Team Notebook

August 3, 2023

Contents

1 DS

1.1 HLD

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

const int N = 1e5 + 9, LG = 18, inf = 1e9 + 9;

struct ST {
#define lc (n << 1)
#define rc ((n << 1) | 1)
  int t[4 * N], lazy[4 * N];
  ST() {
    fill(t, t + 4 * N, -inf);
    fill(lazy, lazy + 4 * N, 0);
  }
  inline void push(int n, int b, int e) {
    if(lazy[n] == 0) return;
    t[n] = t[n] + lazy[n];
    if(b != e) {
      lazy[lc] = lazy[lc] + lazy[n];
      lazy[rc] = lazy[rc] + lazy[n];
    }
    lazy[n] = 0;
  }
  inline int combine(int a, int b) {
    return max(a, b); //merge left and right queries
  }
  inline void pull(int n) {
    t[n] = max(t[lc], t[rc]); //merge lower nodes of the tree
          to get the parent node
  }
  void build(int n, int b, int e) {
    if(b == e) {
      t[n] = 0;
      return;
    }
    int mid = (b + e) >> 1;
    build(lc, b, mid);
    build(rc, mid + 1, e);
    pull(n);
  }
  void upd(int n, int b, int e, int i, int j, int v) {
    push(n, b, e);
    if(j < b || e < i) return;
    if(i <= b && e <= j) {
      lazy[n] += v;
      push(n, b, e);
      return;
    }
    int mid = (b + e) >> 1;
    upd(lc, b, mid, i, j, v);
    upd(rc, mid + 1, e, i, j, v);
    pull(n);
  }
  int query(int n, int b, int e, int i, int j) {
    push(n, b, e);
    if(i > e || b > j) return -inf;
    if(i <= b && e <= j) return t[n];
    int mid = (b + e) >> 1;
    return combine(query(lc, b, mid, i, j), query(rc, mid +
        1, e, i, j));
  }
} t;

vector<int> g[N];
int par[N][LG + 1], dep[N], sz[N];
void dfs(int u, int p = 0) {
  par[u][0] = p;
  dep[u] = dep[p] + 1;
  sz[u] = 1;
  for (int i = 1; i <= LG; i++) par[u][i] = par[par[u][i -
      1]][i - 1];
  if (p) g[u].erase(find(g[u].begin(), g[u].end(), p));
  for (auto &v : g[u]) if (v != p) {
      dfs(v, u);
      sz[u] += sz[v];
      if(sz[v] > sz[g[u][0]]) swap(v, g[u][0]);
    }
}
int lca(int u, int v) {
  if (dep[u] < dep[v]) swap(u, v);
  for (int k = LG; k >= 0; k--) if (dep[par[u][k]] >= dep[v
      ]) u = par[u][k];
  if (u == v) return u;
  for (int k = LG; k >= 0; k--) if (par[u][k] != par[v][k])
      u = par[u][k], v = par[v][k];
  return par[u][0];
}
int kth(int u, int k) {
  assert(k >= 0);
  for (int i = 0; i <= LG; i++) if (k & (1 << i)) u = par[u
      ][i];
  return u;
}
int T, head[N], st[N], en[N];
void dfs_hld(int u) {
  st[u] = ++T;
  for (auto v : g[u]) {
    head[v] = (v == g[u][0] ? head[u] : v);
    dfs_hld(v);
  }
  en[u] = T;
}
int n;
int query_up(int u, int v) {
  int ans = -inf;
  while(head[u] != head[v]) {
    ans = max(ans, t.query(1, 1, n, st[head[u]], st[u]));
    u = par[head[u]][0];
  }
  ans = max(ans, t.query(1, 1, n, st[v], st[u]));
  return ans;
}
int query(int u, int v) {
  int l = lca(u, v);
  int ans = query_up(u, l);
  if (v != l) ans = max(ans, query_up(v, kth(v, dep[v] - dep
      [l] - 1)));
  return ans;
}
int32_t main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);

  cin >> n;
  for (int i = 1; i < n; i++) {
    int u, v;
    cin >> u >> v;
    g[u].push_back(v);
    g[v].push_back(u);
  }
  dfs(1);
  head[1] = 1;
  dfs_hld(1);
  int q;
  cin >> q;
  t.build(1, 1, n);
  while (q--) {
    string ty;
    int u, v;
    cin >> ty >> u >> v;
    if (ty == "add") {
      t.upd(1, 1, n, st[u], en[u], v);
    } else {
      cout << query(u, v) << '\n';
    }
  }
  return 0;
```

