break;
      }
      return ans;
```

```
    }

    int get_min(int x) { // returns minimum of val ^ x
      node* cur = root;
      int ans = 0;
      for (int i = B - 1; i >= 0; i--) {
        int k = x >> i & 1;
        if (cur->nxt[k] && cur->nxt[k]->sz > 0) {
          cur = cur->nxt[k];
          ans <<= 1;
        } else if (cur->nxt[!k] && cur->nxt[!k]->sz > 0) {
          cur = cur->nxt[!k];
          ans = (ans << 1) | 1;
        } else {
          break;
        }
      }
      return ans;
    }

    void del(node* cur) {
      for (int i = 0; i < 2; i++) if (cur->nxt[i]) del(cur->nxt[i]);
      delete(cur);
    }
  } t;
```

2.4 STRING OF INTEGER

```
struct Trie {
    struct node {
        node *nxt[26];
        int pref_cnt=0,end_cnt=0;
        node() {
            for (int i = 0; i < 26; i++) {
                nxt[i] = NULL;
            }
        }
    } *root;

    Trie() {
        root = new node();
    }

    void insert(string &s) {
        node *cur = root;

        for (int i = 0; i < s.size(); i++) {
            int ch = s[i] - 'a';
            if (cur->nxt[ch] == NULL)
                cur->nxt[ch] = new node();
            cur = cur->nxt[ch];
            cur->pref_cnt++;
        }
        cur->end_cnt++;
    }
    int equal_to(string &s) { //words equal to s
        node *cur = root;
        for (int i = 0; i < s.size(); i++) {
            int ch = s[i] - 'a';
            if (cur->nxt[ch] == NULL) return 0;
            cur = cur->nxt[ch];
        }
        return cur->end_cnt;
    }
    int starts_with(string &s) { //words preffix==s
        node *cur = root;
        for (int i = 0; i < s.size(); i++) {
            int ch = s[i] - 'a';
            if (cur->nxt[ch] == NULL) return 0;
            cur = cur->nxt[ch];
        }
        return cur->pref_cnt;
    }
    void erase(string &s) { //erase one instance of s
        node *cur = root;
        for (int i = 0; i < s.size(); i++) {
            int ch = s[i] - 'a';
            if (cur->nxt[ch] == NULL) return;
            cur = cur->nxt[ch];
            cur->pref_cnt--;
        }
        cur->end_cnt--;
    }
    void del(node *cur) {
        for (int i = 0; i < 26; i++) {
            if (cur->nxt[i] != NULL) del(cur->nxt[i]);
        }
        delete cur;
    }

    ~Trie() {
        del(root);
    }
};
// init-> Trie tr;
```

2.5 KADANES ALGO

```
int ans = LONG_LONG_MIN,sum=0;
    for (int i = 0; i < n; i++) {
        sum= max(v[i], sum+v[i]);
        ans = max(ans, sum);
    }
    ans = max(ans, sum);
```

2.6 Merge tree

```
vecl v[4 * (30000 + 1)];
ll a[30000 + 5];

void build(ll node, ll b, ll e) {
    if (b == e) {
        v[node].pb(a[e]);
        return;
    }
    ll l, r, mid;
    mid = (b + e) / 2;
    l = node * 2;
    r = l + 1;

    build(l, b, mid);
    build(r, mid + 1, e);

    ll i = 0, j = 0;
    while (i < v[l].size() && j < v[r].size()) {
        if (v[l][i] <= v[r][j]) {
            v[node].pb(v[l][i]);
            i++;
        } else {
            v[node].pb(v[r][j]);
            j++;
        }
    }
    while (j < v[r].size()) {
        v[node].pb(v[r][j]);
        j++;
    }
    while (i < v[l].size()) {
        v[node].pb(v[l][i]);
        i++;
    }
}

ll query(ll node, ll b, ll e, ll i, ll j, ll k) {
    if (e < i || b > j) return 0;
    if (b >= i && e <= j) {
        ll res = v[node].size() -
                 (upper_bound(v[node].begin(), v[node].end(), k) - v[node].begin());
        return res;
    }

    ll l, r, mid;
    mid = (b + e) / 2;

    l = query(node * 2, b, mid, i, j, k);
    r = query(node * 2 + 1, mid + 1, e, i, j, k);

    return l + r;
}

void akam() {
    ll n, i, j, c = 0, x, y, k, ans = 0, sum = 0;
    cin >> n;
    for (i = 1; i <= n; i++) cin >> a[i];

    build(1, 1, n);

    cin >> k;
    for (i = 0; i < k; i++) {
```

```
        cin >> x >> y >> j;
        ans = query(1, 1, n, x, y, j);
        cout << ans << endl;
    }
}

int main() {
    ios_base::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);

    akam();
    return 0;
}
```

2.7 Monotonic Queue

```
struct monotonic_queue
{
    deque <pair <int, int> > dq;
    void add(int val, int in)
    {
        // strictly increasing order
        while (!dq.empty() && dq.back().first >= val)
            dq.pop_back();
        dq.push_back({val, in});
    }
    void del(int in)
    {
        if (!dq.empty() && dq.front().second == in)
            dq.pop_front();
    }
};
```

2.8 Ordered Set

```
#include<ext/pb_ds/assoc_container.hpp>
#include<ext/pb_ds/tree_policy.hpp>
using namespace __gnu_pbds;
template <typename T> using o_set = tree<T, null_type,
    less<T>, rb_tree_tag,
    tree_order_statistics_node_update>;
template <typename T, typename R> using o_map = tree<T,
     R, less<T>, rb_tree_tag,
    tree_order_statistics_node_update>;
int main() {
  o_set<int>se;
  se.insert(1);
  se.insert(2);
  cout << *se.find_by_order(0) << endl; ///k th element
  cout << se.order_of_key(2) << endl; ///number of
     elements less than k
  o_map<int, int>mp;
  mp.insert({1, 10});
  mp.insert({2, 20});
  cout << mp.find_by_order(0)->second << endl; ///k th
     element
  cout << mp.order_of_key(2) << endl; ///number of
     first elements less than k
  return 0;
}
```

2.9 Segment Tree

```
class sgtree{
    vector<int> seg;
public:
    sgtree(int n) {
        seg.resize(4*n);
    }
    void build(int ind, int low, int high, int arr[]) {
        if(low==high) {
            seg[ind]=arr[low];
            return;
        }
        int mid = (low+high)/2;
        build(2*ind+1, low, mid, arr);
        build(2*ind+2, mid+1, high, arr);
        seg[ind] = min(seg[2*ind+1], seg[2*ind+2]);
    }
    int query(int ind, int low, int high, int l,int r)
     {
        if(r<low || high<l) return INT_MAX;
        if(low>=l && high<=r) return seg[ind];
        int mid = (low+high)/2;
        int left = query(2*ind+1, low, mid, l, r);
        int right = query(2*ind+2, mid+1, high, l, r);
        return min(left, right);
    }
    void update(int ind, int low, int high, int i, int
     val) { //sets val
        if(low==high) {
            seg[ind] = val;
            return;
        }
        int mid = (low+high)/2;
        if(i<=mid) update(2*ind+1, low, mid, i,val);
        else update(2*ind+2, mid+1, high, i, val);
        seg[ind]= min(seg[2*ind+1], seg[2*ind+2]);
    }
};
// init -> sgtree tr(n);
```

2.10 Segment Tree Lazy

