

Team notebook

Team Tapu_Sena

January 1, 2025

Contents

| | |
|--------------------------------|-----------|
| 1 BasicNumberTheory | 1 |
| 2 DP | 1 |
| 2.1 Digitdp | 1 |
| 3 DataStructures | 2 |
| 3.1 BIT | 2 |
| 3.2 DSU | 2 |
| 3.3 Mo's | 3 |
| 3.4 SegTree | 3 |
| 4 Graph | 3 |
| 4.1 Bridges | 3 |
| 4.2 Detectprintcycle | 4 |
| 4.3 Dijkstra | 5 |
| 4.4 Floydwarshall | 5 |
| 4.5 LCA | 6 |
| 4.6 MST | 6 |
| 4.7 SCC | 7 |
| 4.8 Toposort | 7 |
| 4.9 TreesSection | 8 |
| 4.10 Trie | 10 |
| 5 Maths | 10 |
| 5.1 Combwithoutmod | 10 |
| 5.2 Factinverse | 10 |
| 5.3 MatrixExpo | 11 |
| 5.4 NextPermutation | 11 |
| 5.5 SQRT | 11 |

| | |
|----------------------------|-----------|
| 5.6 SieveRelated | 12 |
| 5.7 nCr | 13 |
| 6 Random | 13 |
| 6.1 Misc | 13 |
| 7 Runflag | 14 |
| 8 Stresstest | 14 |
| 8.1 brute | 14 |
| 8.2 code | 14 |
| 8.3 gen | 15 |
| 8.4 stress | 15 |
| 9 Strings | 15 |
| 9.1 FindDiff | 15 |
| 9.2 Hashing | 16 |
| 9.3 KMP | 17 |
| 10 template | 17 |

1 BasicNumberTheory

```
/*-----Number theory
Starts-----*/
ll gcd(ll a, ll b) {if (b > a) {return gcd(b, a);} if (b == 0) {return
a;} return gcd(b, a % b);}
ll expo(ll a, ll b, ll mod) {ll res = 1; while (b > 0) {if (b & 1)res =
(res * a) % mod; a = (a * a) % mod; b = b >> 1;} return res;}
```

```

void extendgcd(ll a, ll b, ll*v) {if (b == 0) {v[0] = 1; v[1] = 0; v[2] = a; return;} extendgcd(b, a % b, v); ll x = v[1]; v[1] = v[0] - v[1] * (a / b); v[0] = x; return;} //pass an array of size 3
ll mminv(ll a, ll b) {ll arr[3]; extendgcd(a, b, arr); return arr[0];}
//for non prime b
ll mminvprime(ll a, ll b) {return expo(a, b - 2, b);}
bool revsort(ll a, ll b) {return a > b;}
ll combination(ll n, ll r, ll m, ll *fact, ll *ifact) {ll val1 = fact[n]; ll val2 = ifact[n - r]; ll val3 = ifact[r]; return ((val1 * val2) % m) * val3 % m;}
void google(int t) {cout << "Case #" << t << ": ";}
vector<ll> sieve(int n) {int*arr = new int[n + 1](); vector<ll> vect; for (int i = 2; i <= n; i++)if (arr[i] == 0) {vect.push_back(i); for (int j = 2 * i; j <= n; j += i)arr[j] = 1;} return vect;}
ll mod_add(ll a, ll b, ll m) {a = a % m; b = b % m; return ((a + b) % m + m) % m;}
ll mod_mul(ll a, ll b, ll m) {a = a % m; b = b % m; return ((a * b) % m + m) % m;}
ll mod_sub(ll a, ll b, ll m) {a = a % m; b = b % m; return ((a - b) % m + m) % m;}
ll mod_div(ll a, ll b, ll m) {a = a % m; b = b % m; return (mod_mul(a, mminvprime(b, m), m) + m) % m;} //only for prime m
ll phin(ll n) {ll number = n; if (n % 2 == 0) {number /= 2; while (n % 2 == 0) n /= 2;} for (ll i = 3; i <= sqrt(n); i += 2) {if (n % i == 0) {while (n % i == 0)n /= i; number = (number / i * (i - 1));}} if (n > 1)number = (number / n * (n - 1)); return number;} //O(sqrt(N))
ll bpow(ll base,ll power){ll res=1;while(power){if(power&1){res *= base;power--;}else{base *= base;power /=2;}}return res;}
bool power_of_two(ll x){ return x && (!(x&(x-1)));}
ll nearest_power_of_two(ll x){ ll ans = 1; while(ans<x){ ans <<= 1;}return ans;}
/*-----Number theory
Ends-----*/