```
}
//https://www.hackerrank.com/challenges/subtrees-and-paths/
    problem
```

1.2 Segment Tree Lazy

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

const int N = 5e5 + 9;
int a[N];
struct ST {
  #define lc (n << 1)
  #define rc ((n << 1) | 1)
  long long t[4 * N], lazy[4 * N];
  ST() {
    memset(t, 0, sizeof t);
    memset(lazy, 0, sizeof lazy);
  }
  inline void push(int n, int b, int e) {
    if (lazy[n] == 0) return;
    t[n] = t[n] + lazy[n] * (e - b + 1);
    if (b != e) {
      lazy[lc] = lazy[lc] + lazy[n];
      lazy[rc] = lazy[rc] + lazy[n];
    }
    lazy[n] = 0;
  }
  inline long long combine(long long a,long long b) {
    return a + b;
  }
  inline void pull(int n) {
    t[n] = t[lc] + t[rc];
  }
  void build(int n, int b, int e) {
    lazy[n] = 0;
    if (b == e) {
      t[n] = a[b];
      return;
    }
    int mid = (b + e) >> 1;
    build(lc, b, mid);
    build(rc, mid + 1, e);
    pull(n);
  }
  void upd(int n, int b, int e, int i, int j, long long v) {
    push(n, b, e);
    if (j < b || e < i) return;
    if (i <= b && e <= j) {
      lazy[n] = v; //set lazy
      push(n, b, e);
      return;
    }
    int mid = (b + e) >> 1;
    upd(lc, b, mid, i, j, v);
    upd(rc, mid + 1, e, i, j, v);
    pull(n);
  }
  long long query(int n, int b, int e, int i, int j) {
    push(n, b, e);
    if (i > e || b > j) return 0; //return null
    if (i <= b && e <= j) return t[n];
    int mid = (b + e) >> 1;
    return combine(query(lc, b, mid, i, j), query(rc, mid +
        1, e, i, j));
  }
};
int32_t main() {

}
```

2 Graph

2.1 2DBFS

```
2D BFS:
bool valid(int x, int y)
{
    return (x>=1&&x<=row&&y>=1&&y<=col);
}
ll bfs(pll src, pll des)
{
    memset(level,-1,sizeof(level));
    // for(ll i=1; i<=row; i++)
    // {
    //     for(ll j=1; j<=col; j++)
    //     {
    //         level[i][j]=inf;
    //     }
    // }
    queue<pll> q;
    q.push(src);
    level[src.ff][src.ss]=0;
    while(!q.empty())
    {
        pll pr=q.front();
        ll x=pr.ff, y=pr.ss;
        q.pop();
        if(x==des.ff && y==des.ss)
        {
            return level[des.ff][des.ss];
        }
        for(ll i=0; i<4; i++)
        {
            ll xx=x+dx[i], yy=y+dy[i];
            // if(valid(xx,yy) && level[x][y]+1<level[xx][yy
                ])
            if(valid(xx,yy) && level[xx][yy]==-1)
            {
                q.push(mp(xx,yy));
                level[xx][yy]=level[x][y]+1;
            }
        }
    }
    return level[des.ff][des.ss];
}
```

2.2 ArticulationPoints

```
int T, low[N], dis[N], art[N];
vector<int> g[N];
void dfs(int u, int pre = 0) {
  low[u] = dis[u] = ++T;
  int child = 0;
  for(auto v: g[u]) {
    if(!dis[v]) {
      dfs(v, u);
      low[u] = min(low[u], low[v]);
      if(low[v] >= dis[u] && pre != 0) art[u] = 1;
      ++child;
    }
    else if(v != pre) low[u] = min(low[u], dis[v]);
  }
  if(pre == 0 && child > 1) art[u] = 1;
}
```