```
// update increases val
class sgtree{
    vector<int> seg,lazy;
public:
    sgtree(int n) {
        seg.resize(4*n+5);
        lazy.resize(4*n+5);
    }
    void build(int ind, int low, int high, int arr[]) {
        if(low==high) {
            seg[ind]=arr[low];
            return;
        }
        int mid = (low+high)/2;
        build(2*ind+1, low, mid, arr);
        build(2*ind+2, mid+1, high, arr);
        seg[ind] = (seg[2*ind+1]+seg[2*ind+2]);
    }
    int query(int ind, int low, int high, int l,int r)
     {
        if(lazy[ind]!=0) {
            seg[ind] += (high-low+1)*lazy[ind]; //
     remove + for set update
            if(low!=high) {
                lazy[2*ind+1] += lazy[ind]; //remove +
     for set update
                lazy[2*ind+2] += lazy[ind]; //remove +
     for set update
            }
            lazy[ind]=0;
        }
        if(r<low || high<l) return 0;
        if(low>=l && high<=r) return seg[ind];
        int mid = (low+high)/2;
        int left = query(2*ind+1, low, mid, l, r);
        int right = query(2*ind+2, mid+1, high, l, r);
        return left+right;
    }
    void update(int ind, int low, int high, int l, int
     r, int val) {
        if(lazy[ind]!=0) {
            seg[ind] += (high-low+1)*lazy[ind]; //
     remove + for set update
            if(low!=high) {
                lazy[2*ind+1] += lazy[ind]; //remove +
     for set update
                lazy[2*ind+2] += lazy[ind]; //remove +
     for set update
            }
            lazy[ind]=0;
        }
        if(high<l || r<low) return;
        if(low>=l && high<=r) {
            seg[ind] += (high - low + 1)*val;
            if(low!=high) {
                lazy[2*ind+1] += val;
                lazy[2*ind+2] += val;
            }
            return;
        }
        int mid = (low + high)/2;
        update(2*ind+1, low, mid, l, r, val);
        update(2*ind+2, mid+1, high, l, r, val);
        seg[ind]  = seg[2*ind+1] + seg[2*ind+2];
    }
};
```

2.11 Sparse Table

```
const int N = 1e5 + 9;
int t[N][18], a[N];
void build(int n) {
  for(int i = 1; i <= n; ++i) t[i][0] = a[i];
  for(int k = 1; k < 18; ++k) {
    for(int i = 1; i + (1 << k) - 1 <= n; ++i) {
      t[i][k] = min(t[i][k - 1], t[i + (1 << (k - 1))][k - 1]);
    }
  }
}

int query(int l, int r) {
  int k = 31 - __builtin_clz(r - l + 1);
  return min(t[l][k], t[r - (1 << k) + 1][k]);
}
```

2.12 We Need to Find an increasing subsequence of b numbers such that their difference is minimized

```
// We Need to Find an increasing subsequence of
// b numbers such that their difference is minimized
int a, b;
cin >> a >> b;
vi v(a);
cinv(v);
vi dp(a, INT_MAX);
for (int k = 2; k <= b; k++) {
    set<int> s;
    for (int i = a - 1; i >= 0; i--) {
        if (dp[i] == INT_MAX) {
            // Not Possible
            continue;
        }
        s.insert(v[i] + dp[i]); // storing last bounds
        auto it = s.upper_bound(v[i] + dp[i]);
        // Guaranteed that we can extend if there exist
        if (it == s.end()) {
            dp[i] = INT_MAX;
        } else {
            dp[i] = *it - v[i]; // new difference
        }
    }
}
int ans = *min_element(all(dp));
if (ans == INT_MAX) ans = -1;
cout << ans << endl;
```

2.13 Ternary Search

```
// ternary seach on integer
// for minimum
while (hi - lo > 4)
{
    int m1 = (hi + lo) >> 1LL;
    int m2 = (hi + lo) / 2 + 1;
    int a1 = koro(v, m1);
    int a2 = koro(v, m2);
    if (a1 > a2) lo = m1;
    else hi = m2;
    // for max
    if (a1 > a2) hi = m1;
    else lo = m2;
}
loop through lo - 10 to hi + 10;
// ternary seach on double
double l = ... , r = ... , EPS = 1e-7;
while (r - l > EPS)
{
    double m1 = l + (r - l) / 3,
           m2 = r - (r - l) / 3;
    if (f(m1) < f(m2))
        l = m1;
    else
        r = m2;
}
f(r) - maximum of function
```

3 Dynamic Programming

3.1 Knapsack

```
for (int i = 1; i <= n; i++)
  for (int j = w[i]; j <= W; j++)
    f[j] = max(f[j], f[j - w[i]] + v[i]);
```

3.2 LCS

```
// In the Name of Allah, the Beneficent, the Merciful.
#include <bits/stdc++.h>
using namespace std;
#define ll long long
#define vl vector<long long>
#define vi vector<int>
#define F first
#define S second
#define inv(a) for (auto &it : a) cin >> it;
#define outv(a) for (auto &it : a) cout << it << " ";
     cout << "\n";
#define outv1(a) for (auto &it : a) cout << it << endl;
#define pb push_back
#define all(v) v.begin(), v.end()
#define rall(v) v.rbegin(), v.rend()
int Set(int N, int pos) { return N = N | (1 << pos); }
int reset(int N, int pos) { return N = N & ~(1 << pos);
     }
bool check(int N, int pos) { return (bool)(N & (1 <<
    pos)); }
#define fastio ios_base::sync_with_stdio(false); cin.
    tie(NULL); cout.tie(NULL)
int gcd(int a, int b) {return __gcd(a,b);};
#define  int long long


void solve() {

    string s,t;
    cin>>s>>t;

    int n = s.size(),m=t.size();

    vector<vi> dp(n+1, vi(m+1,-1)),choice(n+1, vi(m
     +1,-1));

    dp[0][0]=0;
    if(s[0]==t[0]) {
        choice[0][0]=0;
        dp[0][0]++;
    }

    for (int i = 1; i <= n; i++)
    {
        for (int j = 1; j <= m; j++)
        {
            int &ans = dp[i][j];
            if(i-1>=0 && j-1>=0 && (s[i-1]==t[j-1])) {
                ans = max(ans, dp[i-1][j-1]+1);
                choice[i][j]=0;
            }
            else {
                if(dp[i-1][j]>=dp[i][j-1]) {
                    ans = max(ans, dp[i-1][j]);
                    choice[i][j]=1;
                }
                else  {
                    ans = max(ans, dp[i][j-1]);
                    choice[i][j]=2;
                }

            }


        }

    }
    // cout<<dp[n][m]<<endl;

    string res = "";
    int i = n,j=m;
    while (i>=1 && j>=1)
    {
        if(choice[i][j]==0) {
            res+=s[i-1];
            i--,j--;
        }
        else if(choice[i][j]==1) {
            i--;
        }
        else {
```

```
            j--;
        }
    }
    reverse(all(res));
    cout<<res<<endl;

}
```

3.3 LIS

```
ll lis(vector<ll> const &a) {
    ll n = a.size();
    const ll INF = 1e9;
    vector<ll> d(n + 1, INF);
    d[0] = -INF;

    for (ll i = 0; i < n; i++) {
        ll l = upper_bound(d.begin(), d.end(), a[i]) - d.begin();
        if (d[l - 1] < a[i] && a[i] < d[l])
            d[l] = a[i];
    }

    ll ans = 0;
    for (ll l = 0; l <= n; l++) {
        if (d[l] < INF)
            ans = l;
    }

    return ans;
}
```

3.4 Longest Common Subarray

```
#include <bits/stdc++.h>
using namespace std;
// Function to find the maximum length of equal subarray
int FindMaxLength(int A[], int B[], int n, int m) {
    // Auxiliary dp[] vector
    vector<int> dp(m + 1, 0);
    int maxm = 0;
    // Updating the dp[] vector in Bottom-Up approach
    for (int i = n - 1; i >= 0; i--) {
        int prev = 0;
        for (int j = m - 1; j >= 0; j--) {
            int temp = dp[j];
            if (A[i] == B[j]) {
                dp[j] = prev + 1;
                maxm = max(maxm, dp[j]);
            } else {
                dp[j] = 0;
            }
            prev = temp;
        }
    }
    // Return the maximum length
    return maxm;
}
// Driver Code
int main() {
    int A[] = {1, 2, 8, 2, 1};
    int B[] = {8, 2, 1, 4, 7};
    int n = sizeof(A) / sizeof(A[0]);
    int m = sizeof(B) / sizeof(B[0]);