```

2 DP

2.1 Digitdp

```

//(number * 10 + i) % k --> divisibilty check
#define ll long long

```

```

class Solution {

```

```

public:
ll dp[15][15][15][2][24][2];
ll recur(ll pos,ll evenCnt,ll oddCnt,bool leading,ll num,bool tight,string &s,int k){
if(pos>=s.length()){
if(evenCnt==oddCnt && !leading && num==0){
return 1;
}
return 0;
}
if(dp[pos][evenCnt][oddCnt][leading][num][tight]!==-1)
return dp[pos][evenCnt][oddCnt][leading][num][tight];

ll ans=0;
ll ub=((tight)?s[pos]-'0':9);
for(ll dig=0;dig<=ub;++dig){
if(dig==0 && leading){
ans+=recur(pos+1,evenCnt,oddCnt,leading,num,tight&(dig==ub),s,k);
}else if(dig%2){
ans+=recur(pos+1,evenCnt,oddCnt+1,0,(num*10+dig)%k,tight&(dig==ub));
}else if(dig%2==0){
ans+=recur(pos+1,evenCnt+1,oddCnt,0,(num*10+dig)%k,tight&(dig==ub));
}
}
return dp[pos][evenCnt][oddCnt][leading][num][tight]=ans;
}
int numberOfBeautifulIntegers(int low, int high, int k) {
low--;
string l=to_string(low);
string r=to_string(high);
memset(dp,-1,sizeof(dp));

//
ll fr=recur(0,0,0,1,0,1,r,k);
memset(dp,-1,sizeof(dp));
//
ll fl=recur(0,0,0,1,0,1,l,k);
ll ans=fr-fl;
return ans;
}
};

```

3 DataStructures

3.1 BIT

```

class BIT{
    vector<ll> bit;
public:
    BIT(ll n){
        bit.resize(n+1,0);
    }
    void update(ll x, ll val,ll n){
        for(; x <= n; x += x&-x)
            bit[x] += val;
    }
    ll query(ll x){
        ll sum = 0;
        for(; x > 0; x -= x&-x)
            sum += bit[x];
        return sum;
    }
};
//BIT bt(n);
//for(ll i=1;i<=n;++i){
//bt.update(i,temp[i],n);
//1-based indexing so input in vector from 1 to <=n

```

3.2 DSU

```

class DSU{
    vector<ll> par,size;
public:
    ll tot_components;
    DSU(ll n){
        size.resize(n+1,1);
        par.resize(n+1);
        for(ll i=1;i<=n;++i)
            par[i]=i;
        tot_components=n;
    }
    ll findPar(ll node){
        if (node==par[node])
            return node;

```

```

        return par[node]=findPar(par[node]);
    }
    ll getsize(ll node){
        return size[findPar(node)];
    }
    void unite(ll u,ll v){
        ll ult_u=findPar(u);
        ll ult_v=findPar(v);
        if(ult_u==ult_v)return;
        if(size[ult_u]<size[ult_v]){
            size[ult_v]+=size[ult_u];
            par[ult_u]=ult_v;
        }else{
            size[ult_u]+=size[ult_v];
            par[ult_v]=ult_u;
        }
        tot_components--;
    }
};

```

3.3 Mo's

```

const int N = 2e5 + 5;
const int Q = 2e5 + 5;
const int M = 1e6 + 5;
const int SZ = sqrt(N) + 1;
struct var{
    ll l , r , idx;
} qr[Q];
int n , q , a[N]; ll freq[M];
ll ans[Q]; ll cur = 0;
bool comp(var &d1, var &d2){
    int b1 = d1.l / SZ;
    int b2 = d2.l / SZ;
    if(b1 != b2){
        return b1 < b2;
    }else{
        return (b1 & 1) ? d1.r < d2.r : d1.r > d2.r;
    }
}
}
inline void add(ll x){...}
inline void del(ll x){...}

```

```

void mo(){
    cin >> n >> q;
    for(int i = 1; i<= n ; i++){cin >> a[i];
    for(int i = 1; i<= q ; i++){
        cin >> qr[i].l >> qr[i].r;
        qr[i].idx = i;
    }
    sort(qr+1, qr+q+1 , comp);
    for(int i = 1; i<= q ; i++){
        while(l < qr[i].l) remove(a[l++]);
        while(l > qr[i].l) add(a[--l]);
        while(r < qr[i].r) add(a[++r]);
        while(r > qr[i].r) remove(a[r--]);
        ans[qr[i].idx] = cur;
    }
}

```

3.4 SegTree

```

struct segtree{
    vector<ll> v; ll id=0, sz; //id for initialization
    segtree(ll _n){sz=1; while(sz<_n){sz<=<=1; v.assign(2*sz, id);}
    ll func(ll x, ll y){ return x+y; }
    void pull(ll x){ v[x]=func(v[2*x+1], v[2*x+2]); }
    void update(ll i, ll val, ll x, ll lx, ll rx){ // 0-based indexing
        if(rx-lx==1){ v[x]=val; return; }
        ll m=(lx+rx)/2;
        if(i<m){update(i, val, 2*x+1, lx, m); }
        else{update(i, val, 2*x+2, m, rx); }
        pull(x);
    }
    void update(ll i, ll val){ update(i, val, 0, 0, sz); }
    ll query(ll l, ll r, ll x, ll lx, ll rx){ // [l,r] 1->INCLUSIVE/
        r->EXCLUSIVE
        if(lx>=r || l>=rx) return id;
        if(lx>=l && rx<=r) return v[x];
        ll m=(lx+rx)/2;
        ll s1=query(l, r, 2*x+1, lx, m);
        ll s2=query(l, r, 2*x+2, m, rx);
        return func(s1, s2);
    }
    ll query(ll l, ll r){ return query(l, r, 0, 0, sz); }
};

