LCA by Binary Lifting:

```cpp
#define log2(x) (31^__builtin_clz(x))

struct LCA{
    int n, lg;
    vector<int> lev; //depth
    vector<vector<int>> par , adj;

    LCA(vector<vector<int>> adj , int n , int root){
        n++;
        this->adj = adj;
        this->n = n;
        this->lg = log2(n);
        lev.assign(n , 0);
        par.assign(n , vector<int>(lg + 1, -1));
        dfs(root, root, 0);
    }

    void dfs(int u, int p, int d){
        lev[u] = d; par[u][0] = p;
        for (int j = 1; 1 << j < n; j++)
            if (par[u][j - 1] != -1)
                par[u][j] = par[par[u][j - 1]][j - 1];
        for (auto v : adj[u])
            if (v != p)
                dfs(v, u, d + 1);
    }

    int get(int u, int v)
    {
        if (lev[u] < lev[v])
            swap(u, v);
        int diff = lev[u] - lev[v];
        for (int i = lg; i >= 0; i--)
            if (diff & 1 << i)
                u = par[u][i];
        if (u == v)
            return u;
        for (int i = lg; i >= 0; i--)
            if (par[u][i] != -1 && par[u][i] != par[v][i])
                u = par[u][i], v = par[v][i];
        return par[u][0];
    }

    //get jth ancestor of node u;
    int getAncestorJ(int u , int distance){
        for (int i = lg; i >= 0; i--)
            if (distance & 1 << i)
                u = par[u][i];
        return u;
    }
};
```