1 CSES

1.1 CountingTilings

```
//Your task is to count the number of ways you can fill an n×m grid using 1×2 and 2×1 tiles.
int dp[1005][1<<10] = {};
vector<pii>v;
void solve(){
  int n,m;
  cin>>n>>m;
  for(int a = 0;a<(1<<n);++a){
    for(int b = 0;b<(1<<n);++b){
      bool flag = 1;
      for(int i = 0;i<n;++i){
        if(a&(1<<i) and b&(1<<i)){
          if(i==n-1 or !(a&(1<<(i+1))) or !(b&(1<<(i+1))))flag = 0;
          else{
            i++;
            continue;
          }
        }
        if(!(a&(1<<i)) and !(b&(1<<i)))flag = 0;
      }
      if(flag)v.pb({a,b});
    }
  };
  dp[0][(1<<n)-1] = 1;
  for(int i = 1;i<=m;++i){
    for(auto j:v)dp[i][j.S] = (1ll*dp[i-1][j.F]+dp[i][j.S])%mod;
  }
  cout<<dp[m][(1<<n)-1]<<endl;
}
signed main(){
  IOS;
  solve();
}
```

1.2 Sequence1

```
//0, 1, 2, 9, 44, 265, 1854, 14833, 133496, 1334961
#include <bits/stdc++.h>

#pragma GCC optimize("Ofast,unroll-loops,no-stack-protector,fast-math")

using namespace std;

#define fastio ios::sync_with_stdio(false), cin.tie(NULL), cout.tie(NULL)

typedef uint64_t ull;

const int mod = 1e9 + 7, mxN = 1e6 + 1;

int n;
ull dp[mxN];

int main() {
  fastio;
  dp[1] = 0;
  dp[2] = 1;
  for(int i = 3; i < mxN; ++i)
    dp[i] = (i - 1) * (dp[i - 1] + dp[i - 2]) % mod;
  cin >> n;
  cout << dp[n] << "\n";

  return 0;
}
```

1.3 JosephusQueries

```
/*
Consider a game where there are n children (numbered 1,2,…,n) in a circle. During the game, every second child is removed from the circle, until there are no children left.

Your task is to process q queries of the form: "when there are n children, who is the kth child that will be removed?"
*/
#include <bits/stdc++.h>
using namespace std;

int f(int n, int k) {
  if(n == 1) {
    return 0;
  }
  if(k * 2 <= n) {
    return k * 2 - 1;
  }
  int pos = f(n - n / 2, k - n / 2);
  if(pos == 0) {
    return (n % 2 == 1 ? n - 1 : 0);
  }
  return (pos - n % 2) * 2;
}

int main() {
  ios::sync_with_stdio(false);
  cin.tie(0);
  int tt;
  cin >> tt;
  while(tt--) {
    int n, k;
    cin >> n >> k;
    cout << f(n, k) + 1 << "\n";
  }
  return 0;
}
```

1.4 AnotherGame

```
/*
There are n heaps of coins and two players who move alternately. On each move, a player selects some of the nonempty heaps and removes one coin from each heap. The player who removes the last coin wins the game.

Your task is to find out who wins if both players play optimally.
*/
#include <bits/stdc++.h>
using namespace std;

int main() {
  ios::sync_with_stdio(false);
  cin.tie(0);
  int tt;
  cin >> tt;
  while(tt--) {
    int n;
    cin >> n;
    bool b = false;
    for(int i = 0; i < n; ++i) {
      int x;
      cin >> x;
      b = (b || x % 2);
    }
    cout << (b ? "first" : "second") << "\n";
  }
  return 0;
}
```

1.5 CountingCoprimePairs

```
/*
Given a list of n positive integers, your task is to count the number of pairs of integers that are coprime (i.e., their greatest common divisor is one).
*/
#include <bits/stdc++.h>
using namespace std;

const int N = 1e6 + 5;

int main(int argc, char* argv[]) {
  ios::sync_with_stdio(false);
  cin.tie(0);
  vector<bool> isprime(N + 1, true);
  isprime[0] = isprime[1] = false;
  vector<int> prime;
  vector<int> mu(N + 1);
  mu[1] = 1;
  for(int i = 2; i <= N; ++i) {
    if(isprime[i]) {
      mu[i] = -1;
      prime.push_back(i);
    }
    for(int j = 0; j < (int) prime.size() && i * prime[j] <= N; ++j) {
      isprime[i * prime[j]] = false;
      mu[i * prime[j]] = mu[i] * mu[prime[j]];
      if(i % prime[j] == 0) {
        mu[i * prime[j]] = 0;
        break;
      }
    }
```

```cpp
  }
  int n;
  cin >> n;
  vector<int> cnt(N + 1);
  for(int i = 0; i < n; ++i) {
    int x;
    cin >> x;
    cnt[x] += 1;
  }
  long long ans = 0;
  for(int i = 1; i <= N; ++i) {
    long long s = 0;
    for(int j = i; j <= N; j += i) {
      s += cnt[j];
    }
    ans += 1LL * mu[i] * s * (s - 1) / 2;
  }
  cout << ans << "\n";
  return 0;
}
```

1.6 Sequence2

```cpp
//1, 2, 5, 14, 42, 132, 429, 1430, 4862, 16796
#include <bits/stdc++.h>

#pragma GCC optimize("Ofast,unroll-loops,no-stack-protector,fast-math")

using namespace std;

#define fastio ios::sync_with_stdio(false), cin.tie(NULL), cout.tie(NULL)

typedef uint64_t ull;

ull FastPower(ull a, ull b, ull m) {
  a %= m;
  ull ans = 1;
  while(b) {
    if(b & 1)
      ans = ans * a % m;
    a = a * a % m;
    b >>= 1;
  }
  return ans;
}

const int mod = 1e9 + 7, mxN = 2e6 + 1;

ull n, f[mxN];

int main() {
  fastio;
  f[0] = 1;
  for(int i = 1; i < mxN; ++i)
    f[i] = f[i - 1] * i % mod;
  cin >> n;
  if(n & 1) {
    cout << "0\n";
    return 0;
  }
  n >>= 1;
  ull temp = FastPower(f[n], mod - 2, mod);
  cout << f[n << 1] * temp % mod * temp % mod * FastPower(n + 1, mod - 2, mod) % mod << "\n";

  return 0;
}
```

1.7 LongestPalindrome

```cpp
/*
Problem Name: Longest Palindrome
Problem Link: https://cses.fi/problemset/task/1111
Author: Sachin Srivastava (mrsac7)
*/
#include<bits/stdc++.h>
using namespace std;

template<typename... T>
#define error(args...) { string _s = #args; replace(_s.begin(), _s.end(), ',', ' '); stringstream _ss(_s); istream_iterator<string> _it(_ss); err(_it, args); }
void err(istream_iterator<string> it) {}
template<typename T, typename... Args>
void err(istream_iterator<string> it, T a, Args... args) {cerr << *it << "=" << a << ", "; err(++it, args...);}

#define int long long
#define F first
#define S second

const long long inf = 1LL<<62;
const int md = 1000000007;

void solve(){
    string s; cin>>s;
    int n = s.size();
    int dp[n][2] = {0};
    int x1 = 0, y1 = -1;
    int x2 = 0, y2 = -1;
    int mx = 0, ans = 0;
    for (int i = 0; i < n; i++) {
        int k = 0;
        if (i>y1) k = 1;
        else k = min(dp[x1+y1-i][0], y1-i+1);
        while (0<=i-k && i+k<n && s[i-k] == s[i+k]) k++;
        dp[i][0] = k--;
        if (i+k>y1) x1 = i-k, y1 = i+k;
        if (2*dp[i][0] - 1 > mx) ans = i-k, mx = 2*dp[i][0] - 1;
        k = 0;
        if (i<=y2) k = min(dp[x2+y2-i+1][1],y2-i+1);
        while(0<=i-k-1 && i+k<n && s[i-k-1] == s[i+k]) k++;
        dp[i][1] = k--;
        if (i+k>y2) x2 = i-k-1, y2 = i+k;
        if (2*dp[i][1] > mx) ans = i-k-1, mx = 2*dp[i][1];
    }
    cout<<s.substr(ans,mx);
}
signed main(){
    ios_base::sync_with_stdio(false);cin.tie(0);cout.tie(0);
    #ifdef LOCAL
    freopen("input.txt", "r" , stdin);
    freopen("output.txt", "w", stdout);
    #endif
    int t=1;
    //cin>>t;
    for (int i = 1; i <= t; i++) {
        solve();
        cout<<'\n';
    }
}
```

1.8 DistinctSubstrings

```cpp
//Count the number of distinct substrings that appear in a string.
//abaa => 8 : Explanation: the substrings are a, b, aa, ab, ba, aba, baa and abaa.
#include <bits/extc++.h>
#include <bits/stdc++.h>
#pragma gcc optimize("ofast, unroll-loops, no-stack-protector, fast-math")
#define IOS ios::sync_with_stdio(0),cin.tie(0),cout.tie(0)
#define int long long
#define double long double
#define pb push_back
#define sz(x) (int)(x).size()
#define all(v) begin(v),end(v)
#define debug(x) cerr<<#x<<" = "<<x<<'\n'
#define LINE cout<<"\n-----------------\n"
#define endl '\n'
#define VI vector<int>
#define F first
#define S second
#define MP(a,b) make_pair(a,b)
#define rep(i,m,n) for(int i = m;i<=n;++i)
#define res(i,m,n) for(int i = m;i>=n;--i)
#define gcd(a,b) __gcd(a,b)
#define lcm(a,b) a*b/gcd(a,b)
#define Case() int _;cin>>_;for(int Case = 1;Case<=_;++Case)
#define pii pair<int,int>
using namespace __gnu_cxx;
using namespace __gnu_pbds;
using namespace std;
template <typename K, typename cmp = less<K>, typename T = thin_heap_tag> using _heap = __gnu_pbds::priority_queue<K, cmp, T>;
template <typename K, typename M = null_type> using _hash = gp_hash_table<K, M>;
const int N = 1e6+5,L = 20,mod = 1e9+7;
const long long inf = 2e18+5;
const double eps = 1e-7,pi = acos(-1);
mt19937 mt(std::chrono::system_clock::now().time_since_epoch().count());
struct suffix_array{
  int n;
  vector<int>SA,Rank,LCP;
  void counting_sort(vector<int>&v,auto getkey){
    int n = 0;
    for(auto i:v)n = max(n,getkey(i)+1);
    vector<int>bucket(n),ans(v.size());
    for(auto i:v)++bucket[getkey(i)];
    partial_sum(begin(bucket),end(bucket),begin(bucket));
    for(auto ite = v.rbegin();ite!=v.rend();++ite)ans[--bucket[getkey(*ite)]] = move(*ite);
    v.swap(ans);
    return;
  }
  suffix_array(string s):n(s.size()){
    SA.resize(n),Rank.resize(n),LCP.resize(n);
    for(int i = 0;i<n;++i)SA[i] = i;
    sort(SA.begin(),SA.end(),[&](int a,int b){
      return s[a]<s[b];
    });
    for(int i = 0;i<n;++i){
      Rank[SA[i]] = (i?Rank[SA[i-1]]+(s[SA[i]]!=s[SA[i-1]]):SA[0]);
    }
    for(int k = 0;(1<<k)<=n;++k){
      vector<int>idx;
      for(int i = n-(1<<k);i<n;++i)idx.push_back(i);
      for(auto i:SA)if(i>=(1<<k))idx.push_back(i-(1<<k));
      counting_sort(idx,[&](int a){return Rank[a];});
      SA.swap(idx);
      vector<int>new_rank(n);
      new_rank[SA[0]] = 0;
      for(int i = 1;i<n;++i){
        auto cmp = [&](int a,int b){
          return Rank[a]!=Rank[b] or a+(1<<k)>=n or Rank[a+(1<<k)]!=Rank[b+(1<<k)];
        };
        new_rank[SA[i]] = new_rank[SA[i-1]]+cmp(SA[i-1],SA[i]);
      }
      Rank.swap(new_rank);
    }
    for(int i = 0,k = 0;i<n;++i){
      if(Rank[i]==0)continue;
      if(k)--k;
      while(i+k<n and SA[Rank[i]-1]+k<n and s[i+k]==s[SA[Rank[i]-1]+k])++k;
      LCP[Rank[i]] = k;
    }
  }
};
void solve(){
  string s;
  getline(cin,s);
  suffix_array sa(s);
  int n = s.size();
  int ans = n*(n+1)/2;
  for(int i = 1;i<n;++i)ans-=sa.LCP[i];
  cout<<ans<<endl;
}
signed main(){
  IOS;
  solve();
}
```

