BIT Min/Max

```
void build() {
    for(int i=1; i<=n; i++)
        BIT[i+n]=arr[i];
    for(int i = n ; i >= 1 ; --i)
        BIT[i] = min(BIT[i << 1], BIT[(i << 1) | 1]);
}

void update(int k, int x,int n) {
    k += n;
    BIT[k] = x;
    for(k >>= 1 ; k >= 1 ; k >>= 1)
        BIT[k] = min(BIT[k << 1], BIT[(k << 1) | 1]);
}
void update(ll l,ll r,ll val) {
    for(l+=n,r+=n; l < r ; l>>=1,r>>=1) {
        if(l&1) BIT[l] = min(BIT[l],val), l++;
        if(r&1) r--,BIT[r] = min(BIT[r],val);
    }
}

ll query(ll idx) {
    idx += n;
    ll ans = inf;
    while(idx) ans = min(ans,BIT[idx]), idx >>= 1;
    return ans;
}

int query(int a, int b) {
    a += n;
    b += n;
    int res = INT_MAX;
    while(a <= b) {
        if(a & 1)
            res = min(res, BIT[a++]);
        if(!(b & 1))
            res = min(res, BIT[b--]);
        a >>= 1;
        b >>= 1;
    }
    return res;}
```

Physics Formulas

motion

$v = u + at$

$s = ut +(1/2)\ at*t$,

$v*v - u*u = 2*a*s$

Projectile motion

$x = utcos\theta$

$y = utsin\theta - (1/2)\ gt*t$

$y = x\ tan\theta - g*x*x/(\ 2u*u*cos\theta*cos\theta)$

$T = 2u\ sin\theta/g$

$R = u*u*sin2\theta/g$

$H = u*u*sin\theta*sin\theta/2g$

others:

$p=mv$

$v*v/r*g = tan\theta$(**Banking angle)**

$W = F\ S\ cos\ \theta$

$K = (½)mv*v = p*p/\ 2m$

$T = 2*\pi*sqrt(l/g)$

Trigonometry

$sin2\theta=2\ sin\theta\ cos\theta$

$cos2\theta=cos\theta*cos\theta-sin\theta*sin\theta$

$sin3\theta=3sin\theta-4*sin\theta*sin\theta*sin\theta$

$cos3\theta=4*cos\theta*cos\theta*cos\theta-3cos\theta$

For triangle:

$a=bcosC+ccosB$

$b=acosC+ccosA$

$c=bcosA+acosB$

$sin(A+–B)=sinAcosB +– cosAsinB$

$cos(A+–B)=cosAcosB –+ sinAsinB$

Circle Line intersection

```
double r, a, b, c; // given as input
//ax+by+c=0//EPS=1e-9
double x0 = -a*c/(a*a+b*b), y0 = -b*c/(a*a+b*b);
if (c*c > r*r*(a*a+b*b)+EPS)
    puts ("no points");
else if (abs (c*c - r*r*(a*a+b*b)) < EPS){
puts ("1 point");
cout << x0 << ' ' << y0 << '\n';}
else {
double d = r*r - c*c/(a*a+b*b);
double mult = sqrt (d / (a*a+b*b));
double ax, ay, bx, by;
ax = x0 + b * mult;
bx = x0 - b * mult;
ay = y0 - a * mult;
by = y0 + a * mult;
puts ("2 points");
cout << ax << ' ' << ay << '\n' << bx << ' ' << by << '\n';}
```

Point Binary Search

```
//We want to find an element which a<0.0000001
const float target = 0.0000001;
int main(){
    float l=0.00000000,r=100000.00000000;
    cout << l << " " << r;
    while((r-l)>0.0000000001) {
        float mid = (float)((l+r)/2);
        cout << mid << endl;
        if(mid>target) r=mid;
        else l=mid;
    }
    cout << l;
}
```

Binary Index Tree

```
ll query(ll idx) {
    ll ans = 0;
    while(idx) {
        ans += bit[idx];
        idx -= (idx & -idx);
    }
    return ans;
}

void update(ll idx, ll val, ll n) {
    while(idx<=n) {
        bit[idx] += val;
        idx += (idx & -idx);
    }
}

void range_up(int l,int r,int val,int n) {
    update(l,val,n);
    update(r+1,-val,n);
}


cin >> arr[i];
update(i, arr[i], n);

cin >> idx >> val;
update(idx, val-arr[idx], n);
arr[idx] = val;
```

Segment Tree

```
for(ll i=1; i<=n; i++) cin >> arr[i];
        build(1, 1, n);

//reset
for(ll i=0; i<=4*n; i++) seg[i] = 0;
```

```
const ll M = 2e5+10;

ll arr[M];
ll seg[4*M], prop[4*M];

void build(ll node, ll l, ll r) {
    if(l==r) {
        seg[node] = arr[l];
        return;
    }
    ll mid = (l+r)>>1;
    build(node*2, l, mid);
    build(node*2 + 1, mid+1, r);

    seg[node] = seg[node*2] + seg[node*2 + 1];
}

void update(ll node, ll l, ll r, ll idx, ll val) {
    if(l==r) {
        seg[node] = val;
        return;
    }
    ll mid = (l+r)>>1;
    if(idx <= mid) {
        update( node*2, l, mid, idx, val);
    }
    else {
        update(node*2 + 1, mid+1, r, idx, val);
    }

    seg[node] = seg[node*2] + seg[node*2 + 1];
}

ll query(ll node, ll l, ll r, ll i, ll j) {
    if(r<i || l>j) return INT_MAX;
    if(l>=i && r<=j) return seg[node];

    int mid = (l+r)>>1;

    ll q1 = query(node*2, l, mid, i, j);
    ll q2 = query(node*2 + 1, mid+1, r, i, j);
    return q1 + q2;
}
```