    // Function call to find maximum length of subarray
    cout << FindMaxLength(A, B, n, m) << endl;

    return 0;
}
```

3.5 Number of Subsequences Having Product at least K

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

const int N = 1010, mod = 1e9 + 7, SQ = sqrt(mod) + 1;
int a[N], k;
int dp1[N][SQ], dp2[N][SQ];

int mul_back(int i, int p) {
    if (i <= 0) return p >= 1;
    int &ret = dp1[i][p];
    if (ret != -1) return ret;
    ret = mul_back(i - 1, p);
    ret += mul_back(i - 1, p / a[i]);
    if (ret >= mod) ret -= mod;
    return ret;
}

int mul_front(int i, int p) {
    if (i <= 0) return p <= k;
    int &ret = dp2[i][p];
    if (ret != -1) return ret;
    ret = mul_front(i - 1, p);
    if (1LL * a[i] * p < SQ) {
        ret += mul_front(i - 1, p * a[i]);
    } else {
        ret += mul_back(i - 1, k / (1LL * p * a[i]));
    }
    if (ret >= mod) ret -= mod;
    return ret;
}

int32_t main() {
    ios_base::sync_with_stdio(0);
    cin.tie(0);

    memset(dp1, -1, sizeof(dp1));
    memset(dp2, -1, sizeof(dp2));

    int n;
    cin >> n >> k;
    --k;

    for (int i = 1; i <= n; i++) {
        cin >> a[i];
    }

    int ans = 1;
    for (int i = 1; i <= n; i++) {
        ans = (ans + ans) % mod;
    }

    cout << (ans - mul_front(n, 1) + mod) % mod << '\n' ;

    return 0;
}
```

4 Graph Algorithms

4.1 Bellman-Ford Algorithm

```
vector<long long> Node[100005], cost[100005];
long long n, m, i, j, cc = 0, k;
long long dis[100005], parent[100005];
long long inf = 1e10;

void bellmenford(long long s, long long f) {
    // Initialization
    for (i = 1; i <= n; i++) {
        if (i == s) dis[i] = 0;
        else dis[i] = inf;
        parent[i] = -1;
    }
    // Relax edges (n - 1) times
    for (i = 1; i < n; i++) {
        bool done = true;
        for (j = 1; j <= n; j++) {
            for (k = 0; k < Node[j].size(); k++) {
                long long u = j;
                long long v = Node[j][k];
                long long uv = cost[j][k];

                if (dis[u] + uv < dis[v]) {
                    dis[v] = dis[u] + uv;
                    parent[v] = u;
                    done = false;
                }
            }
        }
        if (done) break; // Early stopping if no update
    }
    // Detect negative cycle
    for (i = 1; i <= n; i++) {
        for (j = 0; j < Node[i].size(); j++) {
            long long u = i;
            long long v = Node[i][j];
```

```
            long long uv = cost[i][j];

            if (dis[u] + uv < dis[v]) {
                cout << "Found Negative Cycle" << endl;
                return;
            }
        }
    }
    // Print distances
    for (i = 1; i <= n; i++) {
        cout << "NODE: " << i << " distance: " << dis[i
    ] << endl;
    }
}
```

4.2 Diameter of a Graph

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

const int N = 2e5 + 9;

vector<int> g[N];
int farthest(int s, int n, vector<int> &d) {
  static const int inf = N;
  d.assign(n + 1, inf); d[s] = 0;
  vector<bool> vis(n + 1);
  queue<int> q; q.push(s);
  vis[s] = 1; int last = s;
  while (!q.empty()) {
    int u = q.front(); q.pop();
    for (int v: g[u]) {
      if (vis[v]) continue;
      d[v] = d[u] + 1;
      q.push(v); vis[v] = 1;
    }
    last = u;
  }
  return last;
}
int32_t main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  int n; cin >> n;
  for (int i = 1; i < n; i++) {
    int u, v; cin >> u >> v;
    g[u].push_back(v);
    g[v].push_back(u);
  }
  vector<int> dx, dy;
  int x = farthest(1, n, dx);
  int y = farthest(x, n, dx);
  farthest(y, n, dy);
  for (int i = 1; i <= n; i++) {
    cout << max(dx[i], dy[i]) << ' ';
  }
  cout << '\n';
  return 0;
}
// https://cses.fi/problemset/task/1132
```

4.3 Dijkstra's Algorithm

```
vector<long long> dis;
vector<int> parent;
vector<vector<pair<int, int>>> adjlist;
void Dijkstra(int node, int source = 0) {
    dis.assign(node, LLONG_MAX);
    parent.assign(node, -1);
    dis[source] = 0;

    priority_queue<pair<long long, int>> pq;
    pq.push({0, source});

    bitset<100007> processed;

    while (!pq.empty()) {
        int cur_node = pq.top().second;
        pq.pop();

        if (processed[cur_node])
            continue;

        processed[cur_node] = 1;

        for (auto &i : adjlist[cur_node]) {
            int x = i.first;
            long long w = i.second;

            if (dis[cur_node] + w < dis[x]) {
                dis[x] = dis[cur_node] + w;
                parent[x] = cur_node;
                pq.push({-dis[x], x}); // Negative for
    min-heap behavior
            }
        }
    }
}
```

4.4 Floyd-Warshall Algorithm

```
typedef vector<vector<long long>> v1;
v1 Graph;
long long capacity[1000][1000];
long long n, m;

void init(int N) {
    Graph = v1(N + 1);
}

long long bfs(long long s, long long t, vector<long
    long>& parent) {
    fill(parent.begin(), parent.end(), -1);
    parent[s] = -2;

    queue<pair<long long, long long>> q;
    q.push({s, INT_MAX});

    while (!q.empty()) {
        long long u = q.front().first;
        long long flow = q.front().second;
        q.pop();

        for (long long i = 0; i < Graph[u].size(); i++)
    {
            long long v = Graph[u][i];
            if (parent[v] == -1 && capacity[u][v]) {
                parent[v] = u;
                long long new_flow = min(flow, capacity
    [u][v]);
                cout << v << " ";
                if (v == t) return new_flow;
                q.push({v, new_flow});
            }
        }
    }

    return 0;
}

long long max_flow(long long s, long long t) {
    vector<long long> parent(n + 1);
    long long flow = 0;
    long long new_flow;

    while ((new_flow = bfs(s, t, parent))) {
        cout << endl;
        cout << new_flow << endl;
        flow += new_flow;

        long long u = t;
        while (s != u) {
            long long prev = parent[u];
            capacity[prev][u] -= new_flow;
            capacity[u][prev] += new_flow;
            u = prev;
        }
    }

    return flow;
}
```

4.5 Kruskal's Algorithm

```
const int maX = 1e5 + 5;

long long id[maX], nodes, edges;
pair<long long, pair<long long, long long>> p[maX]; //
    {cost, {u, v}}

void initialize() {
    for (int i = 1; i < maX; i++)
```

```
        id[i] = i;
}

long long root(long long x) {
    while (x != id[x]) {
        id[x] = id[id[x]];  // Path compression
        x = id[x];
    }
    return x;
}

void union1(long long x, long long y) {
    long long p = root(x);
    long long q = root(y);
    id[p] = q;
}

long long kruskal(pair<long long, pair<long long, long long>> p[]) {
    long long x, y, cost, minimumCost = 0;
    for (long long i = 0; i < edges; i++) {
        x = p[i].second.first;
        y = p[i].second.second;
        cost = p[i].first;

        if (root(x) != root(y)) {
            minimumCost += cost;
            union1(x, y);
        }
    }
    return minimumCost;
}
```

4.6 Lowest Common Ancestor (LCA)