1.9 BracketSequencesII

```
//Your task is to calculate the number of valid bracket sequences of length n when a prefix of the sequence is given.
#include <bits/stdc++.h>


#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>

#ifdef _MSC_VER
#include <intrin.h>
#endif



namespace felix {

namespace internal {

// @param m `1 <= m`
// @return x mod m
constexpr long long safe_mod(long long x, long long m) {
  x %= m;
  if(x < 0) {
    x += m;
  }
  return x;
}

// Fast modular multiplication by barrett reduction
// Reference: https://en.wikipedia.org/wiki/Barrett_reduction
// NOTE: reconsider after Ice Lake
class barrett {
public:
  unsigned int m;
  unsigned long long im;

  // @param m `1 <= m < 2^31`
  explicit barrett(unsigned int _m) : m(_m), im((unsigned long long)(-1) / _m + 1) {}

  // @return m
  unsigned int umod() const { return m; }

  // @param a `0 <= a < m`
  // @param b `0 <= b < m`
  // @return `a * b % m`
  unsigned int mul(unsigned int a, unsigned int b) const {
    // [1] m = 1
    // a = b = im = 0, so okay

    // [2] m >= 2
    // im = ceil(2^64 / m)
    // -> im * m = 2^64 + r (0 <= r < m)
    // let z = a*b = c*m + d (0 <= c, d < m)
    // a*b * im = (c*m + d) * im = c*(im*m) + d*im = c*2^64 + c*r + d*im
    // c*r + d*im < m * m + m * im < m * m + 2^64 + m <= 2^64 + m * (m + 1) < 2^64 * 2
    // ((ab * im) >> 64) == c or c + 1
    unsigned long long z = a;
    z *= b;
#ifdef _MSC_VER
    unsigned long long x;
    _umul128(z, im, &x);
#else
    unsigned long long x = (unsigned long long)(((unsigned __int128)(z) * im) >> 64);
#endif
    unsigned int v = (unsigned int)(z - x * m);
    if(m <= v) {
      v += m;
    }
    return v;
  }
};

// @param n `0 <= n`
// @param m `1 <= m`
// @return `(x ** n) % m`
constexpr long long pow_mod_constexpr(long long x, long long n, int m) {
    if(m == 1) return 0;
    unsigned int _m = (unsigned int)(m);
    unsigned long long r = 1;
    unsigned long long y = safe_mod(x, m);
    while(n) {
        if (n & 1) r = (r * y) % _m;
        y = (y * y) % _m;
        n >>= 1;
    }
    return r;
}

// Reference:
// M. Forisek and J. Jancina,
// Fast Primality Testing for Integers That Fit into a Machine Word
// @param n `0 <= n`
constexpr bool is_prime_constexpr(int n) {
    if(n <= 1) return false;
    if(n == 2 || n == 7 || n == 61) return true;
    if(n % 2 == 0) return false;
    long long d = n - 1;
    while(d % 2 == 0) d /= 2;
    constexpr long long bases[3] = {2, 7, 61};
    for(long long a : bases) {
        long long t = d;
        long long y = pow_mod_constexpr(a, t, n);
        while(t != n - 1 && y != 1 && y != n - 1) {
            y = y * y % n;
            t <<= 1;
        }
        if(y != n - 1 && t % 2 == 0) {
            return false;
        }
    }
    return true;
}
template<int n> constexpr bool is_prime = is_prime_constexpr(n);

// @param b `1 <= b`
// @return pair(g, x) s.t. g = gcd(a, b), xa = g (mod b), 0 <= x < b/g
constexpr std::pair<long long, long long> inv_gcd(long long a, long long b) {
    a = safe_mod(a, b);
    if(a == 0) return {b, 0};

    // Contracts:
    // [1] s - m0 * a = 0 (mod b)
    // [2] t - m1 * a = 0 (mod b)
```

```cpp
    // [3] s * |m1| + t * |m0| <= b
    long long s = b, t = a;
    long long m0 = 0, m1 = 1;

    while(t) {
        long long u = s / t;
        s -= t * u;
        m0 -= m1 * u;  // |m1 * u| <= |m1| * s <= b

        // [3]:
        // (s - t * u) * |m1| + t * |m0 - m1 * u|
        // <= s * |m1| - t * u * |m1| + t * (|m0| + |m1| * u)
        // = s * |m1| + t * |m0| <= b

        auto tmp = s;
        s = t;
        t = tmp;
        tmp = m0;
        m0 = m1;
        m1 = tmp;
    }
    // by [3]: |m0| <= b/g
    // by g != b: |m0| < b/g
    if(m0 < 0) m0 += b / s;
    return {s, m0};
}

// Compile time primitive root
// @param m must be prime
// @return primitive root (and minimum in now)
constexpr int primitive_root_constexpr(int m) {
    if(m == 2) return 1;
    if(m == 167772161) return 3;
    if(m == 469762049) return 3;
    if(m == 754974721) return 11;
    if(m == 998244353) return 3;
    int divs[20] = {};
    divs[0] = 2;
    int cnt = 1;
    int x = (m - 1) / 2;
    while(x % 2 == 0) x /= 2;
    for(int i = 3; (long long)(i)*i <= x; i += 2) {
        if(x % i == 0) {
            divs[cnt++] = i;
            while (x % i == 0) {
                x /= i;
            }
        }
    }
    if(x > 1) {
        divs[cnt++] = x;
    }
    for(int g = 2;; g++) {
        bool ok = true;
        for(int i = 0; i < cnt; i++) {
            if(pow_mod_constexpr(g, (m - 1) / divs[i], m) == 1) {
                ok = false;
                break;
            }
        }
        if(ok) return g;
    }
}
template<int m> constexpr int primitive_root = primitive_root_constexpr(m);

// @param n `n < 2^32`
// @param m `1 <= m < 2^32`
// @return sum_{i=0}^{n-1} floor((ai + b) / m) (mod 2^64)
unsigned long long floor_sum_unsigned(unsigned long long n, unsigned long long m, unsigned long long a, unsigned long long b) {
  unsigned long long ans = 0;
  while(true) {
    if(a >= m) {
      ans += n * (n - 1) / 2 * (a / m);
      a %= m;
    }
    if(b >= m) {
      ans += n * (b / m);
      b %= m;
    }
    unsigned long long y_max = a * n + b;
    if(y_max < m) {
      break;
    }
    // y_max < m * (n + 1)
    // floor(y_max / m) <= n
    n = (unsigned long long)(y_max / m);
    b = (unsigned long long)(y_max % m);
    std::swap(m, a);
  }
  return ans;
}

} // namespace internal

} // namespace felix

namespace felix {

namespace internal {

class modint_base {};
class static_modint_base : modint_base {};

template<class T> using is_modint = std::is_base_of<modint_base, T>;
template<class T> using is_modint_t = std::enable_if_t<is_modint<T>::value>;

} // namespace internal

template<int m>
class static_modint : internal::static_modint_base {
public:
  static constexpr int mod() {
    return m;
  }

  static_modint() : value(0) {}

  template<class T>
  static_modint(T v) {
    v %= mod();
    if(v < 0) {
      v += mod();
    }
    value = v;
  }

  const int& operator()() const {
    return value;
  }

  template<class T>
  explicit operator T() const {
    return static_cast<T>(value);
  }

  static_modint& operator+=(const static_modint& rhs) {
    value += rhs.value;
    if(value >= mod()) {
      value -= mod();
    }
    return *this;
  }

  static_modint& operator-=(const static_modint& rhs) {
    value -= rhs.value;
    if(value < 0) {
      value += mod();
    }
    return *this;
  }

  static_modint& operator*=(const static_modint& rhs) {
    value = (long long) value * rhs.value % mod();
    return *this;
  }

  static_modint& operator/=(const static_modint& rhs) {
    auto eg = internal::inv_gcd(rhs.value, mod());
    assert(eg.first == 1);
    return *this *= eg.second;
  }

  template<class T>
  static_modint& operator+=(const T& rhs) {
    return *this += static_modint(rhs);
  }

  template<class T>
  static_modint& operator-=(const T& rhs) {
    return *this -= static_modint(rhs);
  }

  template<class T>
  static_modint& operator*=(const T& rhs) {
    return *this *= static_modint(rhs);
  }

  template<class T>
  static_modint& operator/=(const T& rhs) {
    return *this /= static_modint(rhs);
  }

  static_modint operator+() const {
    return *this;
  }

  static_modint operator-() const {
    return static_modint() - *this;
  }

  static_modint& operator++() {
    return *this += 1;
  }

  static_modint& operator--() {
    return *this -= 1;
  }

  static_modint operator++(int) {
    static_modint res(*this);
    *this += 1;
    return res;
  }

  static_modint operator--(int) {
    static_modint res(*this);
    *this -= 1;
    return res;
  }

  static_modint operator+(const static_modint& rhs) {
    return static_modint(*this) += rhs;
  }

  static_modint operator-(const static_modint& rhs) {
    return static_modint(*this) -= rhs;
  }

  static_modint operator*(const static_modint& rhs) {
    return static_modint(*this) *= rhs;
  }

  static_modint operator/(const static_modint& rhs) {
    return static_modint(*this) /= rhs;
  }

  inline bool operator==(const static_modint& rhs) const {
    return value == rhs();
  }

  inline bool operator!=(const static_modint& rhs) const {
    return !(*this == rhs);
  }

private:
  int value;
};

template<int m, class T> static_modint<m> operator+(const T& lhs, const static_modint<m>& rhs) {
  return static_modint<m>(lhs) += rhs;
}

template<int m, class T> static_modint<m> operator-(const T& lhs, const static_modint<m>& rhs) {
  return static_modint<m>(lhs) -= rhs;
}

template<int m, class T> static_modint<m> operator*(const T& lhs, const static_modint<m>& rhs) {
  return static_modint<m>(lhs) *= rhs;
}

template<int m, class T> static_modint<m> operator/(const T& lhs, const static_modint<m>& rhs) {
  return static_modint<m>(lhs) /= rhs;
}

template<int m>
std::istream& operator>>(std::istream& in, static_modint<m>& num) {
  long long x;
  in >> x;
  num = static_modint<m>(x);
  return in;
}

template<int m>
std::ostream& operator<<(std::ostream& out, const static_modint<m>& num) {
  return out << num();
}

template<int id>
class dynamic_modint : internal::modint_base {
public:
  static int mod() {
    return int(bt.umod());
  }

  static void set_mod(int m) {
    assert(1 <= m);
    bt = internal::barrett(m);
  }

  dynamic_modint() : value(0) {}

  template<class T>
  dynamic_modint(T v) {
    v %= mod();
    if(v < 0) {
      v += mod();
    }
    value = v;
  }

  const unsigned int& operator()() const {
    return value;
  }

  template<class T>
  explicit operator T() const {
    return static_cast<T>(value);
  }

  dynamic_modint& operator+=(const dynamic_modint& rhs) {
    value += rhs.value;
    if(value >= umod()) {
      value -= umod();
    }
    return *this;
  }

  template<class T>
  dynamic_modint& operator+=(const T& rhs) {
    return *this += dynamic_modint(rhs);
  }

  dynamic_modint& operator-=(const dynamic_modint& rhs) {
    value += mod() - rhs.value;
    if(value >= umod()) {
      value -= umod();
    }
    return *this;
  }

  template<class T>
  dynamic_modint& operator-=(const T& rhs) {
    return *this -= dynamic_modint(rhs);
  }

  dynamic_modint& operator*=(const dynamic_modint& rhs) {
    value = bt.mul(value, rhs.value);
    return *this;
  }

  template<class T>
  dynamic_modint& operator*=(const T& rhs) {
    return *this *= dynamic_modint(rhs);
  }

  dynamic_modint& operator/=(const dynamic_modint& rhs) {
    auto eg = internal::inv_gcd(rhs.value, mod());
    assert(eg.first == 1);
    return *this *= eg.second;
  }

  template<class T>
  dynamic_modint& operator/=(const T& rhs) {
    return *this /= dynamic_modint(rhs);
  }

  dynamic_modint operator+() const {
    return *this;
  }

  dynamic_modint operator-() const {
    return dynamic_modint() - *this;
  }

  dynamic_modint& operator++() {
    ++value;
    if(value == umod()) {
      value = 0;
    }
    return *this;
  }

  dynamic_modint& operator--() {
    if(value == 0) {
      value = umod();
    }
    --value;
    return *this;
  }

  dynamic_modint operator++(int) {
    dynamic_modint res(*this);
    ++*this;
    return res;
  }

  dynamic_modint operator--(int) {
    dynamic_modint res(*this);
    --*this;
    return res;
  }

  dynamic_modint operator+(const dynamic_modint& rhs) {
    return dynamic_modint(*this) += rhs;
  }

  dynamic_modint operator-(const dynamic_modint& rhs) {
    return dynamic_modint(*this) -= rhs;
  }

  dynamic_modint operator*(const dynamic_modint& rhs) {
    return dynamic_modint(*this) *= rhs;
  }

  dynamic_modint operator/(const dynamic_modint& rhs) {
    return dynamic_modint(*this) /= rhs;
  }

  inline bool operator==(const dynamic_modint& rhs) const {
    return value == rhs();
  }

  inline bool operator!=(const dynamic_modint& rhs) const {
    return !(*this == rhs);
  }

private:
  unsigned int value;
  static internal::barrett bt;
  static unsigned int umod() { return bt.umod(); }
};

template<int id, class T> dynamic_modint<id> operator+(const T& lhs, const dynamic_modint<id>& rhs) {
  return dynamic_modint<id>(lhs) += rhs;
}

template<int id, class T> dynamic_modint<id> operator-(const T& lhs, const dynamic_modint<id>& rhs) {
  return dynamic_modint<id>(lhs) -= rhs;
}

template<int id, class T> dynamic_modint<id> operator*(const T& lhs, const dynamic_modint<id>& rhs) {
  return dynamic_modint<id>(lhs) *= rhs;
}

template<int id, class T> dynamic_modint<id> operator/(const T& lhs, const dynamic_modint<id>& rhs) {
  return dynamic_modint<id>(lhs) /= rhs;
}

template<int id> internal::barrett dynamic_modint<id>::bt(998244353);

template<int id>
std::istream& operator>>(std::istream& in, dynamic_modint<id>& num) {
  long long x;
  in >> x;
  num = dynamic_modint<id>(x);
  return in;
}

template<int id>
std::ostream& operator<<(std::ostream& out, const dynamic_modint<id>& num) {
  return out << num();
}

using modint998244353 = static_modint<998244353>;
using modint1000000007 = static_modint<1000000007>;

namespace internal {

template <class T>
using is_static_modint = std::is_base_of<static_modint_base, T>;

template <class T>
using is_static_modint_t = std::enable_if_t<is_static_modint<T>::value>;

template <class> struct is_dynamic_modint : public std::false_type {};
template <int id>
struct is_dynamic_modint<dynamic_modint<id>> : public std::true_type {};

template <class T>
using is_dynamic_modint_t = std::enable_if_t<is_dynamic_modint<T>::value>;

} // namespace internal

} // namespace felix

using namespace std;
using namespace felix;

using mint = modint1000000007;

mint C(int n, int k) {
  static vector<mint> fact{1}, inv_fact{1};
  if(k < 0 || k > n) {
    return mint(0);
  }
  while((int) fact.size() <= n) {
    fact.push_back(fact.back() * (int) fact.size());
    inv_fact.push_back(1 / fact.back());
  }
  return fact[n] * inv_fact[k] * inv_fact[n - k];
}

int main() {
  ios::sync_with_stdio(false);
  cin.tie(0);
  int n;
  cin >> n;
  if(n % 2 == 1) {
    cout << "0\n";
    return 0;
  }
  string s;
  cin >> s;
  int m = (int) s.size();
  int delta = 0;
  int left = 0;
  for(char& c : s) {
    delta += (c == '(' ? +1 : -1);
    left += (c == '(');
    if(delta < 0) {
      cout << "0\n";
      return 0;
    }
  }
  left = n / 2 - left;
  mint ans = C(n - m, left) - C(n - m, left + delta + 1);
  cout << ans << "\n";
  return 0;
}
```

1.10 CountingNumbers

```cpp
//Your task is to count the number of integers between a and b where no two adjacent digits are the same.
#include <bits/extc++.h>
#include <bits/stdc++.h>
#pragma gcc optimize("ofast, unroll-loops, no-stack-protector, fast-math")
#define IOS ios::sync_with_stdio(0),cin.tie(0),cout.tie(0)
#define int long long
#define double long double
#define pb push_back
#define sz(x) (int)(x).size()
#define all(v) begin(v),end(v)
#define debug(x) cerr<<#x<<" = "<<x<<'\n'
#define LINE cout<<"\n-----------------\n"
#define endl '\n'
#define VI vector<int>
#define F first
#define S second
#define MP(a,b) make_pair(a,b)
#define rep(i,m,n) for(int i = m;i<=n;++i)
#define res(i,m,n) for(int i = m;i>=n;--i)
#define gcd(a,b) __gcd(a,b)
#define lcm(a,b) a*b/gcd(a,b)
#define Case() int _;cin>>_;for(int Case = 1;Case<=_;++Case)
#define pii pair<int,int>
#define lowbit(x) (x&(-x))
using namespace __gnu_cxx;
using namespace __gnu_pbds;
using namespace std;
template <typename K, typename cmp = less<K>, typename T = thin_heap_tag> using _heap = __gnu_pbds::priority_queue<K, cmp, T>;
template <typename K, typename M = null_type> using _hash = gp_hash_table<K, M>;
const int N = 1e6+5,L = 20,mod = 1e9+7,inf = 2e9+5;
const double eps = 1e-7,pi = acos(-1);
mt19937 mt(std::chrono::system_clock::now().time_since_epoch().count());
int cnt(int x){
  if(x<0)return 0;
  string s = std::to_string(x);
  reverse(all(s));
  int n = s.size(),ans = 0;
  int dp[n][2][10] = {};
  for(int i = 0;i<10;++i){
    dp[0][(i>(s[0]-'0'))][i]++;
  }
  for(int i = 1;i<n;++i){
    for(int j = 0;j<2;++j){
      for(int last = 0;last<10;++last){
        for(int add = 0;add<10;++add){
          if(add==last)continue;
          bool flag = (add>(s[i]-'0')) or (add==(s[i]-'0') and j);
          dp[i][flag][add]+=dp[i-1][j][last];
        }
      }
    }
  }
  for(int i = 0;i<n-1;++i){
    for(int j = 0;j<2;++j){
      for(int k = 1;k<10;++k){
        ans+=dp[i][j][k];
      }
    }
  }
  for(int i = 1;i<10;++i){
    ans+=dp[n-1][0][i];
  }
  return ans+1;
}
void solve(){
  int a,b;
  cin>>a>>b;
  cout<<cnt(b)-cnt(a-1)<<endl;
}
signed main(){
  IOS;
  solve();
}
```

