CoU-Runtime-Terrors

Rashed - Shaiful - Ratul

August 26, 2025

Contents

1 Number Theory

1.1 Miller Rabin

```
using u64 = uint64_t;
using u128 = __uint128_t;

u64 binpower(u64 base, u64 e, u64 mod)
     {
    u64 result = 1;
    base %= mod;
    while (e) {
        if (e & 1)
            result = (u128)result *
                 base % mod;
        base = (u128)base * base % mod
            ;
        e >>= 1;
    }
    return result;
}

bool check_composite(u64 n, u64 a, u64
     d, int s) {
    u64 x = binpower(a, d, n);
    if (x == 1 || x == n - 1)
        return false;
    for (int r = 1; r < s; r++) {
        x = (u128)x * x % n;
        if (x == n - 1)
            return false;
    }
    return true;
};

bool MillerRabin(u64 n, int iter=5) {
    if (n < 4)
        return n == 2 || n == 3;

    int s = 0;
    u64 d = n - 1;
    while ((d & 1) == 0) {
        d >>= 1;
        s++;
    }

    for (int i = 0; i < iter; i++) {
        int a = 2 + rand() % (n - 3);
        if (check_composite(n, a, d, s
            ))
            return false;
    }
    return true;
}
```

1.2 Number of Divisor

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

1.3 Sum of Divisors

```
long long SumOfDivisor(long long n)
{
    long long ans=1;
    for(long long i=0; prime[i]*prime[
        i]<=n;
            i++)
    {
        long long sum=0,p=1;
        while(n%prime[i]==0)
        {
            n/=prime[i];
            p*=prime[i];
            sum+=p;
        }
        ans*=(sum+1);
    }
    if(n>1)
        ans*=(n+1);
    return ans;
}
```

1.4 Eulers Phi

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

1.5 Eulers Phi 1 to N O(nloglog(n))

```
#define MAX 100000
long long phi[MAX + 7];
void generatePhi()
{
    phi[1] = 0;
    for (long long i = 2; i <= MAX; i
        ++)
    {
        if(!phi[i])
        {
            phi[i] = i-1;
            for(long long j=(i<<1); j
                <=MAX; j+=i)
            {
                if(!phi[j])
                    phi[j] = j;
                phi[j] = phi[j] * (i
                    -1) / i;
            }
        }
    }
}
```

1.6 Seive

```
vector<ll>prime;
bool mark[1000003];
void sieve(ll n)
{
    ll i,j;
    mark[1]=1;
    for(i=4; i<=n; i+=2)mark[i]=1;
    prime.push_back(2);
    for(i=3; i<=n; i+=2)
    {
        if(!mark[i])
        {
            prime.push_back(i);
            if(i*i<=n)
            {
                for(j=i*i; j<=n; j+=(i
                    *2))mark[j]=1;
            }
        }
    }
}
```

1.7 Segment Sieve

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

vector<int> simpleSieve(int limit){
    vector<bool> mark(limit+1,true);
    vector<int> primes;
    for(int p=2;p*p<=limit;p++){
        if(mark[p]){
            for(int i=p*p;i<=limit;i+=
                p) mark[i]=false;
        }
    }
    for(int i=2;i<=limit;i++) if(mark[
        i]) primes.push_back(i);
    return primes;
}

void segmentedSieve(ll L, ll R){
    if(R<2) return;
    ll limit = floor(sqrt(R))+1;
    vector<int> primes = simpleSieve(
        limit);
    vector<bool> isPrime(R-L+1,true);
    for(int p:primes){
        ll start = max<ll>((ll)p*(ll)p
            , ((L + p - 1)/p)*p);
        for(ll j=start;j<=R;j+=p)
            isPrime[j-L]=false;
    }
    if(L==1) isPrime[0]=false;
    for(ll i=L;i<=R;i++) if(isPrime[i-
        L]) cout<<i<<"\n";
}

int main(){
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    int t;
    if(!(cin>>t)) return 0;
    while(t--){
        long long L,R;
        cin>>L>>R;
        segmentedSieve(L,R);
        if(t) cout<<"\n";
    }
    return 0;
}
```

1.8 Large Number Divisible

```
    string a; int b; cin >> a >> b;
  int mod = 0;
  for (int i = 0; i < a.size(); i++) {
    mod = (mod * 10LL % b + (a[i] - '0
        ')) % b;
  }
```

1.9 Legendres Formula

```
int legendre(long long n, long long p)
     {
  int ans = 0;
  while (n) {
    ans += n / p;
    n /= p;
  }
  return ans;
}
```

1.10 BigMod / Binary Exponention

```
long long binpow(long long a, long
     long b) {
    long long res = 1;
    while (b > 0) {
        if (b & 1)
            res = res * a;
        a = a * a;
        b >>= 1;
```

```
    }
    return res;
}
```

1.11 Permutation and Combination

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

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

int power(long long n, long long k) {
  int ans = 1 % mod; n %= mod; if (n <
       0) n += mod;
  while (k) {
    if (k & 1) ans = (long long) ans *
         n % mod;
    n = (long long) n * n % mod;
    k >>= 1;
  }
  return ans;
}
int f[N], invf[N];
int nCr(int n, int r) {
  if (n < r or n < 0) return 0;
  return 1LL * f[n] * invf[r] % mod *
       invf[n - r] % mod;
}
int nPr(int n, int r) {
  if (n < r or n < 0) return 0;
  return 1LL * f[n] * invf[n - r] %
       mod;
}
int32_t main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  f[0] = 1;
  for (int i = 1; i < N; i++) {
    f[i] = 1LL * i * f[i - 1] % mod;
  }
  invf[N - 1] = power(f[N - 1], mod -
       2);
  for (int i = N - 2; i >= 0; i--) {
    invf[i] = 1LL * invf[i + 1] * (i +
         1) % mod;
  }
  cout << nCr(6, 2) << '\n';
  cout << nPr(6, 2) << '\n';
  return 0;
}
```

1.12 SPF

```
    const int N = 1e6 + 5;
int spf[N];

void sieve_spf() {
    for (int i = 2; i < N; ++i) {
        if (spf[i] == 0) { // i is
             prime
            for (int j = i; j < N; j
                 += i) {
                if (spf[j] == 0) spf[j
                     ] = i;
            }
        }
    }
}

vector<int> get_factors(int n) {
    vector<int> res;
    while (n > 1) {
        res.push_back(spf[n]);
        n /= spf[n];
    }
    return res;
}
```

1.13 Smallest Number Having Exactly K Divisors

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

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

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

1.14 Sum of divisors 1 to N

```
ll inv2;
long long fun(long long start, long
     long end) {
  return ((((end - start + 1) % mod)
     * ((start + end) % mod) % mod) *
          inv2 %mod);
}
void akam()
{
    ll n,i,j,c=0,x,y,k,ans=0,sum=0;
    cin>>n;
    inv2 = 500000004;
    ll first_same = 1,last_same;

    while(first_same<=n){

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

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

1.15 Max gcd of two element of an array

```
    ll m[1000001];
void akam()
{
    ll n,i,j,c=0,x,y,k,ans=0,sum=0;
    cin>>n;
    ll a[n];
    for(i=0; i<n; i++)
    {
        cin>>a[i];
        m[a[i]]++;
        c = max(c,a[i]);
    }
    for(i=c;i>=1;i--){

        ll cnt = 0;
        for(j=i;j<=c;j+=i){

            if(m[j]>=2){
                cout<<j<<endl;
                return;
            }

            if(m[j]>0){
              cnt++;
            }

            if(cnt>=2){
                cout<<i<<endl;
                return;
            }

        }
    }
}
```

1.16 Product of divisors

```
void akam()
{
    ll prf[n+1],suf[n+1];
    suf[n] = 1;
    prf[n] = 1;
    prf[0] = (v[0].S+1)%(mod-1);
    suf[n-1] = (v[n-1].S+1)%(mod-1);
    j = n-2;
    for(i=1;i<n;i++){

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

    ans = 1;
    for(i=0;i<n;i++){
    /// This part is for product of
         divisors
```

```
        x = ((v[i].S+1)*v[i].S)/2;
        x = x%(mod-1);
        if(i==0){
            y = suf[i+1]%(mod-1);
        }
        else if(i==n-1){
            y = prf[i-1]%(mod-1);
        }
        else {
            y = (prf[i-1]*suf[i+1])%(
                mod-1);
        }
        x = (x*y)%(mod-1);
        y = big(v[i].F,x,mod);
        ans = (ans*y)%mod;
    }
    cout<<ans<<endl;
}
```

1.17 $x^{yk}\%mod$

```
ll po(ll n,ll m, ll md){
    ll res = 1;
    while(m>0){

        if(m&1)res = (res*n)%md;

        n = (n*n)%md;
        m>>=1;
    }
    return res;
}
void akam()
{
    ll n,i,j,c=0,x,y,k,ans=0,sum=0;

    //cout<<"Case "<<tst<<": ";
    cin>>x>>y>>k;
    y = po(y,k,mod-1);
// euler toteint of mod is used to mod
     b^c
    cout<<po(x,y,mod)<<endl;

}
```

1.18 First n digit and last n digit of a^b

```
ll binpow(ll n, ll k){
    ll res = 1;
    while(k>0){
        if(k&1)res = (res * n)%1000;
// here i used 1000 to find last three
, can use 10000 for four 10^n for last
     n
        n = (n*n)%1000;
        k>>=1;
    }
    return res;
}
void akam()
{
    ll n,i,j,c=0,x,y,k,ans=0,sum=0;

    //cout<<"Case "<<tst<<": ";
    cin>>n>>k;
    string last = to_string(binpow(n,k
        ));
    while(last.size()<3)last = '0' +
        last;
    long double  d = k * log10(n) + 1;
    //// here d is the number of digit
        in (n^k)
    ll d_er_floor = floor(d);
    d = d - (d_er_floor + 1 - 3);
// here d k log10(n) - ( [(k log10(n)]
     + 1 - x) ,
//here is the number of digit you need
     from first:
    ll first = pow(10.0,d);
    cout<<first<<"..."<<last<<endl;

}
```

1.19 Maximum Co-Prime Product

```
//shohag's tmplt
    #include <iostream>
#include <vector>
using namespace std;
using ll = long long;
const int N = 1e5;

// credit: mango_lassi
int arr[N + 1];
int u[N + 1];
int cnt[N + 1];

vector<int> d[N + 1];
bool b[N + 1];

bool coprime(int x) {
  int ret = 0;
  for (int i : d[x]) ret += cnt[i] * u
      [i];
  return ret;
}

void update(int x, int a) {
  for (int i : d[x]) cnt[i] += a;
}

int main() {
  for (int i = 1; i <= N; i++) {
    for (int j = i; j <= N; j += i) d[
        j].push_back(i);
    if (i == 1) u[i] = 1;
    else if ((i / d[i][1]) % d[i][1]
         == 0) u[i] = 0;
    else u[i] = -u[i / d[i][1]];
  }

  int n;
  cin >> n;

  ll ans = 0;
  for (int i = 0; i < n; i++) {
    int a;
    cin >> a;
    ans = max(ans, (ll)a);
    b[a] = 1;
  }
  for (int i = 1; i <= N; ++i) {
for (int j = 2;
i * j <= N; ++j) b[i] |= b[i * j];
  }

  vector<int> s;
  for (int i = N; i > 0; --i) {
    if (! b[i]) continue;
    while(coprime(i)) {
      ans = max(ans, (ll)i * s.back())
          ;
      update(s.back(), -1);
      s.pop_back();
    }
    update(i, 1);
    s.push_back(i);
  }
  cout << ans << '\n';
}
```

1.20 Sum of Product of every pair