2.3 BellmanFord

```
nt n,edge,src,edge_cost[N];
int d[N],edge_u[N],edge_v[N];
int main()
{
    int i.step;
    cin>>n>>m>>src;
```

```
    rep(i,n)
        d[i]=99999;
    d[src]=0;
    for(i=1;i<=m;i++)
        cin>>edge_u[i]>>edge_v[i]>>edge_cost[i];
    bool neg_cycle=false;
    for(step=1;step<=n;step++)
    {
        bool updated=false;
        rep(i,m)
        {
            int u=edge_u[i],v=edge_v[i],w=edge_cost[i];
            if(d[u]+edge_cost[i]<d[v])
            {
                updated=true;
                if(step==n)
                    neg_cycle=true;
                d[v]=d[u]+edge_cost[i];
            }
        }
        if(!updated)
            break;
    }
    if(!neg_cycle)
    {
        printf("Distance to node from source: \n");
        rep(i,n)
            cout<<i<<" "<<d[i]<<endl;
    }
    else
        cout<<"Negative cycle detected\n";
}
```

2.4 Bipartite

```
int edge[N][N];
int vis[N];
int color[N];
int m, n;

bool dfs(int u)
{
    for (int i = 0; i < n; i++)
    {
        if (edge[u][i])
        {
            if (!vis[i])
            {
                vis[i] = 1;
                color[i] = !color[u];
                dfs(i);
            }
            else if (color[i] == color[u])
            {
                return false;
            }
        }
    }
    return true;
}
//Bipartite-matching:
const ll N=105;
ll Left[N],Right[N],seen[N];
vll graph[N];
ll n,m;
bool kuhn(ll u)
{
    for(auto v: graph[u])
    {
        if(seen[v]) continue;
        seen[v]=1;
        if(Right[v]==-1 || kuhn(Right[v]))
        {
            Right[v]=u;
            Left[u]=v;
            return true;
        }
    }
    return false;
}
void bipartite_matching()
{
    memo(Left,-1);
    memo(Right,-1);
    ll cnt=0;
    for(ll i=0; i<m; i++) // m = left side er total nodes
    {
        memo(seen,0);
        if(kuhn(i)) cnt++;
    }
    cout<<cnt<<endl;
}
```

2.5 Bridge

```
const int N = 1e5 + 9;
vi adj[N];
int in[N], low[N], vis[N], timer = 0;
bool hasBridge = false;
vector<pii> edges;
void dfs(int u, int par)
{
    vis[u] = 1;
    in[u] = low[u] = timer++;
    for (auto v : adj[u])
    {
        if (v == par)
        {
            continue;
        }
        if (vis[v])
        { // backedge
            low[u] = min(in[v], low[u]);
            if (in[u] > in[v])
            {
                edges.pb({u, v});
            }
        }
        else
        {
            // forwardedge
            dfs(v, u);
            if (low[v] > in[u])
            {
                hasBridge = true;
                return;
                // cout << u << "to" << v << " is bridge" <<
                    endl;
            }
            edges.pb({u, v});
            low[u] = min(low[u], low[v]);
        }
    }
}
//Bridge-Tree
vector <int> g[N], tree[N];
int n, m, in[N], low[N], ptr, compID[N];

void go (int u, int par = -1) {
  in[u] = low[u] = ++ptr;
  for (int v : g[u]) {
    if (in[v]) {
      if (v == par) par = -1;
      else low[u] = min(low[u], in[v]);
    } else {
      go(v, u);
      low[u] = min(low[u], low[v]);
    }
```

```
  }
}

void shrink (int u, int id) {
  compID[u] = id;
  for (int v : g[u]) if (!compID[v]) {
    if (low[v] > in[u]) {
      tree[id].emplace_back(++ptr);
      shrink(v, ptr);
    } else {
      shrink(v, id);
    }
  }
}

int main() {
  cin >> n >> m;
  while (m--) {
    int u, v;
    scanf("%d %d", &u, &v);
    g[u].emplace_back(v);
    g[v].emplace_back(u);
  }
  for (int i = 1; i <= n; ++i) if (!in[i]) go(i);
  vector <int> roots; ptr = 0;
  for (int i = 1; i <= n; ++i) if (!compID[i]) {
    roots.emplace_back(++ptr);
    shrink(i, ptr);
  }
  return 0;
}
```