1.11 WordCombinations

```cpp
//You are given a string of length n and a dictionary containing k words. In how many ways can you create the string using the words?
#include <bits/stdc++.h>

#pragma GCC optimize("Ofast,unroll-loops,no-stack-protector,fast-math")

using namespace std;

#define fastio ios::sync_with_stdio(false), cin.tie(NULL), cout.tie(NULL)

typedef int64_t ll;

const ll A = 912345693, B = 987654327, mxN = 5005, mod = 1e9 + 7;

int n;
string s;
ll h[mxN], p[mxN], dp[mxN];
vector<ll> num[mxN];

ll Get(int a, int b) {
  return ((h[b] - h[a - 1] * p[b - a + 1]) % B + B) % B;
}

int main() {
  fastio;
  p[0] = 1;
  for(int i = 1; i < mxN; ++i)
    p[i] = p[i - 1] * A % B;
  cin >> s >> n;
  s = " " + s;
  h[0] = 0;
  for(int i = 1; i <= s.size(); ++i)
    h[i] = (A * h[i - 1] + s[i]) % B;
  for(int i = 0; i < n; ++i) {
    string temp;
    cin >> temp;
    ll val = 0;
    for(char c : temp)
      val = (val * A + c) % B;
    num[temp.size()].push_back(val);
  }
  n = s.size() - 1;
  dp[0] = 1;
  for(int i = 0; i < n; ++i) {
    for(int j = 1; i + j <= n; ++j) {
      ll val = Get(i + 1, i + j);
      for(ll x : num[j])
        if(val == x)
          dp[i + j] = (dp[i + j] + dp[i]) % mod;
    }
  }
  cout << dp[n] << "\n";

  return 0;
}
```

1.12 SumOfDivisors

```cpp
/*
Let σ(n) denote the sum of divisors of an integer n. For example, σ(12)=1+2+3+4+6+12=28.

Your task is to calculate the sum ∑(i = 1, n) σ(i) modulo 10^9+7.
*/
#include<bits/stdc++.h>
#define int long long
using namespace std;
const int mod = 1e9 + 7;
constexpr long long Pow(long long x, long long n, int m) {
  if(m == 1) return 0;
  unsigned int _m = (unsigned int)(m);
  unsigned long long r = 1;
  x %= m;
  if(x < 0) x += m;
  unsigned long long y = x;
  while(n) {
    if(n & 1) r = (r * y) % _m;
    y = (y * y) % _m;
    n >>= 1;
  }
  return r;
}
signed main(){
  int n;
  cin>>n;
  int ans = 0;
  for(int l = 1, r = n / (n / l); l <= n; l = r + 1){
    r = n / (n / l);
    ans += (((((((l + r) % mod) * ((r - l + 1) % mod)) % mod) * Pow(2, mod - 2, mod)) % mod) * ((n / l) % mod)) % mod;
    ans %= mod;
  }
  cout<<ans<<endl;
}
```

1.13 RemovalGame

```cpp
/*
There is a list of n numbers and two players who move alternately. On each move, a player removes either the first or last number from the list, and their score increases by that number. Both players try to maximize their scores.

What is the maximum possible score for the first player when both players play optimally?
*/
#include <bits/stdc++.h>
using namespace std;

int main() {
  ios::sync_with_stdio(false);
  cin.tie(0);
  int n;
  cin >> n;
  vector<long long> a(n), pref(n + 1);
  for(int i = 0; i < n; ++i) {
    cin >> a[i];
    pref[i + 1] = pref[i] + a[i];
  }
  vector<vector<long long>> dp(n, vector<long long>(n));
  for(int i = 0; i < n; ++i) {
    dp[i][i] = a[i];
  }
  for(int len = 2; len <= n; ++len) {
    for(int i = 0; i + len - 1 < n; ++i) {
      int j = i + len - 1;
      dp[i][j] = pref[j + 1] - pref[i] - min(dp[i + 1][j], dp[i][j - 1]);
    }
  }
  cout << dp[0][n - 1] << "\n";
  return 0;
}
```

1.14 MinimalRotation

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

template<int ALPHABET = 26, char MIN_CHAR = 'a'>
class suffix_automaton {
public:
  struct Node {
    int len;
    int suffLink;
    int go[ALPHABET] = {};

    Node() : Node(0, -1) {}
    Node(int a, int b) : len(a), suffLink(b) {}
  };

  suffix_automaton() : suffix_automaton(string(0, ' ')) {}
  suffix_automaton(const string& s) {
    SA.emplace_back();
    last = 0;
    for(char c : s) {
      add(c - MIN_CHAR);
    }
  }

  void add(int c) {
    int u = newNode();
    SA[u].len = SA[last].len + 1;
    int p = last;
    while(p != -1 && SA[p].go[c] == 0) {
      SA[p].go[c] = u;
      p = SA[p].suffLink;
    }
    if(p == -1) {
      SA[u].suffLink = 0;
      last = u;
      return;
    }
    int q = SA[p].go[c];
    if(SA[p].len + 1 == SA[q].len) {
      SA[u].suffLink = q;
      last = u;
      return;
    }
    int x = newNode();
    SA[x] = SA[q];
    SA[x].len = SA[p].len + 1;
    SA[q].suffLink = SA[u].suffLink = x;
    while(p != -1 && SA[p].go[c] == q) {
      SA[p].go[c] = x;
      p = SA[p].suffLink;
    }
    last = u;
    return;
  }

// private:
  vector<Node> SA;
  int last;

  inline int newNode() {
    SA.emplace_back();
    return (int) SA.size() - 1;
  }
};

int main() {
  ios::sync_with_stdio(false);
  cin.tie(0);
  string s;
  cin >> s;
  int n = (int) s.size();
  suffix_automaton SA(s + s);
  int p = 0;
  for(int i = 0; i < n; ++i) {
    for(int c = 0; c < 26; ++c) {
      if(SA.SA[p].go[c]) {
        cout << char('a' + c);
        p = SA.SA[p].go[c];
        break;
      }
    }
  }
  cout << "\n";
  return 0;
}
```

1.15 Dice

```
//1, 2, 4, 8, 16, 32, 63, 125, 248, 492
#include <bits/stdc++.h>
using namespace std;

typedef uint64_t ull;
const int mod = 1e9 + 7;

struct Matrix {
  ull M[6][6];
  Matrix() {
    memset(M, 0, sizeof(M));
  }
  Matrix operator*(const Matrix& other) {
    Matrix ans;
    for(int i = 0; i < 6; ++i)
      for(int j = 0; j < 6; ++j)
        for(int k = 0; k < 6; ++k)
          ans.M[i][j] = (ans.M[i][j] + M[i][k] * other.M[k][j])
             % mod;
    return ans;
  }
};

Matrix FastPower(Matrix a, ull b) {
  Matrix ans;
  for(int i = 0; i < 6; ++i)
    ans.M[i][i] = 1;
  while(b) {
    if(b & 1)
      ans = ans * a;
    a = a * a;
    b >>= 1;
  }
  return ans;
}

int main() {
  Matrix A;
  for(int i = 0; i < 6; ++i)
    A.M[0][i] = 1;
  for(int i = 1; i < 6; ++i)
    A.M[i][i - 1] = 1;
  ull n;
  cin >> n;
  cout << FastPower(A, n).M[0][0] << "\n";

  return 0;
}
```

2 Data-Structure

2.1 Treap

```
template<class S,
    S (*node_pull)(S, S),
    S (*node_init)(S),
    class T,
    S (*mapping)(S, T),
    T (*tag_pull)(T, T),
    T (*tag_init)()>
struct Treap{
  struct node{
    node *l = NULL,*r = NULL,*p = NULL;
    const int pri = rand();
    int sz = 1;
    S info;
    T tag = tag_init();
    bool rev;
    node(S k) : info(k){}
    ~node(){
      for(auto &i:{l,r})
        delete i;
    }
    void all_apply(T t,bool is_rev){
      if(is_rev){
        swap(l,r);
        rev^=1;
      }
      info = mapping(info, t);
      tag = tag_pull(tag, t);
    }
    void push(){
      for(auto &i:{l,r})
        if(i)i->all_apply(tag, rev);
      tag = tag_init();
      rev = 0;
    }
    void pull(){
      sz = 1,info = node_init(info);
      for(auto &i:{l,r}){
        if(i){
          sz+=i->sz,i->p = this;
          info = node_pull(info,i->info);
        }
      }
    }
  };
  node *root = NULL;
  int size(node *a){
    return a?a->sz:0;
  }
  int size(){
    return size(root);
  }
  node *merge(node *a,node *b){
    if(!a or !b)return a?:b;
    if(a->pri>b->pri){
      a->push();
      a->r = merge(a->r,b);
      a->r->p = a;
      a->pull();
      return a;
    }
    else{
      b->push();
      b->l = merge(a,b->l);
      b->l->p = b;
      b->pull();
      return b;
    }
  }
  void split(node *t, long long k, node *&a, node *&b, const bool &bst){
    if(!t){a = b = NULL;return;}
    t->push();
    if((bst==0 and size(t->l)+1<=k) or (bst==1 and t->info.key<=k)){
      a = t;
      split(t->r, ( bst ? k : k - size(t->l) - 1 ), a->r, b, bst);
      if(b)b->p = NULL;
      a->pull();
    }
    else{
      b = t;
      split(t->l, k, a, b->l, bst);
      if(a)a->p = NULL;
      b->pull();
    }
  }
  node *insert(long long idx, S x,bool bst = 0){
    node *a,*b;
    split(root, idx, a, b, bst);
    node *tmp = new node(x);
    root = merge(a, merge(tmp, b));
    return tmp;
  }
  void erase(long long l,long long r,bool bst = 0){
    node *a,*b,*c;
    split(root, (bst? l-1 : l), a, b, bst);
    split(b, (bst? r : r - l + 1), b, c, bst);
    delete b;
    root = merge(a,c);
  }
  S operator [](int x){
    node *a, *b, *c;
    split(root, x, a, b, 0);
    split(b, 1, b, c, 0);
    assert(b!=NULL);
    S ans = b->info;
    root = merge(a, merge(b, c));
    return ans;
  }
  int rank(long long k){
    node *a, *b;
    split(root, k - 1, a, b, 1);
    int ans = size(a);
    root = merge(a, b);
    return ans;
  }
  S* find_next(long long k){
    node *a, *b, *c;
    split(root, k - 1, a, b, 1);
    split(b, 1, b, c, 0);
    S* ans = NULL;
    if(b)ans = &b->info;
    root = merge(a, merge(b, c));
    return ans;
  }
  S* find_prev(long long k){
    node *a, *b, *c;
    split(root, k , a, b, 1);
    split(a, size(a) - 1, a, c, 0);
    S* ans = NULL;
    if(c)ans = &c->info;
    root = merge(merge(a, c), b);
    return ans;
  }
  void update(long long l,long long r,T t,bool bst = 0){
    node *a, *b, *c;
    split(root, (bst? l - 1 : l), a, b, bst);
    split(b, (bst? r : r - l + 1), b, c, bst);
    if(b)b->all_apply(t, 0);
    root = merge(a, merge(b, c));
  }
  void reverse(long long l,long long r,bool bst = 0){
    node *a, *b, *c;
    split(root, (bst? l - 1 : l), a, b, bst);
    split(b, (bst? r : r - l + 1), b, c, bst);
    if(b)b->all_apply(tag_init(), 1);
    root = merge(a, merge(b, c));
  }
  S query(long long l,long long r,bool bst = 0){
    node *a, *b, *c;
    split(root, (bst? l - 1 : l), a, b, bst);
    split(b, (bst? r : r - l + 1), b, c, bst);
    S ans;
    if(b)ans = b->info;
    root = merge(a, merge(b, c));
    return  ans;
  }
};
```

2.2 Segtree

```
template<class S,
     S (*node_pull)(S, S),
     S (*node_init)(),
     class T,
     S (*mapping)(S, T),
     T (*tag_pull)(T, T),
     T (*tag_init)()>
struct segment_tree{
  struct node{
    S seg;
    T tag = tag_init();
    int l,r;
    node(S _seg = node_init(),int _l = -1,int _r = -1) : seg(_seg), l(_l), r(_r){}
    friend node operator +(const node &lhs,const node &rhs){
      if(lhs.l==-1)return rhs;
      if(rhs.l==-1)return lhs;
      return node(node_pull(lhs.seg,rhs.seg),lhs.l,rhs.r);
    };
  };
  vector<node>arr;
  void all_apply(int idx,T t){
    arr[idx].seg = mapping(arr[idx].seg, t);
    arr[idx].tag = tag_pull(arr[idx].tag, t);
  }
  void push(int idx){
    all_apply(idx<<1, arr[idx].tag);
    all_apply(idx<<1|1, arr[idx].tag);
    arr[idx].tag = tag_init();
  }
  inline void build(const vector<S> &v,const int &l,const int &r,int idx = 1){
    if(idx==1)arr.resize((r-l+1)<<2);
    if(l==r){
      arr[idx].seg = v[l];
      arr[idx].tag = tag_init();
      arr[idx].l = arr[idx].r = l;
      return;
    }
    int m = (l+r)>>1;
    build(v,l,m,idx<<1);
    build(v,m+1,r,idx<<1|1);
    arr[idx] = arr[idx<<1]+arr[idx<<1|1];
  }
  inline void update(const int &ql,const int &qr,T t,int idx = 1){
    assert(ql<=qr);
    if(ql<=arr[idx].l and arr[idx].r<=qr){
      all_apply(idx, t);
      return;
    }
    push(idx);
    int m = (arr[idx].l+arr[idx].r)>>1;
    if(ql<=m)update(ql,qr,t,idx<<1);
    if(qr>m)update(ql,qr,t,idx<<1|1);
    arr[idx] = arr[idx<<1]+arr[idx<<1|1];
  }
  inline S query(const int &ql,const int &qr,int idx = 1){
    assert(ql<=qr);
    if(ql<=arr[idx].l and arr[idx].r<=qr){
      return arr[idx].seg;
    }
    push(idx);
    int m = (arr[idx].l+arr[idx].r)>>1;
    S ans = node_init(),lhs = node_init(),rhs = node_init();
    if(ql<=m)lhs = query(ql,qr,idx<<1);
    if(qr>m)rhs = query(ql,qr,idx<<1|1);
    ans = node_pull(lhs,rhs);
    return ans;
  }
};
```

2.3 BinaryTrie

```
template<class T>
struct binary_trie {
public:
  binary_trie() {
    new_node();
  }

  void clear() {
    trie.clear();
    new_node();
  }

  void insert(T x) {
    for(int i = B - 1, p = 0; i >= 0; i--) {
      int y = x >> i & 1;
      if(trie[p].go[y] == 0) {
        trie[p].go[y] = new_node();
      }
      p = trie[p].go[y];
      trie[p].cnt += 1;
    }
  }

  void erase(T x) {
    for(int i = B - 1, p = 0; i >= 0; i--) {
      p = trie[p].go[x >> i & 1];
      trie[p].cnt -= 1;
    }
  }

  bool contains(T x) {
    for(int i = B - 1, p = 0; i >= 0; i--) {
      p = trie[p].go[x >> i & 1];
      if(trie[p].cnt == 0) {
        return false;
      }
    }
    return true;
  }

  T get_min() {
    return get_xor_min(0);
  }

  T get_max() {
    return get_xor_max(0);
  }

  T get_xor_min(T x) {
    T ans = 0;
    for(int i = B - 1, p = 0; i >= 0; i--) {
      int y = x >> i & 1;
      int z = trie[p].go[y];
      if(z > 0 && trie[z].cnt > 0) {
        p = z;
      } else {
        ans |= T(1) << i;
        p = trie[p].go[y ^ 1];
      }
    }
    return ans;
  }

  T get_xor_max(T x) {
    T ans = 0;
    for(int i = B - 1, p = 0; i >= 0; i--) {
      int y = x >> i & 1;
      int z = trie[p].go[y ^ 1];
      if(z > 0 && trie[z].cnt > 0) {
        ans |= T(1) << i;
        p = z;
      } else {
        p = trie[p].go[y];
      }
    }
    return ans;
  }

private:
  static constexpr int B = sizeof(T) * 8;

  struct Node {
    std::array<int, 2> go = {};
    int cnt = 0;
  };

  std::vector<Node> trie;

  int new_node() {
    trie.emplace_back();
    return (int) trie.size() - 1;
  }
};
```

2.4 DsuUndo

```
struct dsu_undo{
  vector<int>sz,p;
  int comps;
  dsu_undo(int n){
    sz.assign(n+5,1);
    p.resize(n+5);
    for(int i = 1;i<=n;++i)p[i] = i;
    comps = n;
  }
  vector<pair<int,int>>opt;
  int Find(int x){
    return x==p[x]?x:Find(p[x]);
  }
  bool Union(int a,int b){
    int pa = Find(a),pb = Find(b);
    if(pa==pb)return 0;
    if(sz[pa]<sz[pb])swap(pa,pb);
    sz[pa]+=sz[pb];
    p[pb] = pa;
    opt.push_back({pa,pb});
    comps--;
    return 1;
  }
  void undo(){
        auto [pa,pb] = opt.back();
        opt.pop_back();
        p[pb] = pb;
        sz[pa]-=sz[pb];
        comps++;
  }
};
```

2.5 DSU