```
const int LOG = 20;
vector<pair<int, int>> G[N], depth(N, 0);
int up[N][LOG];

void dfs(int a) {
    for (auto it : G[a]) {
        depth[it] = depth[a] + 1;

        // Binary Lifting
        up[it][0] = a;
        for (int j = 1; j < LOG; j++) {
            up[it][j] = up[up[it][j - 1]][j - 1];
        }

        dfs(it);
    }
}
int get_lca(int a, int b) {
    if (depth[a] < depth[b]) swap(a, b);

    int k = depth[a] - depth[b];
    for (int j = LOG - 1; j >= 0; j--) {
        if ((k >> j) & 1)
            a = up[a][j];
    }
    if (a == b) return a;

    for (int j = LOG - 1; j >= 0; j--) {
        if (up[a][j] != up[b][j]) {
            a = up[a][j];
            b = up[b][j];
        }
    }

    return up[a][0];
}
```

4.7 Prim's Algorithm

```
const int maX = 1e5 + 5;
long long nodes, edges;
bool visit[maX];
vector<pair<long long, long long>> adj[maX];

long long prim(long long x) {
    long long minimumCost = 0;

    priority_queue<
        pair<long long, long long>,
        vector<pair<long long, long long>>,
        greater<pair<long long, long long>>
    > Q;

    Q.push({0, x});

    while (!Q.empty()) {
        pair<long long, long long> p = Q.top();
        Q.pop();

        x = p.second;
        if (visit[x]) continue;

        visit[x] = true;
        minimumCost += p.first;

        for (size_t i = 0; i < adj[x].size(); i++) {
            long long y = adj[x][i].second;
            if (!visit[y])
                Q.push(adj[x][i]);
        }
    }

    return minimumCost;
}
```

4.8 Topological Sorting

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

const int N = 1e5 + 9;
vector<int> g[N];
bool vis[N];
vector<int> ord;
void dfs(int u) {
  vis[u] = true;
  for (auto v: g[u]) {
    if (!vis[v]) {
      dfs(v);
    }
  }
  ord.push_back(u);
}
int32_t main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  int n, m; cin >> n >> m;
  while (m--) {
    int u, v; cin >> u >> v;
    g[u].push_back(v);
  }
  for (int i = 1; i <= n; i++) {
    if (!vis[i]) {
      dfs(i);
    }
  }
  reverse(ord.begin(), ord.end());
  // check is feasible
  vector<int> pos(n + 1);
  for (int i = 0; i < (int) ord.size(); i++) {
    pos[ord[i]] = i;
  }
  for (int u = 1; u <= n; u++) {
    for (auto v: g[u]) {
      if (pos[u] > pos[v]) {
        cout << "IMPOSSIBLE\n";
        return 0;
      }
    }
  }

  // print the order
  for (auto u: ord) cout << u << ' ';
  cout << '\n';
  return 0;
}
// https://cses.fi/problemset/task/1679
```

4.9 Warshall's Algorithm

```
long long n, i, j, cc = 0, m, k;
long long adj[100][100];
long long path[100][100];

void floyd_warshall() {
    for (k = 1; k <= n; k++) {
        for (i = 1; i <= n; i++) {
```

```
            for (j = 1; j <= n; j++) {
                if (adj[i][k] + adj[k][j] < adj[i][j])
    {
                    adj[i][j] = adj[i][k] + adj[k][j];
                    path[i][j] = path[i][k];
                }
            }
        }
    }
}
```

5 BIT MAINIPULATION

5.1 All Subset XOR Sum

```
#include <iostream>
using namespace std;

#define ll long long

ll isOn(ll x, ll i) {
    // cout << "X " << i << " " << x << endl;
    return x & (1LL << i);
}

ll flip(ll x, ll i) {
    return x ^ (1LL << i);
}

ll doOff(ll x, ll i) {
    if (isOn(x, i)) return flip(x, i);
    return x;
}

ll power(ll x, ll m) {
    ll res = 1;
    while (m > 0) {
        if (m % 2 == 1) res = res * x;
        x = x * x;
        m /= 2;
    }
    return res;
}

void akam() {
    ll n, i, j, c = 0, x, y, k, ans = 0, sum = 0;
    // cout << "Case " << tst << ": ";
    cin >> n;
    ll a[n], prf[n + 1];

    for (i = 0; i < n; i++) {
        cin >> a[i];
        // sum += a[i];
    }

    // Age array tar prefix xor ber kore
    // nite hobe than all pair xor sum er motoi concept
    prf[0] = 0;
    for (i = 1; i <= n; i++) {
        prf[i] = (prf[i - 1] ^ a[i - 1]);
    }

    prf[0] = 0;
    for (i = 0; i < 32; i++) {
        ll on = 0, off = 0;
        for (j = 0; j <= n; j++) { // shuru korte hbe 0
 theke
            if (isOn(prf[j], i)) on++;
            else off++;
        }
        // cout << on << " " << off << " " << sum <<
 endl;
        sum += (on * off * (1 << i));
    }

    cout << sum << endl;
}

int main() {
    ios_base::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);

    ll tst = 0;
    // test
    akam();
    return 0;
}
```

5.2 Set,reset,check

```
int Set(int N, int pos) { return N = N | (1 << pos); }
int reset(int N, int pos) { return N = N & ~(1 << pos);
     }
bool check(int N, int pos) { return (bool)(N & (1 <<
    pos)); }
```

5.3 Lexicographically Smallest Subsequence with Constraints

```
#include <iostream>
#include <bitset>
#include <algorithm>
using namespace std;

#define ll long long
#define INF 1e18

void akam() {
    ll n, i, j, c = 0, x, y, k, ans = 0, sum = 0;
    // cout << "Case " << tst << ": ";
    cin >> n >> k;

    ll a[k];
    for (i = 0; i < k; i++) cin >> a[i];

    sort(a, a + k);

    for (i = 1; i < (1 << k); i++) {
        bitset<32> b(i);
        ll cnt = 0;
        c = 1;
        // cout << b << endl;

        for (j = 0; j < k; j++) {
            if (b[j] == 1) {
                if (c > (n / a[j])) {
                    c = INF;
                    break;
                }
                c *= a[j];
                cnt++;
            }
        }

        if (c == INF) continue;

        if (cnt & 1) sum += (n / c);
        else sum -= (n / c);
        // cout << sum << endl;
    }
    cout << sum << endl;
}
```

6 String Algorithms

6.1 Knuth-Morris-Pratt (KMP) Algorithm

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

const int N = 3e5 + 9;

// returns the longest proper prefix array of pattern p
// where lps[i]=longest proper prefix which is also
     suffix of p[0...i]
vector<int> build_lps(string p) {
  int sz = p.size();
  vector<int> lps;
  lps.assign(sz + 1, 0);
  int j = 0;
  lps[0] = 0;
  for(int i = 1; i < sz; i++) {
    while(j >= 0 && p[i] != p[j]) {
      if(j >= 1) j = lps[j - 1];
      else j = -1;
    }
    j++;
    lps[i] = j;
  }
```