```
    s = sum(all array)
    s2 = sum(ai * ai)
    sol = .5 * ( s^2 - s2)
```

2 Data Structures

2.1 Fenwick Tree

```
class FenwickTree {
public:
    vector<int> tree;
    int n;

    FenwickTree(int size) : n(size) {
        tree.resize(n + 1, 0);
    }

    // Add value to element at index i
    void update(int i, int delta) {
        while (i <= n) {
            tree[i] += delta;
            i += i & -i;
        }
    }

    // Get prefix sum from 1 to i
    int query(int i) {
        int sum = 0;
        while (i > 0) {
            sum += tree[i];
            i -= i & -i;
        }
        return sum;
    }

    // Get sum in range [left, right]
    int rangeQuery(int left, int right
        ) {
        return query(right) - query(
            left - 1);
    }
};
```

2.2 Segment Tree with Lazy

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

#define ll long long
#define vi vector<ll>

const int N = 1e5 + 5;
ll tree[4 * N], lazy[4 * N];
int n;

// Build the tree from the initial
     array
void build(int node, int l, int r,
     const vi &arr) {
    if (l == r) {
        tree[node] = arr[l];
        return;
    }
    int mid = (l + r) / 2;
    build(2 * node, l, mid, arr);
    build(2 * node + 1, mid + 1, r,
         arr);
    tree[node] = tree[2 * node] + tree
         [2 * node + 1];
}

// Propagate pending updates
void propagate(int node, int l, int r)
     {
    if (lazy[node] != 0) {
        tree[node] += (r - l + 1) *
             lazy[node]; e
        if (l != r) {
            lazy[2 * node] += lazy[
                 node];
            lazy[2 * node + 1] += lazy
                 [node];
        }
        lazy[node] = 0;
    }
}

// Range update: add val to all
     elements in [ql, qr]
void update(int node, int l, int r,
int ql, int qr, ll val) {
    propagate(node, l, r);
    if (qr < l || r < ql) return; //
         no overlap
    if (ql <= l && r <= qr) { // total
          overlap
        lazy[node] += val;
        propagate(node, l, r);
        return;
    }

    int mid = (l + r) / 2;
    update(2 * node, l, mid, ql, qr,
         val);
```

```cpp
    update(2 * node + 1, mid + 1, r,
        ql, qr, val);
    tree[node] = tree[2 * node] + tree
        [2 * node + 1];
}

// Range query: sum of elements in [ql
    , qr]
ll query(int node, int l, int r, int
    ql, int qr) {
    propagate(node, l, r);
    if (qr < l || r < ql) return 0; //
         no overlap
    if (ql <= l && r <= qr) return
        tree[node];

    int mid = (l + r) / 2;
    return query(2 * node, l, mid, ql,
         qr) +
           query(2 * node + 1, mid +
               1, r, ql, qr);
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    cin >> n;
    vi arr(n + 1); // 1-based indexing

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

    build(1, 1, n, arr);

    int q;
    cin >> q;
    while (q--) {
        int type;
        cin >> type;
        if (type == 1) {
            int l, r;
            cin >> l >> r;
            cout << query(1, 1, n, l,
                r) << '\n';
        } else if (type == 2) {
            int l, r;
            ll val;
            cin >> l >> r >> val;
            update(1, 1, n, l, r, val)
                ;
        }
    }

    return 0;
}
```

2.3 Oredered Set

```cpp
    #include<bits/stdc++.h>
#include<ext/pb_ds/assoc_container.hpp
    >
#include<ext/pb_ds/tree_policy.hpp>
using namespace __gnu_pbds;
using namespace std;

template <typename T> using o_set =
tree<T, null_type, less<T>,
rb_tree_tag,
    tree_order_statistics_node_update
    >;

int32_t main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  o_set<int> se;
  se.insert(4);
  //se.order_of_key(val)
  //se.find_by_order(idx)
  return 0;
}
```

3 Graph and Tree

3.1 BFS

```cpp
    vector<int> bfs(int start) {
    int n = graph.size();
    vector<int> dist(n, -1);
    queue<int> q;
    dist[start] = 0;
    q.push(start);
    while (!q.empty()) {
        int u = q.front(); q.pop();
        for (int v : graph[u]) {
            if (dist[v] == -1) {
                dist[v] = dist[u] + 1;
                q.push(v);
            }
        }
    }
    return dist;
}
```

3.2 Floyd$_{Warshall}$

```cpp
    //Floyd-Warshall Algorithm
vector<vector<ll>> floyd_warshall() {
    int n = adj.size();
    vector<vector<ll>> dist = adj;
    for (int k = 0; k < n; k++) {
    for (int i = 0; i < n; i++) {
    for (int j = 0; j < n; j++) {
    if (dist[i][k] != LLONG_MAX &&
        dist[k][j] != LLONG_MAX) {
    dist[i][j] = min(dist[i][j], dist[
        i][k] + dist[k][j]);
                }
            }
        }
    }
    return dist;
}
```

3.3 Strongly Connected Component

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

struct SCC {
    int n;
    vector<vector<int>> g, rg;  //
        graph, reverse graph
    vector<int> vis, comp, order;

    SCC(int n): n(n), g(n), rg(n), vis
        (n), comp(n) {}

    void add(int u,int v){
        g[u].push_back(v);
        rg[v].push_back(u);
    }

    void dfs1(int u){
        vis[u]=1;
        for(int v:g[u]) if(!vis[v])
            dfs1(v);
        order.push_back(u);
    }

    void dfs2(int u,int c){
        comp[u]=c;
        for(int v:rg[u]) if(comp[v
            ]==-1) dfs2(v,c);
    }

    int build(){
        for(int i=0;i<n;i++) if(!vis[i
            ]) dfs1(i);
        reverse(order.begin(),order.
            end());
        fill(comp.begin(),comp.end()
            ,-1);
        int j=0;
        for(int u:order) if(comp[u
            ]==-1) dfs2(u,j++);
        return j;  // total SCC count
    }
};

int main(){
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n,m;
    cin>>n>>m;
    SCC scc(n);

    for(int i=0;i<m;i++){
        int u,v;
        cin>>u>>v;
        scc.add(u,v);
    }

    int cnt = scc.build();  // run
        Kosaraju
    cout<<"Total SCCs = "<<cnt<<"\n";

    // print which component each node
         belongs to
    for(int i=0;i<n;i++){
        cout<<"Node "<<i<<" is in SCC
            "<<scc.comp[i]<<"\n";
    }
}
```

3.4 Articulation Points

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

const int N = 1e5 + 5;

vector<int> adj[N];
bool visited[N], isArticulation[N];
int tin[N], low[N], timer;

void dfs(int u, int parent = -1) {
    visited[u] = true;
    tin[u] = low[u] = timer++;
    int children = 0;

    for (int v : adj[u]) {
        if (v == parent) continue;
        if (visited[v]) {
            // Back edge
            low[u] = min(low[u], tin[v
                ]);
        } else {
            // Tree edge
            dfs(v, u);
            low[u] = min(low[u], low[v
                ]);
            if (low[v] >= tin[u] &&
                parent != -1) {
                isArticulation[u] =
                    true;
            }
            ++children;
        }
    }

    if (parent == -1 && children > 1)
        {
        isArticulation[u] = true; //
            root case
    }
}

void findArticulationPoints(int n) {
    timer = 0;
    fill(visited, visited + n + 1,
        false);
    fill(isArticulation,
    isArticulation + n + 1, false);

    for (int i = 1; i <= n; ++i) {
        if (!visited[i]) dfs(i);
    }

    cout << "Articulation Points:\n";
    for (int i = 1; i <= n; ++i) {
        if (isArticulation[i]) cout <<
            i << " ";
    }
    cout << endl;
}

int main() {
    int n, m;
    cin >> n >> m; // number of nodes
        and edges

    for (int i = 0; i < m; ++i) {
        int u, v;
        cin >> u >> v;
        adj[u].push_back(v);
        adj[v].push_back(u);
    }
```

```
    findArticulationPoints(n);
    return 0;
}
```

3.5 Bellman Ford

```
vector<int> bellmanFord(int n, s, int
     start) {
    vector<int> dist(n, INT_MAX);
    dist[start] = 0;

    // Relax all edges n-1 times
    for (int i = 0; i < n - 1; i++) {
        for (auto edge : edges) {
            int u = edge[0],
            v = edge[1], weight = edge
                [2];
            if (dist[u] != INT_MAX &&
            dist[u] + weight < dist[v
                ]) {
                dist[v] = dist[u] +
                    weight;
            }
        }
    }

    // Check for negative-weight
        cycles
    for (auto edge : edges) {
        int u = edge[0], v = edge[1],
        weight = edge[2];
        if (dist[u] != INT_MAX
        && dist[u] + weight < dist[v])
            {
            cout << "Graph contains
            a negative-weight cycle"
                << endl;
            return {};
        }
    }

    return dist;
}
```

3.6 Dijkstra

```
    vector<int> dijkstra(int start) {
    int n = graph.size();
    vector<int> dist(n, INT_MAX);
    dist[start] = 0;


    priority_queue<pair<int, int>,
    vector<pair<int, int>>,
    greater<pair<int, int>>> pq;

    pq.push({0, start});

    while (!pq.empty()) {
        int u = pq.top().second;
        int d = pq.top().first;
        pq.pop();


        if (d > dist[u]) continue;

        // Traverse all adjacent nodes
        for (auto edge : graph[u]) {
            int v = edge.first;
            int weight = edge.second;


            if (dist[u] + weight <
                dist[v]) {
                dist[v] = dist[u] +
                    weight;
                pq.push({dist[v], v});
            }
        }
    }

    return dist;
}
```

3.7 DSU

```
    struct DSU {
    vector<int> parent, size;

    // Constructor
    DSU(int n) {
        parent.resize(n + 1);
        size.resize(n + 1, 1); //
            Initially, size of each
            set is 1
        for (int i = 1; i <= n; ++i) {
            parent[i] = i; //
                Initially, each node
                is its own parent
        }
    }

    // Find with path compression
    int find(int v) {
        if (parent[v] == v)
            return v;
        return parent[v] = find(parent
            [v]);
    }

    // Union by size
    void unite(int a, int b) {
        a = find(a);
        b = find(b);
        if (a != b) {
            if (size[a] < size[b])
                swap(a, b); // Make
                    sure a has bigger
                    size
            parent[b] = a;
            size[a] += size[b];
        }
    }

    // Check if two nodes are in same
        component
    bool same(int a, int b) {
        return find(a) == find(b);
    }

    // Get size of the component of a
        node
    int getSize(int v) {
        return size[find(v)];
    }
};
```