LazyProp:-

```
void update(ll node, ll l, ll r, ll i, ll j, ll val) {
    if(prop[node]!=0) {
        seg[node] += prop[node] * (r-l+1);
        if(l!=r) {
            prop[2*node] += prop[node];
            prop[2*node+1] += prop[node];
        }
        prop[node] = 0;
    }

    if(r<i || l>j) return;
    if(l>=i && r<=j) {
        seg[node] += val*(r-l+1);
        if(l!=r) {
            prop[2*node] += val;
            prop[2*node+1] += val;
        }
        return;
    }
    ll mid = (l+r)>>1;
    update(node*2, l, mid, i, j, val);
    update(node*2 + 1, mid+1, r, i, j, val);

    seg[node] = seg[node*2] + seg[node*2 + 1];
}

ll query(ll node, ll l, ll r, ll i, ll j) {
    if(prop[node]!=0) {
        seg[node] += prop[node] * (r-l+1);
        if(l!=r) {
            prop[2*node] += prop[node];
            prop[2*node+1] += prop[node];
        }
        prop[node] = 0;
    }

    if(r<i || l>j) return 0;
    if(l>=i && r<=j) return seg[node];

    ll mid = (l+r)>>1;
    ll q1 = query( node*2, l, mid, i, j);
    ll q2 = query(node*2 + 1, mid+1, r, i, j);

    return q1+q2; }
```

Maximum Subarray Sum in Range Query

```
ll arr[M];
struct Node {
    ll sum, pref, suf, ans;
} seg[4*M];

Node ck(Node l, Node r) {
    Node res;
    res.sum = l.sum + r.sum;
    res.pref = max(l.pref, l.sum + r.pref);
    res.suf = max(r.suf, r.sum + l.suf);
    res.ans = max({l.ans, r.ans, l.suf + r.pref});
    return res;
}

Node make(ll val) {
    Node m;
    m.sum = val;
    m.pref = m.suf = m.ans = max(0LL, val);
    return m;
}

void build(ll node, ll l, ll r) {
    if(l==r) {
        seg[node] = make(arr[l]);
        return;
    }
    ll mid = (l+r)>>1;

    build(node*2, l, mid);
    build(node*2 + 1, mid+1, r);

    seg[node] = ck(seg[node*2], seg[node*2 + 1]);
}

void update(ll node, ll l, ll r, ll idx, ll val) {
    if(l==r) {
        seg[node] = make(val);
        return;
    }
    ll mid = (l+r)>>1;

    if(idx <= mid)  update(node*2, l, mid, idx, val);
    else  update(node*2 + 1, mid+1, r, idx, val);

    seg[node] = ck(seg[node*2], seg[node*2 + 1]);
}

Node query(ll node, ll l, ll r, ll i, ll j) {
    if(j<l || i>r) return make(0);
    if(l>=i && r<=j) return seg[node];

    ll mid = (l+r)>>1;
    Node q1 = query(node*2, l, mid, i, j);
    Node q2 = query(node*2 + 1, mid+1, r, i, j);
    return ck(q1, q2);
}
```

PBDS

```
#include<ext/pb_ds/assoc_container.hpp>
#include<ext/pb_ds/tree_policy.hpp>
using namespace __gnu_pbds;
template <typename T> using ordered_set = tree<T, null_type,
less<T>, rb_tree_tag, tree_order_statistics_node_update>;

void example() {
    ordered_set<int>os;
    os.insert(1);
    os.insert(2);
    cout << *os.find_by_order(2) << endl;
    cout << os.order_of_key(7) << endl;
}
```

Seive

```
void seive() {
    for(ll i=2; i*i<M; i++)
        if(is_prime[i]==0)
            for(ll j=i*i; j<M; j+=i) is_prime[j] = 1;

    for(ll i=2; i<M; i++)
        if(is_prime[i]==0) prime.push_back(i);
}

ll noOfdiv(ll n) {
    ll ans = 1;
    for(ll i=0; i<prime.size() && prime[i]*prime[i]<=n; i++){
        ll cnt = 0;
        while(n%prime[i] == 0){
            cnt++;
            n /= prime[i];
            if(n<=1) break;
        }
        ans *= (cnt+1);
    }
    if(n>1) return ans*2;
    else return ans;
}
```

NOD : $$d(n) = (e_1 + 1) \cdot (e_2 + 1) \cdots (e_k + 1)$$

SOD : $$\sigma(n) = \frac{p_1^{e_1+1} - 1}{p_1 - 1} \cdot \frac{p_2^{e_2+1} - 1}{p_2 - 1} \cdots \frac{p_k^{e_k+1} - 1}{p_k - 1}$$

Phi(n): $$= n \cdot \left(1 - \frac{1}{p_1}\right) \cdot \left(1 - \frac{1}{p_2}\right) \cdots \left(1 - \frac{1}{p_k}\right)$$

Divisor

```
void divisor() {
    for(ll i=1; i<N; i++)
        for(ll j=i; j<N; j+=i)
            divi[j].push_back(i);
}
```

Cum SOD

```
ll csod(ll n){
    ll ans = 0;
    for(ll i=1; i*i<=n; i++){
        ll t1 = ((n/i)-i+1)*i;
        ll t2 = sum(n/i) - sum(i);
        ans += t1+t2;
    }
    ans -= sum(n); ans -= n-1;
    return ans;
}
```

N! Variations

```
int trailingZeroes(int n) {
    int c = 0, f = 5;
    while(f <= n)  {
        c += n/f; f *= 5;  }
    return c;
}

int factDigitCnt(int n) {
    if(n<=1)     return n;
    double digits = 0;
    for(int i=2; i<=n; i++) digits += log10(i);
    return floor(digits)+1;
}

ll factDivisorsCnt(ll n) {
    ll res = 1;
    for(int i=0; primes[i]<=n; i++) {
        ll exp = 0;  ll p = primes[i];
        while(p <= n) {
            exp += (n/p);  p *= primes[i];
        }
        res *= (exp+1);
    }
    return res;
}
```