```

4 Graph

4.1 Bridges

```

class Solution {
public:
    // to find all the bridges in the graph - tarjan's algorithm
    int timer=0;
    void dfs(int node, int par, vector<vector<int>> &adj, vector<int>
        &vis, vector<int> &tin, vector<int> &lowestTime, vector<vector<int>>
        &allBridges){
        vis[node]=1;
        timer++;
        for(auto &adjNode: adj[node]){
            if(!vis[adjNode]){
                tin[adjNode]=timer;
                lowestTime[adjNode]=timer;
                dfs(adjNode, node, adj, vis, tin, lowestTime, allBridges);
            }
            if(adjNode!=par && lowestTime[adjNode]>tin[node]){
                allBridges.push_back({node, adjNode});
            }
            else if(adjNode!=par){
                lowestTime[node]=min(lowestTime[node], lowestTime[adjNode]);
            }
        }
    }
    vector<vector<int>> criticalConnections(int n, vector<vector<int>> &
        connections) {
        vector<vector<int>> adj(n+1);
        vector<int> vis(n+1);
        vector<int> tin(n+1), lowestTime(n+1);
        vector<vector<int>> allBridges;
        int len=connections.size();
        for(int x=0; x<len; ++x){
            int node1=connections[x][0];
            int node2=connections[x][1];
            adj[node1].push_back(node2);
            adj[node2].push_back(node1);
        }
        for(int x=0; x<n; ++x){
            if(!vis[x]){
                tin[x]=timer;
                lowestTime[x]=timer;
                dfs(x, -1, adj, vis, tin, lowestTime, allBridges);
            }
        }
    }
};

```

```

    }
}
// for(int x=1;x<=n;++x){
//     cout<<tin[x]<<" "<<lowestTime[x]<<endl;
// }
return allBridges;
}
};

```

4.2 Detectprintcycle

//Applicable only for Without Loops and Multiple Edged Graphs -
undirected graphs

```

int n,m;
vector<vector<int>> adj;
vector<bool> visited,vis2;
vector<int> parent,dis;
vector<int> st;
int cycle_start, cycle_end;

bool dfs(int v, int par) { // passing vertex and its parent vertex
    visited[v] = true;
    for (int u : adj[v]) {
        if(u == par) continue; // skipping edge to parent vertex
        if (visited[u]) {
            cycle_end = v;
            cycle_start = u;
            return true;
        }
        parent[u] = v;
        if (dfs(u, parent[u]))
            return true;
    }
    return false;
}

void find_cycle() {
    visited.assign(n, false);
    parent.assign(n, -1);
    cycle_start = -1;

    for (int v = 0; v < n; v++) {
        if (!visited[v] && dfs(v, parent[v]))

```

```

        break;
    }

    if (cycle_start == -1) {
        cout << "IMPOSSIBLE" << endl;
    } else {
        vector<int> cycle;
        cycle.push_back(cycle_start);
        for (int v = cycle_end; v != cycle_start; v = parent[v])
            cycle.push_back(v);
        cycle.push_back(cycle_start);

        cout << "Cycle found: ";
        for (int v : cycle)
            st.pb(v);
        cout << endl;
    }
}

//Directed graph
vector<ll> path,vis;
vector<vector<ll>> adj;
stack<ll> st;
void dfs(ll node){
    if(path.size()!=0) return;
    st.push(node);
    vis[node]=1;
    for(auto &child:adj[node]){
        if(!vis[child]){
            dfs(child);
        }else if(vis[child]==1 && path.size()==0){
            path.pb(child);
            while(st.top()!=child){
                path.pb(st.top());
                st.pop();
            }
            path.pb(child);
            return;
        }
    }
    vis[node]=2;
    st.pop();
}

```

4.3 Dijkstra

```
vector<ll> dijkstra(vector<vector<pair<int,int>>> &adj,int n,int S){
    priority_queue<pair<ll, ll>, vector<pair<ll, ll>>, greater<pair<ll,
        ll>>> pq;
    vector<ll> dis(n + 1, 1e18);
    vector<ll> parent(n + 1);
    for (ll i = 1; i <= n; ++i)
        parent[i] = i;

    dis[S] = 0;
    pq.push({0, S});

    while (!pq.empty()){
        ll distance = pq.top().first;
        ll node = pq.top().second;
        pq.pop();
        //trying to add the set erase functionality somehow (though tc
        //remains the same for both pq and set)
        if(distance>dis[node]) continue;

        for (auto &i : adj[node]){
            ll childNode = i.first;
            ll edgeWeight = i.second;
            if (distance + edgeWeight < dis[childNode]){
                dis[childNode] = distance + edgeWeight;
                pq.push({dis[childNode], childNode});
                parent[childNode] = node;
            }
        }
    }
}
/*
if (dis[n] == 1e18)
    return {-1};
vector<ll> ans;
ll tNode = n;
ans.push_back(tNode);
while (parent[tNode] != tNode){
    ans.push_back(parent[tNode]);
    tNode = parent[tNode];
}
reverse(ans.begin(), ans.end());
return ans;
*/
return dis;
```

```
}
```