3.8 Findigh Bridges

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

const int N = 1e5 + 5; // adjust size
    as needed
vector<int> adj[N];
bool visited[N];
int tin[N], low[N];
int timer;
vector<pair<int, int>> bridges;

void dfs(int u, int p = -1) {
    visited[u] = true;
    tin[u] = low[u] = timer++;

    for (int v : adj[u]) {
        if (v == p) continue;
        if (visited[v]) {
            // Back edge
            low[u] = min(low[u], tin[v
                ]);
        } else {
            // Tree edge
            dfs(v, u);
            low[u] = min(low[u], low[v
                ]);

            if (low[v] > tin[u]) {
                // Bridge found
                bridges.emplace_back(u
                    , v);
            }
        }
    }
}

void find_bridges(int n) {
    timer = 0;
    bridges.clear();
    fill(visited, visited + n + 1,
        false);
    for (int i = 1; i <= n; ++i) {
        if (!visited[i]) {
            dfs(i);
        }
    }
}

int main() {
    int n, m;
    cin >> n >> m; // n = number of
        nodes, m = number of edges
    for (int i = 0; i < m; ++i) {
        int u, v;
        cin >> u >> v;
        adj[u].push_back(v);
        adj[v].push_back(u);
    }

    find_bridges(n);

    cout << "Bridges:\n";
    for (auto [u, v] : bridges) {
        cout << u << " - " << v << "\n
            ";
    }

    return 0;
}
```

3.9 Topological Sort

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

const int N = 1e5 + 9;
int indeg[N];
vector<int> g[N];
bool vis[N];
int32_t main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  int n, m; cin >> n >> m;
  while (m--) {
    int u, v; cin >> u >> v;
    indeg[v]++;
    g[u].push_back(v);
  }
  vector<int> z;
  for (int i = 1; i <= n; i++) {
    if (indeg[i] == 0) {
      z.push_back(i);
      vis[i] = true;
    }
  }
  vector<int> ans;
  while (ans.size() < n) {
    if (z.empty()) {
      cout << "IMPOSSIBLE\n";
      return 0;
    }
    int cur = z.back();
    z.pop_back();
    ans.push_back(cur);
    for (auto v: g[cur]) {
      indeg[v]--;
      if (!vis[v] and indeg[v] == 0) {
        z.push_back(v);
        vis[v] = true;
      }
    }
  }
  for (auto x: ans) cout << x << ' ';
  return 0;
}
```

4 String

4.1 Hashing

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

const int N = 1e6 + 9;
const int p1 = 137, mod1 = 127657753,
p2 = 277, mod2 = 987654319;
```

```cpp
int ip1, ip2;
pair<int, int> pw[N], ipw[N];
void prec() {
  pw[0] =  {1, 1};
  for (int i = 1; i < N; i++) {
    pw[i].first =
    1LL * pw[i - 1].first * p1 % mod1;
    pw[i].second =
    1LL * pw[i - 1].second * p2 % mod2;
  }
  ip1 = power(p1, mod1 - 2, mod1);
  ip2 = power(p2, mod2 - 2, mod2);
  ipw[0] =  {1, 1};
  for (int i = 1; i < N; i++) {
    ipw[i].first =
    1LL * ipw[i - 1].first * ip1 % mod1;
    ipw[i].second =
    1LL * ipw[i - 1].second * ip2 % mod2;
  }
}

pair<int, int> string_hash(string s) {
  int n = s.size();
  pair<int, int> hs({0, 0});
  for (int i = 0; i < n; i++) {
    hs.first +=
    1LL * s[i] * pw[i].first % mod1;
    hs.first %= mod1;
    hs.second +=
    1LL * s[i] * pw[i].second % mod2;
    hs.second %= mod2;
  }
  return hs;
}
pair<int, int> pref[N];
void build(string s) {
  int n = s.size();
  for (int i = 0; i < n; i++) {
    pref[i].first =
    1LL * s[i] * pw[i].first % mod1;
    if (i) pref[i].first =
    (pref[i].first + pref[i - 1].first) % mod1;
    pref[i].second =
    1LL * s[i] * pw[i].second % mod2;
    if (i) pref[i].second =
    (pref[i].second + pref[i - 1].second) % mod2;
  }
}
pair<int, int> get_hash(int i, int j) {
  assert(i <= j);
  pair<int, int> hs({0, 0});
  hs.first = pref[j].first;
  if (i) hs.first =
  (hs.first - pref[i - 1].first + mod1) % mod1;
  hs.first =
  1LL * hs.first * ipw[i].first % mod1;
  hs.second =
  pref[j].second;
  if (i) hs.second =
  (hs.second - pref[i - 1].second + mod2) % mod2;
  hs.second =
  1LL * hs.second * ipw[i].second % mod2;
  return hs;
}

int32_t main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  prec();
  string a, b; cin >> a >> b;
  build(a);
  int ans = 0, n = a.size(), m = b.size();
  auto hash_b = string_hash(b);
  for (int i = 0; i + m - 1 < n; i++) {
    ans += get_hash(i, i + m - 1) == hash_b;
  }
  cout << ans << '\n';
  return 0;
}
```

4.2 Number of Divisors

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

int32_t main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  string s; cin >> s;
  int ans = 0;
  int n = s.size();
  for (int len = 1; len <= n / 2; len++) {
    bool ok = true;
    for (int i = 0; i + len - 1 < n; i += len) {
      ok &= get_hash(i, i + len - 1) == get_hash(0, len - 1);
    }
    ans += ok;
  }
  return 0;
}
```

4.3 LCP of two substring

```cpp
int lcp(int i, int j, int x, int y) { // O(log n)
  int l = 1, r = min(j - i + 1, y - x + 1), ans = 0;
  while (l <= r) {
    int mid = l + r >> 1;
    if (get_hash(i, i + mid - 1) == get_hash(x, x + mid - 1)) {
      ans = mid;
      l = mid + 1;
    }
    else {
      r = mid - 1;
    }
  }
  return ans;
}
```

4.4 Largest substring occurs more than k times

```cpp
int n;
int max_oc(int len) {
  map<pair<int, int>, int> mp;
  for (int i = 0; i + len - 1 < n; i++) {
    mp[get_hash(i, i + len - 1)]++;
  }
  int ans = 0;
  for (auto [x, y]: mp) {
    ans = max(ans, y);
  }
  return ans;
}
int32_t main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  prec();
  string s; cin >> s;
  build(s);
  int k; cin >> k;
  n = s.size();
  int l = 1, r = s.size(), ans = -1;
  while (l <= r) {
    int mid = (l + r) >> 1;
    if (max_oc(mid) >= k) {
      ans = mid;
      l = mid + 1;
    }
    else {
      r = mid - 1;
    }
  }
  cout << ans << '\n';
  return 0;
}
```

5 Bit Manipulation

5.1 Everything

```cpp
// odd/even check
bool isOdd(int n){ return n&1; }

// set ith bit =1
int setBit(int n,int i){ return n|(1<<i); }

// clear ith bit =0
int clearBit(int n,int i){ return n&~(1<<i); }

// toggle ith bit
int toggleBit(int n,int i){ return n^(1<<i); }

// check ith bit
bool isBitSet(int n,int i){ return n&(1<<i); }

// count set bits
int countSetBits(int n){
    int c=0; while(n){ c+=n&1; n>>=1; } return c;
}

// Kernighan method
int countSetBitsK(int n){
    int c=0; while(n){ n&=(n-1); c++; } return c;
}

// lowest set bit
int lowestSetBit(int n){ return n&-n; }

// clear lowest set bit
int clearLowestSetBit(int n){ return n&(n-1); }

// power of two?
bool isPow2(int n){ return n>0&&(n&(n-1))==0; }

// pos of MSB (1-based)
int msbPos(int n){
    int p=0; while(n){ n>>=1; p++; } return p;
}

// reverse 32-bit
unsigned revBits(unsigned n){
    unsigned r=0; for(int i=0;i<32;i++)
    { r<<=1; r|=(n&1); n>>=1; }
    return r;
}

// xor 1..n
int xor1toN(int n){
    if(n%4==0) return n;
    if(n%4==1) return 1;
    if(n%4==2) return n+1;
    return 0;
}

// single unique element
int findUnique(const vector<int>&a){
    int u=0; for(int x:a) u^=x; return u;
}

// two unique elements
pair<int,int> findTwoUnique(const vector<int>&a){
    int x=0,y=0,xorv=0;
    for(int v:a) xorv^=v;
    int sb=xorv&-xorv;
    for(int v:a){ if(v&sb) x^=v; else y^=v; }
    return {x,y};
}

// generate subsets
void subsets(const vector<int>&a){
    int n=a.size();
    for(int m=0;m<(1<<n);m++){
        vector<int> s;
        for(int i=0;i<n;i++) if(m&(1<<i))
        s.push_back(a[i]);
    }
```

```
}

// next num same setbits
int nextSameSetBits(int n){
    int c=n&-n, r=n+c;
    return (((r^n)>>2)/c)|r;
}

/* builtins:
__builtin_popcount(x)  // count 1s
__builtin_clz(x)       // leading 0s
__builtin_ctz(x)       // trailing 0s
__builtin_parity(x)    // parity
    bitcount
__builtin_ffs(x)       // first 1-pos
__builtin_bswap32(x)   // byte swap
__builtin_expect(x,v)  // branch hint
__builtin_uadd_overflow(x,y,&r) //
    detect overflow
*/
```

6 Dynamic Programming

6.1 Number of Subsequences Having Product at least K.

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

const int N = 1010, mod = 1e9 + 7, SQ
    = sqrt(mod) + 1;

int a[N], k;
int dp1[N][SQ], dp2[N][SQ];
int mul_back(int i, int p) {
  if (i <= 0) return p >= 1;
  int &ret = dp1[i][p];
  if (ret != -1) return ret;
  ret = mul_back(i - 1, p);
  ret += mul_back(i - 1, p / a[i]);
  if (ret >= mod) ret -= mod;
  return ret;
}
int mul_front(int i, int p) {
  if (i <= 0) return p <= k;
  int &ret = dp2[i][p];
  if (ret != -1) return ret;
  ret = mul_front(i - 1, p);
if(1LL * a[i]*p<SQ)ret+mul_front(i-1,p
    *a[i]);
else ret += mul_back(i - 1, k / (1LL *
    p * a[i]));
  if (ret >= mod) ret -= mod;
  return ret;
int32_t main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  memset(dp1, -1, sizeof dp1);
  memset(dp2, -1, sizeof dp2);
  int n; cin >> n >> k;
  --k;
  for (int i = 1; i <= n; i++) {
    cin >> a[i];
  }
  int ans = 1;
  for (int i = 1; i <= n; i++) {
    ans = (ans + ans) % mod;
  }
  cout << (ans - mul_front(n, 1)+mod %
      mod<<'\n';
  return 0;
}
```

6.2 LCS

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

int LCS(const string &A, const string
    &B) {
    int n = A.size(), m = B.size();
    vector<vector<int>> dp(n+1, vector
        <int>(m+1, 0));
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= m; j++) {
            if (A[i-1] == B[j-1])
                dp[i][j] = dp[i-1][j
                    -1] + 1;
            else
                dp[i][j] = max(dp[i-1][j],
                    dp[i][j-1]);
        }
    }
    return dp[n][m];
}

int main() {
    string A, B;
    cin >> A >> B;
    cout << "LCS length = " << LCS(A,
        B) << "\n";
}

//Reconstruction
string reconstructLCS(const string &A,
const string &B, vector<vector<int>> &
 dp) {
    int i = A.size(), j = B.size();
    string res;
    while(i > 0 && j > 0){
        if(A[i-1] == B[j-1]){
            res += A[i-1];
            i--; j--;
        } else if(dp[i-1][j] > dp[i][j
         -1])
            i--;
        else
            j--;
    }
    reverse(res.begin(), res.end());
    return res;
}
```