NCR & NPR

```
ll binExp(ll a, ll b){
    ll ans = 1;
    while(b > 0){
        if(b & 1){
            ans *= a;  ans %= MOD;
        }
        a *= a; a %= MOD; b >>= 1;
    }
    return ans; }

ll modInv(ll a, ll b){
            return binExp(a, b - 2);
}

ll nCr(int x, int y) {
if(x<0 || y<0 || x<y) return 0;
return fact[x]*(inv[y]*inv[x - y] % MOD) % MOD; }

ll nPr(int x, int y) {
if(x<0 || y<0 || x<y)return 0;
return (fact[x] *inv[x - y] ) % MOD; }

ll nCr(int n, int r){
            if(n < r) return 0;
            return ((((fact[n]) * (modInv(fact[r], mod))) % mod) *
(modInv(fact[n - r], mod))) % mod;
}
void pre_cal() {
    fact[0]=1;

    for(int i=1; i<=MAX; i++) {
        fact[i]=fact[i-1]*1LL*i%MOD;
    }

    inv[MAX]=biExp(fact[MAX],MOD-2);
    for(int i=MAX; i>0; i--) {
        inv[i-1]=i* 1LL*inv[i] % MOD; }}
```

SOD of n^m:

```
ll primeFact(ll n,int m) {
    ll sum = 1;
    for(int i=0; i<primes.size() && primes[i]<=n; i++) {
        ll cnt = 0, p = primes[i];
        if(n%p == 0) {
            while(n%p == 0)
                cnt++, n /= p;
            cnt = cnt*m+1;
            ll calc = (biExpo(p,cnt,MOD)+MOD-1)%MOD;
            calc *= biExpo(p-1,MOD-2,MOD);
            calc %= MOD;
            sum = (sum*calc)%MOD;
        }
    }
    if(n > 1) {
        ll calc = (biExpo(n,1+m,MOD)+MOD-1)%MOD;
        calc *= biExpo(n-1,MOD-2,MOD);
        calc %= MOD;
        sum = (sum*calc)%MOD;
    }
    return sum;
}
```

Bits Related

```
int SET(int cur, int pos) { return cur | (1LL << pos); }
bool check(ll n,ll pos) { return n & (1ll<<pos); }
int CLEAR(int n,int pos) { return n & ~(1<<pos); }
```

Check if a subset sum exits:

```
bitset<MAX>bs;
bool check(int sum) { bs.reset(); bs[0]=1;
for(int i=1;i<=n;i++) bs |= bs << arr[i]; return bs[sum]; }
```

Dijkstra

```
#define INF (1<<30)
typedef pair<int, int> pii;
const int mx = 1e6+5;
int parrent[mx];
long long dist[mx];
vector<pair<int,ll>> g[mx];

void dijkstra(int src, int n) {
    for(int i=1; i<=n; i++) dist[i] = INF;
    priority_queue<pii, vector<pii>, greater<pii> > pq;
    pq.push({0, src});
    dist[src] = 0;
    while(!pq.empty()) {
        int cur_n = pq.top().second;
        ll cur_d = pq.top().first;
        pq.pop();
        if(dist[cur_n] < cur_d) continue;
        for(auto child : g[cur_n]) {
            if(cur_d + child.second < dist[child.first]) {
                parrent[child.first] = cur_n;
                dist[child.first] = cur_d + child.second;
                pq.push({dist[child.first], child.first});
            }
        }
    }
}

vector<int> ans = {n};
    int x = n;
    while(parrent[x]!=0) {
        ans.push_back(parrent[x]);
        x = parrent[x];
    }
    reverse(all(ans));
```

Gird minpath Dijkstra

```
#define INF (1<<30)
typedef pair<int, pair<int, int>> pipii;
const int M = 1000;
int g[M][M];
int dist[M][M];
int n, m;
int dx[] = {1, 0, -1, 0};
int dy[] = {0, -1, 0, 1};
bool valid(int i, int j) {
    return(i>=0 && j>=0 && i<n && j<m);
}
void dijkstra(int n, int m) {
    for(int i=0; i<n; i++)
        for(int j=0; j<m; j++)  dist[i][j] = INF;

    priority_queue<pipii, vector<pipii>, greater<pipii>> pq;
    dist[0][0] = g[0][0];
    pq.push({g[0][0], {0,0}});

    while(!pq.empty()) {
        pair<int, int>node =  pq.top().second;
        int x = node.first;
        int y = node.second;
        int cost = pq.top().first;
        pq.pop();
        if(dist[x][y] < cost) continue;
        for(int i=0; i<4; i++) {
            int new_x = x+dx[i];
            int new_y = y+dy[i];
            if(valid(new_x, new_y)) {
                if(g[new_x][new_y]+cost < dist[new_x][new_y]) {
                    dist[new_x][new_y] = g[new_x][new_y]+cost ;
                    pq.push({dist[new_x][new_y], {new_x, new_y}});
                } }
            }
        }
}
```

Bipartet

```
int vis[M], clr[M], f = 1;
void dfs(int src, int c) {
    if(vis[src]) {
        if(clr[src]!=c) f = 0;
        return;
    }
    vis[src] = 1;
    clr[src] = c;
    for(auto child : g[src])
        dfs(child, clr[src]^1);
}
```

Cycle detect directed

```
bool present_vis[N];
bool previous_vis[N];
bool iscycle(ll src) {
    previous_vis[src] = true;
    if(!present_vis[src]) {
        present_vis[src] = true;
        for(auto child : g[src]) {
            if(!present_vis[child] && iscycle(src)) return 1;
            if(previous_vis[src]) return true;
        }
    }
    previous_vis[src] = false;
    return false;
}
bool cycle = false;
    for(ll i=1; i<=n; i++) {
        if(!present_vis[i] && iscycle(i)) {
            cycle = true;
        }
    }
```

Cycle detect Undirected

```
bool iscycle(ll src, ll par) {
    vis[src] = true;
    for(auto child : g[src]) {
        if(child != par) {
            if(vis[child]) return true;
            if(!vis[child] && iscycle(child, src)) return 1;
        }
    }
    return false;
}
 bool cycle = false;
 for(ll i=1; i<=n; i++) {
     if(!vis[i] && iscycle(i, -1)) {
          cycle = true;  }
 }
```