2.6 CycleDetection

```
vector<ll> graph[100000+5];
bool vis[100000+5];
bool vis2[100000+5];
ll n,m;
bool cycle=0;
void cycleDFS(ll u)
{
    vis[u]=1;
    vis2[u]=1;
    for(auto v: graph[u])
    {
        if(!vis[v])
        {
            cycleDFS(v);
        }
        else if(vis2[v])
        {
            cycle=1;
        }
    }
    vis2[u]=0;
}
```

2.7 DSU

```
int parent[N];
int sz[N];

void make_set(int n){
    for(int i = 1; i <= n; i++){
        parent[i] = i;
        sz[i] = 1;
    }
}

int find_set(int u){
    if(parent[u]==u) return u;
    return parent[u] = find_set(parent[u]);
}

void union_set(int u, int v){
    int a = find_set(u);
    int b = find_set(v);

    if(sz[a]<sz[b]) swap(a, b);
    if(a!=b){
        parent[b] = a;
        sz[a] += sz[b];
    }
}
```

2.8 Dijkstra

```
int n, m;
vector<ar<int, 2>> adj[MAX_N];
vector<ll> dist;

void dijkstra(int s)
{
    dist.assign(n + 1, LINF);
    priority_queue<ar<ll, 2>, vector<ar<ll, 2>>, greater<ar<
        ll, 2>>> pq;
    dist[s] = 0;
    pq.push({0, s});
    while (pq.size())
    {
        auto [d, u] = pq.top();
        pq.pop();
        if (d > dist[u])
            continue;
        for (auto [v, w] : adj[u])
        {
            if (dist[v] > dist[u] + w)
            {
                dist[v] = dist[u] + w;
                pq.push({dist[v], v});
            }
        }
    }
}

//Second Shortest
ll dis1[n+5],disn[n+5];
    dij(1);
    for(ll i=1; i<=n; i++) dis1[i]=dis[i];
    dij(n);
    for(ll i=1; i<=n; i++) disn[i]=dis[i];
    ll minn2=inf;
    for(ll u=1; u<=n; u++)
    {
        for(auto v: graph[u])
        {
            ll dist=dis1[u]+disn[v.ss];
            dist+=v.ff;
            if(dist>dis1[n])
            {
                minn2=min(minn2,dist);
            }
        }
    }
    cout<<"Second shortest distance: "<<minn2<<endl;
```

2.9 KrushkalMST

```
struct dsu {
  vector<int> par, rnk, size; int c;
  dsu(int n) : par(n+1), rnk(n+1,0), size(n+1,1), c(n) {
    for (int i = 1; i <= n; ++i) par[i] = i;
  }
  int find(int i) { return (par[i] == i ? i : (par[i] = find
      (par[i]))); }
```

```
  bool same(int i, int j) { return find(i) == find(j); }
  int get_size(int i) { return size[find(i)]; }
  int count() { return c; } //connected components
  int merge(int i, int j) {
    if ((i = find(i)) == (j = find(j))) return -1; else --c;
    if (rnk[i] > rnk[j]) swap(i, j);
    par[i] = j; size[j] += size[i];
    if (rnk[i] == rnk[j]) rnk[j]++;
    return j;
  }
};

int32_t main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  int n, m; cin >> n >> m;
  vector<array<int, 3>> ed;
  for(int i = 1; i <= m; i++){
    int u, v, w; cin >> u >> v >> w;
    ed.push_back({w, u , v});
  }
  sort(ed.begin(), ed.end());
  long long ans = 0;
  dsu d(n);
  for (auto e: ed){
    int u = e[1], v = e[2], w = e[0];
    if (d.same(u, v)) continue;
    ans += w;
    d.merge(u, v);
  }
  cout << ans << '\n';
  return 0;
}
```