6.3 LCSubstring

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

int LCSubstring(const string &A, const
     string &B) {
    int n = A.size(), m = B.size();
    vector<vector<int>> dp(n+1, vector
        <int>(m+1, 0));
    int maxLen = 0;

    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= m; j++) {
            if (A[i-1] == B[j-1]) {
                dp[i][j] = dp[i-1][j
                    -1] + 1;
                maxLen = max(maxLen,
                    dp[i][j]);
            } else {
                dp[i][j] = 0;
            }
        }
    }
    return maxLen;
}

int main() {
    string A, B;
    cin >> A >> B;
    cout << "Longest common substring
        length = "
    << LCSubstring(A, B) << "\n";
}
```

6.4 Longest Increasing Subsequence

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

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

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

\subsection{Bitmask Dp}
\begin{lstlisting}
    #include <bits/stdc++.h>
using namespace std;

const int N = 20;
const int INF = 1e9;

int n;                      // Number
    of elements
vector<int> arr;             // Example
     array
int memo[1 << N];            //
    Memoization array

// Recursive DP function
int solve(int mask) {
    if (mask == (1 << n) - 1) return
        0;  // Base case: all
        elements taken

    if (memo[mask] != -1) return memo[
        mask];

    int ans = INF;
    for (int i = 0; i < n; i++) {
        if (!(mask & (1 << i))) {
            // If i-th element
        not in subset
            int next_mask = mask | (1
                << i);
            ans = min(ans, arr[i] +
                solve(next_mask));
                // Example: min-sum
                DP
        }
    }

    return memo[mask] = ans;
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    cin >> n;
    arr.resize(n);
    for (int i = 0; i < n; i++) cin >>
        arr[i];

    memset(memo, -1, sizeof(memo));
    cout << solve(0) << "\n";

    return 0;
}
```

6.5 Digit Dp

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

const int MAX_DIGITS = 20;
long long memo[MAX_DIGITS][2][100];
// Example: pos, tight, sum (or other
    state)
vector<int> digits;

// Example: count numbers
with sum of digits = target
int target;

long long dp(int pos, bool tight, int
    sum) {
    if (pos == digits.size()) {
        return (sum == target); //
            Base case
    }

    if (memo[pos][tight][sum] != -1)
    return memo[pos][tight][sum];
```

```
    int limit = tight ? digits[pos] :
        9;
    long long ans = 0;

    for (int d = 0; d <= limit; d++) {
        ans += dp(pos + 1, tight &&
        (d == limit), sum + d);
    }

    return memo[pos][tight][sum] = ans
        ;
}

// Convert number to digits
vector<int> getDigits(long long x) {
    vector<int> v;
    while (x) {
        v.push_back(x % 10);
        x /= 10;
    }
    reverse(v.begin(), v.end());
    return v;
}

long long solve(long long x) {
    digits = getDigits(x);
    memset(memo, -1, sizeof(memo));
    return dp(0, true, 0);
}

int main() {
    long long l, r;
    cin >> l >> r >> target;

    // Count numbers in [l, r]
    with sum of digits = target
    long long ans = solve(r) - solve(l
         - 1);
    cout << ans << "\n";

    return 0;
}
```

6.6 Minimizing Coins

```
    void solve() {

    int n,x;
    cin>>n>>x;

    vector<int>dp(x+1,mx);
    vector<int>coins(n);
    rep(i,0,n)
    cin>>coins[i];

    dp[0] = 0;

    for(int i=1; i<=x; i++){
        for(int j=0; j<n; j++){
            if(i>=coins[j])
                dp[i] = min(dp[i],dp[i
                    -coins[j]]+1);
        }
    }

  cout<<(dp[x]==mx? -1:dp[x])<<endl;
}
```

6.7 Ways to make sum using any number of time

```
    void solve() {

    int n,x;
    cin>>n>>x;

    vector<int>dp(x+1);
    vector<int>coins(n);
    rep(i,0,n)
    cin>>coins[i];

    dp[0] = 1;

    for(int i=1; i<=x; i++){
        for(int j=0; j<n; j++){
            if(i>=coins[j])
                dp[i] = (dp[i] + dp[i-
                    coins[j]]) % mod;
        }
    }

    cout<<dp[x]<<endl;
}
```

```
        {
            cin>>x>>y;
            update(1,1,n,x,x,y);
        }
        else
        {
            cin>>x>>y;
            Tr tr = query(1,1,n,x,y);
            cout<<tr.ms<<endl;
        }
    }
}
int main()
{
    ios_base::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
    ll tst =0;

    // test
    akam();

    return 0;

}
```

7.2 Interactive

```
char ask(ll a,ll b,ll c,ll d){
cout<<"? "<<a<<" "<<b<<" "<<c<<" "<<" "<<d<<endl;
    fflush(stdout);
    char ch;
    cin>>ch;

    return ch;
}
void akam()
{
    ll n,i,j,c=0,x,y,k,ans=0,sum=0;

    //cout<<"Case "<<tst<<": ";
    cin>>n;
    ll mi = 0;
    for(i=1;i<n;i++){
        char ch = ask(mi,mi,i,i);

        if(ch=='<')mi = i;
    }
   // cout<<mi<<" SDF"<<endl;
    ll mj = -1;
    if(mj==mi)mj++;
    for(i=0;i<n;i++){
        if(i==mi)continue;
        if(mj==-1){
            mj = i;
            continue;
        }
        char ch = ask(mj,mi,i,mi);
        if(ch=='<')mj = i;

        if(ch=='='){

            ch = ask(mj,mj,i,i);

            if(ch=='>')mj = i;
        }
    }
    cout<<"! "<<mi<<" "<<mj<<endl;
    fflush(stdout);


}
```

7.3 nth Fibonacci number

```
ll fib(ll n)
{
    if(n==1)return 0;
    if(n==2)return 1;
    ll b = n-2;
    ll x,y,z,w;
    ll f[2][2] = {{1,1},{1,0}};
    ll r[2][2] = {{1,0},{0,1}};
    if(b<0)
    {
        return 0;
    }
    while(b>0)
    {
```

```
        if(b&1)
        {
x=((r[0][0]*f[0][0])%MAX+(r[0][1]*f[1][0])%MAX)%MAX;
y=((r[0][0]*f[0][1])%MAX+(r[0][1]*f[1][1])%MAX)%MAX;
w=((r[1][0]*f[0][0])%MAX+(r[1][1]*f[1][0])%MAX)%MAX;
z=((r[1][0]*f[0][1])%MAX+(r[1][1]*f[1][1])%MAX)%MAX;
            r[0][0] = x;
            r[0][1] = y;
            r[1][0] = w;
            r[1][1] = z;
//cout<<r[0][0]<<" r"<<endl;
        }
// cout<<" b "<<b<<endl;
x=((f[0][0]*f[0][0])%MAX+(f[0][1]*f[1][0])%MAX)%MAX;
y=((f[0][0]*f[0][1])%MAX+(f[0][1]*f[1][1])%MAX)%MAX;
w=((f[1][0]*f[0][0])%MAX+(f[1][1]*f[1][0])%MAX)%MAX;
z=((f[1][0]*f[0][1])%MAX+(f[1][1]*f[1][1])%MAX)%MAX;
// cout<<"X "<<x<<" y "<<y<<" w "<<w<<" "<<z<<endl;
        f[0][0] = x;
        f[0][1] = y;
        f[1][0] = w;
        f[1][1] = z;
// cout<<"f[0][0] "<<f[0][0]<< " "<<f[0][1]<<endl;
        b>>=1;
    }
    return r[0][0];
}
```

8 Template Stress Test

8.1 Main function

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

#define rep(i, a, b) for (int i = a; i < (b); ++i)
#define all(x) x.begin(), x.end()
#define UNIQUE(X) (X).erase(unique(all(X)), (X)).end()
#define endl '\n'
#define int long long
#define yes cout << "YES\n"
#define no cout << "NO\n"

mt19937 rng(chrono::steady_clock::now().time_since_epoch().count());
int my_rand(int l, int r)
{
    return uniform_int_distribution<int>(l, r) (rng);
}
void solve() {
    int n = my_rand(5,5);
    cout<<n<<endl;
    for(int i=1; i<=n; i++){
        int x = my_rand(1,5);
        cout<<x<<' ';
    }
    cout<<endl;
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(NULL);cout.tie(NULL);

    //int t; t = my_rand(1,1);cout<<t<<endl; while(t--)
    solve();
    return 0;
}
```

8.2 Stress Test for Windows

```
 @echo off

if [%1]==[] (set /A numLoop = 100)
else (set /A numLoop = %1)
if [%2]==[] (set /A doComp = 1)
else (set /A doComp = %2)

if %doComp% equ 1 (
    echo Compiling solution, gen, brute...

    g++ -std=c++17 gen.cpp -o gen
    g++ -std=c++17 solution.cpp -o solution
    g++ -std=c++17 brute.cpp -o brute

    echo Done compiling.
)

set "diff_found="
```

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

    rem add \f after "fc" to
    ignore trailing whitespaces and to convert
    rem multiple whitespaces into one space
    fc output.out output2.out > diagnostics
    if errorlevel 1 (
        set "diff_found=y"
        goto :break
    )
)

:break

if defined diff_found (
    echo A difference has been found.
    echo Input:
    type input.in
    echo.
    echo.

    echo Output:
    type output.out
    echo.

    echo Expected:
    type output2.out
    echo.
) else (
    echo All tests passed :D
)

del input.in
del output.out
del output2.out
\\save it as .bat file
```

8.3 Stress test for Linux

```
set -e
g++ code.cpp -o code
g++ gen.cpp -o gen
g++ brute.cpp -o brute
for((i = 1; ; ++i)); do
./gen $i > input_file
./code < input_file > myAnswer
./brute < input_file >correctAnswer
diff -Z myAnswer correctAnswer > /dev/null || break
echo "Passed test: " $i
done
echo "WA on the following test:"
cat input_file
echo "Your answer is:"
cat myAnswer
echo "Correct answer is:"
cat correctAnswer
//Should save the file with extension .sh
//File name run.sh
//In Terminal bash run.sh
```

8.4 Some Syntax

```
int main()
{
// Tuple Declared and Initialized using make_tuple()
auto t1 = make_tuple(1, "GeeksforGeeks", 'g');

// Tuple Printed
cout <<"Tuple: "<< get<0>(t1)<< ", "<< get<1>(t1)
<< ", " << get<2>(t1);
    return 0;
}


//// Getline GetChar
int main()
{
    string str;
    cout << "Please enter your name: \n";
    cin.ignore();
    getline(cin, str);  // Reads the entire line
cout << "Hello, " << str << " welcome to GfG !\n";
    return 0;
}
```

8.5 Debugger