DSU

```
int parent[N];
int sz[N];
void make(int v) {
    parent[v] = v;
    sz[v] = 1;
}
int find(int v) {
    if(v==parent[v]) return v;
    return parent[v] = find(parent[v]);
}
void Union(int a, int b) {
    a = find(a), b = find(b);
    if(a!=b) {
        if(sz[a] > sz[b])  swap(a, b);
            parent[a] = b;
            sz[b] += sz[a];
    }
}

//for(int i=1; i<=n; i++) make(i);
```

MST

```
//1st. make, find , union from DSU, then
pair<ll, vector<pair<ll, ll>>> find_mst(vector<pair<ll, pair<ll,
ll>>> &vp) {
    ll tot = 0;
    vector<pair<ll, ll>> mst;
    sort(all(vp);
    for(auto u:vp) {
        ll a = u.second.first;
        ll b = u.second.second;
        if(find_set(a) != find_set(b)) {
            tot += u.first;
            union_set(a, b);
            mst.push_back({a, b});
        }
    }
    return {tot, mst};
}
```

Floyd Warshal

```
const int N = 250, INF = 1e15;
int dist[N][N];
void make_graph()
    for(ll i=0; i<=N; i++)
        for(ll j=0; j<=N; j++)
            if(i==j) dist[i][j] = 0;
            else dist[i][j] = INF;

void floyd_warshall(ll n)
    for(int k=1; k<=n; k++)
        for(int i=1; i<=n; i++)
            for(int j=1; j<=n; j++)
                if(dist[i][k]!=INF && dist[k][j]!=INF)
                    dist[i][j] = min(dist[i][j], dist[i][k]+dist[k][j]);

make_graph();
```

Diameter

```
int depth[N];

void dfs(int vertex, int par=0){
   for(auto child : g[vertex]){
      if(child == par) continue;
      depth[child] = depth[vertex] + 1;
      dfs(child , vertex);
   }
}
//1st  dfs(1) then get mx_depth_node, then again
dfs(mx_depth_node) and get diameter of tree
```

Height_Depth

```
void dfs(int vertex, int par=0) {
    for(auto child : g[vertex]) {
        if(child == par) continue;
        depth[child] = depth[vertex] + 1;
        dfs(child, vertex);
        height[vertex] = max(height[vertex],
height[child]+1);
    }
}
```

PreCompution

```
void dfs(int vertex, int par=0){
   if(vertex&1) odd_cnt[vertex]++;
   else even_cnt[vertex]++;

   subtree_sum[vertex] += vertex;
   //subtree_sum[vertex] += val[vertex];
   for(auto child : g[vertex]){
      if(child == par) continue;
      dfs(child , vertex);
      subtree_sum[vertex] += subtree_sum[child];
      even_cnt[vertex] += even_cnt[child];
      odd_cnt[vertex] += odd_cnt[child];
   }}
```

LCA

```
const int N = 1e5 + 10;
int tin[N], tout[N];
int up[N][32];
vector<int> adj[N];
int n, lg, timer;
void dfs(int src, int par){
    tin[src] = ++timer;
    up[src][0] = par;
    for(int i = 1;i <= lg;i++){
        up[src][i] = up[up[src][i - 1]][i - 1];
    }
    for(auto child : adj[src]){
        if(child != par)
            dfs(child, src);
    }
    tout[src] = ++timer;
}

bool is_ancestor(int parent, int child){
    return tin[parent] <= tin[child] && tout[parent] >=
tout[child];
}
int lca(int u, int v){
    if(is_ancestor(u, v)) return u;
    if(is_ancestor(v, u)) return v;

    for(int i = lg;i >= 0;i--){
        if(!is_ancestor(up[u][i], v))
            u = up[u][i];
    }
    return up[u][0];
}

void pre_process(int root){
    timer = 0;
    lg = ceil(log2(n));
    dfs(root, root);}
```

```
int main(){
    cin >> n;
    for(int i = 1;i < n;i++){
        int u, v;
        cin >> u >> v;
        adj[u].push_back(v);
        adj[v].push_back(u);
    }
    pre_process(1);
    int q;
    cin >> q;
    while(q--){
        int u, v;
        cin >> u >> v;
        cout << lca(u, v) << endl;
    }
}
```

Binary Lifting

```
const int N = 2e5 + 10;
int up[N][32];
vector<int> adj[N];
int n, lg;
void dfs(int src, int par){
    up[src][0] = par;
    for(int i = 1;i <= lg;i++){
        up[src][i] = up[up[src][i - 1]][i - 1];
    }
    for(auto child : adj[src]){
        if(child != par)
            dfs(child, src);
    }
}
```

```
int kth_ancestor(int u, int k){
    int i = 0;
    while(k){
        if(k & 1) u = up[u][i];
        i++;
        k >>= 1;
    }
    return u;
}
void pre_process(int root){
    lg = ceil(log2(n));
    dfs(root, 0);
}
int main(){
    int q;
    cin >> n >> q;
    for(int i = 2;i <= n;i++){
        int x;
        cin >> x;
        adj[i].push_back(x);
        adj[x].push_back(i);
    }
    pre_process(1);

while(q--){
        int u, k;
        cin >> u >> k;
        int ans = kth_ancestor(u, k);
        cout << (ans > 0 ? ans : -1) << endl;
    }
}
```

LIS

```
int lis[MAX],store_lis[MAX];
int LIS(int n) {
    for(int i=0; i<n; i++)
        lis[i]=INF;
    int pos, cnt = 0 ;
    lis[0] = -INF ;
    lis[n] = INF ;
    for (int i = 0 ; i < n ; i++ ) {
        pos = lower_bound( lis, lis+n+1,
                        arr[i] ) - lis;
        lis[pos] = arr[i] ;
        cnt = max ( cnt, pos );
        store_lis[i] = cnt ;
    }
    return cnt;
}
```