4.4 Floydwarshall

```
void shortest_distance(vector<vector<ll>> &matrix){
    for(int i=1;i<=n;++i){
        for(int j=1;j<=n;++j){
            if(matrix[i][j]==-1)
                matrix[i][j]=1e18;
        }
    }

    for(int val=1;val<=n;++val){
        for(int i=1;i<=n;++i){
            for(int j=1;j<=n;++j){
                matrix[i][j]=min(matrix[i][j],matrix[i][val]+matrix[val][j])
            }
        }
    }

    for(int i=1;i<=n;++i){
        for(int j=1;j<=n;++j){
            if(matrix[i][j]==1e18)
                matrix[i][j]=-1;
        }
    }
}
```

4.5 LCA

```
struct LCA{
    vector<vector<int>> up;
    vector<int> tin, tout, distance;
    int timer;
    vector<vector<int>>store;
    LCA(int n) {
        timer = 0;
        tin.resize(n);
        tout.resize(n);
        up.assign(n, vector<int>(21, -1));
        distance.assign(n, 0);
        store.assign(n, {});
    }
};
```

```

}
void dfs(int v, int p, vector<vector<int>> &adj, int
    dis,vector<int>count) {
    distance[v]=dis;
    tin[v] = ++timer;
    up[v][0] = p;
    for (int i = 1; i < 21; i++)
        up[v][i] = up[up[v][i - 1]][i - 1];

    for(int i=0;i<30;i++){
        if(cost[v]&(1<<i)) count[i]++;
    }
    store[v]=count;
    for (int u : adj[v]) {
        if (u != p)
            dfs(u, v, adj, dis+1,count);
    }
    tout[v] = ++timer;
}
bool is_ancestor(int u, int v) {
    return tin[u] <= tin[v] && tout[u] >= tout[v];
}
int lca(int u, int v) {
    if (is_ancestor(u, v))
        return u;
    if (is_ancestor(v, u))
        return v;
    for (int i = 20; i >= 0; i--) {
        if (!is_ancestor(up[u][i], v))
            u = up[u][i];
    }
    return up[u][0];
}
int dist(int u, int v) {
    int w = lca(u, v);
    return abs(distance[u] + distance[v] - 2*distance[w]);
}
};

```

4.6 MST

```

void solve(){
    ll n,m;

```

```

    cin>>n>>m;
    vector<vector<pll>> adj(n+1);
    fr(i,m){
        ll x,y,wt;
        cin>>x>>y>>wt;
        adj[x].pb({y,wt});
        adj[y].pb({x,wt});
    }
    priority_queue<pair<ll,pll>,vector<pair<ll,pll>>,greater<pair<ll,pll>>>
        pq;
    pq.push({0,{1,0}});
    ll ans=0;
    vector<ll> vis(n+1);
    while(!pq.empty()){
        ll wt=pq.top().ff;
        ll node=pq.top().ss.ff;
        ll par=pq.top().ss.ss;
        pq.pop();
        if(vis[node]) continue;
        ans+=wt;
        vis[node]=1;
        for(auto &i: adj[node]){
            if(!vis[i.ff]){
                pq.push({i.ss,{i.ff,node}});
            }
        }
    }
    for(ll i=1;i<=n;++i){
        if(!vis[i]){
            cout<<"IMPOSSIBLE"<<endl;
            return;
        }
    }
    cout<<ans<<endl;
}

```

4.7 SCC

```

vector<vector<ll>> adj,adjRev;
vector<ll> vis,order,ans;
void dfs(ll node,ll pass,ll component){
    vis[node]=1;
    vector<ll> newAdj=((!pass)?adj[node]:adjRev[node]);

```

```

    for(auto &i:newAdj){
        if(!vis[i]) dfs(i,pass,component);
    }
    if(!pass)order.pb(node);
    ans[node]=component;
}
void solve(){
    ll n,m;
    cin>>n>>m;
    adj.resize(n+1);
    adjRev.resize(n+1);
    vis.resize(n+1);
    ans.resize(n+1);
    for(ll i=0;i<m;i++){
        ll u,v;
        cin>>u>>v;
        adj[u].pb(v);
        adjRev[v].pb(u);
    }
    for(ll i=1;i<=n;++i){
        if(!vis[i]){
            dfs(i,0,0);
        }
    }
    reverse(all(order));
    vis.assign(n+1,0);
    ll components=1;
    for(auto &node:order){
        if(!vis[node]){
            dfs(node,2,components);
            components++;
        }
    }
    cout<<components-1<<endl;
    for(ll i=1;i<=n;++i){
        cout<<ans[i]<<" ";
    }
    cout<<endl;
}

```

4.8 Toposort

```

vector<int> findOrder(int N, vector<vector<int>> prerequisites) {
    vector<vector<int>> adjList(N+1);
    for(auto &i:prerequisites){
        adjList[i[0]].push_back(i[1]);
    }

    //using toposort bfs to detect the presence of a cycle
    vector<int> indegree(N+1,0);
    for(auto &i:adjList){
        for(auto &j:i)
            indegree[j]++;
    }
    queue<int> q;
    for(int i=1;i<=N;++i){
        if(indegree[i]==0)
            q.push(i);
    }
    vector<int> topo;
    while(!q.empty()){
        int node=q.front();
        q.pop();
        topo.push_back(node);
        for(auto &i:adjList[node])
        {
            indegree[i]--;
            if(indegree[i]==0)
                q.push(i);
        }
    }
    if(topo.size()==N)
        return topo;
    else
        return {};
}