```
    // #define cerr cout
namespace __DEBUG_UTIL__
{
    /* Primitive Datatypes Print */
    void print(const char *x) { cerr << x; }
    void print(bool x) { cerr << (x ? "T" : "F"); }
    void print(char x) { cerr << '\'' << x << '\''; }
    void print(signed short int x) { cerr << x; }
    void print(unsigned short int x) { cerr << x; }
    void print(signed int x) { cerr << x; }
    void print(unsigned int x) { cerr << x; }
    void print(signed long int x) { cerr << x; }
    void print(unsigned long int x) { cerr << x; }
    void print(signed long long int x) { cerr << x; }
    void print(unsigned long long int x) { cerr << x; }
    void print(float x) { cerr << x; }
    void print(double x) { cerr << x; }
    void print(long double x) { cerr << x; }
    void print(string x) { cerr << '\"' << x << '\"'; }
    template <size_t N>
    void print(bitset<N> x) { cerr << x; }
    void print(vector<bool> v)
    { /* Overloaded this because stl optimizes vector<bool> by using
          _Bit_reference instead of bool to conserve space. */
        int f = 0;
        cerr << '{';
        for (auto &&i : v)
            cerr << (f++ ? "," : "") << (i ? "T" : "F");
        cerr << "}";
    }
    /* Templates Declarations to support nested datatypes */
    template <typename T>
    void print(T &&x);
    template <typename T>
    void print(vector<vector<T>> mat);
    template <typename T, size_t N, size_t M>
    void print(T (&mat)[N][M]);
    template <typename F, typename S>
    void print(pair<F, S> x);
    template <typename T, size_t N>
    struct Tuple;
    template <typename T>
    struct Tuple<T, 1>;
    template <typename... Args>
    void print(tuple<Args...> t);
    template <typename... T>
    void print(priority_queue<T...> pq);
    template <typename T>
    void print(stack<T> st);
    template <typename T>
    void print(queue<T> q);
    /* Template Datatypes Definitions */
    template <typename T>
    void print(T &&x)
    {
        /*  This works for every container that supports range-based loop
            i.e. vector, set, map, oset, omap, dequeue */
        int f = 0;
        cerr << '{';
        for (auto &&i : x)
            cerr << (f++ ? "," : ""), print(i);
        cerr << "}";
    }
    template <typename T>
    void print(vector<vector<T>> mat)
    {
        int f = 0;
        cerr << "\n~~~~~\n";
        for (auto &&i : mat)
        {
            cerr << setw(2) << left << f++, print(i), cerr << "\n";
        }
        cerr << "~~~~~\n";
    }
    template <typename T, size_t N, size_t M>
    void print(T (&mat)[N][M])
    {
        int f = 0;
        cerr << "\n~~~~~\n";
        for (auto &&i : mat)
        {
            cerr << setw(2) << left << f++, print(i), cerr << "\n";
        }
        cerr << "~~~~~\n";
    }
    template <typename F, typename S>
    void print(pair<F, S> x)
    {
        cerr << '(';
        print(x.first);
        cerr << ',';
        print(x.second);
        cerr << ')';
    }
```

```cpp
    template <typename T, size_t N>
    struct Tuple
    {
        static void printTuple(T t)
        {
            Tuple<T, N - 1>::printTuple(t);
            cerr << ",", print(get<N - 1>(t));
        }
    };
    template <typename T>
    struct Tuple<T, 1>
    {
        static void printTuple(T t) { print(get<0>(t)); }
    };
    template <typename... Args>
    void print(tuple<Args...> t)
    {
        cerr << "(";
        Tuple<decltype(t), sizeof...(Args)>::printTuple(t);
        cerr << ")";
    }
    template <typename... T>
    void print(priority_queue<T...> pq)
    {
        int f = 0;
        cerr << '{';
        while (!pq.empty())
            cerr << (f++ ? "," : ""), print(pq.top()), pq.pop();
        cerr << "}";
    }
    template <typename T>
    void print(stack<T> st)
    {
        int f = 0;
        cerr << '{';
        while (!st.empty())
            cerr << (f++ ? "," : ""), print(st.top()), st.pop();
        cerr << "}";
    }
    template <typename T>
    void print(queue<T> q)
    {
        int f = 0;
        cerr << '{';
        while (!q.empty())
            cerr << (f++ ? "," : ""), print(q.front()), q.pop();
        cerr << "}";
    }
    /* Printer functions */
    void printer(const char *) {} /* Base Recursive */
    template <typename T, typename... V>
    void printer(const char *names, T &&head, V &&...tail)
    {
        /* Using && to capture both lvalues and rvalues */
        int i = 0;
        for (int bracket = 0; names[i] != '\0' and (names[i] != ',' or bracket > 0); i++)
            if (names[i] == '(' or names[i] == '<' or names[i] == '{')
                bracket++;
            else if (names[i] == ')' or names[i] == '>' or names[i] == '}')
                bracket--;
        cerr.write(names, i) << " = ";
        print(head);
        if (sizeof...(tail))
            cerr << " ||", printer(names + i + 1, tail...);
        else
            cerr << "]\n";
    }
    /* PrinterArr */
    void printerArr(const char *) {} /* Base Recursive */
    template <typename T, typename... V>
    void printerArr(const char *names, T arr[], size_t N, V... tail)
    {
        size_t ind = 0;
        for (; names[ind] and names[ind] != ','; ind++)
            cerr << names[ind];
        for (ind++; names[ind] and names[ind] != ','; ind++)
            ;
        cerr << " = {";
        for (size_t i = 0; i < N; i++)
            cerr << (i ? "," : ""), print(arr[i]);
        cerr << "}";
        if (sizeof...(tail))
            cerr << " ||", printerArr(names + ind + 1, tail...);
        else
            cerr << "]\n";
    }
}
#ifndef ONLINE_JUDGE
#define debug(...) cerr << __LINE__ << ": [", __DEBUG_UTIL__::printer(#__VA_ARGS__, __VA_ARGS__)
#define debugArr(...) cerr << __LINE__ << ": [", __DEBUG_UTIL__::printerArr(#__VA_ARGS__, __VA_ARGS__)
#else
#define debug(...)
#define debugArr(...)
#endif
#endif
```

```
//*******Main******* */

#include <bits/stdc++.h>
using namespace std;
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace __gnu_pbds;
template<class T>
using oset = tree<T, null_type, less<T>, rb_tree_tag, tree_order_statistics_node_update>; // find_by_order,
    order_of_key

#define FastIO ios::sync_with_stdio(false);cin.tie(nullptr);

#ifndef ONLINE_JUDGE
#include "dbgtest.h"
#else
#define debug(...)
#define debugArr(...)
#endif
```

9 Miscellaneous Formulation

9.1 Some of mine