LDS

```
int lds[MAX],store_lds[MAX];
int LDS(int n) {
    for(int i=0; i<n; i++)
        lds[i]=INF;
    int pos, cnt= 0 ;
    lds[0] = -INF ;
    lds[n] = INF;
    for(int i = n-1; i>=0; i--) {
        pos = lower_bound( lds, lds+n+1,
                        arr[i] ) - lds;
        lds[pos] = arr[i] ;
        cnt = max ( cnt, pos );
        store_lds[i] = cnt ;
    }
    return cnt;
```

Kadane

```
ll max_sum = 0, cur_sum = 0, minus_cnt = 0;
for(ll i=0; i<n; i++) {
    if(arr[i] < 0) minus_cnt++;
    cur_sum += arr[i];
    if(max_sum < cur_sum) max_sum = cur_sum;
    if(cur_sum < 0) cur_sum = 0;
}
```

Digit Dp

```
const long long mod = 1e9 + 7;
long long dp[50][900][2][2];

long long func(int idx, int sum, bool tight, int str, string &num){
        if(~dp[idx][sum][tight][str]) return dp[idx][sum][tight][str];
        if(sum < 0) return 0;
        if(idx == num.size()) return sum == 0;

        long long ans = 0;
        int end = (tight ? num[idx] - '0' : 9);
        for(int i = 0;i <= end;i++){
                bool t = (tight && i == num[idx] - '0');
                ans += func(idx + 1, sum - i, t, str, num) %
mod;
        }
        return dp[idx][sum][tight][str] = ans;
}

void solve(int cs){
        memset(dp, -1, sizeof(dp));
        string num1, num2;
        int sum;
        cin >> num1 >> num2 >> sum;
        num1[num1.size() - 1]--;
        long long c1 = func(0, sum, 1, 0, num1);
        long long c2 = func(0, sum, 1, 1, num2);
        cout << ((c2 - c1) + mod) % mod << endl;
}
```

0-1 KnapSack

```
int row = n+1, col = sz+1;
    int mat[row][col];
    memset(mat, 0, sizeof(mat));

    for(int i=1; i<row; i++){
        for(int j=1; j<col; j++){
            if(j-arr[i].weight < 0) mat[i][j] = mat[i-1][j];
            else mat[i][j] = max(mat[i-1][j],
mat[i-1][j-arr[i].weight]+arr[i].price);
        }
    }
row--, col--;
    vector<int> ans;
    while(row>0 && col>0){
        if(mat[row][col]==mat[row-1][col]) row--;
        else{
            ans.push_back(arr[row].name);
            col -= arr[row].weight;
            row--;
        }
    }
    reverse(all(ans));
```

Coin Change Ways

```
int row = n;
int col = amount+1;
int mat[row][col];
for(int i=0; i<row; i++) mat[i][0] = 1;
for(int i=0; i<row; i++) {
    for(int j=1; j<col; j++) {
        if(i==0) {
            if(j%coin[i]) mat[i][j] = 0;
            else mat[i][j] = 1;
        }
        else if(j-coin[i]<0) mat[i][j] = mat[i-1][j];
        else mat[i][j] = mat[i-1][j] + mat[i][j-coin[i]];
    }}
```

Coin Change Minimum

```
int row = n+1;
    int col = amount+1;
    int mat[row][col];
    memset(mat, 0, sizeof(mat));
    for(int i=1; i<col; i++) mat[0][i] = amount+1;
    for(int i=1; i<row; i++) {
        for(int j=1; j<col; j++) {
            if((j-coin[i] < 0)) mat[i][j] = mat[i-1][j];
            else mat[i][j] = min(mat[i-1][j],
mat[i][j-coin[i]]+1);
        }
    }

    vector<int> ans;
    row--, col--;
    while(row>0 && col>0) {
        if(mat[row][col] == mat[row-1][col]) row--;
        else {
            ans.push_back(coin[row]);
            col -= coin[row];
        }
    }
    reverse(ans.begin(), ans.end());
```

LCS

```
int row = s1.size()+1;
    int col = s2.size()+1;

    int arr[row][col];
    char tracker[row][col];
    memset(arr, 0, sizeof(arr));
    for(int i=1; i<row; i++) {
        for(int j=1; j<col; j++) {
            if(s1[i-1] == s2[j-1]) {
                arr[i][j] = arr[i-1][j-1]+1;
                tracker[i][j] = 'D';
            }
            else {
                if(arr[i-1][j] >= arr[i][j-1]) {
                    arr[i][j] = arr[i-1][j];
                    tracker[i][j] = 'U';
                }
                else {
                    arr[i][j] = arr[i][j-1];
                    tracker[i][j] = 'L';
                }
            }
        }
    }

    cout << arr[row-1][col-1] << endl;
    row--, col--;

    string ans = "";
    while(row>0 && col>0){
        if(tracker[row][col]=='D'){
            ans += s1[row-1];
            row--;
            col--;
        }
        else if(tracker[row][col]=='U') row--;
        else col--;
    }
    reverse(ans.begin(), ans.end());
}
```

N -Queen

```
#include <bits/stdc++.h>
using namespace std;
#define N 4
int ld[30] = { 0 };
int rd[30] = { 0 };
int cl[30] = { 0 };

int board[N][N] = { { 0, 0, 0, 0 },
                    { 0, 0, 0, 0 },
                    { 0, 0, 0, 0 },
                    { 0, 0, 0, 0 } };

bool solveNQUtil(int board[N][N], int col){
    if (col >= N)
        return true;
    for (int i = 0; i < N; i++) {
        if ((ld[i - col + N - 1] != 1 && rd[i + col] != 1)
            && cl[i] != 1) {
            board[i][col] = 1;
            ld[i - col + N - 1] = rd[i + col] = cl[i] = 1;
            if (solveNQUtil(board, col + 1))
                return true;
            ld[i - col + N - 1] = rd[i + col] = cl[i] = 0;
        }
    }
    return false;
}

bool solveNQ(){
    if (solveNQUtil(board, 0) == false) {
        cout << "Solution does not exist";
        return false;
    }

    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++)
            cout << " " << (board[i][j]==1?"Q":".") << " ";
        cout << endl;
```

```
  }
   return true;
}

int main(){
   solveNQ();
   return 0;
}
```