```
struct DSU{
  vector<int>sz;
  int n;
  DSU(int _n):n(_n){
    sz.assign(n+1,-1);
  }
  int Find(int x){
    return sz[x]<0?x:sz[x] = Find(sz[x]);
  }
  bool Union(int a,int b){
    int pa = Find(a),pb = Find(b);
    if(pa==pb)return 0;
    if((-sz[pa])<(-sz[pb]))swap(pa,pb);
    sz[pa]+=sz[pb];
    sz[pb] = pa;
    return 1;
  }
};
```

2.6 Fenwick

```
template<class T>struct fenwick_tree{
  int n;
  vector<T>arr;
  inline int lowbit(int x){
    return x&(-x);
  }
  fenwick_tree(int _n) : n(_n){
    arr.assign(n+5,0);
  }
  T query(int x){
    T ans = 0;
    for(int i = x;i>0;i-=lowbit(i)){
      ans+=arr[i];
    }
    return ans;
  }
  void update(int x,T y){
    for(int i = x;i<=n;i+=lowbit(i)){
      arr[i]+=y;
    }
  }
};
```

2.7 Persistent DSU

```
int rk[200001] = {};
struct Persistent_DSU{
  rope<int>*p;
  int n;
  Persistent_DSU(int _n = 0):n(_n){
    if(n==0)return;
    p = new rope<int>;
    int tmp[n+1] = {};
    for(int i = 1;i<=n;++i)tmp[i] = i;
    p->append(tmp,n+1);
  }
  Persistent_DSU(const Persistent_DSU &tmp){
    p = new rope<int>(*tmp.p);
    n = tmp.n;
  }
  int Find(int x){
    int px = p->at(x);
    return px==x?x:Find(px);
  }
  bool Union(int a,int b){
    int pa = Find(a),pb = Find(b);
    if(pa==pb)return 0;
    if(rk[pa]<rk[pb])swap(pa,pb);
    p->replace(pb,pa);
    if(rk[pa]==rk[pb])rk[pa]++;
    return 1;
  }
};
```

2.8 TimingSegtree

```
template<class T,class D>struct timing_segment_tree{
  struct node{
    int l,r;
    vector<T>opt;
  };
  vector<node>arr;
  void build(int l,int r,int idx = 1){
    if(idx==1)arr.resize((r-l+1)<<2);
    if(l==r){
      arr[idx].l = arr[idx].r = l;
      arr[idx].opt.clear();
      return;
    }
    int m = (l+r)>>1;
    build(l,m,idx<<1);
    build(m+1,r,idx<<1|1);
    arr[idx].l = l,arr[idx].r = r;
    arr[idx].opt.clear();
  }
  void update(int ql,int qr,T k,int idx = 1){
    if(ql<=arr[idx].l and arr[idx].r<=qr){
      arr[idx].opt.push_back(k);
      return;
    }
    int m = (arr[idx].l+arr[idx].r)>>1;
    if(ql<=m)update(ql,qr,k,idx<<1);
    if(qr>m)update(ql,qr,k,idx<<1|1);
  }
  void dfs(D &d,vector<int>&ans,int idx = 1){
    int cnt = 0;
    for(auto [a,b]:arr[idx].opt){
      if(d.Union(a,b))cnt++;
    }
    if(arr[idx].l==arr[idx].r)ans[arr[idx].l] = d.comps;
    else{
      dfs(d,ans,idx<<1);
      dfs(d,ans,idx<<1|1);
    }
    while(cnt--)d.undo();
  }
};
```

2.9 AreaOfRectangles

```
long long AreaOfRectangles(vector<tuple<int,int,int,int>>v){
  vector<tuple<int,int,int,int>>tmp;
  int L = INT_MAX,R = INT_MIN;
  for(auto [x1,y1,x2,y2]:v){
    tmp.push_back({x1,y1+1,y2,1});
    tmp.push_back({x2,y1+1,y2,-1});
    R = max(R,y2);
    L = min(L,y1);
  }
  vector<long long>seg((R-L+1)<<2),tag((R-L+1)<<2);
  sort(tmp.begin(),tmp.end());
  function<void(int,int,int,int,int,int)>update = [&](int ql,
      int qr,int val,int l,int r,int idx){
    if(ql<=l and r<=qr){
      tag[idx]+=val;
      if(tag[idx])seg[idx] = r-l+1;
      else if(l==r)seg[idx] = 0;
      else seg[idx] = seg[idx<<1]+seg[idx<<1|1];
      return;
    }
    int m = (l+r)>>1;
    if(ql<=m)update(ql,qr,val,l,m,idx<<1);
    if(qr>m)update(ql,qr,val,m+1,r,idx<<1|1);
    if(tag[idx])seg[idx] = r-l+1;
    else seg[idx] = seg[idx<<1]+seg[idx<<1|1];
  };
  long long last_pos = 0,ans = 0;
  for(auto [pos,l,r,val]:tmp){
    ans+=(pos-last_pos)*seg[1];
    update(l,r,val,L,R,1);
    last_pos = pos;
  }
  return ans;
}
```

2.10 SparseTable

```
template<class T,T (*op)(T,T)>struct sparse_table{
  int n;
  vector<vector<T>>mat;
  sparse_table(): n(0){}
  sparse_table(const vector<T>&v){
    n = (int)(v.size());
    mat.resize(30);
    mat[0] = v;
    for(int i = 1;(1<<i)<=n;++i){
      mat[i].resize(n-(1<<i)+1);
      for(int j = 0;j<n-(1<<i)+1;++j){
        mat[i][j] = op(mat[i-1][j],mat[i-1][j+(1<<(i-1))]);
      }
    }
  }
  T query(int ql,int qr){
    int k = __lg(qr-ql+1);
    return op(mat[k][ql],mat[k][qr-(1<<k)+1]);
  }
};
```

2.11 VEBTree

```
// Can correctly work with numbers in range [0; MAXN]
// Supports all std::set operations in O(1) on random queries /
      dense arrays, O(log_64(N)) in worst case (sparce array).
// Count operation works in O(1) always.
template<unsigned int MAXN>
class fast_set {
private:
  static const unsigned int PREF = (MAXN <= 64 ? 0 :
              MAXN <= 4096 ? 1 :
              MAXN <= 262144 ? 1 + 64 :
              MAXN <= 16777216 ? 1 + 64 + 4096 :
              MAXN <= 1073741824 ? 1 + 64 + 4096 + 262144 :
                  227) + 1;
  static constexpr unsigned long long lowest_bitsll[] = {0ULL,
       1ULL, 3ULL, 7ULL, 15ULL, 31ULL, 63ULL, 127ULL, 255ULL,
       511ULL, 1023ULL, 2047ULL, 4095ULL, 8191ULL, 16383ULL,
       32767ULL, 65535ULL, 131071ULL, 262143ULL, 524287ULL,
       1048575ULL, 2097151ULL, 4194303ULL, 8388607ULL, 16777215
       ULL, 33554431ULL, 67108863ULL, 134217727ULL, 268435455ULL
       , 536870911ULL, 1073741823ULL, 2147483647ULL, 4294967295
       ULL, 8589934591ULL, 17179869183ULL, 34359738367ULL,
       68719476735ULL, 137438953471ULL, 274877906943ULL,
       549755813887ULL, 1099511627775ULL, 2199023255551ULL,
       4398046511103ULL, 8796093022207ULL, 17592186044415ULL,
       35184372088831ULL, 70368744177663ULL, 140737488355327ULL,
        281474976710655ULL, 562949953421311ULL, 1125899906842623
       ULL, 2251799813685247ULL, 4503599627370495ULL,
       9007199254740991ULL, 18014398509481983ULL,
       36028797018963967ULL, 72057594037927935ULL,
       144115188075855871ULL, 288230376151711743ULL,
       576460752303423487ULL, 1152921504606846975ULL,
       2305843009213693951ULL, 4611686018427387903ULL,
       9223372036854775807ULL, 18446744073709551615ULL};
  static const unsigned int SZ = PREF + (MAXN + 63) / 64 + 1;
  unsigned long long m[SZ] = {0};

  inline unsigned int left(unsigned int v) const {
    return (v - 62) * 64;
  }

  inline unsigned int parent(unsigned int v) const {
    return v / 64 + 62;
  }

  inline void setbit(unsigned int v) {
    m[v >> 6] |= 1ULL << (v & 63);
  }

  inline void resetbit(unsigned int v) {
    m[v >> 6] &= ~(1ULL << (v & 63));
  }
```

```cpp
  inline unsigned int getbit(unsigned int v) const {
    return m[v >> 6] >> (v & 63) & 1;
  }

  inline unsigned long long childs_value(unsigned int v) const {
    return m[left(v) >> 6];
  }

  inline int left_go(unsigned int x, const unsigned int c) const {
    const unsigned long long rem = x & 63;
    unsigned int bt = PREF * 64 + x;
    unsigned long long num = m[bt >> 6] & lowest_bitsll[rem + c];
    if(num) {
      return (x ^ rem) | __lg(num);
    }
    for(bt = parent(bt); bt > 62; bt = parent(bt)) {
      const unsigned long long rem = bt & 63;
      num = m[bt >> 6] & lowest_bitsll[rem];
      if(num) {
        bt = (bt ^ rem) | __lg(num);
        break;
      }
    }
    if(bt == 62) {
      return -1;
    }
    while(bt < PREF * 64) {
      bt = left(bt) | __lg(m[bt - 62]);
    }
    return bt - PREF * 64;
  }

  inline int right_go(unsigned int x, const unsigned int c) const {
    const unsigned long long rem = x & 63;
    unsigned int bt = PREF * 64 + x;
    unsigned long long num = m[bt >> 6] & ~lowest_bitsll[rem + c];
    if(num) {
      return (x ^ rem) | __builtin_ctzll(num);
    }
    for(bt = parent(bt); bt > 62; bt = parent(bt)) {
      const unsigned long long rem = bt & 63;
      num = m[bt >> 6] & ~lowest_bitsll[rem + 1];
      if(num) {
        bt = (bt ^ rem) | __builtin_ctzll(num);
        break;
      }
    }
    if(bt == 62) {
      return -1;
    }
    while(bt < PREF * 64) {
      bt = left(bt) | __builtin_ctzll(m[bt - 62]);
    }
    return bt - PREF * 64;
  }

public:
  fast_set() {
    assert(PREF != 228);
    setbit(62);
  }

  bool empty() const {return getbit(63);}

  void clear() {
    fill(m, m + SZ, 0);
    setbit(62);
  }

  bool count(unsigned int x) const {
    return m[PREF + (x >> 6)] >> (x & 63) & 1;
  }

  void insert(unsigned int x) {
    for(unsigned int v = PREF * 64 + x; !getbit(v); v = parent(v)) {
      setbit(v);
    }
  }

  void erase(unsigned int x) {
    if(!getbit(PREF * 64 + x)) {
      return;
    }
    resetbit(PREF * 64 + x);
    for(unsigned int v = parent(PREF * 64 + x); v > 62 && !childs_value(v); v = parent(v)) {
      resetbit(v);
    }
  }

  int find_next(unsigned int x) const {
    return right_go(x, 0);
  }

  int find_prev(unsigned int x) const {
    return left_go(x, 1);
  }
};
```

2.12 DynamicSegtree

```cpp

template<class T>struct dynamic_segment_tree{
  struct node{
    node *l = NULL,*r = NULL;
    T sum;
    node(T k = 0): sum(k){}
    node(node *p){if(p)*this = *p;}
    ~node(){
      for(auto &i:{l,r})
        if(i)delete i;
    }
    void pull(){
      sum = 0;
      for(auto i:{l,r})
        if(i)sum+=i->sum;
    }
  }*root = NULL;
  int n;
  dynamic_segment_tree(){}
  dynamic_segment_tree(const dynamic_segment_tree<T>&tmp){root = new node(tmp.root);}
  void update(node *&t,int pos,T k,int l,int r){
    if(!t)t = new node();
    if(l==r)return t = new node(k),void();
    int m = (l+r)>>1;
    t = new node(t);
    if(pos<=m)update(t->l,pos,k,l,m);
    else update(t->r,pos,k,m+1,r);
    t->pull();
  }void update(int pos,T k,int l = -1e9,int r = 1e9){update(root,pos,k,l,r);}
  T query(node *&t,int ql,int qr,int l,int r){
    if(!t)return 0;
    if(ql<=l and r<=qr)return t->sum;
    int m = (l+r)>>1;
    T ans = 0;
    if(ql<=m)ans+=query(t->l,ql,qr,l,m);
    if(qr>m)ans+=query(t->r,ql,qr,m+1,r);
    return ans;
  }T query(int ql,int qr,int l = -1e9,int r = 1e9){return query(root,ql,qr,l,r);}
};
```

2.13 ZkwSegtree

```cpp
template<class S,
         S (*node_pull)(S, S),
         S (*node_init)(),
         class F,
         S (*mapping)(S, F),
         F (*tag_pull)(F, F),
         F (*tag_init)()>
class segment_tree {
public:
  segment_tree() : segment_tree(0) {}
  explicit segment_tree(int _n) : segment_tree(vector<S>(_n, node_init())) {}
  explicit segment_tree(const vector<S>& v) : n((int) v.size()) {
    log = std::__lg(2 * n - 1);
    size = 1 << log;
    d = vector<S>(size << 1, node_init());
    lz = vector<F>(size, tag_init());
    for(int i = 0; i < n; i++) {
      d[size + i] = v[i];
    }
    for(int i = size - 1; i; --i) {
      update(i);
    }
  }
  void set(int p, S x) {
    assert(0 <= p && p < n);
    p += size;
    for(int i = log; i; --i) {
      push(p >> i);
    }
    d[p] = x;
    for(int i = 1; i <= log; ++i) {
      update(p >> i);
    }
  }
  S get(int p) {
    assert(0 <= p && p < n);
    p += size;
    for(int i = log; i; i--) {
      push(p >> i);
    }
    return d[p];
  }
  S operator[](int p) {
    return get(p);
  }
  S query(int l, int r) {
    r++;
    assert(l<=r);
    l += size;
    r += size;
    for(int i = log; i; i--) {
      if(((l >> i) << i) != l) {
        push(l >> i);
      }
      if(((r >> i) << i) != r) {
        push(r >> i);
      }
    }
    S sml = node_init(), smr = node_init();
    while(l < r) {
      if(l & 1) {
        sml = node_pull(sml, d[l++]);
      }
      if(r & 1) {
        smr = node_pull(d[--r], smr);
      }
      l >>= 1;
      r >>= 1;
    }
    return node_pull(sml, smr);
  }
  void apply(int p, F f) {
    assert(0 <= p && p < n);
    p += size;
    for(int i = log; i; i--) {
      push(p >> i);
    }
    d[p] = mapping(f, d[p]);
    for(int i = 1; i <= log; i++) {
      update(p >> i);
    }
  }
  void update(int l, int r, F f) {
    r++;
    assert(l<=r);
    l += size;
    r += size;
    for(int i = log; i; i--) {
      if(((l >> i) << i) != l) {
        push(l >> i);
      }
      if(((r >> i) << i) != r) {
        push((r - 1) >> i);
      }
    }
    {
      int l2 = l, r2 = r;
      while(l < r) {
        if(l & 1) {
          all_apply(l++, f);
        }
        if(r & 1) {
          all_apply(--r, f);
        }
        l >>= 1;
        r >>= 1;
      }
      l = l2;
      r = r2;
    }
    for(int i = 1; i <= log; i++) {
      if(((l >> i) << i) != l) {
        update(l >> i);
      }
      if(((r >> i) << i) != r) {
        update((r - 1) >> i);
      }
    }
  }
private:
  int n, size, log;
  vector<S> d;
  vector<F> lz;
  inline void update(int k) { d[k] = node_pull(d[k << 1], d[k << 1 | 1]); }
  void all_apply(int k, F f) {
    d[k] = mapping(d[k], f);
    if(k < size) {
      lz[k] = tag_pull(lz[k], f);
    }
  }
  void push(int k) {
    all_apply(k << 1, lz[k]);
    all_apply(k << 1 | 1, lz[k]);
    lz[k] = tag_init();
  }
};
```

2.14 MoAlgo