2.10 SCC

```
// given a directed graph return the minimum number of edges
     to be added so that the whole graph become an SCC

bool vis[N];
vector<int> g[N], r[N], G[N], vec; //G is the condensed
    graph
void dfs1(int u) {
  vis[u] = 1;
  for(auto v: g[u]) if(!vis[v]) dfs1(v);
  vec.push_back(u);
}

vector<int> comp;
void dfs2(int u) {
  comp.push_back(u);
  vis[u] = 1;
  for(auto v: r[u]) if(!vis[v]) dfs2(v);
}

int idx[N], in[N], out[N];
int main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);

  int n, m;
  cin >> n >> m;
  for(int i = 1; i <= m; i++) {
    int u, v;
    cin >> u >> v;
    g[u].push_back(v);
    r[v].push_back(u);
  }
  for(int i = 1; i <= n; i++) if(!vis[i]) dfs1(i);
  reverse(vec.begin(), vec.end());
  memset(vis, 0, sizeof vis);
  int scc = 0;
  for(auto u: vec) {
    if(!vis[u]) {
      comp.clear();
      dfs2(u);
      scc++;
      for(auto x: comp) idx[x]=scc;
    }
  }
  for(int u = 1; u <= n; u++) {
    for(auto v: g[u]) {
      if(idx[u] != idx[v]) {
        in[idx[v]]++, out[idx[u]]++;
        G[idx[u]].push_back(idx[v]);
      }
    }
  }
  int needed_in=0, needed_out=0;
  for(int i = 1; i <= scc; i++) {
    if(!in[i]) needed_in++;
    if(!out[i]) needed_out++;
  }
  int ans = max(needed_in, needed_out);
  if(scc == 1) ans = 0;
  cout << ans << '\n';
  return 0;
}
```

3 Ishraqfatin7-template

```
#include <bits/stdc++.h>
using namespace std;
using ll = long long;
#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define FAST                          \
    ios_base::sync_with_stdio(false); \
    cin.tie(0);
int dx[] = {1, 0, -1, 0};
int dy[] = {0, 1, 0, -1};
//PBDS
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace __gnu_pbds;
template <class T>
using ordered_set = tree<T, null_type, less<T>, rb_tree_tag,
     tree_order_statistics_node_update>;
template <class T>
using multi_ordered_set = tree<T, null_type, less_equal<T>,
    rb_tree_tag, tree_order_statistics_node_update>;
template <class T>
using r_ordered_set = tree<T, null_type, greater<T>,
    rb_tree_tag, tree_order_statistics_node_update>;
template <class T>
using r_multi_ordered_set = tree<T, null_type, greater_equal
    <T>, rb_tree_tag, tree_order_statistics_node_update>;
// s.order_of_key(k); --> number of items strictly smaller
    than k
// s.find_by_order(k); --> k-th item in set (0-indexing) (
    returns iterator)
//horse
// ll dx[] = {-1, 1, -1, 1, -2, 2, -2, 2};
// ll dy[] = {2, 2, -2, -2, 1, 1, -1, -1};
```

4 NumberTheory

4.1 CRT

```
ll mod_expo(ll x, ll y, ll m)
{
    if(y==0)return 1;
    ll ans=mod_expo(x,y/2,m);
    ans*=ans;
    ans%=m;
    if(y%2==0)return ans%m;
```

```
    else return ans*x%m;
}
int main()
{
    ll n;
    cin>>n;
    ll a[n],m[n],M,M1[n],x,M1_inv[n];
    M=1;
    for(ll i=0; i<n; i++)
    {
        cin>>m[i]>>a[i];
        M*=m[i];
    }
    for(ll i=0; i<n; i++)
    {
        M1[i]=M/m[i];
        ll y=M1[i]*a[i]%M,z=0;
        for(ll j=1;; j++)
        {
            ll ans=M1[i]*j%m[i];
            z=(z+y)%M;phi
            if(ans==1)
            {
                M1[i]=z;
                break;
            }
        }
    }
    x=0;
    for(ll i=0; i<n; i++)
    {
        x=(x+M1[i])%M;
    }
    cout<<x;
    return 0;
}
```