Geometry

Triangle:

- To form: a+b>c, b+c>a, c+a>b
- Check if 3 points form triangle: |(x2-x1)(y3-y1)-(y2-y1)(x3-x1)| >0

Perimeter: p = a+b+c
Area: 1) (a*b)/2
2) (abSinC)/2
3) sqrt(s(s-a)(s-b)(s-c)); ///**s=(p/2)**
4) (sqrt(3)/4)*a*a; ///equi triangle
5) (b*sqrt(4*a*a-b*b))/4; ///isosceles
Pythagoras: a*a+b*b = c*c
SineRule: a/SinA=b/SinB=c/SinC
CosineRule: CosA = (b*b+c*c-a*a)/2bc
Center: x=(x1+x2+x3)/3, y=(y1+y2+y3)/3
Median: AD=sqrt((2*b*b+2*c*c-a*a)/4)
Centroid: AG=sqrt(2*b*b-2*c*c-a*a)/3
Inradius: A/s
Circumradius: a/(2*sinA)
r=abc/sq((a+b+c)(a+b-c)(a+c-b)(b+c-a))

Circle:

Distance: sqrt((x2-x1)^2 + (y2-y1)^2)
Check if 3 points are in same line:
x1*(y2-y3)-x2(y1-y3)+x3(y1-y2) = 0
Find a circle that covers 2 given:
x3 = (x1+x2)/2 , y3 = (y1+y2)/2
r = dist(x1,y1,x2,y2)
Lattice Points:1+gcd(|x1-x2|,|y1-y2|)
Slope formed by 2 points:(y2-y1)/(x2-x1)
Area of sector of circle: ½ r^2*
θ
Arc Length: r*θ

Parallelogram:

Given 3 points find 4th point:
Dx = Ax + (Cx-Bx)
Dy = Ay + (Cy-By)
Area: |½((Ax*By+Bx*Cy+Cx*Dy+Dx*Ay)
-(Ay*Bx + By*Cx + Cx*Dx + Dy*Ax))|

Trapezium:

Area:(a+b)/(a-b) * sqrt((s-a)(s-b) (s-b-c)(s-b-d))
-> s = (a+b+c+d)/2
-> a = long parallel side
-> b = short parallel side
-> c,d = non-parallel side
Area:h*((b1+b2)/2)
H:sq(b^2-(b^2-d^2+(a-c)^2)/2(a-c))^2)

Right Circular Cone:

Volume:(pie*h/3)*(R^2+R*r+r^2)
Lateral surface Area:pie(r+R)*S
Area of the base:pie*r^2
Lateral area:pie*r*L
Total Surface A:pie*r^2+pie*r*s
Volume:⅓*pie*r^2*h
s=sq(r^2+h^2)

Truncated Cone:

z=(H*r2^2)(r1^2-r2^2)
R=(½*r1^2(z+h))/(H+z)
Volume:⅓*pie*h*(R^2+(R*r2)+r2^2)
Volume of a cylinder: pi*r*r*h
Volume of a triangular prism: .5*b*h*H

Combinatorics:

Summation of squares of n natural numbers:
(n*(n+1)*(2n+1))/6
P(n,r): n! / (r! * (n-r)!)
C(n,r): (n*(n-1)*..*(n-r+1)) / r!
P(n,k): n! / (n-k)!
-> nCk = nCn-k
-> Ways to go from (0,0) to (r,c):
(r+c)Cr or (r+c)Cc
-> Ways to go from (0,0,0) to (x,y,z): (x+y+z)Cx * (y+z)Cy
-> a1+a2+.+an = k , ai>=0: C(k+n-1,n-1)
-> Catalan Numbers:
C(n) = (2n)! / ((n+1)! * r!)

Others:
Decider for a point located left or right of a line:
d=(x3-x2)*(y2-y1)-(y3-y2)*(x2-x1)

Number of digits: log10(n)+1
Depth of road water: (s^2-h^2)/2h
//sum of series n/1+n/2+n/3+....n/n
ll root=sqrt(n);
for(int i=1; i<=root; i++)
sum+=n/i;
sum=(2*sum)-(root*root);

count the numbers that are divisible by given number in a certain range:a=2,b=3,c=7;
low=(a+b-1)/a;
high=c/a;
total=high-low+1;
Euler Constant:γ≈0.5772156649
#Number of squares in a n*n grid:
S=(n*(n+ 1)*(2*n + 1))/6;
#Number of rectangle in a n*n grid:
R=(n+1)*n/2*(n+1)*n/2 - S;
#Total number of rectangle and square in a n*n grid:
ans=[(n^2 + n)^2]/4
#Number of squares in a n*m grid exp:6*4
S=6*4+5*3+4*2+3*1=50
#Number of rectangles in n*m grid
R=m(m+1)n(n+1)/4
#Number of cubes in a n*n*n grid formula:n^k-(n-2)^k
C=n*(n+1)/2*n*(n+1)/2;
#Number of boxes in a n*n*n grid:
B=(n+1)*n/2*(n+1)*n/2*(n+1)*n/2-C;
#Number of hypercube in a n^4grid:
start a loop from 1 to <=n;
HC=0;
for(i=1;i<=n;i++)
HC+=i*i*i*i;
#Number of hyper box in a n^4 grid:
HB=(n+1)*n/2*(n+1)*n/2*(n+1)*n/2*(n+1)*n/2 - HC;