```cpp
struct qry{
  int ql,qr,id;
};
template<class T>struct Mo{
  int n,m;
  vector<pii>ans;
  Mo(int _n,int _m): n(_n),m(_m){
    ans.resize(m);
  }
  void solve(vector<T>&v,vector<qry>&q){
    int l = 0,r = -1;
    vector<int>cnt,cntcnt;
    cnt.resize(n+5);
    cntcnt.resize(n+5);
    int mx = 0;
    function<void(int)>add = [&](int pos){
      cntcnt[cnt[v[pos]]]--;
      cnt[v[pos]]++;
      cntcnt[cnt[v[pos]]]++;
      mx = max(mx,cnt[v[pos]]);
    };
    function<void(int)>sub = [&](int pos){
      if(!--cntcnt[cnt[v[pos]]] and cnt[v[pos]]==mx)mx--;
      cnt[v[pos]]--;
      cntcnt[cnt[v[pos]]]++;
      mx = max(mx,cnt[v[pos]]);
    };
    sort(all(q),[&](qry a,qry b){
      static int B = max((int)1,n/max((int)sqrt(m),(int)1));
      if(a.ql/B!=b.ql/B)return a.ql<b.ql;
      if((a.ql/B)&1)return a.qr>b.qr;
      return a.qr<b.qr;
    });
    for(auto [ql,qr,id]:q){
      while(l>ql)add(--l);
      while(r<qr)add(++r);
      while(l<ql)sub(l++);
      while(r>qr)sub(r--);
      ans[id] = {mx,cntcnt[mx]};
    }
  }
};
```

2.15 Hash

```cpp
struct custom_hash {
  static uint64_t splitmix64(uint64_t x) {
    x += 0x9e3779b97f4a7c15;
    x = (x ^ (x >> 30)) * 0xbf58476d1ce4e5b9;
    x = (x ^ (x >> 27)) * 0x94d049bb133111eb;
    return x ^ (x >> 31);
  }
  size_t operator()(uint64_t x) const {
    static const uint64_t FIXED_RANDOM = chrono::steady_clock::now().time_since_epoch().count();
    return splitmix64(x + FIXED_RANDOM);
  }
  size_t operator()(pair<uint64_t,uint64_t> x) const {
    static const uint64_t FIXED_RANDOM = chrono::steady_clock::now().time_since_epoch().count();
    return splitmix64(3*x.first + x.second + FIXED_RANDOM);
  }
};
template<class T,class U>using hash_map = gp_hash_table<T,U,custom_hash>;
```

2.16 RedBlackTree

```cpp
template<class T, typename cmp=less<>>struct _tree{//#include<bits/extc++.h>
  tree<pair<T,int>,null_type,cmp,rb_tree_tag,tree_order_statistics_node_update>st;
  int id = 0;
  void insert(T x){st.insert({x,id++});}
  void erase(T x){st.erase(st.lower_bound({x,0}));}
  int order_of_key(T x){return st.order_of_key(*st.lower_bound({x,0}));}
  T find_by_order(int x){return st.find_by_order(x)->first;}
  T lower_bound(T x){return st.lower_bound({x,0})->first;}
  T upper_bound(T x){return st.upper_bound({x,(int)1e9+7})->first;}
  T smaller_bound(T x){return (--st.lower_bound({x,0}))->first;}
};
```

3 Geometry

3.1 Theorem

- Pick′ s Theorem

  $A = I + \frac{B}{2} - 1$

  $A := Area$

  $i := PointsInside$

  $B := PointsBoundary$

3.2 PointInPolygon

```
template<class T>
int PointInPolygon(const vector<Point<T>> &Poly, const Point<T> p){
  int ans = 0;
  for(auto a = --Poly.end(),b = Poly.begin();b!=Poly.end();a = b++){
    if(PointOnSegment(*a,*b,p)){
      return -1;
    }
    if(seg_intersect(p,p+Point<T>(2e9+7,1),*a,*b)){
      ans = !ans;
    }
  }
  return ans;
}
```

3.3 PointInConvex

```
template<class T>
int PointInConvex(const vector<Point<T>>&C,const Point<T>&p){
  if(btw(C[0],C[1],p) || btw(C[0],C.back(),p))return -1;
  int l = 0,r = (int)C.size()-1;
  while(l<=r){
    int m = (l+r)>>1;
    auto a1 = (C[m]-C[0])^(p-C[0]);
    auto a2 = (C[(m+1)%C.size()]-C[0])^(p-C[0]);
    if(a1>=0 and a2<=0){
      auto res = (C[(m+1)%C.size()]-C[m])^(p-C[m]);
      return res > 0 ? 1 : (res >= 0 ? -1 : 0);
    }
    if(a1 < 0) r = m-1;
    else l = m+1;
  }
  return 0;
}
```

3.4 MaximumDistance

```
template<class T>
T MaximumDistance(vector<Point<T>>&p){
  vector<Point<T>>C = ConvexHull(p,0);
  int n = C.size(),t = 2;
  T ans = 0;
  for(int i = 0;i<n;i++){
    while(((C[i] - C[t]) ^ (C[(i+1)%n] - C[t])) < ((C[i] - C[(t+1)%n]) ^ (C[(i+1)%n] - C[(t+1)%n]))) t = (t + 1)%n;
    ans = max({ans, abs2(C[i] - C[t]), abs2(C[(i+1)%n] - C[t])});
  }
  return ans;
}
```

3.5 PolarAngleSort

```
template<class T>
bool cmp(const Point<T> &a,const Point<T> &b){
  int lhs = (a.y < 0 || a.y==0 && a.x > 0) ? 0 : (1 + (a.x != 0 || a.y != 0));
  int rhs = (b.y < 0 || b.y==0 && b.x > 0) ? 0 : (1 + (b.x != 0 || b.y != 0));
  if(lhs != rhs) {
    return lhs < rhs;
  }
  long long area = (a^b);
  return area ? area > 0 : abs(a.x) + abs(a.y) < abs(b.x) + abs(b.y);
}
```

3.6 MinimumDistance

```
template<class T>
T MinimumDistance(vector<Point<T>>&p,int l = -1,int r = -1){
  if(l==-1 and r==-1){
    sort(p.begin(),p.end(),[](Point<T> a,Point<T> b){
      if(a.x!=b.x)return a.x<b.x;
      return a.y<b.y;
    });
    p.erase(unique(p.begin(),p.end()),p.end());
    return MinimumDistance(p,0,p.size()-1);
  }
  if(l==r)return numeric_limits<T>::max();
  int m = (l+r)>>1,mid_pos = p[m].x;
  T ans = min(MinimumDistance(p,l,m),MinimumDistance(p,m+1,r));
  vector<Point<T>>tmp((r-l+1),Point<T>(0,0));
  merge(p.begin()+l,p.begin()+m+1, p.begin()+m+1,p.begin()+r+1, tmp.begin(), [](Point<T> a,Point<T> b){return a.y<b.y;});
  for(int i = l;i<=r;++i)p[i] = tmp[i-l];
  tmp.clear();
  for(int i = l;i<=r;++i){
    if((p[i].x-mid_pos)*(p[i].x-mid_pos)<ans){
      tmp.push_back(p[i]);
    }
  }
  int n = tmp.size();
  for(int i = 0;i<n;++i){
    for(int j = i+1;j<n;++j){
      ans = min(ans,abs2(tmp[i]-tmp[j]));
      if(((tmp[i].y-tmp[j].y)*(tmp[i].y-tmp[j].y))>ans){
        break;
      }
    }
  }
  return ans;
}
```

3.7 ConvexHull

```
template<class T>
vector<Point<T>> ConvexHull(vector<Point<T>> v,bool Boundary = 1){
  sort(begin(v),end(v),[&](Point<T> &a,Point<T> &b){
    if(a.x!=b.x)return a.x<b.x;
    return a.y<b.y;
  });
  vector<Point<T>>ans;
  int t = 1;
  auto add = [&](Point<T> &p){
    while(ans.size() > t and ((p - ans[ans.size() - 2])^(ans.back() - ans[ans.size() - 2])) > (Boundary ? 0 : 0-eps))
      ans.pop_back();
    ans.push_back(p);
  };
  for(int i = 0; i < v.size(); ++i) add(v[i]);
  t = ans.size();
  for(int i = (int)(v.size())-2; i >= 0; --i) add(v[i]);
  if(v.size() > 1) ans.pop_back();
  return ans;
}
```

3.8 Template

```
template<class T>
struct Point{
  T x,y;
  Point(T x = 0,T y = 0) : x(x), y(y) {}
  Point operator + (const Point &b) const {
    return Point(x + b.x,y + b.y);
  }
  Point operator - (const Point &b) const {
    return Point(x - b.x,y - b.y);
  }
  Point operator * (T b) const {
    return Point(x*b,y*b);
  }
  Point operator / (T b) const {
    return Point(x/b,y/b);
  }
  T operator * (const Point &b) const {
    return x * b.x + y * b.y;
  }
  T operator ^ (const Point &b) const {
    return x * b.y - y * b.x;
  }
};
int sign(double a){
  return fabs(a) < eps ? 0 : a > 0 ? 1 : -1;
}
template<class T>
double abs(const Point<T>&p){
  return sqrtl(p*p);
}
template<class T>
T abs2(const Point<T>&p){
  return p*p;
}
template<class T>
int ori(Point<T> a,Point<T> b,Point<T> c){
  return sign((b-a)^(c-a));
}
template<class T>
bool collinearity(Point<T> p1,Point<T> p2,Point<T> p3){
  return sign((p1-p3)^(p2-p3)) == 0;
}
template<class T>
bool btw(Point<T> p1,Point<T> p2,Point<T> p3) {
  if(!collinearity(p1, p2, p3)) return 0;
  return sign((p1-p3)*(p2-p3)) <= 0;
}
template<class T>
bool PointOnSegment(const Point<T> &p1,const Point<T> &p2,const Point<T> &p3){
  return collinearity(p1,p2,p3) && btw(p1,p2,p3);
}
template<class T>
bool seg_intersect(Point<T> p1, Point<T> p2, Point<T> p3, Point<T> p4) {
  int a123 = ori(p1, p2, p3);
  int a124 = ori(p1, p2, p4);
  int a341 = ori(p3, p4, p1);
  int a342 = ori(p3, p4, p2);
  if(a123 == 0 && a124 == 0)
    return btw(p1, p2, p3) || btw(p1, p2, p4) || btw(p3, p4, p1) || btw(p3, p4, p2);
  return a123 * a124 <= 0 && a341 * a342 <= 0;
}
template<class T>
double area(vector<Point<T>> v){
  if(v.size()<=2)return 0;
  double ans = 0;
  for(int i = 1;i<v.size()-1;++i){
    ans+=((v[i]-v[0])^(v[i+1]-v[0]));
  }
  return abs(ans)/2.;
}
```

4 Graph

4.1 HLD

```
struct heavy_light_decomposition{
  int n;
  vector<int>dep,father,sz,mxson,topf,id;
  vector<vector<int>>g;
  heavy_light_decomposition(int _n = 0) : n(_n) {
    g.resize(n+5);
    dep.resize(n+5);
    father.resize(n+5);
    sz.resize(n+5);
    mxson.resize(n+5);
    topf.resize(n+5);
    id.resize(n+5);
  }
  void add_edge(int u, int v){
    g[u].push_back(v);
    g[v].push_back(u);
  }
  void dfs(int u,int p){
    dep[u] = dep[p]+1;
    father[u] = p;
    sz[u] = 1;
    mxson[u] = 0;
    for(auto v:g[u]){
      if(v==p)continue;
      dfs(v,u);
      sz[u]+=sz[v];
      if(sz[v]>sz[mxson[u]])mxson[u] = v;
    }
  }
  void dfs2(int u,int top){
    static int idn = 0;
    topf[u] = top;
    id[u] = ++idn;
    if(mxson[u])dfs2(mxson[u],top);
    for(auto v:g[u]){
      if(v!=father[u] and v!=mxson[u]){
        dfs2(v,v);
      }
    }
  }
  void build(int root){
    dfs(root,0);
    dfs2(root,root);
  }
  vector<pair<int, int>> path(int u,int v){
    vector<pair<int, int>>ans;
    while(topf[u]!=topf[v]){
      if(dep[topf[u]]<dep[topf[v]])swap(u,v);
      ans.push_back({id[topf[u]], id[u]});
      u = father[topf[u]];
    }
    if(id[u]>id[v])swap(u,v);
    ans.push_back({id[u], id[v]});
    return ans;
  }
};
```

4.2 Bridges

```
vector<pii> findBridges(const vector<vector<int>>& g) {
  int n = (int) g.size();
  vector<int> id(n, -1), low(n);
  vector<pii> bridges;
  function<void(int, int)> dfs = [&](int u, int p) {
    static int cnt = 0;
    id[u] = low[u] = cnt++;
    for(auto v : g[u]) {
      if(v == p) continue;
      if(id[v] != -1) low[u] = min(low[u], id[v]);
      else {
        dfs(v, u);
        low[u] = min(low[u], low[v]);
        if(low[v] > id[u]) bridges.EB(u, v);
      }
    }
  };
  for(int i = 0; i < n; ++i) {
    if(id[i] == -1) dfs(i, -1);
  }
  return bridges;
}
```

4.3 TwoSat

```
struct two_sat{
  SCC s;
  vector<bool>ans;
  int have_ans = 0;
  int n;
  two_sat(int _n) : n(_n) {
    ans.resize(n+1);
    s = SCC(2*n);
  }
  int inv(int x){
    if(x>n)return x-n;
    return x+n;
  }
  void add_or_clause(int u, bool x, int v, bool y){
    if(!x)u = inv(u);
    if(!y)v = inv(v);
    s.add_edge(inv(u), v);
    s.add_edge(inv(v), u);
  }
  void check(){
    if(have_ans!=0)return;
    s.build();
    for(int i = 0;i<=n;++i){
      if(s.scc[i]==s.scc[inv(i)]){
        have_ans = -1;
        return;
      }
      ans[i] = (s.scc[i]<s.scc[inv(i)]);
    }
    have_ans = 1;
  }
};
```

4.4 MCMF

```cpp
template<class Cap_t, class Cost_t>
class MCMF {
public:
  struct Edge {
    int from;
    int to;
    Cap_t cap;
    Cost_t cost;
    Edge(int u, int v, Cap_t _cap, Cost_t _cost) : from(u), to(
         v), cap(_cap), cost(_cost) {}
  };

  static constexpr Cap_t EPS = static_cast<Cap_t>(1e-9);

  int n;
  vector<Edge> edges;
  vector<vector<int>> g;
  vector<Cost_t> d;
  vector<bool> in_queue;
  vector<int> previous_edge;

  MCMF() {}
  MCMF(int _n) : n(_n+1), g(_n+1), d(_n+1), in_queue(_n+1),
       previous_edge(_n+1) {}

  void add_edge(int u, int v, Cap_t cap, Cost_t cost) {
    assert(0 <= u && u < n);
    assert(0 <= v && v < n);
    g[u].push_back(edges.size());
    edges.emplace_back(u, v, cap, cost);
    g[v].push_back(edges.size());
    edges.emplace_back(v, u, 0, -cost);
  }

  bool spfa(int s, int t) {
    bool found = false;
    fill(d.begin(), d.end(), numeric_limits<Cost_t>::max());
    d[s] = 0;
    in_queue[s] = true;
    queue<int> que;
    que.push(s);
    while(!que.empty()) {
      int u = que.front();
      que.pop();
      if(u == t) {
        found = true;
      }
      in_queue[u] = false;
      for(auto& id : g[u]) {
        const Edge& e = edges[id];
        if(e.cap > EPS && d[u] + e.cost < d[e.to]) {
          d[e.to] = d[u] + e.cost;
          previous_edge[e.to] = id;
          if(!in_queue[e.to]) {
            que.push(e.to);
            in_queue[e.to] = true;
          }
        }
      }
    }
    return found;
  }

  pair<Cap_t, Cost_t> flow(int s, int t, Cap_t f =
       numeric_limits<Cap_t>::max()) {
    assert(0 <= s && s < n);
    assert(0 <= t && t < n);
    Cap_t cap = 0;
    Cost_t cost = 0;
    while(f > 0 && spfa(s, t)) {
      Cap_t send = f;
      int u = t;
      while(u != s) {
        const Edge& e = edges[previous_edge[u]];
        send = min(send, e.cap);
        u = e.from;
      }
      u = t;
      while(u != s) {
        Edge& e = edges[previous_edge[u]];
        e.cap -= send;
        Edge& b = edges[previous_edge[u] ^ 1];
        b.cap += send;
        u = e.from;
      }
      cap += send;
      f -= send;
      cost += send * d[t];
    }
    return make_pair(cap, cost);
  }
};
```

4.5 LCA