```

4.9 TreesSection

```

//diameter of a tree
void dfs(ll node,ll par){
    for(auto &child:adj[node]){
        if(child==par)continue;
        depth[child]=depth[node]+1;
        dfs(child,node);
    }
}

```



```

}
void solve(){
    ll n;
    cin>>n;
    fr(i,n-1){
        ll x,y;
        cin>>x>>y;
        adj[x].pb(y);
        adj[y].pb(x);
    }
    ll max_depth=0;
    dfs(1,-1);
    ll ind=-1;
    for(ll i=1;i<=n;++i){
        if(depth[i]>max_depth){
            max_depth=depth[i];
            ind=i;
        }
        depth[i]=0;
    }
    max_depth=0;
    dfs(ind,-1);
    for(ll i=1;i<=n;++i){
        max_depth=max(max_depth,depth[i]);
    }
    cout<<max_depth<<endl;
}

// approach in-out dp on trees
ll dfs1(ll node,ll par){
    ll dis=0;
    sib_cnt[node]=1;
    for(auto child: adj[node]){
        if(child!=par){
            dfs1(child,node);
            sib_cnt[node]+=sib_cnt[child];
            dis+=in[child]+sib_cnt[child];
        }
    }
    in[node]=dis;
    return dis;
}

```

```

void dfs2(ll node,ll par){
    ll val=0;
    for(auto child: adj[node]){
        if(child!=par){
            val+=in[child]+sib_cnt[child]*2;
        }
    }
    for(auto &child:adj[node]){
        if(child!=par){
            out[child]=out[node]+(n-sib_cnt[node]+1)+val-in[child]-sib_cnt[child]*2;
            dfs2(child,node);
        }
    }
}

//BINARY-LIFTING
ll LOG;
vector<ll> depth;
vector<vector<ll>> up;
void precal(ll n,vector<ll> &parent){
    LOG=0;
    while((1<<LOG)<=n){
        LOG++;
    }
    up.resize(n+1,vector<ll>(LOG+1));
    depth.resize(n+1);
    for(ll i=1;i<LOG;++i){
        for(ll j=0;j<n;++j){
            up[j][0]=parent[j];
            if(j!=0){
                depth[j]=depth[parent[j]]+1;
            }
            up[j][i]=up[up[j][i-1]][i-1];
        }
    }
}

void solve(){
    ll n,q;
    cin>>n>>q;
    vector<ll> parent(n+1);
    for(ll i=1;i<n;++i){
        ll x;

```

```

    cin>>x;
    x--;
    parent[i]=x;
}
//printv(parent);
precal(n,parent);

for(ll i=0;i<q;++i){
    // line
    ll node,k;
    cin>>node>>k;
    node--;
    ll ans;
    if(depth[node]<k){
        cout<<-1<<endl;
    }else{
        for(ll i=0;i<LOG;++i){
            if(k&(1<<i)){
                node=up[node][i];
            }
        }
        cout<<node+1<<endl;
    }
}
}

//Euler Tour Technique -> to flatten the tree into an array
//Binary Indexed Tree (Fenwick Tree) -> to perform the update and query
operations on the flattened tree
vector<ll> start(200005),endd(200005);
vector<ll> val;ll timer = 1;

class BIT{
    vector<ll> bit;
public:
    BIT(ll n){
        bit.resize(n+1,0);
    }
    void update(ll x, ll val,ll n){
        for(; x <= n; x += x&-x)
            bit[x] += val;
    }
    ll query(ll x){
        ll sum = 0;

```

```

        for(; x > 0; x -= x&-x)
            sum += bit[x];
        return sum;
    }
};

void euler(ll node,ll par=-1){
    start[node]=timer++;
    for(auto child:adj[node]){
        if(child!=par){
            euler(child,node);
        }
    }
    endd[node]=timer-1;
}

```

4.10 Trie

```

class trie{
public:
    struct node{
        map<char, node*> children;
        int prefix;
        vector<string> wend;
        node(){
            prefix=0;
        }
    };
    node *root;
    trie(){
        root = new node();
    }
    void insert(string s){
        node* nd = root;
        int i=0;
        while(i<s.length()){
            if(nd->children[s[i]]!=NULL){
                nd = nd->children[s[i]];
                nd->prefix++;
                i++;
                continue;
            }
            nd->children[s[i]] = new node();

```

```

        nd = nd->children[s[i]];
        nd->prefix++;
        i++;
    }
    nd->wend.push_back(s);
}

int search_word(string s){
    node *nd = root;
    int i=0;
    while(i<s.length()){
        if(nd->children[s[i]]!=NULL){
            nd = nd->children[s[i]];
            if(nd->prefix==1)return i;
            i++;
        }
    }
    return i;
}
};

```

5 Maths

5.1 Combwithoutmod

```

ll nCr(int n, int r) {
    long double sum=1;
    for(int i = 1; i <= r; i++){
        sum = sum * (n - r + i) / i;
    }
    return sum;
}