```
  return lps;
}
vector<int>ans;
// returns matches in vector ans in 0-indexed
void kmp(vector<int> lps, string s, string p) {
  int psz = p.size(), sz = s.size();
  int j = 0;
  for(int i = 0; i < sz; i++) {
    while(j >= 0 && p[j] != s[i])
      if(j >= 1) j = lps[j - 1];
      else j = -1;
    j++;
    if(j == psz) {
      j = lps[j - 1];
      // pattern found in string s at position i-psz+1
      ans.push_back(i - psz + 1);
    }
    // after each loop we have j=longest common suffix
     of s[0..i] which is also prefix of p
  }
}

int main() {
  int i, j, k, n, m, t;
  cin >> t;
  while(t--) {
    string s, p;
    cin >> s >> p;
    vector<int>lps = build_lps(p);
    kmp(lps, s, p);
    if(ans.empty()) cout << "Not Found\n";
    else {
      cout << ans.size() << endl;
      for(auto x : ans) cout << x << ' ';
      cout << endl;
    }
    ans.clear();
    cout << endl;
  }
  return 0;
}
```

6.2 Manacher's Algorithm

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

struct Manacher {
  vector<int> p[2];
  // p[1][i] = (max odd length palindrome centered at i
     ) / 2 [floor division]
  // p[0][i] = same for even, it considers the right
     center
  // e.g. for s = "abbabba", p[1][3] = 3, p[0][2] = 2
  Manacher(string s) {
    int n = s.size();
    p[0].resize(n + 1);
    p[1].resize(n);
    for (int z = 0; z < 2; z++) {
      for (int i = 0, l = 0, r = 0; i < n; i++) {
        int t = r - i + !z;
        if (i < r) p[z][i] = min(t, p[z][l + t]);
        int L = i - p[z][i], R = i + p[z][i] - !z;
        while (L >= 1 && R + 1 < n && s[L - 1] == s[R +
       1])
          p[z][i]++, L--, R++;
        if (R > r) l = L, r = R;
      }
    }
  }
  bool is_palindrome(int l, int r) {
    int mid = (l + r + 1) / 2, len = r - l + 1;
    return 2 * p[len % 2][mid] + len % 2 >= len;
  }
};

int32_t main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  string s; cin >> s;
  Manacher M(s);
  int n = s.size();
  for (int i = 0; i < n; i++) {
    cout << 2 * M.p[1][i] + 1 << ' ';
    if (i + 1 < n) cout << 2 * M.p[0][i + 1] << ' ';
  }
  cout << '\n';
  return 0;
}
// https://judge.yosupo.jp/problem/
     enumerate_palindromes
```

6.3 Maximum Repeating Balanced Substring Range

```
string s;
const int N = 1e6+2;
struct node {
    int open,close,full;
};
node tree[4*N];
node merge(node l, node r) {
    node notun;
    notun.full = l.full+r.full+min(l.open, r.close);
    notun.open = l.open+r.open-min(l.open, r.close);
    notun.close = l.close+r.close-min(l.open, r.close);
    return notun;
}
node query(int ind, int b, int e, int i, int j) {
    if(j<b || e<i) {
        node ans;
        ans.open=0,ans.close=0,ans.full=0;
        return ans;
    }
    if(b>=i && e<=j) return tree[ind];
    int mid = (b+e)>>1;
    return merge(
        query(2*ind+1, b, mid, i, j),
        query(2*ind+2, mid+1, e, i, j)
    );
}
void build(int ind, int b, int e) {
    if(b==e) {
        if(s[b]=='(') tree[ind].open=1,tree[ind].close
     =0,tree[ind].full=0;
        else tree[ind].open=0,tree[ind].close=1,tree[
     ind].full=0;
        return;
    }
    int left = ind*2+1;
    int right = ind*2+2;
    int mid = (b+e)>>1;
    build(left, b,mid);
    build(right, mid+1,e);
    tree[ind]=merge(tree[left],tree[right]);
}
int32_t main() {
    #ifndef ONLINE_JUDGE
    freopen("input.txt", "r", stdin);
    freopen("output.txt", "w", stdout);
    #endif

    cin>>s;
    int n;
    n=s.size();
    build(0,0,n-1);
    node ans = query(0,0,n-1,0,n-1);
    cout<<ans.full*2<<endl;

    return 0;
}
```

6.4 Number of Palindromic Substrings in Range

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

const int N = 1e5 + 9;
vector<int> d1, d2;
void manachers(string &s) {
  int n = s.size();
  d1 = vector<int>(n); // maximum odd length palindrome
     centered at i
  // here d1[i]=the palindrome has d1[i]-1 right
     characters from i
  // e.g. for aba, d1[1]=2;
  for (int i = 0, l = 0, r = -1; i < n; i++) {
    int k = (i > r) ? 1 : min(d1[l + r - i], r - i);
    while (0 <= i - k && i + k < n && s[i - k] == s[i +
     k]) {
      k++;
```

```
    }
    d1[i] = k--;
    if (i + k > r) {
      l = i - k;
      r = i + k;
    }
  }
  d2 = vector<int>(n); // maximum even length palindrome centered at i
  // here d2[i]=the palindrome has d2[i]-1 right characters from i
  // e.g. for abba, d2[2]=2;
  for (int i = 0, l = 0, r = -1; i < n; i++) {
    int k = (i > r) ? 0 : min(d2[l + r - i + 1], r - i + 1);
    while (0 <= i - k - 1 && i + k < n && s[i - k - 1] == s[i + k]) {
      k++;
    }
    d2[i] = k--;
    if (i + k > r) {
      l = i - k - 1;
      r = i + k ;
    }
  }
}
const int MAXN = (int)5100;
const int MAXV = (int)5100;// maximum value of any element in array

// array values can be negative too, use appropriate minimum and maximum value
struct wavelet_tree {
  int lo, hi;
  wavelet_tree *l, *r;
  int *b, *c, bsz, csz; //  c holds the prefix sum of elements

  wavelet_tree() {
    lo = 1;
    hi = 0;
    bsz = 0;
    csz = 0, l = NULL;
    r = NULL;
  }