```
    Some Properties/Techniques of Number
Theory
1. How many numbers are coprime, from 1 to N*M,
where N and M are coprime ,GCD(N,M)=1
1. With N but not with M
2. With M but not with N
3. With both M and N
Ans1 = (phi(N)*M)-phi(N*M)
Ans2 = (phi(M)*N)-phi(N*M)
Ans3 = phi(M*N)
2. Divisors sum of every number from1 to 2*10^9 ?
Ans = sod(1) + sod(2)+sod(3) + sod(4)+ +sod(n);
long long sum_all_divisors(long long num)
{
 long long sum = 0;
 for (long long i = 1; i <= sqrt(num); i++) {
 long long t1 = i * (num / i - i + 1);
 long long t2 = (((num / i) * (num / i + 1)) / 2)
 - ((i * (i + 1)) / 2);
 sum += t1 + t2;
 }
 return sum;
}
3. If gcd(x,n) = 1 then gcd(n-x,n) = 1;
4.logk(number) =log10(number)/log10(k)
// This is used for base
conversion from decimal to k base.
```

9.2 Combinatorics

1. $\sum_{0 \le k \le n} n - kk = Fib_{n+1}$
2. $nk = nn - k$
3. $nk + nk + 1 = n + 1k + 1$
4. $knk = nn - 1k - 1$
5. $nk = \frac{n}{k} n - 1k - 1$
6. $\sum_{i=0}^{n} ni = 2^n$
7. $\sum_{i \ge 0} n2i = 2^{n-1}$
8. $\sum_{i \ge 0} n2i + 1 = 2^{n-1}$
9. $\sum_{i=0}^{k} (-1)^i ni = (-1)^k n - 1k$
10. $\sum_{i=0}^{k} n + ii = \sum_{i=0}^{k} n + in = n + k + 1k$
11. $1n1 + 2n2 + 3n3 + \ldots + nnn = n2^{n-1}$
12. $1^2 n1 + 2^2 n2 + 3^2 n3 + \ldots + n^2 nn = \left(n + n^2\right) 2^{n-2}$
13. Vandermonde's Identify: $\sum_{k=0}^{r} mknr - k = m + nr$
14. Hockey-Stick Identify: $n, r \in N, n > r, \sum_{i=r}^{n} ir = n + 1r + 1$

15. $\sum_{i=0}^{k} ki^2 = 2kk$

16. $\sum_{k=0}^{n} nknn - k = 2nn$

17. $\sum_{k=q}^{n} nkkq = 2^{n-q}nq$

18. $\sum_{i=0}^{n} k^i ni = (k+1)^n$

19. $\sum_{i=0}^{n} 2ni = 2^{2n-1} + \frac{1}{2}2nn$

20. $\sum_{i=1}^{n} nin - 1i - 1 = 2n - 1n - 1$

21. $\sum_{i=0}^{n} 2ni^2 = \frac{1}{2}\left(4n2n + 2nn^2\right)$

22. Highest Power of 2 that divides ${}^{2n}C_n$: Let x be the number of 1 s in the binary representation. Then the number of odd terms will be 2^x. Let it form a sequence. The n-th value in the sequence (starting from $n = 0$) gives the highest power of 2 that divides ${}^{2n}C_n$.

9.3 Pascal Triangle

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

24. An integer $n \geq 2$ is prime if and only if all the intermediate binomial coefficients $n1, n2, \ldots, nn - 1$ are divisible by n.

25. Kummer's Theorem: For given integers $n \geq m \geq 0$ and a prime number p, the largest power of p dividing nm is equal to the number of carries when m is added to $n - m$ in base p. For implementation take inspiration from lucas theorem.

26. Number of different binary sequences of length n such that no two 0 's are adjacent $= Fib_{n+1}$

27. Combination with repetition: Let's say we choose k elements from an n-element set, the order doesn't matter and each element can be chosen more than once. In that case, the number of different combinations is: $n + k - 1k$

28. Number of ways to divide n persons in $\frac{n}{k}$ equal groups i.e. each having size k is

$$\frac{n!}{k!^{\frac{n}{k}}\left(\frac{n}{k}\right)!} = \prod_{n \geq k}^{n-=k} n - 1k - 1$$

29. The number non-negative solution of the equation: $x_1 + x_2 + x_3 + \ldots + x_k = n$ is $n + k - 1n$

30. Number of ways to choose n ids from 1 to b such that every id has distance at least k $= \left(\frac{b-(n-1)(k-1)}{n}\right)$

31. $\sum_{i=1,3,5,\ldots(a-b)^n)}^{i \leq n} nia^{n-i}b^i = \frac{1}{2}\left((a+b)^n - \right.$

32. $\sum_{i=0}^{n} \frac{ki}{ni} = \frac{n+1n-k+1}{nk}$

33. Derangement: a permutation of the elements of a set, such that no element appears in its original position. Let $d(n)$ be the number of derangements of the identity permutation fo size n. $d(n) = (n-1) \cdot (d(n-1) + d(n-2))$ where $d(0)$

34. Involutions: permutations such that $p^2 =$ identity permutation. $a_0 = a_1 = 1$ and $a_n = a_{n-1} + (n-1)a_{n-2}$ for $n > 1$.

35. Let $T(n,k)$ be the number of permutations of size n for which all cycles have length $\leq k.T(n,k) = \left\{ \begin{array}{l} n! \\ n \end{array} \right.$ $\{n \cdot T(n - 1, k) - F(n-1,k) \cdot T(n$ Here $F(n,k) = n \cdot (n-1) \cdot \ldots \cdot (n- \ k+1)$

9.4 Lucas Theorem

- If p is prime, then $\binom{p^a}{k} \equiv 0(\bmod p)$
- For non-negative integers m and n and a prime p, the following congruence relation holds: $\binom{m}{n} \equiv \prod_{i=0}^{k}\binom{m_i}{n_i}$ (mod p), where, $m = m_kp^k + m_{k-1}p^{k-1} + \ldots + m_1p + m_0$, and $n = n_kp^k + n_{k-1}p^{k-1} + \ldots + n_1p + n_0$ are the base p expansions of m and n respectively. This uses the convention that $\binom{m}{n} = 0$, when $m < n$.

37. $\sum_{i=0}^{n} ni \cdot i^k = \sum_{i=0}^{n} ni \cdot \sum_{j=0}^{k} \left\{ \begin{array}{c} k \\ j \end{array} \right\} \cdot i^j$
$= \sum_{i=0}^{n} ni \cdot \sum_{j=0}^{k} \left\{ \begin{array}{c} k \\ j \end{array} \right\} \cdot j!ni = \left(\sum_{i=0}^{n} \frac{n!}{(\ln d(1i))!} \cdot \sum_{j=0}^{k} \left\{ \begin{array}{c} k \\ j \end{array} \right\} \cdot \frac{1}{(i-j)!} =\right.$
$$\sum_{i=0}^{n}\sum_{j=0}^{k} \frac{n!}{(n-i)!} \cdot \left\{ \begin{array}{c} k \\ j \end{array} \right\} \cdot \frac{1}{(i-j)!} =$$
n! $\sum_{i=0}^{n} \sum_{j=0}^{k} \left\{ \begin{array}{c} k \\ j \end{array} \right\} \cdot \frac{1}{(n-i)!} \cdot \frac{1}{(i-j)!} =$
n! $\sum_{i=0}^{n} \sum_{j=0}^{k} \left\{ \begin{array}{c} k \\ j \end{array} \right\} \cdot n - jn - i \cdot \frac{1}{(n-j)!}$
$=$ n! $\sum_{j=0}^{k} \left\{ \begin{array}{c} k \\ j \end{array} \right\} \cdot \frac{1}{(n-j)!} \sum_{i=0}^{n} \cdot n - jn - i$
$= \sum_{j=0}^{k} \left\{ \begin{array}{c} k \\ j \end{array} \right\} \cdot n^{\underline{j}} \cdot 2^{n-j}$ Here $n^{\underline{j}} =$
$P(n, j) = \frac{n!}{(n-j)!}$ and $\left\{ \begin{array}{c} k \\ j \end{array} \right\}$ is stirling number of the second kind. So, instead of $O(n)$, now you can calculate the original equation in $O\left(k^2\right)$ or even in $O\left(k \log^2 n\right)$ using NTT.

38. $\sum_{i=0}^{n-1} ijx^i = x^j(1-x)^{-j-1}$
$\left(1 - x^n \sum_{i=0}^{j} nix^{j-i}(1-x)^i\right.$

39. $x_0, x_1, x_2, x_3, \ldots, x_n x_0 + x_1, x_1 + x_2, x_2 + x_3, \ldots x_n \ldots$ If we continuously do this n times then the polynomial of the first column of the n-th row will be
$$p(n) = \sum_{k=0}^{n} nk \cdot x(k)$$

40. If $P(n) = \sum_{k=0}^{n} nk \cdot Q(k)$, then,
$$Q(n) = \sum_{k=0}^{n} (-1)^{n-k} nk \cdot P(k)$$

41. If $P(n) = \sum_{k=0}^{n} (-1)^k nk \cdot Q(k)$,
then,
$$Q(n) = \sum_{k=0}^{n} (-1)^k nk \cdot P(k)$$

9.5 Catalan Numbers

42. $C_n = \frac{1}{n+1} 2nn$

43. $C_0 = 1, C_1 = 1$ and $C_n = \sum_{k=0}^{n-1} C_k C_{n-1-k}$

44. Number of correct bracket sequence consisting of n opening and n closing brackets.

45. The number of ways to completely parenthesize $n + 1$ factors.

46. The number of triangulations of a convex polygon with $n + 2$ sides (i.e. the number of partitions of polygon into disjoint triangles by using the diagonals).

47. The number of ways to connect the $2n$ points on a circle to form n disjoint i.e. non-intersecting chords.

48. The number of monotonic lattice paths from point $(0,0)$ to point (n,n) in a square lattice of size $n \times n$, which do not pass above the main diagonal (i.e. connecting $(0,0)$ to (n,n)).

49. The number of rooted full binary trees with $n+1$ leaves (vertices are not numbered). A rooted binary tree is full if every vertex has either two children or no children.

50. Number of permutations of $1, \ldots, n$ that avoid the pattern 123 (or any of the other patterns of length 3); that is, the number of permutations with no three-term increasing sub-sequence. For $n=3$, these permutations are 132, 213, 231, 312 and 321.Forn $=4$,theyare1432, 2143, 2413, 2431, 3 and 4321.

51. Balanced Parentheses count with prefix: The count of balanced parentheses sequences consisting of $n+k$ pairs of parentheses where the first k symbols are open brackets. Let the number be $C_n^{(k)}$, then

$$C_n^{(k)} = \frac{k+1}{n+k+1} 2n + kn$$

9.6 Narayana numbers

52. $N(n,k) = \frac{1}{n}\binom{n}{k}\binom{n}{k-1}$

53. The number of expressions containing n pairs of parentheses, which are correctly matched and which contain k distinct nestings. For instance, $N(4,2)=6$ as with four pairs of parentheses six sequences can be created which each contain two times the sub-pattern '()'. Stirling numbers of the first kind

54. The Stirling numbers of the first kind count permutations according to their number of cycles (counting fixed points as cycles of length one).

9.7 Stirling numbers of the first kind

55. $S(n,k)$ counts the number of permutations of n elements with k disjoint cycles.

56. $S(n,k) = (n-1) \cdot S(n-1,k) +$ $S(n-1,k-1)$, where, $S(0,0)=1, S(n,0)=S(0,n)=0$
42, 321m, 3241, 3412, 3421, 4132, 4213, 42 57. $\sum_{k=0} S(n,k) = n\,!$

57. The unsigned Stirling numbers may also be defined algebraically, as the coefficient of the rising factorial: $x^{\bar{n}} = x(x+1)\ldots(x+n-1) = \sum_{k=0}^{n} S(n,k$

58. Lets $[n,k]$ be the stirling number of the first kind, then

$$[n-k] = \sum_{0 \le i_1 < i_2 < i_k < n} i_1 i_2 \ldots i_k$$

9.8 Stirling numbers of the second kind

60. Stirling number of the second kind is the number of ways to partition a set of n objects into k non-empty subsets.

61. $S(n,k) = k \cdot S(n-1,k) + S(n-1,k-\ 1)$, where $S(0,0)=1, S(n,0)=S(0,n)=0$

62. $S(n,2) = 2^{n-1} - 1$

63. $S(n,k) \cdot k! =$ number of ways to color n nodes using colors from 1 to k such that each color is used at least once.