```

5.2 Factinverse

```

const ll N = 100005;
vector<ll> f(N), invf(N);

ll power(ll a, ll b)

```

```

{
    if(b==0)
        return 1;
    else
    {
        ll x=power(a,b/2);
        ll y=(x*x)%modval;
        if(b%2)
            y=(y*a)%modval;
        return y;
    }
}

ll inverse(ll a)
{
    return power(a,modval-2);
}

void precompute()
{
    f[0]=1;
    for(int i=1;i<N;i++)
        f[i]=(f[i-1]*i)%modval;
    for(int i=0;i<N;i++)
        invf[i]=inverse(f[i]);
}

```

5.3 MatrixExpo

```

struct Matrix{
    vector<vector<ll>> a;
    Matrix(ll n,ll m){
        a.resize(n,vector<ll>(m,0));
    }
    Matrix operator *(const Matrix& other){
        ll x=a.size();
        ll y=a[0].size();
        ll z=other.a[0].size();
        Matrix product=Matrix(x,z);

        for(int i=0;i<x;i++){
            for(int j=0;j<y;j++){
                for(int k=0;k<z;k++){

```

```

        product.a[i][k]+=a[i][j]*other.a[j][k];
        product.a[i][k]%=modval;
    }
}
return product;
}
};

Matrix expo_power(Matrix a, ll k){
    Matrix product=Matrix(n,n);
    for(ll i=0;i<n;++i){
        product.a[i][i]=1;
    }
    while(k){
        if(k&1){
            product=product*a;
        }
        a=a*a;
        k>>=1;
    }
    return product;
}
//Matrix mat=Matrix(n,n);
//Matrix res=expo_power(mat,k);
//mat.a[i][j]=1;
//O(k^3logN)

```

5.4 NextPermutation

```

vector<vector<ll>> generate_all(vector<ll>v){
    vector<vector<ll>>ans;
    do{
        ans.pb(v);
    }while(next_permutation(all(v)));
    return ans;
}

```

5.5 SQRT

```

ll sqrt(ll a, ll p) {
    a %= p; if (a < 0) a += p;
    if (a == 0) return 0;
    assert(modpow(a, (p-1)/2, p) == 1); // else no solution
    if (p % 4 == 3) return modpow(a, (p+1)/4, p);
    // a^(n+3)/8 or 2^(n+3)/8 2^(n1)/4 works if p % 8 == 5
    ll s = p - 1, n = 2;
    int r = 0, m;
    while (s % 2 == 0)
        ++r, s /= 2;
    while (modpow(n, (p - 1) / 2, p) != p - 1) ++n;
    ll x = modpow(a, (s + 1) / 2, p);
    ll b = modpow(a, s, p), g = modpow(n, s, p);
    for (; r = m) {
        ll t = b;
        for (m = 0; m < r && t != 1; ++m)
            t = t * t % p;
        if (m == 0) return x;
        ll gs = modpow(g, 1LL << (r - m - 1), p);
        g = gs * gs % p;
        x = x * gs % p;
        b = b * g % p;
    }
}

```

5.6 SieveRelated

```

// prints all the prime numbers of a number
void printprimefactors(ll n){
    if(n<=1) return;
    while(n%2==0){
        cout<<2;
        n=n/2;
    }
    while(n%3==0){
        cout<<3;
        n=n/3;
    }
    for (ll i = 5; i*i<=n; i=i+6){
        while (n%i==0){
            cout<<i;
            n=n/i;
        }
    }
}

```

```

        while(n%(i+2)==0){
            cout<<i+2;
            n=n/(i+2);
        }
    }
    if(n>3) cout<<n;
    return;
    //i/p-->450
    //o/p-->23355
}
//tc-theta(sqrt(n))

// checks if a number is prime or not
bool isPrime(ll n){
    if(n==1) return false;
    if(n==2 || n==3) return true;
    if(n%2==0 || n%3==0) return false;
    for(ll i=5;i*i<=n;i=i+6)
        if(n%i==0 || n%(i+2)==0)
            return false;
    return true;
}

// finds the shortest prime factor for all numbers
const ll MAXN = 1e6+5;
vector<ll> spf(MAXN,1);

void sieve() {
    spf[0] = 0;
    for (int i = 2; i <= MAXN; i++) {
        if (spf[i] == 1) {
            for (int j = i; j <= MAXN; j += i) {
                if (spf[j] == 1)
                    spf[j] = i;
            }
        }
        // cout<<spf[i]<<" ";
    }
}

// stores all the prime numbers till n
vector<ll> sv(ll n){
    int *arr = new int[n+1]();

```

```

vector<ll> vect;
for(int i = 2 ; i<= n ; i++){
    if(arr[i] == 0){
        vect.push_back(i);
        for(int j = 2*i ; j <= n ; j += i){
            arr[j] = 1;
        }
    }
}
return vect;
}

void divisors(ll n){
    for (ll i = 1; i*i <=n; i++){
        if(n%i==0){
            cout<<i<<" ";
            if(i!=n/i)
                cout<<n/i<<" ";
        }
    }
}