```cpp
vector<vector<int>>g,dp;
vector<int>deep;
void build(int root,int n){
  dp.assign(25,vector<int>(n+5));
  deep.assign(n+5,0);
  function<void(int,int,int)>dfs = [&](int u,int p,int dis){
    dp[0][u] = p;
    deep[u] = dis;
    for(auto v:g[u]){
      if(v==p)continue;
      dfs(v,u,dis+1);
    }
  };
  dfs(root,0,1);
  for(int i = 1;i<=20;++i){
    for(int j = 1;j<=n;++j){
      dp[i][j] = dp[i-1][dp[i-1][j]];
    }
  }
}
int LCA(int u,int v){
  if(deep[u]<deep[v])swap(u,v);
  for(int i = 20;i>=0;--i){
    if(deep[dp[i][u]]>=deep[v])
      u = dp[i][u];
  }
  if(u==v)return u;
  for(int i = 20;i>=0;--i){
    if(dp[i][u]!=dp[i][v])u = dp[i][u],v = dp[i][v];
  }
  return dp[0][u];
}
```

4.6 CentroidDecomposition

```cpp
vector<vector<int>>g;
vector<int>sz,tmp;
vector<bool>vis;//visit_centroid
int tree_centroid(int u,int n){
  function<void(int,int)>dfs1 = [&](int u,int p){
    sz[u] = 1;
    for(auto v:g[u]){
      if(v==p)continue;
      if(vis[v])continue;
      dfs1(v,u);
      sz[u]+=sz[v];
    }
  };
  function<int(int,int)>dfs2 = [&](int u,int p){
    for(auto v:g[u]){
      if(v==p)continue;
      if(vis[v])continue;
      if(sz[v]*2<n)continue;
      return dfs2(v,u);
    }
    return u;
  };
  dfs1(u,-1);
  return dfs2(u,-1);
}
int cal(int u,int p = -1,int deep = 1){
  int ans = 0;
  tmp.pb(deep);
  sz[u] = 1;
  for(auto v:g[u]){
    if(v==p)continue;
    if(vis[v])continue;
    ans+=cal(v,u,deep+1);
    sz[u]+=sz[v];
  }
  //calcuate the answer
  return ans;
}
int centroid_decomposition(int u,int tree_size){
  int center = tree_centroid(u,tree_size);
  vis[center] = 1;
  int ans = 0;
  for(auto v:g[center]){
    if(vis[v])continue;
    ans+=cal(v);
    for(int i = sz(tmp)-sz[v];i<sz(tmp);++i){
      //update
    }
  }
  while(!tmp.empty()){
    //roll_back(tmp.back())
    tmp.pop_back();
  }
  for(auto v:g[center]){
    if(vis[v])continue;
    ans+=centroid_decomposition(v,sz[v]);
  }
  return ans;
}
```

4.7 BCC AP

```
struct BCC_AP{
  int dfn_cnt = 0,bcc_cnt = 0,n;
  vector<int>dfn,low,ap,bcc_id;
  stack<int>st;
  vector<bool>vis,is_ap;
  vector<vector<int>>bcc;
  BCC_AP(int _n):n(_n){
    dfn.resize(n+5),low.resize(n+5),bcc.resize(n+5),vis.resize(n+5),is_ap.resize(n+5),bcc_id.resize(n+5);
  }
  inline void build(const vector<vector<int>>&g,int u,int p = -1){
    int child = 0;
    dfn[u] = low[u] = ++dfn_cnt;
    st.push(u);
    vis[u] = 1;
    if(g[u].empty() and p==-1){
      bcc_id[u] = ++bcc_cnt;
      bcc[bcc_cnt].push_back(u);
      return;
    }
    for(auto v:g[u]){
      if(v==p)continue;
      if(!dfn[v]){
        build(g,v,u);
        child++;
        if(dfn[u]<=low[v]){
          is_ap[u] = 1;
          bcc_id[u] = ++bcc_cnt;
          bcc[bcc_cnt].push_back(u);
          while(vis[v]){
            bcc_id[st.top()] = bcc_cnt;
            bcc[bcc_cnt].push_back(st.top());
            vis[st.top()] = 0;
            st.pop();
          }
        }
        low[u] = min(low[u],low[v]);
      }
      low[u] = min(low[u],dfn[v]);
    }
    if(p==-1 and child<2)is_ap[u] = 0;
    if(is_ap[u])ap.push_back(u);
  }
};
```

4.8 SCC

```
struct SCC{
  int n,cnt = 0,dfn_cnt = 0;
  vector<vector<int>>g;
  vector<int>sz,scc,low,dfn;
  stack<int>st;
  vector<bool>vis;
  SCC(int _n = 0) : n(_n){
    sz.resize(n+5),scc.resize(n+5),low.resize(n+5),dfn.resize(n+5),vis.resize(n+5);
    g.resize(n+5);
  }
  inline void add_edge(int u, int v){
    g[u].push_back(v);
  }
  inline void build(){
    function<void(int, int)>dfs = [&](int u,int dis){
      low[u] = dfn[u] = ++dfn_cnt,vis[u] = 1;
      st.push(u);
      for(auto v:g[u]){
        if(!dfn[v]){
          dfs(v, dis+1);
          low[u] = min(low[u],low[v]);
        }
        else if(vis[v]){
          low[u] = min(low[u],dfn[v]);
        }
      }
      if(low[u]==dfn[u]){
        ++cnt;
        while(vis[u]){
          auto v = st.top();
          st.pop();
          vis[v] = 0;
          scc[v] = cnt;
          sz[cnt]++;
        }
      }
    };
    for(int i = 0;i<=n;++i){
      if(!scc[i]){
        dfs(i, 1);
      }
    }
  }
  vector<vector<int>> compress(){
    vector<vector<int>>ans(cnt+1);
    for(int u = 0;u<=n;++u){
      for(auto v:g[u]){
        if(scc[u] == scc[v]){
          continue;
        }
        ans[scc[u]].push_back(scc[v]);
      }
    }
    for(int i = 0;i<=cnt;++i){
      sort(ans[i].begin(), ans[i].end());
      ans[i].erase(unique(ans[i].begin(), ans[i].end()), ans[i].end());
    }
    return ans;
  }
};
```

4.9 LineContainer

```
template<class T>
T floor_div(T a, T b) {
  return a / b - ((a ^ b) < 0 && a % b != 0);
}

template<class T>
T ceil_div(T a, T b) {
  return a / b + ((a ^ b) > 0 && a % b != 0);
}

namespace line_container_internal {

struct line_t {
  mutable long long k, m, p;

  inline bool operator<(const line_t& o) const { return k < o.k; }
  inline bool operator<(long long x) const { return p < x; }
};

} // line_container_internal

template<bool MAX>
struct line_container : std::multiset<line_container_internal::line_t, std::less<>> {
  static const long long INF = std::numeric_limits<long long>::max();

  bool isect(iterator x, iterator y) {
    if(y == end()) {
      x->p = INF;
      return 0;
    }
    if(x->k == y->k) {
      x->p = (x->m > y->m ? INF : -INF);
    } else {
      x->p = floor_div(y->m - x->m, x->k - y->k);
    }
    return x->p >= y->p;
  }

  void add_line(long long k, long long m) {
    if(!MAX) {
      k = -k;
      m = -m;
    }
    auto z = insert({k, m, 0}), y = z++, x = y;
    while(isect(y, z)) {
      z = erase(z);
    }
    if(x != begin() && isect(--x, y)) {
      isect(x, y = erase(y));
    }
    while((y = x) != begin() && (--x)->p >= y->p) {
      isect(x, erase(y));
    }
  }

  long long get(long long x) {
    assert(!empty());
    auto l = *lower_bound(x);
    return (l.k * x + l.m) * (MAX ? +1 : -1);
  }
};
```

4.10 Dinic

```cpp
template<class T>
struct Dinic{
  struct edge{
    int from, to;
    T cap;
    edge(int _from, int _to, T _cap) : from(_from), to(_to), cap(_cap) {}
  };
  int n;
  vector<edge> edges;
  vector<vector<int>> g;
  vector<int> cur, h;
  Dinic(int _n) : n(_n+1), g(_n+1) {}
  void add_edge(int u, int v, T cap){
    g[u].push_back(edges.size());
    edges.push_back(edge(u, v, cap));
    g[v].push_back(edges.size());
    edges.push_back(edge(v, u, 0));
  }
  bool bfs(int s,int t){
    h.assign(n, -1);
    h[s] = 0;
    queue<int> que;
    que.push(s);
    while(!que.empty()) {
      int u = que.front();
      que.pop();
      for(auto id : g[u]) {
        const edge& e = edges[id];
        int v = e.to;
        if(e.cap > 0 && h[v] == -1) {
          h[v] = h[u] + 1;
          if(v == t) {
            return 1;
          }
          que.push(v);
        }
      }
    }
    return 0;
  }
  T dfs(int u, int t, T f) {
    if(u == t) {
      return f;
    }
    T r = f;
    for(int& i = cur[u]; i < (int) g[u].size(); ++i) {
      int id = g[u][i];
      const edge& e = edges[id];
      int v = e.to;
      if(e.cap > 0 && h[v] == h[u] + 1) {
        T send = dfs(v, t, min(r, e.cap));
        edges[id].cap -= send;
        edges[id ^ 1].cap += send;
        r -= send;
        if(r == 0) {
          return f;
        }
      }
    }
    return f - r;
  }
  T flow(int s, int t, T f = numeric_limits<T>::max()) {
    T ans = 0;
    while(f > 0 && bfs(s, t)) {
      cur.assign(n, 0);
      T send = dfs(s, t, f);
      ans += send;
      f -= send;
    }
    return ans;
  }
  vector<pair<int,int>> min_cut(int s) {
    vector<bool> vis(n);
    vis[s] = true;
    queue<int> que;
    que.push(s);
    while(!que.empty()) {
      int u = que.front();
      que.pop();
      for(auto id : g[u]) {
        const auto& e = edges[id];
        int v = e.to;
        if(e.cap > 0 && !vis[v]) {
          vis[v] = true;
          que.push(v);
        }
      }
    }
    vector<pair<int,int>> cut;
    for(int i = 0; i < (int) edges.size(); i += 2) {
      const auto& e = edges[i];
      if(vis[e.from] && !vis[e.to]) {
        cut.push_back(make_pair(e.from, e.to));
      }
    }
    return cut;
  }
};
```

5 Math

5.1 Numbers

- Bernoulli numbers

  $B_0 - 1, B_1^{\pm} = \pm\frac{1}{2}, B_2 = \frac{1}{6}, B_3 = 0$

  $$\sum_{j=0}^{m} \binom{m+1}{j} B_j = 0, \text{EGF is } B(x) = \tfrac{x}{e^x - 1} = \sum_{n=0}^{\infty} B_n \frac{x^n}{n!}.$$

  $$S_m(n) = \sum_{k=1}^{n} k^m = \frac{1}{m+1} \sum_{k=0}^{m} \binom{m+1}{k} B_k^+ n^{m+1-k}$$

- Stirling numbers of the second kind Partitions of n distinct elements into exactly k groups.

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

  $S(n,k) = \frac{1}{k!}\sum_{i=0}^{k}(-1)^{k-i}\binom{k}{i}i^n$

  $x^n = \sum_{i=0}^{n} S(n,i)(x)_i$

- Pentagonal number theorem

  $$\prod_{n=1}^{\infty}(1-x^n) = 1 + \sum_{k=1}^{\infty}(-1)^k\left(x^{k(3k+1)/2} + x^{k(3k-1)/2}\right)$$

- Catalan numbers

  $$C_n^{(k)} = \frac{1}{(k-1)n+1}\binom{kn}{n}$$

  $C^{(k)}(x) = 1 + x[C^{(k)}(x)]^k$

- Eulerian numbers

  Number of permutations $\pi \in S_n$ in which exactly k elements are greater than the previous element. k j:s s.t. $\pi(j) > \pi(j+1)$, $k+1$ j:s s.t. $\pi(j) \geq j$, k j:s s.t. $\pi(j) > j$.

  $E(n,k) = (n-k)E(n-1,k-1) + (k+1)E(n-1,k)$

  $E(n,0) = E(n,n-1) = 1$

  $E(n,k) = \sum_{j=0}^{k}(-1)^j\binom{n+1}{j}(k+1-j)^n$

5.2 ExtendGCD

```cpp
// @return $x, y$ s.t. $ax + by = \gcd(a, b)$
#define ll long long
ll ext_gcd(ll a, ll b, ll& x, ll& y) {
  if(b == 0) {
    x = 1; y = 0;
    return a;
  }
  ll x2, y2;
  ll c = a % b;
  if(c < 0) c += b;
  ll g = ext_gcd(b, c, x2, y2);
  x = y2;
  y = x2 - (a / b) * y2;
  return g;
}
//a^{-1} % p = x % p
```

5.3 InvGCD