64. An r-associated Stirling number of the second kind is the number of ways to partition a set of n objects into k subsets, with each subset containing at least r elements. It is denoted by $S_r(n,k)$ and obeys the recurrence relation. $S_r(n+1,k) = kS_r(n,k) +$ $nr-1S_r(n-r+1,k-1)$

65. Denote the n objects to partition by the integers $1,2,\ldots,n$. Define the reduced Stirling numbers of the second kind, denoted $S^d(n,k)$, to be the number of ways to partition the integers $1,2,\ldots,n$ into k nonempty subsets such that all elements in each subset have pairwise distance at least d. That is, for any integers i and j in a given subset, it is required that $|i-j| \ge d$. It has been shown that these numbers satisfy, $S^d(n,k) = S(n-d+1,k-d+1), n \ge k \ge d$

9.9 Bell number

66. Counts the number of partitions of a set.

67. $B_{n+1} = \sum_{k=0}^{n} \binom{n}{k} \cdot B_k$

68. $B_n = \sum_{k=0}^{n} S(n,k)$, where $S(n,k)$ is stirling number of second kind.

9.10 Math

69. $ab \bmod ac = a(b \bmod c)$

70. $\sum_{i=1}^{n} i^2 = 1^2 + 2^2 + 3^2 + \ldots + n^2 = \frac{n \cdot (n+1) \cdot (2n+1)}{6}$

71. $\sum_{i=1}^{n} i^3 = 1^3 + 2^3 + 3^3 + \ldots + n^3 = \left(\frac{n \cdot (n+1)}{2}\right)^2$

72. $\sum_{i=0}^{n} i \cdot i! = (n+1)! - 1.$

73. $a^k - b^k = (a-b) \cdot \left(a^{k-1}b^0 + a^{k-2}b^1 + \ \ldots + a^0 b^{k-1}\right)$

74. $\min(a+b, c) = a + \min(b, c-a)$

75. $|a-b| + |b-c| + |c-a| = 2(\max(a,b,c) - \min(a,b,c))$

76. $a \cdot b \le c \to a \le \left\lfloor \frac{c}{b} \right\rfloor$ is correct

77. $a \cdot b < c \to a < \left\lfloor \frac{c}{b} \right\rfloor$ is incorrect

78. $a \cdot b \ge c \to a \ge \left\lfloor \frac{c}{b} \right\rfloor$ is correct

79. $a \cdot b > c \to a > \left\lfloor \frac{c}{b} \right\rfloor$ is correct

80. For positive integer n, and arbitrary real numbers m, x, $\left\lfloor \frac{\lfloor x/m \rfloor}{n} \right\rfloor \left\lfloor \frac{x}{mn} \right\rfloor \left\lceil \frac{\lceil x/m \rceil}{n} \right\rceil = \left\lceil \frac{x}{mn} \right\rceil$

81. Lagrange's identity:
$\left(\sum_{k=1}^{n} a_k^2\right)\left(\sum_{k=1}^{n} b_k^2\right) - \left(\sum_{k=1}^{n} a_k b_k\right)^2$
$\sum_{i=1}^{n} i a^i = \frac{a\left(n a^{n+1} - (n+1)a^n + 1\right)}{(a-1)^2}$

82. Vieta's formulas: Any general polynomial of degree n
$p(x) = a_n x^n + a_{n-1}x^{n-1} + \ldots + a_1 x + a_0$
(with the coefficients being real or complex numbers and $a_n \neq 0$) is known by the fundamental theorem of algebra to have n (not necessarily distinct) complex roots $r_1, r_2, \ldots, r_n$.
$$\begin{cases} r_1 + r_2 + \ldots + r_{n-1} + r_n = -\frac{a_{n-1}}{a_n} \\ (r_1 r_2 + r_1 r_3 + \ldots + r_1 r_n) + (r_2 r_3 \\ \vdots \\ r_1 r_2 \ldots r_n = (-1)^n \frac{a_0}{a_n}. \end{cases}$$
Vieta's formulas can equivalently be written as
$\sum_{1 \le i_1 < i_2 < \ldots < i_k \le n} \left(\prod_{j=1}^{k} r_{i_j}\right) = (-1)^k \frac{q_{n-k}}{a_n},$

83. We are given n numbers $a_1, a_2, \ldots, a_n$ and our task is to find a value x that minimizes the sum,
$|a_1 - x| + |a_2 - x| + \ldots + |a_n - x|$
optimal x = median of the array. if n is even x = [left median,right median] $\sum^{n-1}$ ie^n every number $_2$ in this range will $= \frac{1}{2}\sum_{i=1}^{n}\left(a_1 \sum_{j=1, j \neq i}^{n}\right)^2 \left(d_i \left(a_2 - a_j b_i^2\right)^2 \ldots + (a_n - x)^2\right.$
$optimal x = \frac{(a_1 + a_2 + \ldots + a_n)}{n}$

84. Given an array a of n non-negative integers. The task is to find the sum of the product of elements of all the possible subsets. It is equal to the product of $(a_i + 1)$ for all a_i

85. Pentagonal number theorem: In mathematics, the pentagonal number theorem states that

$$\prod_{n=1}^{\infty}(1 - x^n) = \prod_{k=-\infty}^{\infty}(-1)^k x^{\frac{k(3k-1)}{2}} \ 4 + \ldots + r_2 r_n) + \ldots + r_{n-1} r_n = \frac{a_{n-2}}{a_n}\left(x^{\frac{k(3k+1)}{2}} + x^{\frac{k(3k-1)}{2}}\right)$$

In other words,
$(1-x)\left(1-x^2\right)\left(1-x^3\right)\cdots = 1 - x - x^2 +$

The exponents $1, 2, 5, 7, 12, \cdots$ on the right hand side are given by the formula $g_k = \frac{k(3k-1)}{2}$ for $k = 1, -1, 2, -2, 3, \cdots$ and are called (generalized) pentagonal numbers. It is useful to find the partition number in $O(n\sqrt{n})$

9.11 Fibonacci Number

87. $F_0 = 0, F_1 = 1$ and $F_n = F_{n-1} + F_{n-2}$

88. $F_n = \sum_{k=0}^{\lfloor \frac{n-1}{2} \rfloor} n - k - 1k$

89. $F_n = \frac{1}{\sqrt{5}} \left(\frac{1+\sqrt{5}}{2}\right)^n$
$\frac{1}{\sqrt{5}} \left(\frac{1-\sqrt{5}}{2}\right)^n$

90. $\sum_{i=1}^{n} F_i = F_{n+2} - 1$

91. $\sum_{i=0}^{n-1} F_{2i+1} = F_{2n}$

92. $\sum_{i=1}^{n} F_{2i} = F_{2n+1} - 1$

93. $\sum_{i=1}^{n} F_i^2 = F_n F_{n+1}$

94. $F_m F_{n+1} - F_{m-1} F_n = (-1)^n F_{m-n}$ $F_{2n} = F_{n+1}^2 - F_{n-1}^2 = F_n \left(F_{n+1} + F_{n-1}\right)$

95. $F_m F_n + F_{m-1} F_{n-1} = F_{m+n-1}$ $F_m F_{n+1} + F_{m-1} F_n = F_{m+n}$

96. A number is Fibonacci if and only if one or both of $\left(5 \cdot n^2 + 4\right)$ or $\left(5 \cdot n^2 - 4\right)$ is a perfect square

97. Every third number of the sequence is even and more generally, every k^{th} number of the sequence is a multiple of F_k

98. $gcd\left(F_m, F_n\right) = F_{gcd(m,n)}$

99. Any three consecutive Fibonacci numbers are pairwise coprime, which means that, for every n, $gcd\left(F_n, F_{n+1}\right) = gcd\left(F_n, F_{n+2}\right), gcd\left(F_{n+1}, F_{n+2}\right) = 1$

100. If the members of the Fibonacci sequence are taken mod$n$, the resulting sequence is periodic with period at most $6n$.

9.12 Pythagorean Triples

101. A Pythagorean triple consists of three positive integers a, b, and C, such that $a^2 + b^2 = c^2$. Such a triple is commonly written $(\ a, b, c\ )$

102. Euclid's formula is a fundamental formula for generating Pythagorean triples given an arbitrary pair of integers m and n with $m > n > 0$. The formula states that the integers
$a = m^2 - n^2, b = 2mn, c = m^2 + n^2$
form a Pythagorean triple. The triple generated by Euclid's formula is primitive if and only if m and n are coprime and not both odd. When both m and n are odd, then a, b, and c will be even, and the triple will not be primitive; however, dividing a, b, and c by 2 will yield a primitive triple when m and n are coprime and both odd.

103. The following will generate all Pythagorean triples uniquely:
$a = k \cdot \left(m^2 - n^2\right), b = k \cdot (2mn), c =$ where m, n, and k are positive integers with $m > n$, and with m and n coprime and not both odd.

104. Theorem:The number of Pythagorean triples a,b, n with $maxa, b, n = n$ is given by

$$\frac{1}{2}\left(\prod_{p^\alpha || n} (2\alpha + 1) - 1\right)$$

where the product is over all prime divisors p of the form $4k + 1$. The notation $p^\alpha || n$ stands for the highest exponent α for which p^α divides n Example: For $n = 2 \cdot 3^2 \cdot 5^3 \cdot 7^4 \cdot 11^5 \cdot 13^6$, the number of Pythagorean triples with hypotenuse n is $\frac{1}{2}(7.13 - 1) = 45$. To obtain a formula for the number of Pythagorean triples with hypotenuse less than a specific positive integer N , we may add the numbers corresponding to each $n < N$ given by the Theorem. There is no simple way to compute this as a function of N .

9.13 Sum of Squares Function

105. The function is defined as $r_k(n) =| (a_1, a_2, \ldots, a_k) \in \mathbf{Z^k} : n =$

106. The number of ways to write a natural number as sum of two squares is given by $r_2(n)$. It is given explicitly by $r_2(n) = 4\left(d_1(n) - d_3(n)\right)$
where d1(n) is the number of divisors of n which are congruent with $\big(m^2 1n^2$ modulo 4 and d3(n) is the number ^{n} ol divisors of n which are congruent with 3 modulo 4 . The prime factorization $n = 2^g p_1^{f_1} p_2^{f_2} \ldots q_1^{h_1} q_2^{h_2} \ldots$, where p_i are the prime factors of the form $p_i \equiv 1(\bmod 4)$, and q_i are the prime factors of the form $q_i \equiv 3(\bmod\ 4)$ gives another formula $r_2(n) = 4\left(f_1 + 1\right)\left(f_2 + 1\right)\ldots$, if all exponents $h_1, h_2, \ldots$ are even. If one or more h_i are odd, then $r_2(n) = 0$.

107. The number of ways to represent n as the sum of four squares is eight times the sum of all its divisors which are not divisible by 4 , i.e. $r_4(n) = 8\sum d \mid n; 4ddr8(n) = 16\sum_{d|n}(-1)^{n+d}d^3$

9.14 Number Theory

108. for $i > j, gcd(i, j) = gcd(i - j, j) \leq (i - j)$

109. $\sum_{x=1}^{n}\left[d \mid x^k\right] = \left\lfloor \frac{n}{\left\lceil \prod_{i=0} p_i^{\frac{e_i}{k}} \right\rceil} \right\rfloor$, where
$d = \prod_{i=0} p_i^{e_i}$. Here, $[a \mid b]$ means if a divides b then it is 1 , otherwise it is 0 .

110. The number of lattice points on segment (x_1, y_1) to (x_2, y_2) is $gcd\left(abs\left(x_1 - x_2\right), abs\left(y_1 - y_2\right)\right) + 1$

111. $(n-1)!$ mod $n = n - 1$ if n is prime, 2 if $n = 4, 0$ otherwise.

112. A number has odd number of divisors if it is perfect square

113. The sum of all divisors of a natural number n is odd if and only if $n = 2^r \cdot k^2$ where r is non-negative and k is positive integer.

114. Let a and b be coprime positive integers, and find integers $a\prime$ and $b\prime$ such that $aa\prime \equiv 1 \bmod b$ and $bb\prime \equiv 1 \bmod a$. Then the number of representations of a positive integers (n) as a non negative linear combination of a and b is

$$\frac{n}{ab} - \left\{\frac{b\prime n}{a}\right\} - \left\{\frac{a\prime n}{b}\right\} + 1$$

Here, x denotes the fractional part of x.

115. $\sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{k=1}^{c} d(i \cdot j \cdot k) = \sum_{gcd(i,j)=gcd(j,k)=gcd(k,i)=1}$
Here, $d(x) =$ number of divisors of x.

116. Gauss's generalization of Wilson's theorem: Gauss proved that,