```

5.7 nCr

```

struct nCr{
    ll maxx , md;
    vll fact, ifact;
    inline ll mul(ll a, ll b) { return a *1LL* b % md ;}
    ll power(ll a, ll n) {
        if(n == 0) return 1 ;
        int p = power(a, n/2) % md ;
        p = mul(p, p) ;
        return n & 1 ? mul(p, a) : p ;
    }
    int invMod(int a) {return power(a,md-2);}
    void pre() {
        fact[0] = 1 ;
        for(int i = 1;i< maxx;++i) fact[i] = mul(i, fact[i-1]) ;
        ifact[maxx-1] = invMod(fact[maxx-1]) ;
        for(int i = maxx-1 ; i>0 ;--i) ifact[i-1] = mul(ifact[i], i) ;
    }
    nCr(int _mxN, int _M) {
        maxx = _mxN + 1;

```

```

    md = _M ;
    fact.resize(maxx) ;
    ifact.resize(maxx) ;
    pre() ;
}
ll C(ll n, ll r) {
    if (n < r || r < 0 || n < 0) return 0;
    return mul(fact[n], mul(ifact[r], ifact[n-r])) ;
}
};
//maxx N we need
//const int N = 100;
// initialise nCr struct
// nCr comb(N , mod);

```

6 Random

6.1 Misc

```

// dec to bin
string dectoBin(ll n){
    bitset<64> b(n);
    string s=b.to_string();
    reverse(all(s));
    while(s.size()>0 && s[s.size()-1]=='0')
        s.pop_back();
    reverse(all(s));
    return s;
}

//dec to hex
string dectoHex(ll n){
    string hex="";
    while(n>15){
        ll rem=n%16;
        if(rem<10)
            hex+=to_string(n%16);
        else{
            hex+=((rem==10)?"A":(rem==11)?"B":(rem==12)?"C":(rem==13)?"D":(rem==14)?"E":"F");
        }
        if(n<=15)break;
        n/=16;
    }
}

```

```

}
if(n<10)hex+=to_string(n);
else{
    hex+=((n==10)?"A":(n==11)?"B":(n==12)?"C":(n==13)?"D":(n==14)?"E":"F");
}
reverse(all(hex));
return hex;
}

// lcs in nlogn
int LCS(vector<int>& firstArr,vector<int>& secondArr){
    unordered_map<int, int> mp;
    for (int i = 0; i < firstArr.size(); i++) {
        mp[firstArr[i]] = i + 1;
    }
    vector<int> tempArr;
    for (int i = 0; i < secondArr.size(); i++) {
        // If current element exists in the Map
        if (mp.find(secondArr[i]) != mp.end()) {
            tempArr.push_back(mp[secondArr[i]]);
        }
    }
    vector<int> tail;
    tail.push_back(tempArr[0]);
    for (int i = 1; i < tempArr.size(); i++) {
        if (tempArr[i] > tail.back())
            tail.push_back(tempArr[i]);
        else if (tempArr[i] < tail[0])
            tail[0] = tempArr[i];
        else {
            auto it = lower_bound(tail.begin(),
                                tail.end(),
                                tempArr[i]);
            *it = tempArr[i];
        }
    }
    return (int)tail.size();
}

//Mex
ll MEX(vector<ll>&v){
    ll n = v.size();
    map<ll, ll>m;
    for(int i = 0; i <= n; ++i){

```

```

        m[i]++;
    }
    for(int i = 0; i < n; ++i){
        m.erase(v[i]);
    }
    return m.begin()->first;
}

//maximum subarray sum
ll maximum_subarray_sum(vector<ll> &v){
    ll ans=0;
    ll var=INT_MIN;
    for(i,v.size()){
        var=max(v[i],var+v[i]);
        ans=max(ans,var);
    }
    return ans;
}

```

7 Runflag

```

code -r ~/.bashrc
source ~/.bashrc

run(){
    g++ $1.cpp -std=c++17 -O2 -Wall -o $1.out && ./$1.out < in.txt >
        out.txt && rm $1.out
}

```

8 Stresstest

8.1 brute

```

#include<bits/stdc++.h>
using namespace std;
int main(){
    int a;
    cin>>a;
    cout<<2*a<<endl;
    return 0;
}

```

```

}

```

8.2 code

```

#include<bits/stdc++.h>
using namespace std;
int main(){
    int a;
    cin>>a;
    cout<<2*a<<endl;
    return 0;
}

```

8.3 gen

```

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

mt19937 rng(chrono::steady_clock::now().time_since_epoch().count());

int RANDOM(int a, int b){
    return uniform_int_distribution<int>(a, b)(rng);
}

int main(){
    cout<<RANDOM(1, 1000000000)<<endl;
    return 0;
}

```

8.4 stress

```

set -e
g++ code.cpp -o code
g++ gen.cpp -o gen
g++ brute.cpp -o brute
for((i = 1;i<10000 ; ++i)); do
    ./gen $i > input_file
    ./code < input_file > myAnswer
    ./brute < input_file > correctAnswer
done

```

```

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
## if ! diff myAnswer correctAnswer > /dev/null; then
##     break
## fi
## echo "Passed test: " $i