Template

```
struct Point
{   double x,y;
    Point() {}
    Point(double x, double y):x(x),y(y) {}
    Point(const Point &p):x(p.x),y(p.y) {}
    void input()
    {
       scanf("%lf%lf",&x,&y);
    }
    Point operator + (const Point &p)
    const
    {
       return Point(x+p.x, y+p.y);
    }
    Point operator - (const Point &p)
    const
    {
       return Point(x-p.x, y-p.y);
    }
    Point operator * (double c) const
    {
       return Point(x*c, y*c);
    }
    Point operator / (double c) const
    {
       return Point(x/c, y/c);
    }
};
```

```
// returns distance between two point
double dist(Point A, Point B)
{ return sqrt(dot(A-B,A-B)); }
// Determine if Line AB and CD are parallel or collinear
bool LinesParallel(Point A, Point B, Point C, Point D)
{return fabs(cross(B-A,C-D))<EPS;}
// Determine if Line AB and CD are collinear
bool LinesCollinear(Point A, Point B, Point C, Point D)
{ return LinesParallel(A,B,C,D) &&
fabs(cross(A-B,A-C))<EPS &&
fabs(cross(C-D,C-A))<EPS;}

//checks if AB intersect with CD
bool SegmentIntersect(Point A, Point B, Point C, Point D)
{ if(LinesCollinear(A,B,C,D))
 { if(dist2(A,C)<EPS || dist2(A,D)<EPS ||
dist2(B,C)<EPS || dist2(B,D)<EPS)
  return true;
if(dot(C-A,C-B) > 0 && dot(D-A,D-B) > 0 &&
dot(C-B,D-B) > 0)
  return false;
       return true;
 }if(cross(D-A,B-A) * cross(C-A,B-A) > 0)
       return false;
  if(cross(A-C,D-C) * cross(B-C,D-C) > 0)
       return false;
       return true;
}
```

```
// Compute the coordinates where AB
and CD intersect
Point ComputeLineIntersection(Point A, Point B, Point
C, Point D)
{   double a1,b1,c1,a2,b2,c2;
    a1=A.y-B.y;
    b1=B.x-A.x;
    c1=cross(A,B);
    a2=C.y-D.y;
    b2=D.x-C.x;
    c2=cross(C,D);
    double Dist=a1*b2-a2*b1;
    return Point((b1*c2-b2*c1)/Dist,(c1*a2-c2*a1)/Dist);

// return the minimum distance from
      a point C to a line AB
      double DistancePointSegment(Point A, Point B,
            Point C)
      {  return distBetweenPoint
      (C,ProjectPointSegment(A,B,C)); }
```

Hashing

```
const int Max = 2e5+10;
const int base = 331;
const int Mod = 1e9+7;
ll pw[Max], Hash[Max];
void pre_power() {
    pw[0] = 1;
    for(int i=1; i<Max; i++)
        pw[i] = (pw[i-1]*base) %Mod;
}
void get_hash(string s) {
    ll hash_val = 0;
    for(int i=0; i<s.size(); i++) {
        hash_val = (hash_val*base + (s[i]-'a'+1)) %Mod;
        Hash[i+1] = hash_val;
    }}

ll SubStringHash(int l, int r) {
    return ((Hash[r] - Hash[l-1]*pw[r-l+1]) %Mod + Mod) %Mod;
}
int main() {
    pre_power();
    string s;
    cin >> s;
    get_hash(s);
    cout << SubStringHash(1, s.size()) << endl;
    return 0;
}
```

KMP

```
vector<int> get_lps(string b){
    vector<int> lps(b.length());
    int i = 0;
    lps[0] = 0;
    for(int j = 1;j < b.length(); ){
        if(b[i] == b[j]){
            lps[j] = i + 1;
            i++, j++;
        }
        else{
            if(i != 0) i = lps[i - 1];
            else lps[j] = 0, j++;
        }
    }
    return lps;
}

int kmp(string a, string b){
    int cnt = 0;
    vector<int> lps = get_lps(b);
    int i = 0, j = 0;
    while(i < a.length()){
        if(a[i] == b[j]){
            i++, j++;
        }
        else{
            if(j != 0) j = lps[j - 1];
            else i++;
        }
        if(j == b.length()) cnt++, j = lps[j - 1];
    }
    return cnt;
}

void solve(int cs){
    string a, b;
    cin >> a >> b;
    cout << "Case " << cs << ": " << kmp(a, b) << endl;
}
```

Longest Palindromic Subsequence

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

const int Max = 2e5+10;
const int base = 331;
const int Mod = 1e9+7;

ll pw[Max], Hash[Max], rHash[Max];

void pre_power() {
    pw[0] = 1;
    for(int i=1; i<Max; i++)
        pw[i] = (pw[i-1]*base) %Mod;
}

void get_hash(string s, string r) {
    ll hash_val = 0, rhash_val = 0;
    for(int i=0; i<s.size(); i++) {
        hash_val = ((hash_val*base) + s[i]) %Mod;
        Hash[i+1] = hash_val;

        rhash_val = ((rhash_val*base) + r[i]) %Mod;
        rHash[i+1] = rhash_val;
    }
}

ll SubStringHash(int l, int r) {
    return (Hash[r] - (Hash[l-1]*pw[r-l+1]) %Mod + Mod)
%Mod;
}

ll rSubStringHash(int l, int r) {
    return (rHash[r] - (rHash[l-1]*pw[r-l+1]) %Mod +
Mod) %Mod;
}

int main() {
    pre_power();

    int n;
    string s;
    cin >> n >> s;

    string r = s;
    reverse(r.begin(), r.end());

    get_hash(s, r);

    int lo = 1, hi = s.size(), ans = 0;
    if(!(s.size()&1)) hi--;

    while(lo<=hi) {
        int mid = (lo+hi) >> 1;
        if(!(mid&1)) mid++;

        bool ok = false;
        for(int i=0; i<s.size()-mid+1; i++) {
            ll SubHash = SubStringHash(i+1, i+mid);
            ll rSubHash = rSubStringHash(s.size()-(i+mid)+1,
s.size()-i);
            if(SubHash == rSubHash) {
                ok = true;
                break;
            }
        }
        if(ok) ans = max(ans, mid), lo = mid+2;
        else hi = mid-2;
    }