$\prod_{k=1 gcd(k,m)=1}^{m} k \equiv \begin{cases} -1 & (\bmod m) \quad if \\ 1 & (\bmod m) \quad ot \end{cases}$
where p represents an odd prime and α a positive integer. The values of m for which the product is -1 are precisely the ones where there is a primitive root modulo m.

9.15 Divisor Function

117. $\sigma_x(n) = \sum_{d|n} d^x$

118. It is multiplicative i.e if $gcd(a, b) = 1 \to \sigma_x(ab) = \sigma_x(a)\sigma_x(b)$.

119.

$$\sigma_x(n) = \prod_{i=1}^{\tau} \frac{p_i^{(a_i+1)x} - 1}{p_i^x - 1}$$

9.16 Divisor Summatory Function

- Let $\sigma_0(k)$ be the number of divisors of k.
- $D(x) = \sum_{n \leq x} \sigma_0(n)$
- $D(x) = \sum_{k=1}^{x} \left\lfloor \frac{x}{k} \right\rfloor = 2 \sum_{k=1}^{u} \left\lfloor \frac{x}{k} \right\rfloor - u^2$, where $u = \sqrt{x}$
- $D(n) =$ Number of increasing arithmetic progressions where $\left|\frac{q}{h}\right| + \frac{b_1}{-1}$ s^c the second or later kerm, (i.e? The last term, starting term can be any positive integer $\leq n$. For example, $D(3) = 5$ and there are 5 such arithmetic progressions: $(1,2,3,4); (2,3,4); (1,4); (2,4)$

1241, plet $2q_n k$) be the sum of divisors of k .
Then, $\sum_{k=1}^{n} \sigma_1(k) = \sum_{k=1}^{n} k \left\lfloor \frac{n}{k} \right\rfloor$
122. $\prod_{d|n} d = n^{\frac{\sigma_0}{2}}$ if n is not a perfect square, and $= \sqrt{n} \cdot n^{\frac{\sigma_0 - 1}{2}}$ if n is a perfect square.

9.17 Euler's Totient function

123. The function is multiplicative. This means that if $gcd(m,n) = 1, \phi(m.\ n) = \phi(m) \cdot \phi(n)$.

124. $\phi(n) = n \prod_{p|n} \left(1 - \frac{1}{p}\right)$

125. If p is prime and (k $\geq$ 1), then, $\phi\left(p^k\right) = p^{k-1}(p-1) = p^k \left(1 - \frac{1}{p}\right)$

126. $J_k(n)$, the Jordan totient function, is the number of k-tuples of positive integers all less than or equal to n that form a coprime $(k+1)$-tuple together with n. It is a generalization of Euler's totient, $\phi(n) = J_1(n). J_k(n) = n^k \prod_{p|n} \left(1 - \frac{1}{p^k}\right)$

127. $\sum_{d|n} J_k(d) = n^k$

128. $\sum_{d|n} \phi(d) = n$
129.$\phi(n) = \sum_{d|n} \mu(d) \cdot \frac{n}{d} = n \sum_{d|n} \frac{\mu(d)}{d}$

129. $\phi(n) = \sum_{d|n} d \cdot \mu\left(\frac{n}{d}\right)$

130. $a|b \rightarrow \varphi(a)|\varphi(b)$

131. $n \mid \varphi\left(a^n - 1\right)$ for $a, n > 1$

132. $\varphi(mn) = \varphi(m)\varphi(n) \cdot \frac{d}{\varphi(d)}$ where
$d = gcd(m,n)$ Note the special cases

$$\varphi(2m) = \{\, 2\ \varphi(m); if m is even \varphi(m); if m is odd \varphi\left(n^m\right) = n^{m-1}\varphi(n)$$

134. $\varphi(lcm(m,n)) \cdot \varphi(gcd(m,n)) = \varphi(m) \cdot \varphi(n)$ Compare this to the formula $lcm(m,n) \cdot gcd(m,n) = m \cdot n$

135. $\varphi(n)$ is even for $n \geq 3$. Moreover, if if n has r distinct odd prime factors, $2^r \mid \varphi(n)$

136. $\sum_{d|n} \frac{\mu^2(d)}{\varphi(d)} = \frac{n}{\varphi(n)}$

137. $\sum_{1 \leq k \leq n, gcd(k,n)=1} k = \frac{1}{2} n \varphi(n)$ for $n > 1$

138. $\frac{\varphi(n)}{n} = \frac{\varphi(rad(n))}{rad(n)}$ where $rad(n) = \prod_{p|n, pprime} p$

139. $\phi(m) \geq \log_2 m$

140. $\phi(\phi(m)) \leq \frac{m}{2}$

141. When $x \geq \log_2 m$, then
$n^x \mod m = n^{\phi(m)+x} \mod \phi(m)$

142. $\sum gcd(k-1,n) = 1 \leq k \leq n, gcd(k,n) = 1\ \varphi(n)d(n)$ where $d(n)$ is number of divisors. Same equation for $gcd(a \cdot k - 1, n)$ where a and n are coprime.

143. For every n there is at least one other integer $m \neq n$ such that $\varphi(m) = \varphi(n)$.

144. $\sum_{i=1}^{n} \varphi(i) \cdot \left\lfloor \frac{n}{i} \right\rfloor = \frac{n*(n+1)}{2}$

145. If gcd(i, n) = d; where $1 \leq i \leq n-1$ then,there are $\varphi(n/d)$ possible values of i.

146. $\sum_{i=1, i\%2 \neq 0}^{n} \varphi(i) \cdot \left\lfloor \frac{n}{i} \right\rfloor = \sum_{k \geq 1} \left[\frac{n}{2^k}\right]^2$. Note that [] is used here to denote round operator not floor or ceil

147. $\sum_{i=1}^{n} \sum_{j=1}^{n} ij[gcd(i,j) = 1] = \sum_{i=1}^{n} \varphi(i) i^2$
148. Average of coprimes of n which are less than n is $\frac{n}{2}$.

9.18 Mobius Function and Inversion

149. For any positive integer n, define $\mu(n)$ as the sum of the primitive n^{th} roots of unity. It has values in $-1, 0, 1$ depending on the factorization of n into prime factors:
    - $\mu(n) = 1$ if n is a square-free positive integer with an even number of prime factors.
    - $\mu(n) = -1$ if n is a squarefree positive integer with an odd number of prime factors.
    - $\mu(n) = 0$ if n has a squared prime factor.
    - It is a multiplicative function.

150.
$$\sum_{d|n} \mu(d) = \{\, 1\ ; n = 10; n > 0$$

151. $\sum_{n=1}^{N} \mu^2(n) = \sum_{n=1}^{\sqrt{N}} \mu(k) \cdot \left\lfloor \frac{N}{k^2} \right\rfloor$ This is also the number of square-free numbers $\leq n$

152. Mobius inversion theorem: The classic version states that if g and f are arithmetic functions satisfying $g(n) = \sum_{d|n} f(d)$ for every integer $n \geq 1$ then $g(n) = \sum_{d|n} \mu(d) g\left(\frac{n}{d}\right)$ for every integer $n \geq 1$

153. If $F(n) = \prod_{d|n} f(d)$, then $F(n) = \prod_{d|n} F\left(\frac{n}{d}\right)^{\mu(d)}$

154. $\sum_{d|n} \mu(d)\phi(d) = \prod_{j=1}^{K} (2 - P_j)$ where p_j is the primes factorization of d

155. If $F(n)$ is multiplicative, $F \not\equiv 0$, then $\sum_{d|n} \mu(d) f(d) = \prod_{i=1} (1 - f(P_i))$. where p_i are primes of n.

9.19 GCD and LCM

156. $gcd(a, 0) = a$

157. $gcd(a, b) = gcd(b, a \bmod b)$

158. Every common divisor of a and b is a divisor of $gcd(a, b)$.

159. if m is any integer, then $gcd(a + m \cdot b, b) = gcd(a, b)$

160. The gcd is a multiplicative function in the following sense: if a_1 and a_2 are relatively prime, then $gcd(a_1 \cdot a_2, b) = gcd(a_1, b) \cdot gcd(a_2, b)$.

161. $gcd(a, b) \cdot lcm(a, b) = |a \cdot b|$

162. $gcd(a, lcm(b, c))$
$lcm(gcd(a, b), gcd(a, c))$.

163. $lcm(a, gcd(b, c))\ gcd(lcm(a, b), lcm(a, c))$.

164. For non-negative integers a and b, where a and b are not both zero, $gcd\left(n^a - 1, n^b - 1\right) = n^{gcd(a,b)} - 1$

165. $gcd(a, b) = \sum_{k|aandk|b} \phi(k)$

166. $\sum_{i=1}^{n} [gcd(i, n) = k] = \phi\left(\frac{n}{k}\right)$

167. $\sum_{k=1}^{n} gcd(k, n) = \sum_{d|n} d \cdot \phi\left(\frac{n}{d}\right)$

168. $\sum_{k=1}^{n} x^{gcd(k,n)} = \sum_{d|n} x^d \cdot \phi\left(\frac{n}{d}\right)$

169. $\sum_{k=1}^{n} \frac{1}{gcd(k,n)} = \sum_{d|n} \frac{1}{d} \cdot \phi\left(\frac{n}{d}\right) =$
$\frac{1}{n} \sum_{d|n} d \cdot \phi(d)$

170. $\sum_{k=1}^{n} \frac{k}{gcd(k,n)} = \frac{n}{2} \cdot \sum_{d|n} \frac{1}{d} \cdot \phi\left(\frac{n}{d}\right) = \frac{n}{2} \cdot \frac{1}{n} \cdot \sum_{d|n} d \cdot \phi(d)$

171. $\sum_{k=1}^{n} \frac{n}{gcd(k,n)} = 2 * \sum_{k=1}^{n} \frac{k}{gcd(k,n)} - 1$, for $n > 1$

172. $\sum_{i=1}^{n} \sum_{j=1}^{n} [gcd(i, j) = 1] = \sum_{d=1}^{n} \mu(d) \left\lfloor \frac{n}{d} \right\rfloor^2$

173. $\sum_{i=1}^{n} \sum_{j=1}^{n} gcd(i, j) = \sum_{d=1}^{n} \phi(d) \left\lfloor \frac{n}{d} \right\rfloor^2$

174. $\sum_{i=1}^{n} \sum_{j=1}^{n} i \cdot j[gcd(i, j) = 1] = \sum_{i=1}^{n} \phi(i) i^2$

175. $F(n) = \sum_{i=1}^{n} \sum_{j=1}^{n} lcm(i, j) = \sum_{l=1}^{n} \left(\frac{(1 + \lfloor \frac{n}{l} \rfloor)(\lfloor \frac{n}{l} \rfloor)}{2} \right)^2 \sum_{d|l} \mu(d) l d$

176. $gcd(A_L, A_{L+1}, \ldots, A_R)\ gcd(A_L, A_{L+1} - A_L, \ldots, A_R - A_{R-1})$.'

177. Given n, If $SUM = LCM(1, n) + LCM(2, n) + \ldots + LCM(n, n)$ then SUM $= \frac{n}{2} \left(\sum_{d|n} (\phi(d) \times d) + 1 \right.$

9.20 Extra Miscellaneous

178. $a + b = a \oplus b + 2(a\&b)$.

179. $a + b = a \mid b + a\&b$

180. $a \oplus b = a \mid b - a\&b$

181. k_{th} bit is set in x iff $x \bmod 2^{k-1} - x \bmod 2^k \neq 0 \left(= 2^k \text{ to be exact}\right)$. It comes handy when you need to look at the bits of the numbers which are pair sums or subset sums etc.

182. $n \bmod 2^i = n\& \left(2^i - 1\right)$

183. $1 \oplus 2 \oplus 3 \oplus \cdots \oplus (4k - 1) = 0$ for any $k \geq 0$

184. Erdos Gallai Theorem: The degree sequence of an undirected graph is the non-increasing sequence of its vertex degrees A sequence of non-negative integers $d_1 \geq d_2 \geq \cdots \geq d_n$ can be represented as the degree sequence of finite simple graph on n vertices if and only if $d_1 + d_2 + \cdots + d_n$ is even and $\sum_{i=1}^{k} d_i \leq k(k-1) + \sum_{i=k+1}^{n} \min(d_i, k)$ holds for every k in $1 \leq k \leq n$.

9.21 Properties of mod:

185. If $ac \equiv bc (\bmod m)$, then $a \equiv b \ (\bmod m/gcd(c, m))$.

186. If $mn \equiv 0 (\bmod n)$, then the smallest number m is equal to $lcm(n, d)$.

187. If p is prime, then $(x + y)^p \equiv x^p + y^p \ (\bmod p)$.

188. $ab \equiv a(b \bmod c)(\bmod ac)$.

9.22 Area Formulas

- Ellipse: $A = \pi ab$
- a, b : Semi-axes
- Regular Polygon: $A = \frac{1}{2} nRr$
- n : Sides
- R: Circumradius
- r : Apothem

9.23 Volume Formulas

- Rectangular Prism: $V = lwh$
- l, w, h : Dimensions
- Sphere: $V = \frac{4}{3}\pi r^3$
- r : Radius
- Cone: $V = \frac{1}{3}\pi r^2 h$

$$-r\text{: } Radius - h\text{: } Height$$

- Pyramid: $V = \frac{1}{3} bh$
- b : Base area
- h : Height

189. Sector Area: area $= \frac{\theta}{2} \cdot r^2$ (angle in radians)

190. Chord Length: $d = 2 \cdot r \cdot \sin\left(\frac{\theta}{2}\right)$ (angle in radians)
$d = 2 \cdot \sqrt{r^2 - x^2}$ (where x is the perpendicular distance from the center to the chord)

191. Law of Cosines: $c^2 = a^2 + b^2 - 2ab\cos\gamma$

192. Law of Sines: $\frac{a}{\sin\alpha} = \frac{b}{\sin\beta} = \frac{c}{\sin\gamma}$