```cpp
pair<long long, long long> inv_gcd(long long a, long long b) {
  a %= b;
  if(a < 0) a += b;
  if(a == 0) return {b, 0};
  long long s = b, t = a;
  long long m0 = 0, m1 = 1;
  while(t) {
    long long u = s / t;
    s -= t * u;
    m0 -= m1 * u;
    swap(s, t);
    swap(m0, m1);
  }
  if(m0 < 0) m0 += b / s;
  return {s, m0};
}
```

5.4 GeneratingFunctions

- Ordinary Generating Function $A(x) = \sum_{i \geq 0} a_i x^i$
  - $A(rx) \Rightarrow r^n a_n$
  - $A(x) + B(x) \Rightarrow a_n + b_n$
  - $A(x)B(x) \Rightarrow \sum_{i=0}^{n} a_i b_{n-i}$
  - $A(x)^k \Rightarrow \sum_{i_1+i_2+\cdots+i_k=n} a_{i_1} a_{i_2} \ldots a_{i_k}$
  - $xA(x)' \Rightarrow na_n$
  - $\frac{A(x)}{1-x} \Rightarrow \sum_{i=0}^{n} a_i$
- Exponential Generating Function $A(x) = \sum_{i \geq 0} \frac{a_i}{i!} x_i$
  - $A(x) + B(x) \Rightarrow a_n + b_n$
  - $A^{(k)}(x) \Rightarrow a_{n+k}$
  - $A(x)B(x) \Rightarrow \sum_{i=0}^{n} \binom{n}{i} a_i b_{n-i}$
  - $A(x)^k \Rightarrow \sum_{i_1+i_2+\cdots+i_k=n} \binom{n}{i_1, i_2, \ldots, i_k} a_{i_1} a_{i_2} \ldots a_{i_k}$
  - $xA(x) \Rightarrow na_n$
- Special Generating Function
  - $(1+x)^n = \sum_{i \geq 0} \binom{n}{i} x^i$
  - $\frac{1}{(1-x)^n} = \sum_{i \geq 0} \binom{i}{n-1} x^i$

5.5 Theorem

- Modular Arithmetic

$$(a + b) \bmod m = (a \bmod m + b \bmod m) \bmod m$$

$$(a - b) \bmod m = (a \bmod m - b \bmod m) \bmod m$$

$$(a \cdot b) \pmod{m} = ((a \bmod m) \cdot (b \bmod m)) \bmod m$$

$$a^b \bmod m = (a \bmod m)^{b \bmod m-1} \bmod m$$

- Cramer's rule

$$\begin{array}{l} ax + by = e \\ cx + dy = f \end{array} \Rightarrow \begin{array}{l} x = \dfrac{ed - bf}{ad - bc} \\ y = \dfrac{af - ec}{ad - bc} \end{array}$$

- Kirchhoff's Theorem

  Denote L be a $n \times n$ matrix as the Laplacian matrix of graph G, where $L_{ii} = d(i)$, $L_{ij} = -c$ where c is the number of edge (i, j) in G.

  - The number of undirected spanning in G is $|\det(\tilde{L}_{11})|$.
  - The number of directed spanning tree rooted at r in G is $|\det(\tilde{L}_{rr})|$.

- Tutte's Matrix

  Let D be a $n \times n$ matrix, where $d_{ij} = x_{ij}$ (x_{ij} is chosen uniformly at random) if $i < j$ and $(i, j) \in E$, otherwise $d_{ij} = -d_{ji}$. $\frac{rank(D)}{2}$ is the maximum matching on G.

- Cayley's Formula
  - Given a degree sequence $d_1, d_2, \ldots, d_n$ for each labeled vertices, there are $\frac{(n-2)!}{(d_1-1)!(d_2-1)!\cdots(d_n-1)!}$ spanning trees.
  - Let $T_{n,k}$ be the number of labeled forests on n vertices with k components, such that vertex $1, 2, \ldots, k$ belong to different components. Then $T_{n,k} = kn^{n-k-1}$.

- Erd□s–Gallai theorem

  A sequence of nonnegative integers $d_1 \geq \cdots \geq d_n$ can be represented as the degree sequence of a finite simple graph on n vertices if and only if $d_1 + \cdots + d_n$ is even and $\sum\limits_{i-1}^{k} d_i \leq k(k-1) + \sum\limits_{i=k+1}^{n} \min(d_i, k)$ holds for every $1 \leq k \leq n$.

- Gale–Ryser theorem

  A pair of sequences of nonnegative integers $a_1 \geq \cdots \geq a_n$ and $b_1, \ldots, b_n$ is bigraphic if and only if $\sum\limits_{i=1}^{n} a_i = \sum\limits_{i=1}^{n} b_i$ and $\sum\limits_{i=1}^{k} a_i \leq \sum\limits_{i=1}^{n} \min(b_i, k)$ holds for every $1 \leq k \leq n$.

- Fulkerson–Chen–Anstee theorem

  A sequence $(a_1, b_1), \ldots, (a_n, b_n)$ of nonnegative integer pairs with $a_1 \geq \cdots \geq a_n$ is digraphic if and only if $\sum\limits_{i=1}^{n} a_i = \sum\limits_{i=1}^{n} b_i$ and $\sum\limits_{i=1}^{k} a_i \leq \sum\limits_{i=1}^{k} \min(b_i, k-1) + \sum\limits_{i=k+1}^{n} \min(b_i, k)$ holds for every $1 \leq k \leq n$.

- M□bius inversion formula
  - $f(n) = \sum_{d|n} g(d) \Leftrightarrow g(n) = \sum_{d|n} \mu(d) f(\frac{n}{d})$
  - $f(n) = \sum_{n|d} g(d) \Leftrightarrow g(n) = \sum_{n|d} \mu(\frac{d}{n}) f(d)$

- Spherical cap
  - A portion of a sphere cut off by a plane.
  - r: sphere radius, a: radius of the base of the cap, h: height of the cap, θ: $\arcsin(a/r)$.
  - Volume $= \pi h^2(3r - h)/3 = \pi h(3a^2 + h^2)/6 = \pi r^3(2 + \cos\theta)(1 - \cos\theta)^2/3$.
  - Area $= 2\pi rh = \pi(a^2 + h^2) = 2\pi r^2(1 - \cos\theta)$.

5.6 FloorSum

```cpp
//f(a, b, c, n) = \sum_{i = 0}^{n - 1} \lfloor \frac{ai + b}{c}\rfloor
long long floor_sum(long long a, long long b, long long c, long long n) {
  long long ans = 0;
  if(a >= c) {
    ans += (n - 1) * n * (a / c) / 2;
    a %= c;
  }
  if(b >= c) {
    ans += n * (b / c);
    b %= c;
  }
  long long y_max = (a * n + b) / c;
  long long x_max = y_max * c - b;
  if(y_max == 0) {
    return ans;
  }
  ans += (n - (x_max + a - 1) / a) * y_max;
  return ans + floor_sum(c, (a - x_max % a) % a, a, y_max);
}
```

5.7 GuessKth

```cpp
template <typename Tfield>
std::pair<int, std::vector<Tfield>> find_linear_recurrence(const std::vector<Tfield> &S) {
    int N = S.size();
    using poly = std::vector<Tfield>;
    poly C_reversed{1}, B{1};
    int L = 0, m = 1;
    Tfield b = 1;

    // adjust: C(x) <- C(x) - (d / b) x^m B(x)
    auto adjust = [](poly C, const poly &B, Tfield d, Tfield b, int m) -> poly {
        C.resize(std::max(C.size(), B.size() + m));
        Tfield a = d / b;
        for (unsigned i = 0; i < B.size(); i++) C[i + m] -= a * B[i];
        return C;
    };

    for (int n = 0; n < N; n++) {
        Tfield d = S[n];
        for (int i = 1; i <= L; i++) d += C_reversed[i] * S[n - i];

        if (d == 0)
            m++;
        else if (2 * L <= n) {
            poly T = C_reversed;
            C_reversed = adjust(C_reversed, B, d, b, m);
            L = n + 1 - L;
            B = T;
            b = d;
            m = 1;
        } else
            C_reversed = adjust(C_reversed, B, d, b, m++);
    }
    return std::make_pair(L, C_reversed);
}

// Calculate $x^N \bmod f(x)$
// Known as `Kitamasa method`
// Input: f_reversed: monic, reversed (f_reversed[0] = 1)
// Complexity: $O(K^2 \log N)$ ($K$: deg. of $f$)
// Example: (4, [1, -1, -1]) -> [2, 3]
//         ( x^4 = (x^2 + x + 2)(x^2 - x - 1) + 3x + 2 )
// Reference: http://misawa.github.io/others/fast_kitamasa_method.html
//            http://sugarknri.hatenablog.com/entry/2017/11/18/233936
template <typename Tfield>
std::vector<Tfield> monomial_mod_polynomial(long long N, const std::vector<Tfield> &f_reversed) {
    assert(!f_reversed.empty() and f_reversed[0] == 1);
    int K = f_reversed.size() - 1;
    if (!K) return {};
    int D = 64 - __builtin_clzll(N);
    std::vector<Tfield> ret(K, 0);
    ret[0] = 1;
    auto self_conv = [](std::vector<Tfield> x) -> std::vector<Tfield> {
        int d = x.size();
        std::vector<Tfield> ret(d * 2 - 1);
        for (int i = 0; i < d; i++) {
            ret[i * 2] += x[i] * x[i];
            for (int j = 0; j < i; j++) ret[i + j] += x[i] * x[j] * 2;
        }
        return ret;
    };
    for (int d = D; d--;) {
        ret = self_conv(ret);
        for (int i = 2 * K - 2; i >= K; i--) {
            for (int j = 1; j <= K; j++) ret[i - j] -= ret[i] * f_reversed[j];
        }
        ret.resize(K);
        if ((N >> d) & 1) {
            std::vector<Tfield> c(K);
            c[0] = -ret[K - 1] * f_reversed[K];
            for (int i = 1; i < K; i++) { c[i] = ret[i - 1] - ret[K - 1] * f_reversed[K - i]; }
            ret = c;
        }
    }
    return ret;
}

// Guess k-th element of the sequence, assuming linear recurrence
// initial_elements: 0-ORIGIN
// Verify: abc198f https://atcoder.jp/contests/abc198/submissions/21837815
template <typename Tfield>
Tfield guess_kth_term(const std::vector<Tfield> &initial_elements, long long k) {
    assert(k >= 0);
    if (k < static_cast<long long>(initial_elements.size())) return initial_elements[k];
    const auto f = find_linear_recurrence<Tfield>(initial_elements).second;
    const auto g = monomial_mod_polynomial<Tfield>(k, f);
    Tfield ret = 0;
    for (unsigned i = 0; i < g.size(); i++) ret += g[i] * initial_elements[i];
    return ret;
}
```

5.8 PowMod

```
constexpr long long Pow(long long x, long long n, int m) {
  if(m == 1) return 0;
  unsigned int _m = (unsigned int)(m);
  unsigned long long r = 1;
  x %= m;
  if(x < 0) x += m;
  unsigned long long y = x;
  while(n) {
    if(n & 1) r = (r * y) % _m;
    y = (y * y) % _m;
    n >>= 1;
  }
  return r;
}
```

5.9 ModInt

```
template<int id>
struct modint {
public:
  static constexpr int mod() { return id; }

  constexpr modint() : value(0) {}
  modint(long long x) : value(x % mod()) {
    if(value < 0) value += mod();
  }

  constexpr int val() const { return value; }

  constexpr modint inv() const {
    return Pow(value, mod()-2, mod());
  }

  constexpr modint& operator+=(const modint& rhs) & {
    value += rhs.value;
    if(value >= mod()) {
      value -= mod();
    }
    return *this;
  }

  constexpr modint& operator-=(const modint& rhs) & {
    value -= rhs.value;
    if(value < 0) {
      value += mod();
    }
    return *this;
  }

  constexpr modint& operator*=(const modint& rhs) & {
    value = 1LL * value * rhs.value % mod();
    return *this;
  }

  constexpr modint& operator/=(const modint& rhs) & {
    return *this *= rhs.inv();
  }

  friend constexpr modint operator+(modint lhs, modint rhs) { return lhs += rhs; }
  friend constexpr modint operator-(modint lhs, modint rhs) { return lhs -= rhs; }
  friend constexpr modint operator*(modint lhs, modint rhs) { return lhs *= rhs; }
  friend constexpr modint operator/(modint lhs, modint rhs) { return lhs /= rhs; }
  constexpr modint operator+() const { return *this; }
  constexpr modint operator-() const { return modint() - *this; }
  constexpr bool operator==(const modint& rhs) const { return value == rhs.value; }
  constexpr bool operator!=(const modint& rhs) const { return value != rhs.value; }


  int value;
};
using mint = modint<mod>;
```

5.10 CRT

```
//#include "InvGCD.h"
// @return
//  $\text{remainder, modulo}$
//      or
//  $0, 0$ if do not exist
pair<long long, long long> crt(const vector<long long>& r, const vector<long long>& m) {
  assert(r.size()==m.size());
  int n = r.size();
  // Contracts: 0 <= r0 < m0
  long long r0 = 0, m0 = 1;
  for(int i = 0; i < n; i++) {
    assert(1 <= m[i]);
    long long r1 = r[i] % m[i];
    if(r1 < 0) r1 += m[i];
    long long m1 = m[i];
    if(m0 < m1) {
      swap(r0, r1);
      swap(m0, m1);
    }
    if(m0 % m1 == 0) {
      if(r0 % m1 != r1) return {0, 0};
      continue;
    }
    long long g, im;
    tie(g, im) = inv_gcd(m0, m1);
    long long u1 = (m1 / g);
    if((r1 - r0) % g) return {0, 0};
    long long x = (r1 - r0) / g % u1 * im % u1;
    r0 += x * m0;
    m0 *= u1;
    if(r0 < 0) r0 += m0;
  }
  return {r0, m0};
}
```

5.11 DiscreteLog

```
//give you $a, b, m$ find $x$ such that $a^x \equiv m (\mod m)$
#line 2 "library/math/discrete-log.hpp"
#include <vector>
#include <cmath>
#include <cassert>
#line 2 "library/data-structure/pbds.hpp"
#include <ext/pb_ds/assoc_container.hpp>
#line 2 "library/random/splitmix64.hpp"
#include <chrono>
```

```cpp

namespace felix {

namespace internal {

struct splitmix64_hash {
  // http://xoshiro.di.unimi.it/splitmix64.c

  static unsigned long long splitmix64(unsigned long long x) {
    x += 0x9e3779b97f4a7c15;
    x = (x ^ (x >> 30)) * 0xbf58476d1ce4e5b9;
    x = (x ^ (x >> 27)) * 0x94d049bb133111eb;
    return x ^ (x >> 31);
  }

  unsigned long long operator()(unsigned long long x) const {
    static const unsigned long long FIXED_RANDOM = std::chrono::steady_clock::now().time_since_epoch().count();
    return splitmix64(x + FIXED_RANDOM);
  }
};

} // namespace internal


} // namespace felix

#line 4 "library/data-structure/pbds.hpp"

namespace felix {

template<class T, class U, class H = internal::splitmix64_hash> using hash_map = __gnu_pbds::gp_hash_table<T, U, H>;
template<class T, class H = internal::splitmix64_hash> using hash_set = hash_map<T, __gnu_pbds::null_type, H>;

template<class T, class Comp = std::less<T>> using ordered_set = __gnu_pbds::tree<T, __gnu_pbds::null_type, Comp, __gnu_pbds::rb_tree_tag, __gnu_pbds::tree_order_statistics_node_update>;
template<class T> using ordered_multiset = ordered_set<T, std::less_equal<T>>;

} // namespace felix
#line 2 "library/modint/barrett.hpp"

namespace felix {

namespace internal {

// Fast modular multiplication by barrett reduction

// Reference: https://en.wikipedia.org/wiki/Barrett_reduction

struct barrett {
  unsigned int m;
  unsigned long long im;

  explicit barrett(unsigned int _m) : m(_m), im((unsigned long long)(-1) / _m + 1) {}

  unsigned int umod() const { return m; }

  unsigned int mul(unsigned int a, unsigned int b) const {
    unsigned long long z = a;
    z *= b;
#ifdef _MSC_VER
    unsigned long long x;
    _umul128(z, im, &x);
#else
    unsigned long long x = (unsigned long long)(((unsigned __int128)(z) * im) >> 64);
#endif
    unsigned long long y = x * m;
    return (unsigned int)(z - y + (z < y ? m : 0));
  }
};

} // namespace internal


} // namespace felix

#line 2 "library/math/binary-gcd.hpp"

namespace felix {

template<class T>
inline T binary_gcd(T a, T b) {
  if(a == 0 || b == 0) {
    return a | b;
  }
  int8_t n = __builtin_ctzll(a);
  int8_t m = __builtin_ctzll(b);
  a >>= n;
  b >>= m;
  while(a != b) {
    T d = a - b;
    int8_t s = __builtin_ctzll(d);
    bool f = a > b;
    b = f ? b : a;
    a = (f ? d : -d) >> s;
  }
  return a << (n < m ? n : m);
}

} // namespace felix

#line 8 "library/math/discrete-log.hpp"

namespace felix {

int discrete_log(int a, int b, int m) {
  assert(b < m);
  if(b == 1 || m == 1) {
    return 0;
  }
  int n = (int) std::sqrt(m) + 1, e = 1, f = 1, j = 1;
  hash_map<int, int> baby;
  internal::barrett bt(m);
  while(j <= n && (e = f = bt.mul(e, a)) != b) {
    baby[bt.mul(e, b)] = j++;
  }
  if(e == b) {
    return j;
  }
  if(binary_gcd(m, e) == binary_gcd(m, b))  {
    for(int i = 2; i < n + 2; i++) {
      e = bt.mul(e, f);
      if(baby.find(e) != baby.end()) {
        return n * i - baby[e];
      }
    }
  }
  return -1;
}