4.2 Pollard-Rho

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

using ll = long long;
namespace PollardRho {
  mt19937 rnd(chrono::steady_clock::now().time_since_epoch()
       .count());
  const int P = 1e6 + 9;
  ll seq[P];
  int primes[P], spf[P];

  inline ll add_mod(ll x, ll y, ll m) {
    return (x += y) < m ? x : x - m;
  }
  inline ll mul_mod(ll x, ll y, ll m) {
    ll res = __int128(x) * y % m;
    return res;
    // ll res = x * y - (ll)((long double)x * y / m + 0.5) *
         m;
    // return res < 0 ? res + m : res;
  }
  inline ll pow_mod(ll x, ll n, ll m) {
    ll res = 1 % m;
    for (; n; n >>= 1) {
      if (n & 1) res = mul_mod(res, x, m);
      x = mul_mod(x, x, m);
    }
    return res;
  }
  // O(it * (logn)^3), it = number of rounds performed
  inline bool miller_rabin(ll n) {
    if (n <= 2 || (n & 1 ^ 1)) return (n == 2);
    if (n < P) return spf[n] == n;
    ll c, d, s = 0, r = n - 1;
    for (; !(r & 1); r >>= 1, s++) {}
    // each iteration is a round
    for (int i = 0; primes[i] < n && primes[i] < 32; i++) {
      c = pow_mod(primes[i], r, n);
      for (int j = 0; j < s; j++) {
        d = mul_mod(c, c, n);
        if (d == 1 && c != 1 && c != (n - 1)) return false;
        c = d;
      }
      if (c != 1) return false;
    }
    return true;
  }
  void init() {
    int cnt = 0;
    for (int i = 2; i < P; i++) {
      if (!spf[i]) primes[cnt++] = spf[i] = i;
      for (int j = 0, k; (k = i * primes[j]) < P; j++) {
        spf[k] = primes[j];
        if (spf[i] == spf[k]) break;
      }
    }
  }
  // returns O(n^(1/4))
  ll pollard_rho(ll n) {
    while (1) {
      ll x = rnd() % n, y = x, c = rnd() % n, u = 1, v, t =
           0;
      ll *px = seq, *py = seq;
      while (1) {
        *py++ = y = add_mod(mul_mod(y, y, n), c, n);
        *py++ = y = add_mod(mul_mod(y, y, n), c, n);
        if ((x = *px++) == y) break;
        v = u;
        u = mul_mod(u, abs(y - x), n);
        if (!u) return __gcd(v, n);
        if (++t == 32) {
          t = 0;
          if ((u = __gcd(u, n)) > 1 && u < n) return u;
        }
      }
      if (t && (u = __gcd(u, n)) > 1 && u < n) return u;
    }
  }
  vector<ll> factorize(ll n) {
    if (n == 1) return vector <ll>();
    if (miller_rabin(n)) return vector<ll> {n};
    vector <ll> v, w;
    while (n > 1 && n < P) {
      v.push_back(spf[n]);
      n /= spf[n];
    }
    if (n >= P) {
      ll x = pollard_rho(n);
      v = factorize(x);
      w = factorize(n / x);
      v.insert(v.end(), w.begin(), w.end());
    }
    return v;
  }
}
int32_t main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  PollardRho::init();
  int t; cin >> t;
  while (t--) {
    ll n; cin >> n;
    auto f = PollardRho::factorize(n);
    sort(f.begin(), f.end());
    cout << f.size() << ' ';
    for (auto x: f) cout << x << ' '; cout << '\n';
  }
  return 0;
}
// https://judge.yosupo.jp/problem/factorize
```

4.3 SOD

```
//SNOD:
ll SNOD(ll n)
{
   ll sum=0;
   for(ll i=1; i<=n; i++)
   {
      sum+=(n/i);
   }
   return sum;
}

//SOD:
// SOD(12)=1+2+3+4+6+12,
// SOD(12)=(2^03^0)+(2^1 3 ^0)+(2^0 3 ^1)+(2^2 3 ^0)+(2^1 3 ^1)
     +(2^2 3 ^1),
// SOD(12)=2^0(3^0+3^1)+2^1(3^0+3^1)+2^2(3^0+3^1),
// SOD(12)=(2^0+2^1+2^2)(3^0+3^1)
// SOD(N)=(p1^0+p1^1+..+p1^a1)*(p2^0+p2^1+..+p2^a2)*...*(pk
     ^0+pk^1+..+pk^ak)
// p1^0+p1^1+..+p1^a1 = (p1^(a1+1)-1)/(p1-1)
ll modInverse(ll a)
{
    return fastExpo(a, MOD - 2);
}
ll rangeSumModulo(ll a, ll b)
{
    return ((((b - a + 1) % MOD) * ((a + b) % MOD)) % MOD) *
         modInverse(2) % MOD;
}
void solve()
{
    ll n;
    cin >> n;
    ll sum = 0;
    ll curr = 1;
    while (curr <= n)
    {
        ll divs = n / curr;
        ll next = n / divs + 1;
        sum += (divs % MOD) * (rangeSumModulo(curr, next - 1)
              % MOD) % MOD;
        sum %= MOD;
        curr = next;
    }
    cout << sum << endl;
}
```

4.4 euler-phi