```

9 Strings

9.1 FindDiff

```

// make sure str1 is not smaller
string findDiff(string str1, string str2) {
    string str = "";
    int n1 = str1.length(), n2 = str2.length();
    reverse(str1.begin(), str1.end());
    reverse(str2.begin(), str2.end());
    int carry = 0;
    for (int i = 0; i < n2; i++) {
        int sub = ((str1[i] - '0') - (str2[i] - '0') - carry);
        if (sub < 0) {
            sub = sub + 10;
            carry = 1;
        }
        else
            carry = 0;
        str.push_back(sub + '0');
    }
    for (int i = n2; i < n1; i++) {
        int sub = ((str1[i] - '0') - carry);
        if (sub < 0) {
            sub = sub + 10;
            carry = 1;
        }
        else

```

```

        carry = 0;
        str.push_back(sub + '0');
    }
    reverse(str.begin(), str.end());
    return str;
}

```

9.2 Hashing

```

//simple
vector<ll> hash1, power1(2e6+10, 1);
vector<ll> invMod1(2e6+10, 1);

ll radix=31;
ll mod=1e9+7;

ll expo(ll b, ll e, ll md){
    ll ans=1;
    while(e){
        if(e&1) ans=(ans*b)%md;
        b=(b*b)%md;
        e/=2;
    }
    return ans;
}

void precal(string &s){
    int n=s.size();
    hash1.resize(n+5);
    ll a=1;
    for(int x=0; x<n; ++x){
        hash1[x+1]=(hash1[x]+(s[x]-'a'+1)*a)%mod;
        a=(a*radix)%mod;
    }
}

ll invmod(ll p){
    return expo(p, mod-2, mod);
}

void preInv(){
    for(int i=1; i<=1e6+1; i++){
        power1[i]=(power1[i-1]*radix)%mod;
        invMod1[i]=invmod(power1[i]);
    }
}

```



```

}

ll calc(ll left, ll right){
    ll val=(hash1[right]-hash1[left-1]+mod)%mod;
    ll ans=(val*invMod1[left-1])%mod;
    return ans;
}

//double-hashing
ll radix=31;
ll mod=1e9+7;

vector<ll> hash1,bhash,power1(5e6+30,1);
void precal(string &s){
    int n=s.size();
    hash1.resize(n+5);
    ll a=1;
    for(int x=0;x<n;++x){
        hash1[x+1]=(hash1[x]+(s[x])*a)%mod;
        a=(a*radix)%mod;
    }
}

void preInv(){
    for(int i=1;i<=5e6+10;i++){
        power1[i]=(power1[i-1]*radix)%mod;
        // invMod1[i]=invmod(power1[i]);
    }
}

void back_precal(string &s){
    int n=s.size();
    bhash.resize(n+3);
    ll a=1;
    for(int x=n;x>=1;--x){
        bhash[x]=(bhash[x+1]+(s[x-1])*a)%mod;
        a=(a*radix)%mod;
    }
}

ll back_calc(ll left, ll right){
    ll val=(bhash[left]-bhash[right+1]+mod)%mod;
    // ll ans=(val*invMod1[bhash.size()-right-1])%mod;
    return val;
}

```

```

// 1 2 3 4 5
// 1 2 3 4 5 6
void solve(){
    string s;
    cin>>s;
    ll n=s.size();
    preInv();
    precal(s);
    back_precal(s);
    ll ans=1;
    vector<ll> mp(n+3);
    mp[1]=1;
    for(ll i=1;i<n;++i){
        ll ind=i+1;
        ll half=ind/2;
        ll left=(hash1[half]*power1[n-ind])%mod;
        if(ind%2)half++;
        ll right=back_calc(half+1,ind);
        //cout<<left<<" "<<right<<endl;
        if(left==right){
            if(ind%2)half--;
            mp[ind]=mp[half]+1;
        }
        ans+=mp[ind];
    }
    // cout<<ans<<endl;
    cout<<ans<<endl;
}

```

9.3 KMP

```

void kmp(string s,string t,set<int> &stt){
    int n=s.size();
    int m=t.size();
    vector<int> lps(m);
    int i=1,j=0;
    while(i<m){
        if(t[i]==t[j]){
            lps[i]=j+1;
            i++;
            j++;
        }
    }
}

```

```

else{
    if(j!=0){
        j=lps[j-1];
    }
    else{
        lps[i]=0;
        i++;
    }
}
}
i=0,j=0;
while(i<n){
    if(s[i]==t[j]){
        i++;
        j++;
    }
    if(j==m){
        stt.insert(i-t.size());
        j=lps[j-1];
    }
    else if(i<n && s[i]!=t[j]){
        if(j!=0){
            j=lps[j-1];
        }
        else{
            i++;
        }
    }
}

```

```

    }
}
}

```

10 template

```

mt19937 rng(chrono::steady_clock::now().time_since_epoch()
.count());
ll uid(ll l, ll r) {return uniform_int_distribution<ll>(l, r)(rng);}
#define fast_io ios_base::sync_with_stdio(false);cin.tie(NULL);
ios::sync_with_stdio(0);
cin.tie(0);
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>

using namespace std;
using namespace __gnu_pbds;
typedef tree<long long, null_type, less<long long>, rb_tree_tag,
tree_order_statistics_node_update> pbds;
// order_of_key(k) : Number of items strictly smaller than k
// find_by_order(k) : K-th element in a set (counting from 0)

```