} // namespace felix
```

5.12 LinearSieve

```cpp
vector<bool> is_prime;
vector<int> primes, phi, mobius, least;
void linear_sieve(int n) {
  n += 1;
  is_prime.resize(n);
  least.resize(n);
  fill(2 + begin(is_prime),end(is_prime), true);
  phi.resize(n); mobius.resize(n);
  phi[1] = mobius[1] = 1;
  least[0] = 0,least[1] = 1;
  for(int i = 2; i < n; ++i) {
    if(is_prime[i]) {
      primes.push_back(i);
      phi[i] = i - 1;
      mobius[i] = -1;
      least[i] = i;
    }
    for(auto j : primes) {
      if(i * j >= n) break;
      is_prime[i * j] = false;
      least[i * j] = j;
      if(i % j == 0) {
        mobius[i * j] = 0;
        phi[i * j] = phi[i] * j;
        break;
      } else {
        mobius[i * j] = mobius[i] * mobius[j];
        phi[i * j] = phi[i] * phi[j];
      }
    }
  }
}
```

6 Misc

6.1 FastIO

```cpp
inline char gc() {
    static const int BUF_SIZE = 1 << 22;
    static int Counts = 1 << 23;
    static char Buffer[BUF_SIZE];
    static char *Pointer = Buffer, *End = Buffer;
    if(Pointer == End) {
        if(Counts < BUF_SIZE) {
            return EOF;
        }
        Counts = fread(Buffer, 1, BUF_SIZE, stdin);
        Pointer = Buffer;
        End = Buffer + Counts;
    }
    return *(Pointer++);
}

template<class T>
inline void read(T& x) {
    static char c;
    do {
        c = gc();
    } while(c < '0' && c != '-');
    bool neg = (c == '-');
    if(!neg) {
        x = c - '0';
    } else x = 0;
    while((c = gc()) >= '0') {
        x = (x << 3) + (x << 1) + (c & 15);
    }
    if(neg) {
        x = -x;
    }
}

template<class T, class... U>
inline void read(T& a, U&... b) {
    read(a);
    read(b...);
}

template<class T>
inline void write(T temp, char end = '\n') {
    static short digits[20], P;
    if(temp == 0) {
        putchar_unlocked('0');
        putchar_unlocked(end);
        return;
    }
    if(temp < 0) {
        putchar_unlocked('-');
        write(-temp,end);
        return;
    }
    P = -1;
    while(temp) {
        digits[++P] = temp % 10;
        temp /= 10;
    }
    while(P >= 0) {
        putchar_unlocked(digits[P--] + '0');
    }
    putchar_unlocked(end);
    return;
}
```

6.2 Debug

```
#ifdef LOCAL
  #define eprintf(...) { fprintf(stderr, __VA_ARGS__); fflush(stderr); }
#else
  #define eprintf(...) 42
#endif
```

6.3 Discrete

```
template<class T>
vector<int> Discrete(const vector<T>&v){
  vector<int>ans;
  vector<T>tmp(v);
  sort(begin(tmp),end(tmp));
  tmp.erase(unique(begin(tmp),end(tmp)),end(tmp));
  for(auto i:v)ans.push_back(lower_bound(begin(tmp),end(tmp),i)-tmp.begin()+1);
  return ans;
}
```

6.4 DuiPai

```
#include<bits/stdc++.h>
using namespace std;
int main(){
  string sol,bf,make;
  cout<<"Your solution file name :";
  cin>>sol;
  cout<<"Brute force file name :";
  cin>>bf;
  cout<<"Make data file name :";
  cin>>make;
  system(("g++ "+sol+" -o sol").c_str());
  system(("g++ "+bf+" -o bf").c_str());
  system(("g++ "+make+" -o make").c_str());
  for(int t = 0;t<10000;++t){
    system("./make > ./1.in");
    double st = clock();
        system("./sol < ./1.in > ./1.ans");
        double et = clock();
        system("./bf < ./1.in > ./1.out");
        if(system("diff ./1.out ./1.ans")) {
      printf("\033[0;31mWrong Answer\033[0m on test #%d",t);
            return 0;
        }
    else if(et-st>=2000){
      printf("\033[0;32mTime limit exceeded\033[0m on test #%d, Time %.0lfms\n",t,et-st);
      return 0;
    }
    else {
            printf("\033[0;32mAccepted\033[0m on test #%d, Time %.0lfms\n", t, et - st);
        }
  }
}
```

6.5 Timer

```
const clock_t startTime = clock();
inline double getCurrentTime() {
  return (double) (clock() - startTime) / CLOCKS_PER_SEC;
}
```

6.6 TenarySearch

```
// return the maximum of $f(x)$ in $[l, r]$
double ternary_search(double l, double r) {
  while(r - l > EPS) {
    double m1 = l + (r - l) / 3;
    double m2 = r - (r - l) / 3;
    double f1 = f(m1), f2 = f(m2);
    if(f1 < f2) l = m1;
    else r = m2;
  }
  return f(l);
}

// return the maximum of $f(x)$ in $(l, r]$
int ternary_search(int l, int r) {
  while(r - l > 1) {
    int mid = (l + r) / 2;
    if(f(m) > f(m + 1)) r = m;
    else l = m;
  }
  return r;
}
```

7 Setup

7.1 Template

```
#include <bits/extc++.h>
#include <bits/stdc++.h>
#pragma GCC optimize("O3,unroll-loops")
#pragma GCC target("avx2,bmi,bmi2,lzcnt,popcnt")
#define IOS ios::sync_with_stdio(0),cin.tie(0),cout.tie(0)
#define int long long
#define double long double
#define pb push_back
#define sz(x) (int)(x).size()
#define all(v) begin(v),end(v)
#define debug(x) cerr<<#x<<" = "<<x<<'\n'
#define LINE cout<<"\n-----------------\n"
#define endl '\n'
#define VI vector<int>
#define F first
#define S second
#define MP(a,b) make_pair(a,b)
#define rep(i,m,n) for(int i = m;i<=n;++i)
#define res(i,m,n) for(int i = m;i>=n;--i)
#define gcd(a,b) __gcd(a,b)
#define lcm(a,b) a*b/gcd(a,b)
```

```cpp
#define Case() int _;cin>>_;for(int Case = 1;Case<=_;++Case)
#define pii pair<int,int>
using namespace __gnu_cxx;
using namespace __gnu_pbds;
using namespace std;
template <typename K, typename cmp = less<K>, typename T = thin_heap_tag> using _heap = __gnu_pbds::priority_queue<K, cmp, T>;
template <typename K, typename M = null_type> using _hash = gp_hash_table<K, M>;
const int N = 1e6+5,L = 20,mod = 1e9+7;
const long long inf = 2e18+5;
const double eps = 1e-7,pi = acos(-1);
void solve(){
}
signed main(){
  IOS;
  solve();
}
```

8 String

8.1 RollingHash

```cpp
template<int HASH_COUNT>
struct RollingHash {

  static const int MAX_HASH_PAIRS = 10;

  // {mul, mod}
  const vector<pair<int, int>> HASH_PAIRS = {{827167801, 999999937}, {998244353, 999999929}, {146672737, 922722049}, {204924373, 952311013}, {585761567, 955873937}, {484547929, 901981687}, {856009481, 987877511}, {852853249, 996724213}, {937381759, 994523539}, {116508269, 993179543}};

  int n;
  vector<int> POW[MAX_HASH_PAIRS];
  array<vector<int>, HASH_COUNT> pref;

  int substr(int k, int l, int r) {
    const auto& p = HASH_PAIRS[k];
    if(l == r) {
      return 0;
    }
    int res = pref[k][r - 1];
    if(l > 0) {
      res -= 1LL * pref[k][l - 1] * get_power(k, r - l) % p.second;
    }
    if(res < 0) {
      res += p.second;
    }
    return res;
  }

  // build powers up to x^k
  void build_powers(int k) {
    for(int i = 0; i < HASH_COUNT; ++i) {
      const auto& p = HASH_PAIRS[i];
      int sz = (int) POW[i].size();
      if(sz > k) {
        continue;
      }
      if(sz == 0) {
        POW[i].push_back(1);
        sz = 1;
      }
      while(sz <= k) {
        POW[i].push_back(1LL * POW[i].back() * p.first % p.second);
        sz += 1;
      }
    }
  }

  int get_power(int a, int b) {
    build_powers(b);
    return POW[a][b];
  }

  RollingHash() : RollingHash("") {}

  RollingHash(const string& s) : n(s.size()) {
    //static_assert(HASH_COUNT > 0 && HASH_COUNT <= MAX_HASH_PAIRS);
    for(int i = 0; i < HASH_COUNT; ++i) {
      const auto& p = HASH_PAIRS[i];
      pref[i].resize(n);
      pref[i][0] = s[0];
      for(int j = 1; j < n; ++j) {
        pref[i][j] = (1LL * pref[i][j - 1] * p.first + s[j]) % p.second;
      }
    }
    build_powers(n);
  }

  void add_char(char c) {
    for(int i = 0; i < HASH_COUNT; ++i) {
      const auto& p = HASH_PAIRS[i];
      pref[i].push_back((1LL * (n == 0 ? 0 : pref[i].back()) * p.first + c) % p.second);
    }
    n += 1;
    build_powers(n);
  }

  // Return hash values for [l, r)
  array<int, HASH_COUNT> substr(int l, int r) {
    array<int, HASH_COUNT> res{};
    for(int i = 0; i < HASH_COUNT; ++i) {
      res[i] = substr(i, l, r);
    }
    return res;
  }

  array<int, HASH_COUNT> merge(const vector<pair<int, int>>& seg) {
    array<int, HASH_COUNT> res{};
    for(int i = 0; i < HASH_COUNT; ++i) {
      const auto& p = HASH_PAIRS[i];
      for(auto [l, r] : seg) {
        res[i] = (1LL * res[i] * get_power(i, r - l) + substr(i, l, r)) % p.second;
      }
    }
    return res;
  }

  inline int size() const {
    return n;
  }
};
```

8.2 Z

```cpp
//z[i] := LCP(s, s[i, n)), z[0] is dont care
template<class T>
vector<int> Z(const vector<T>& a) {
  int n = (int) a.size();
  vector<int> z(n);
  for(int i = 1, j = 0; i < n; ++i) {
    if(i <= j + z[j]) {
      z[i] = min(z[i - j], j + z[j] - i);
    }
    while(i + z[i] < n && a[i + z[i]] == a[z[i]]) {
      z[i] += 1;
    }
    if(i + z[i] > j + z[j]) {
      j = i;
    }
  }
  return z;
}

vector<int> Z(const string& s) {
  return Z(vector<int>(s.begin(), s.end()));
}
```

8.3 KMP

```cpp
#line 2 "library/string/kmp.hpp"
template<class T>
vector<int> KMP(const vector<T>& a) {
  int n = (int) a.size();
  vector<int> k(n);
  for(int i = 1; i < n; ++i) {
    int j = k[i - 1];
    while(j > 0 && a[i] != a[j]) {
      j = k[j - 1];
    }
    j += (a[i] == a[j]);
    k[i] = j;
  }
  return k;
}

vector<int> KMP(const std::string& s) {
  return KMP(vector<int>(s.begin(), s.end()));
}
```

8.4 SuffixArray

```
struct suffix_array{
  int n;
  vector<int>SA,Rank,LCP;
  void counting_sort(vector<int>&v,auto getkey){
    int n = 0;
    for(auto i:v)n = max(n,getkey(i)+1);
    vector<int>bucket(n),ans(v.size());
    for(auto i:v)++bucket[getkey(i)];
    partial_sum(begin(bucket),end(bucket),begin(bucket));
    for(auto ite = v.rbegin();ite!=v.rend();++ite)ans[--bucket[getkey(*ite)]] = move(*ite);
    v.swap(ans);
    return;
  }
  suffix_array(string s):n(s.size()){
    SA.resize(n),Rank.resize(n),LCP.resize(n);
    for(int i = 0;i<n;++i)SA[i] = i;
    sort(SA.begin(),SA.end(),[&](int a,int b){
      return s[a]<s[b];
    });
    for(int i = 0;i<n;++i){
      Rank[SA[i]] = (i?Rank[SA[i-1]]+(s[SA[i]]!=s[SA[i-1]]):SA[0]);
    }
    for(int k = 0;(1<<k)<=n;++k){
      vector<int>idx;
      for(int i = n-(1<<k);i<n;++i)idx.push_back(i);
      for(auto i:SA)if(i>=(1<<k))idx.push_back(i-(1<<k));
      counting_sort(idx,[&](int a){return Rank[a];});
      SA.swap(idx);
      vector<int>new_rank(n);
      new_rank[SA[0]] = 0;
      for(int i = 1;i<n;++i){
        auto cmp = [&](int a,int b){
          return Rank[a]!=Rank[b] or a+(1<<k)>=n or Rank[a+(1<<k)]!=Rank[b+(1<<k)];
        };
        new_rank[SA[i]] = new_rank[SA[i-1]]+cmp(SA[i-1],SA[i]);
      }
      Rank.swap(new_rank);
    }
    for(int i = 0,k = 0;i<n;++i){
      if(Rank[i]==0)continue;
      if(k)--k;
      while(i+k<n and SA[Rank[i]-1]+k<n and s[i+k]==s[SA[Rank[i]-1]+k])++k;
      LCP[Rank[i]] = k;
    }
  }
};
```

8.5 Trie

```
template<int ALPHABET = 26, char MIN_CHAR = 'a'>
class trie {
public:
  struct Node {
    int go[ALPHABET];
    Node() {
      memset(go, -1, sizeof(go));
    }
  };

  trie() {
    newNode();
  }

  inline int next(int p, int v) {
    return nodes[p].go[v] != -1 ? nodes[p].go[v] : nodes[p].go[v] = newNode();
  }

  inline void insert(const vector<int>& a, int p = 0) {
    for(int v : a) {
      p = next(p, v);
    }
  }

  inline void clear() {
    nodes.clear();
    newNode();
  }

  inline int longest_common_prefix(const vector<int>& a, int p = 0) const {
    int ans = 0;
    for(int v : a) {
      if(nodes[p].go[v] != -1) {
        ans += 1;
        p = nodes[p].go[v];
      } else {
        break;
      }
    }
    return ans;
  }

private:
  vector<Node> nodes;

  inline int newNode() {
    nodes.emplace_back();
    return (int) nodes.size() - 1;
  }
};
```

ACM ICPC Team Reference - LeeJiaHuaPlayMinecraft

Contents

ACM ICPC Judge Test - LeeJiaHuaPlayMinecraft

C++ Resource Test

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

namespace system_test {

const size_t KB = 1024;
const size_t MB = KB * 1024;
const size_t GB = MB * 1024;

size_t block_size, bound;
void stack_size_dfs(size_t depth = 1) {
  if (depth >= bound)
    return;
  int8_t ptr[block_size]; // 若無法編譯將 block_size 改成常數
  memset(ptr, 'a', block_size);
  cout << depth << endl;
  stack_size_dfs(depth + 1);
}

void stack_size_and_runtime_error(size_t block_size, size_t bound = 1024) {
  system_test::block_size = block_size;
  system_test::bound = bound;
  stack_size_dfs();
}

double speed(int iter_num) {
  const int block_size = 1024;
  volatile int A[block_size];
  auto begin = chrono::high_resolution_clock::now();
  while (iter_num--)
    for (int j = 0; j < block_size; ++j)
      A[j] += j;
  auto end = chrono::high_resolution_clock::now();
  chrono::duration<double> diff = end - begin;
  return diff.count();
}

void runtime_error_1() {
  // Segmentation fault
  int *ptr = nullptr;
  *(ptr + 7122) = 7122;
}

void runtime_error_2() {
  // Segmentation fault
  int *ptr = (int *)memset;
  *ptr = 7122;
}

void runtime_error_3() {
  // munmap_chunk(): invalid pointer
  int *ptr = (int *)memset;
  delete ptr;
}

void runtime_error_4() {
  // free(): invalid pointer
  int *ptr = new int[7122];
  ptr += 1;
  delete[] ptr;
}

void runtime_error_5() {
  // maybe illegal instruction
  int a = 7122, b = 0;
  cout << (a / b) << endl;
}

void runtime_error_6() {
  // floating point exception
  volatile int a = 7122, b = 0;
  cout << (a / b) << endl;
}

void runtime_error_7() {
  // call to abort.
  assert(false);
}

} // namespace system_test

#include <sys/resource.h>
void print_stack_limit() { // only work in Linux
  struct rlimit l;
  getrlimit(RLIMIT_STACK, &l);
  cout << "stack_size = " << l.rlim_cur << " byte" << endl;
}
```