    lo = 2, hi = s.size();
    if(s.size()&1) hi--;

    while(lo<=hi) {
        int mid = (lo+hi) >> 1;
        if(mid&1) mid++;
 int now = 0;
 bool ok = false;
        for(int i=0; i<s.size()-mid+1; i++) {
            ll SubHash = SubStringHash(i+1, i+mid);
            ll rSubHash =
rSubStringHash(s.size()-(i+mid)+1, s.size()-i);
            if(SubHash == rSubHash) {
                ok = true;
                break;
            }
        }
        if(ok) ans = max(ans, mid), lo = mid+2;
        else hi = mid - 2;
    }

    cout << ans << endl;


    return 0;
}
```

Trie

```
const int M = 5e6+10;

struct node {
    bool is_word;
    int next[2];
    int cnt;
    node() {
        memset(next, -1, sizeof(next));
        is_word = false;
        cnt = 0;
    }
};

node trie[M];
int sz = 0, ans = 0;

void clear() {
    sz = 0;
    for(int i=0; i<M; i++) {
        trie[i].is_word = false;
        trie[i].cnt = 0;
        memset(trie[i].next, -1, sizeof(trie[i].next));
    }
    ans = 0;
}

void Insert(string s) {
    int now = 0;
    for(int i=0; i<s.size(); i++) {
        int d = s[i]-'0';
        if(trie[now].next[d]==-1)
            trie[now].next[d] = ++sz;
        now = trie[now].next[d];
    }
    trie[now].is_word = true;
}

string SearchMa(string s) {
    string ans;
    for(int i=0; i<s.size(); i++) {
        int d = s[i]-'0';
        if(d==0) {
            if(trie[now].next[1]==-1) {
                ans += '0';
                now = trie[now].next[0];
            }
            else {
                ans += '1';
                now = trie[now].next[1];
            }
        }
        else {
            if(trie[now].next[0]==-1) {
                ans += '1';
                now = trie[now].next[1];
            }
            else {
                ans += '0';
                now = trie[now].next[0];
            }
        }
    }
    return ans;
}

string SearchMi(string s) {
    int now = 0;
    string ans;
    for(int i=0; i<s.size(); i++) {
        int d = s[i]-'0';
        if(d==0) {
            if(trie[now].next[0]==-1) {
                ans += '1';
                now = trie[now].next[1];
            }
            else {
                ans += '0';
                now = trie[now].next[0];
            }
        }
        else {
            if(trie[now].next[1]==-1) {
                ans += '0';
                now = trie[now].next[0];
            }
            else {
                ans += '1';
                now = trie[now].next[1];
            }
        }
    }
    return ans;
}

string dec_to_bi(int n) {
    string ans;
    while(n) {
        ans.push_back(n%2 + '0');
        n/=2;
    }
    while(ans.size()<31) ans.push_back('0');
    reverse(ans.begin(), ans.end());
    return ans;
}

int bi_to_dec(string s) {
    int ans = 0;
    reverse(s.begin(), s.end());
    for(int i=0; i<s.size(); i++)
        if(s[i]=='1') ans += (1<<i);
    return ans;
}
```

```
  void Delete(string s) {
    ll now = 0;
    for(ll i=0; i<s.size(); i++) {
        int d = s[i]-'0';
        now = trie[now].next[d];
        trie[now].cnt--;
    }
    now = 0;
    for(int i=0; i<s.size(); i++) {
        int d = s[i]-'0';
        int ck = now;
        now = trie[now].next[d];
        if(trie[now].cnt==0)  trie[ck].next[d] = -1;
    }
}
```

Sliding Window maximum

```
vector<int> maxSlidingWindow(vector<int>& nums, int
k) {
        int n = nums.size();
        deque<int> dq;
        vector<int> ans;
        for(int i = 0;i < n;i++){
            while(dq.size() > 0 && nums[dq.back()] <
nums[i]) dq.pop_back();
            dq.push_back(i);
            if(i >= k - 1){
                ans.push_back(dq.front());
                if(dq.front() < i - k + 2) dq.pop_front();
            }
        }
        for(int i = 0;i < ans.size();i++){
            ans[i] = nums[ans[i]];
        }
        return ans;
    }
```

String Division

```
ll Remainder(string dividend, ll divisor) {
    ll remainder = 0;
    for(ll i=0; i<dividend.size(); i++) {
        if(dividend[i]=='-') continue;
        remainder = remainder*10 + dividend[i]-'0';
        remainder %= divisor;
    }
    return remainder;
}

string Quotient(string dividend, ll divisor) {
    string quotient;
    ll idx = 0;
    ll temp = dividend[idx]-'0';
    while(temp < divisor) temp = temp*10 + dividend[++idx]-'0';
    while(idx < dividend.size()) {
        quotient += (temp / divisor) + '0';
        temp = (temp%divisor)*10 + dividend[++idx]-'0';
    }

    if(quotient.size()==0) return "0";
    else return quotient;
}

void solve(ll cs) {
    string dividend;
    ll divisor;
    cin >> dividend >> divisor;
    cout << "Quotient : " << Quotient(dividend, divisor) << endl;
    cout << "Remainder : " << Remainder(dividend, abs(divisor))
<< endl;
}
```

Large Number Multiplication

Mul1:

```
void solve(ll cs) {
    ll n, m, mod;
    cin >> n >> m >> mod;
    __int128 a = n;
    __int128 b = m;
    __int128 mod1 = mod;
    __int128 c = (a%mod1 * b%mod1) %mod1;
    ll ans = c;
    cout << ans << endl;
}
```

Mul2:

```
ll mul(ll n, ll r, ll m){
    if(r==0) return 0;
    if(r&1) return (mul(n, r-1, m)%m + n%m) %m;
    else return (mul(n, r/2, m)%m + mul(n, r/2, m)%m)
%m;
}

cout << mul(n, r, m) << endl;
```