  void init(int *from, int *to, int x, int y) {
    lo = x, hi = y;
    if(from >= to) return;
    int mid = (lo + hi) >> 1;
    auto f = [mid](int x) {
      return x <= mid;
    };
    b = (int*)malloc((to - from + 2) * sizeof(int));
    bsz = 0;
    b[bsz++] = 0;
    c = (int*)malloc((to - from + 2) * sizeof(int));
    csz = 0;
    c[csz++] = 0;
    for(auto it = from; it != to; it++) {
      b[bsz] = (b[bsz - 1] + f(*it));
      c[csz] = (c[csz - 1] + (*it));
      bsz++;
      csz++;
    }
    if(hi == lo) return;
    auto pivot = stable_partition(from, to, f);
    l = new wavelet_tree();
    l->init(from, pivot, lo, mid);
    r = new wavelet_tree();
    r->init(pivot, to, mid + 1, hi);
  }
  // kth smallest element in [l, r]
  // for array [1,2,1,3,5] 2nd smallest is 1 and 3rd smallest is 2
  int kth(int l, int r, int k) {
    if(l > r) return 0;
    if(lo == hi) return lo;
    int inLeft = b[r] - b[l - 1], lb = b[l - 1], rb = b[r];
    if(k <= inLeft) return this->l->kth(lb + 1, rb, k);
    return this->r->kth(l - lb, r - rb, k - inLeft);
  }
  // count of numbers in [l, r] Less than or equal to k
  int LTE(int l, int r, int k) {
    if(l > r || k < lo) return 0;
    if(hi <= k) return r - l + 1;
    int lb = b[l - 1], rb = b[r];
    return this->l->LTE(lb + 1, rb, k) + this->r->LTE(l - lb, r - rb, k);
  }
  // count of numbers in [l, r] equal to k
  int count(int l, int r, int k) {
    if(l > r || k < lo || k > hi) return 0;
    if(lo == hi) return r - l + 1;
    int lb = b[l - 1], rb = b[r];
    int mid = (lo + hi) >> 1;
    if(k <= mid) return this->l->count(lb + 1, rb, k);
    return this->r->count(l - lb, r - rb, k);
  }
  // sum of numbers in [l ,r] less than or equal to k
  int sum(int l, int r, int k) {
    if(l > r or k < lo) return 0;
    if(hi <= k) return c[r] - c[l - 1];
    int lb = b[l - 1], rb = b[r];
    return this->l->sum(lb + 1, rb, k) + this->r->sum(l - lb, r - rb, k);
  }
  ~wavelet_tree() {
    delete l;
    delete r;
  }
};
int get(int l, int r) {
  return r * (r + 1) / 2 - (l - 1) * l / 2;
}
wavelet_tree oddl, oddr;
int odd(int l, int r) {
  int m = (l + r) / 2;
  int c = 1 - l;
  int less_ = oddl.LTE(l, m, c);
  int ansl = get(l, m) + oddl.sum(l, m, c) + (m - l + 1 - less_) * c;
  c = 1 + r;
  less_ = oddr.LTE(m + 1, r, c);
  int ansr = -get(m + 1, r) + oddr.sum(m + 1, r, c) + (r - m - less_) * c;
  return ansl + ansr;
}
wavelet_tree evenl, evenr;
int even(int l, int r) {
  int m = (l + r) / 2;
  int c = -l;
  int less_ = evenl.LTE(l, m, c);
  int ansl = get(l, m) + evenl.sum(l, m, c) + (m - l + 1 - less_) * c;
  c = 1 + r;
  less_ = evenr.LTE(m + 1, r, c);
  int ansr = -get(m + 1, r) + evenr.sum(m + 1, r, c) + (r - m - less_) * c;
  return ansl + ansr;
}
int a[N], b[N], c[N], d[N];
int sc() {
  int c = getchar();
  int x = 0;
  int neg = 0;
  for(; ((c < 48 || c > 57) && c != '-'); c = getchar());
  if(c == '-') {
    neg = 1;
    c = getchar();
  }
  for(; c > 47 && c < 58; c = getchar()) {
    x = (x << 1) + (x << 3) + c - 48;
  }
  if(neg) x = -x;
  return x;
}
inline void out(int n) {
  int N = n < 0 ? -n : n, rev, cnt = 0;
  rev = N;
  if (N == 0) {
    putchar('0');
    putchar('\n');
    return ;
  }
  while ((rev % 10) == 0) {
    cnt++;
    rev /= 10;
  }
```

```
  if(n < 0) putchar('-');
  rev = 0;
  while (N != 0) {
    rev = (rev << 3) + (rev << 1) + N % 10;
    N /= 10;
  }
  while (rev != 0) {
    putchar(rev % 10 + '0');
    rev /= 10;
  }
  while (cnt--) putchar('0');
  putchar('\n');
  return;
}
int main() {
  int i, j, k, n, m, q, l, r;
  string s;
  cin >> s;
  n = s.size();
  manachers(s);
  for(i = 1; i <= n; i++) a[i] = d1[i - 1] - i;
  oddl.init(a + 1, a + n + 1, -MAXV, MAXV);
  for(i = 1; i <= n; i++) b[i] = d1[i - 1] + i;
  oddr.init(b + 1, b + n + 1, -MAXV, MAXV);
  for(i = 1; i <= n; i++) c[i] = d2[i - 1] - i;
  evenl.init(c + 1, c + n + 1, -MAXV, MAXV);
  for(i = 1; i <= n; i++) d[i] = d2[i - 1] + i;
  evenr.init(d + 1, d + n + 1, -MAXV, MAXV);

  q = sc();
  for(i = 0; i < q; i++) {
    l = sc();
    r = sc();
    out(odd(l, r) + even(l, r));
  }
  return 0;
}
```

6.5 String Divisibility

```
string longDivision(string number, int divisor) {
    // As result can be very large, store it in a
     string
    string ans;

    // Find prefix of number that is larger than
     divisor
    int idx = 0;
    int temp = number[idx] - '0';
    while (idx < (number.size() - 1) && temp < divisor)
        temp = temp * 10 + (number[++idx] - '0');

    // Repeatedly divide divisor with temp
    // After every division, update temp to include one
      more digit
    while (idx < number.size() - 1) {
        // Store result in answer i.e. temp / divisor
        ans += (temp / divisor) + '0';

        // Take next digit of number
        temp = (temp % divisor) * 10 + (number[++idx] -
      '0');
    }
    ans += (temp / divisor) + '0';
    // If divisor is greater than number
    if (ans.length() == 0)
        return "0";

    // Else return ans
    return ans;
}
```

6.6 String Hashing

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

const int N = 1e6 + 9;

int power(long long n, long long k, const int mod) {
  int ans = 1 % mod;
  n %= mod;
  if (n < 0) n += mod;
  while (k) {
    if (k & 1) ans = (long long) ans * n % mod;
    n = (long long) n * n % mod;
    k >>= 1;
  }
  return ans;
}

const int MOD1 = 127657753, MOD2 = 987654319;
const int p1 = 137, p2 = 277;
int ip1, ip2;
pair<int, int> pw[N], ipw[N];
void prec() {
  pw[0] =  {1, 1};
  for (int i = 1; i < N; i++) {
    pw[i].first = 1LL * pw[i - 1].first * p1 % MOD1;
    pw[i].second = 1LL * pw[i - 1].second * p2 % MOD2;
  }
  ip1 = power(p1, MOD1 - 2, MOD1);
  ip2 = power(p2, MOD2 - 2, MOD2);
  ipw[0] =  {1, 1};
  for (int i = 1; i < N; i++) {
    ipw[i].first = 1LL * ipw[i - 1].first * ip1 % MOD1;
    ipw[i].second = 1LL * ipw[i - 1].second * ip2 %
     MOD2;
  }

}
struct Hashing {
  int n;
  string s; // 0 - indexed
  vector<pair<int, int>> hs; // 1 - indexed
  Hashing() {}
  Hashing(string _s) {
    n = _s.size();
    s = _s;
    hs.emplace_back(0, 0);
    for (int i = 0; i < n; i++) {
      pair<int, int> p;
      p.first = (hs[i].first + 1LL * pw[i].first * s[i]
      % MOD1) % MOD1;
      p.second = (hs[i].second + 1LL * pw[i].second * s
     [i] % MOD2) % MOD2;
      hs.push_back(p);
    }
  }
  pair<int, int> get_hash(int l, int r) { // 1 -
     indexed
    assert(1 <= l && l <= r && r <= n);
    pair<int, int> ans;
    ans.first = (hs[r].first - hs[l - 1].first + MOD1)
     * 1LL * ipw[l - 1].first % MOD1;
    ans.second = (hs[r].second - hs[l - 1].second +
     MOD2) * 1LL * ipw[l - 1].second % MOD2;
    return ans;
  }
  pair<int, int> get_hash() {
    return get_hash(1, n);
  }
};
int32_t main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  prec();
  int n;
  while (cin >> n) {
    string s, p;
    cin >> p >> s;
    Hashing h(s);
    auto hs = Hashing(p).get_hash();
    for(int i = 1; i + n - 1 <= s.size(); i++) {
      if (h.get_hash(i, i + n - 1) == hs) cout << i - 1
      << '\n';
    }
    cout << '\n';
  }
  return 0;
}
```