```
const int n = 10;
vector<int> phi(n + 1, 0);
void phi_1_to_N()
{
    phi[0] = 0;
    phi[1] = 1;
    for (int i = 2; i <= n; i++)
        phi[i] = i;
    for (int i = 2; i <= n; i++)
    {
        if (phi[i] == i)
        {
            for (int j = i; j <= n; j += i)
            {
                phi[j] -= phi[j] / i;
            }
        }
    }
}
int PHI(int n)
{
    int result = n;
    for (int i = 2; i * i <= n; i++)
    {
        if (n % i == 0)
        {
            while (n % i == 0)
            {
                n /= i;
            }
            result -= result / i;
        }
    }
    if (n > 1)
        result -= result / n;
    return result;
}
```

4.5 gcd-template

```
#include <bits/stdc++.h>
using namespace std;
#define ar array
#define ll long long
const int MAX_N = 1e5 + 5;
const ll MOD = 1e9 + 7;
const ll INF = 1e9;
int gcd(int a, int b)
{
    return b ? gcd(b, a % b) : a;
}
// extended version to find x, y such that ax + by = gcd(a,
     b)
ll gcd(ll a, ll b, ll &x, ll &y)
{
    if (b == 0)
    {
        x = 1, y = 0;
        return a;
    }
    ll x1, y1, d = gcd(b, a % b, x1, y1);
    x = y1;
    y = x1 - y1 * (a / b);
    return d;
}
// find a solution of a Linear Diophantine Equation
bool lde(ll a, ll b, ll c, ll &x, ll &y)
{
    ll d = gcd(abs(a), abs(b), x, y);
    if (c % d)
        return false;
    x *= c / d;
    y *= c / d;
    if (a < 0)
        x = -x;
    if (b < 0)
        y = -y;
    return true;
}
void shift(ll a, ll b, ll &x, ll &y, ll cnt)
{
    x += cnt * b;
    y -= cnt * a;
}
ll inv_mod(ll a, ll m)
{
    ll x, y;
    gcd(a, m, x, y);
    return (m + x % m) % m;
}
// solve ax = b (mod m)
ll lce(ll a, ll b, ll m)
{
    ll d = gcd(a, m);
    if (d != 1)
    {
```

```
        if (b % d)
            return -1;
        a /= d;
        b /= d;
        m /= d;
    }
    return b * inv_mod(a, m) % m;
}
```

4.6 n!primeFact

```
for(auto x: primes)
{
    ll y = x;
    ll curr = 0;
    while(y <= n)
    {
       curr += n/y;
       y *= x;
    }
    ans *= (((curr+1)*(curr+2))/2)%MOD;
    ans %= MOD;
}
```

4.7 ncr-npr

```
void calcFact(ll n)
{
    F[0]=F[1]=1;
    for(ll i=2; i<=n; i++)
    {
        F[i]=(F[i-1]*i)%MOD;
    }
}
ll C(ll n, ll r)
{
    if(r>n) return 0;
    ll ret=F[n];
    ret=(ret*fastExpo(F[r],MOD-2,MOD))%MOD;
    ret=(ret*fastExpo(F[n-r],MOD-2,MOD))%MOD;
    return ret;
}
ll P(ll n, ll r)
{
    if(r>n) return 0;
    ll ret=F[n];
    ret=(ret*fastExpo(F[n-r],MOD-2,MOD))%MOD;
    return ret;
}
```

4.8 nth-permutation

```
vector<int>nth_permutation(int cnt,int n)
{
    vector<int>idx(cnt),per(cnt),fac(cnt);
    for(int i=0;i<cnt;i++)idx[i]=i;
    for(int i=1;i<=cnt;i++)
    {
        fac[i-1]=n%i;
        n/=i;
    }
    for(int i=cnt-1; i>=0; i--)
    {
        per[cnt-i-1]=idx[fac[i]],
       idx.erase(idx.begin()+fac[i]);
    }
    return per;
}
```

4.9 segmentedSieve

```
#define N 50000
vector<int>prime;
bool flag[100005],check[N];
void sieve()
{
     int i,j;
     prime.pb(2);
     for(i=3;i*i<=N;i+=2)
     {
          if(check[i]==0)
          {
              prime.pb(i);
 for(j=(i*i);j<=N;j+=(2*i))
      check[j]=1;
          }
     }
     for(;i<=N;i+=2)
     {
        if(check[i]==0)
            prime.pb(i);
     }
}
int seg(ll a,ll b)
{
    int i,ans=0;
    for(i=0;i<(b-a+1);i++)
        flag[i]=0;
    if(a<2)
        a=2;
    for(i=0;(ll)(prime[i]* prime[i])<=b && i<prime.size();i
         ++)
    {
        ll j=(prime[i]*(a/prime[i]));
        if(j<a)
            j+=prime[i];
        if(j<(ll)(prime[i]+prime[i]))
            j=prime[i]+prime[i];
        for(;j<=b;j+=prime[i])
            flag[j-a]=1;
    }
    for(i=0;i<(b-a+1);i++)
    {
        if(flag[i]==0)
            ans++;
    }
    return ans;
}
}
Int main()
{
    sieve();
    seg(a,b);
}
```

4.10 spf-gpf

```
#include <bits/stdc++.h>
using namespace std; // for
#define FAST                          \
    ios_base::sync_with_stdio(false); \
    cin.tie(0);
using ll = long long;
const int N = 1e6 + 9;
ll lpf, gpf = 0, dpf = 0, npf = 0, ndiv = 1, sdiv = 1;
ll spf[N];
ll arr[N];
int mark[N];
void sieve()
{

    for (ll i = 2; i * i <= N; i++)
    {
        if (!mark[i])
```

```
        {
            spf[i] = i;
            for (ll j = i * 2; j <= N; j += i)
            {
                mark[j] = 1;
                if (spf[j] == 0)
                    spf[j] = i;
            }
        }
    }
    for (ll i = 2; i <= N; i++)
        if (spf[i] == 0)
            spf[i] = i;
}
ll fastExpo(ll a, ll b)
{
    ll res = 1;
    while (b)
    {
        if (b & 1)
        {
            res = (res * a);
        }
        a = (a * a);
        b >>= 1;
    }
    return res;
}

int main()
{
    FAST;
    sieve();
    int n;
    cin >> n; // read
    int arr[n];
    for (int i = 0; i < n; i++)
        cin >> arr[i];

    for (int i = 0; i < n; i++)
    {
        ll x = arr[i];
        lpf = spf[x], dpf = 0, gpf = 0, npf = 0, ndiv = 1,
            sdiv = 1;
        while (x > 1)
        {
            gpf = max(gpf, spf[x]);
            ll d = spf[x];
            ll count = 0;
            while (!(x % d))
            {
                x /= d;
                count++;
                npf++;
            }
            ndiv *= (count + 1);
            sdiv *= (fastExpo(d, count + 1) - 1) / (d - 1);
            dpf++;
        }
        printf("%lld %lld %lld %lld %lld %lld\n", lpf, gpf,
            dpf, npf, ndiv, sdiv);
    }
}
```

4.11 subset

```
#include <bits/stdc++.h>

using namespace std;

int main()
{
    cout << __builtin_parity(15) << '\n';
    cout << __builtin_parity(31) << '\n';
    int l, r, k;
    cin >> l >> r >> k;
    vector<int> prime(k);
    /// prime = {2, 3, 5} --> k = 3
    for (int &x : prime)
        cin >> x;
    int ans = 0;
    for (int mask = 1; mask < 1 << k; ++mask)
    {
        int prod = 1;
        for (int i = 0; i < k; ++i)
        {
            if (mask & 1 << i)
                prod *= prime[i];
        }
        int here = r / prod - (l - 1) / prod;
        if (__builtin_parity(mask))
            ans += here;
        else
            ans -= here;
    }
    cout << ans << '\n';
    return 0;
}
```