6.7 Large Number String Multiplication

```
string multiply(string a, int b) {
    int carry = 0;
    string ans = "";

    for (int i = a.size() - 1; i >= 0; i--) {
        carry = (a[i] - '0') * b + carry;
        ans += (carry % 10) + '0';
```

```
        carry /= 10;
    }

    while (carry != 0) {
        ans += (carry % 10) + '0';
        carry /= 10;
    }

    reverse(ans.begin(), ans.end()); // To correct the
     reversed order
    return ans;
}
```

6.8 Z-Algorithm (Z-Function)

```
#include<bits/stdc++.h>
using namespace std;
// An element Z[i] of Z array stores length of the
     longest substring
// starting from str[i] which is also a prefix of str
     [0..n-1].
// The first entry of Z array is meaning less as
     complete string is always prefix of itself.
// Here Z[0]=0.
vector<int> z_function(string s) {
  int n = (int) s.length();
  vector<int> z(n);
  for (int i = 1, l = 0, r = 0; i < n; ++i) {
    if (i <= r)
      z[i] = min (r - i + 1, z[i - l]);
    while (i + z[i] < n && s[z[i]] == s[i + z[i]])
      ++z[i];
    if (i + z[i] - 1 > r)
      l = i, r = i + z[i] - 1;
  }
  return z;
}
int32_t main() {
  string s;
  cin >> s;
  vector<int>  ans = z_function(s);
  for(auto x : ans) cout << x << ' ';
  return 0;
}
```

7 Stress Testing

7.1 Main Stress Test Driver

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

7.2 Stress Test for Linux

```
#!/bin/bash

set -e

g++ code.cpp -o code
g++ gen.cpp -o gen
g++ brute.cpp -o brute

for ((i = 1; ; ++i)); do
    ./gen $i > input_file
    ./code < input_file > myAnswer
    ./brute < input_file > correctAnswer

    diff -Z myAnswer correctAnswer > /dev/null || break

    echo "Passed test: $i"
done

echo "WA on the following test:"
cat input_file
echo "Your answer is:"
cat myAnswer
echo "Correct answer is:"
cat correctAnswer
```

7.3 Stress Test for Windows

```
@echo off

if [%1] == [] (
    set /A numLoop = 100
) else (
    set /A numLoop = %1
)

if [%2] == [] (
    set /A doComp = 1
) else (
    set /A doComp = %2
)

if %doComp% equ 1 (
    echo Compiling solution, gen, brute ...
    g++ -std=c++17 gen.cpp -o gen
    g++ -std=c++17 solution.cpp -o solution
    g++ -std=c++17 brute.cpp -o brute
    echo Done compiling.
)

set "diff_found="

for /l %%x in (1, 1, %numLoop%) do (
    echo %%x
    gen > input.in
    solution < input.in > output.out
    brute < input.in > output2.out

    rem Add \f after "fc" to ignore trailing
     whitespaces and convert multiple whitespaces into
     one space
    fc output.out output2.out > diagnostics

    if errorlevel 1 (
        set "diff_found=y"
        goto :break
    )
)

:break

if defined diff_found (
    echo A difference has been found.
    echo Input:
    type input.in
    echo.
    echo Output:
    type output.out
    echo.
    echo Expected:
    type output2.out
    echo.
) else (
    echo All tests passed :D
)

del input.in
del output.out
del output2.out
```

8 Special Topics

8.1 Maximum Sum Subarray

```
struct {
    ll ls = 0, rs = 0, s = 0, ms = 0;
} tree[N * 4];

struct Tr {
    ll ls = INT_MIN, rs = INT_MIN, s = INT_MIN, ms =
     INT_MIN;
};

ll a[N];

void build(ll nd, ll b, ll e) {
    if (b == e) {
        tree[nd].s = a[b];
        tree[nd].ls = a[b];
        tree[nd].rs = a[b];
        tree[nd].ms = a[b];
        return;
    }

```

```
    ll mid = (b + e) / 2;
    build(lf, b, mid);
    build(rt, mid + 1, e);

    tree[nd].s = tree[lf].s + tree[rt].s;
    tree[nd].ls = max(tree[lf].s + tree[rt].ls, tree[lf].ls);
    tree[nd].rs = max(tree[lf].rs + tree[rt].s, tree[rt].rs);
    tree[nd].ms = max({tree[lf].rs + tree[rt].ls, tree[lf].ms, tree[rt].ms});
}

void update(ll nd, ll b, ll e, ll l, ll r, ll val) {
    if (b > r || l > e) return;

    if (b >= l && e <= r) {
        tree[nd].s = val;
        tree[nd].ls = val;
        tree[nd].rs = val;
        tree[nd].ms = val;
        return;
    }

    ll mid = (b + e) / 2;
    update(lf, b, mid, l, r, val);
    update(rt, mid + 1, e, l, r, val);

    tree[nd].s = tree[lf].s + tree[rt].s;
    tree[nd].ls = max(tree[lf].s + tree[rt].ls, tree[lf].ls);
    tree[nd].rs = max(tree[lf].rs + tree[rt].s, tree[rt].rs);
    tree[nd].ms = max({tree[lf].rs + tree[rt].ls, tree[lf].ms, tree[rt].ms});
}

Tr query(ll nd, ll b, ll e, ll l, ll r) {
    if (b > r || l > e) {
        Tr tr;
        return tr;
    }

    if (b >= l && e <= r) {
        Tr tr;
        tr.s = tree[nd].s;
        tr.ls = tree[nd].ls;
        tr.rs = tree[nd].rs;
        tr.ms = tree[nd].ms;
        return tr;
    }

    ll mid = (b + e) / 2;
    Tr trr1, trr2, trr3;
    trr1 = query(lf, b, mid, l, r);
    trr2 = query(rt, mid + 1, e, l, r);

    trr3.s = trr1.s + trr2.s;
    trr3.ls = max({trr1.ls, trr1.s + trr2.ls});
    trr3.rs = max({trr2.rs, trr2.s + trr1.rs});
    trr3.ms = max({trr1.ms, trr2.ms, trr1.rs + trr2.ls});

    return trr3;
}

void akam() {
    ll n, i, j, c = 0, x, y, k, ans = 0, sum = 0;
    cin >> n;

    for (i = 1; i <= n; i++) cin >> a[i];

    build(1, 1, n);
    cin >> k;

    for (i = 0; i < k; i++) {
        cin >> x;
        if (x == 0) {
            cin >> x >> y;
            update(1, 1, n, x, x, y);
        } else {
            cin >> x >> y;
            Tr tr = query(1, 1, n, x, y);
            cout << tr.ms << endl;
        }
    }
}

int main() {
    ios_base::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);

    akam();
    return 0;
}
```

8.2 Nth Fibonacci Number (Matrix Exponentiation)

```
ll fib(ll n) {
    if (n == 1) return 0;
    if (n == 2) return 1;

    ll b = n - 2;
    ll x, y, z, w;
    ll f[2][2] = {{1, 1}, {1, 0}};
    ll r[2][2] = {{1, 0}, {0, 1}};

    if (b < 0) {
        return 0;
    }

    while (b > 0) {
        if (b & 1) {
            x = ((r[0][0] * f[0][0]) % MAX + (r[0][1] * f[1][0]) % MAX) % MAX;
            y = ((r[0][0] * f[0][1]) % MAX + (r[0][1] * f[1][1]) % MAX) % MAX;
            w = ((r[1][0] * f[0][0]) % MAX + (r[1][1] * f[1][0]) % MAX) % MAX;
            z = ((r[1][0] * f[0][1]) % MAX + (r[1][1] * f[1][1]) % MAX) % MAX;

            r[0][0] = x;
            r[0][1] = y;
            r[1][0] = w;
            r[1][1] = z;
        }

        x = ((f[0][0] * f[0][0]) % MAX + (f[0][1] * f[1][0]) % MAX) % MAX;
        y = ((f[0][0] * f[0][1]) % MAX + (f[0][1] * f[1][1]) % MAX) % MAX;
        w = ((f[1][0] * f[0][0]) % MAX + (f[1][1] * f[1][0]) % MAX) % MAX;
        z = ((f[1][0] * f[0][1]) % MAX + (f[1][1] * f[1][1]) % MAX) % MAX;

        f[0][0] = x;
        f[0][1] = y;
        f[1][0] = w;
        f[1][1] = z;

        b >>= 1; // Equivalent to b = b / 2
    }

    return r[0][0];
}
```

9 Some Properties

9 Techniques of Number Theory

Some Properties / Techniques of Number Theory

1. Coprime Counts from 1 to $N \times M$ where $\gcd(N, M) = 1$

- Numbers coprime with N but not with M:

$$\text{Ans}_1 = (\phi(N) \times M) - \phi(N \times M)$$

- Numbers coprime with M but not with N:

$$\text{Ans}_2 = (\phi(M) \times N) - \phi(N \times M)$$

- Numbers coprime with both N and M:

$$\text{Ans}_3 = \phi(N \times M)$$

2. Sum of Divisors from 1 to 2×10^9

$$\text{Ans} = \sum_{i=1}^{n} \text{sod}(i)$$

Code:

```
long long sum_all_divisors(long long num) {
    long long sum = 0;
    for (long long i = 1; i <= sqrt(num); i++) {
        long long t1 = i * (num / i - i + 1);
        long long t2 = (((num / i) * (num / i + 1)) / 2)
                     - ((i * (i + 1)) / 2);
        sum += t1 + t2;
    }
    return sum;
}
```

Listing 1: Efficient sum of divisors from 1 to n

3. Coprime Symmetry Property

$$\text{If } \gcd(x, n) = 1 \text{ then } \gcd(n - x, n) = 1$$

4. Base Conversion Using Logarithms

$$\log_k(\text{number}) = \frac{\log_{10}(\text{number})}{\log_{10}(k)}$$

This identity is useful for converting a number from decimal to base k.

9.1 Combinatorics

Combinatorics Identities

1. $\displaystyle\sum_{0 \le k \le n} \binom{n-k}{k} = F_{n+1}$ (Fibonacci)
2. $\displaystyle\binom{n}{k} = \binom{n}{n-k}$
3. $\displaystyle\binom{n}{k} + \binom{n}{k+1} = \binom{n+1}{k+1}$
4. $\displaystyle k \cdot \binom{n}{k} = n \cdot \binom{n-1}{k-1}$
5. $\displaystyle\binom{n}{k} = \frac{n}{k} \cdot \binom{n-1}{k-1}$
6. $\displaystyle\sum_{i=0}^{n} \binom{n}{i} = 2^n$
7. $\displaystyle\sum_{i \ge 0} \binom{n}{2i} = 2^{n-1}$
8. $\displaystyle\sum_{i \ge 0} \binom{n}{2i+1} = 2^{n-1}$
9. $\displaystyle\sum_{i=0}^{k} (-1)^i \binom{n}{i} = (-1)^k \binom{n-1}{k}$
10. $\displaystyle\sum_{i=0}^{k} \binom{n+i}{i} = \sum_{i=0}^{k} \binom{n+i}{n} = \binom{n+k+1}{k}$
11. $\displaystyle 1 \cdot \binom{n}{1} + 2 \cdot \binom{n}{2} + \cdots + n \cdot \binom{n}{n} = n \cdot 2^{n-1}$
12. $\displaystyle 1^2 \binom{n}{1} + 2^2 \binom{n}{2} + \cdots + n^2 \binom{n}{n} = (n + n^2) \cdot 2^{n-2}$
13. $\displaystyle\sum_{k=0}^{r} \binom{m}{k} \binom{n}{r-k} = \binom{m+n}{r}$ (Vandermonde's Identity)
14. $\displaystyle\sum_{i=r}^{n} \binom{i}{r} = \binom{n+1}{r+1}$ (Hockey-Stick Identity)
15. $\displaystyle\sum_{i=0}^{k} \binom{k}{i}^2 = \binom{2k}{k}$
16. $\displaystyle\sum_{k=0}^{n} \binom{n}{k} \binom{n}{n-k} = \binom{2n}{n}$
17. $\displaystyle\sum_{k=q}^{n} \binom{n}{k} \binom{k}{q} = 2^{n-q} \binom{n}{q}$
18. $\displaystyle\sum_{k}^{n} k^i \binom{n}{k} = (k+1)^n$

9.2 Pascals Triangle

article amsmath, amssymb

Pascal's Triangle and Related Theorems

- In a row p where p is a prime number, all the terms in that row except the 1s are multiples of p.
- **Parity:** To count odd terms in row n, convert n to binary. Let x be the number of 1s in the binary representation. Then the number of odd terms will be 2^x.
- Every entry in row $2^n - 1$, $n \ge 0$, is odd.
- An integer $n \ge 2$ is prime if and only if all the intermediate binomial coefficients

$$\binom{n}{1}, \binom{n}{2}, \ldots, \binom{n}{n-1}$$

  are divisible by n.

- **Kummer's Theorem:** For given integers $n \ge m \ge 0$ and a prime number p, the largest power of p dividing $\binom{n}{m}$ is equal to the number of carries when m is added to $n - m$ in base p. For implementation, take inspiration from Lucas' Theorem.
- Number of different binary sequences of length n such that no two 0s are adjacent is F_{n+1} (Fibonacci number).
- **Combination with repetition:** Choosing k elements from an n-element set, where order doesn't matter and each element can be chosen more than once:

$$\binom{n+k-1}{k}$$

- Number of ways to divide n persons into $\frac{n}{k}$ equal groups (each of size k):

$$\frac{n!}{k!^{n/k}\left(\frac{n}{k}\right)!} \quad \text{(valid when } n \geq k \text{ and } k \mid n\text{)}$$

- The number of non-negative integer solutions of the equation:

$$x_1 + x_2 + \cdots + x_k = n$$

is

$$\binom{n+k-1}{n}$$

- Number of ways to choose n IDs from 1 to b such that every ID has distance at least k:

$$\binom{b-(n-1)(k-1)}{n}$$

-

$$\sum_{\substack{i=1 \\ i \text{ odd}}}^{n} \binom{n}{i} a^{n-i} b^i = \frac{1}{2}\left((a+b)^n - (a-b)^n\right)$$

-

$$\sum_{i=0}^{n} \binom{k}{i}\binom{n}{i} = \binom{n+k}{n}$$

- **Derangement:** A permutation of a set where no element appears in its original position. Let $d(n)$ be the number of derangements of size n:

$d(n) = (n-1)(d(n-1)+d(n-2)), \quad d(0) = 1, d(1) = 0$

- **Involutions:** Permutations such that $p^2 = \text{identity}$. Let a_n be the number of involutions:

$a_0 = a_1 = 1, \quad a_n = a_{n-1}+(n-1)a_{n-2}, \quad \text{for } n > 1$

- Let $T(n,k)$ be the number of permutations of size n for which all cycles have length $\leq k$:

$T(n,k) = n \cdot T(n-1,k) - F(n-1,k) \cdot T(n-k,k)$

where

$$F(n,k) = n(n-1)\ldots(n-k+1)$$