LCA by sparse table:

```
struct LCA{
    vector<int> first , occur , depth;
    vector<vector<int>> st , adj;

    LCA(vector<vector<int>> adj , int n , int root){
        this->adj = adj;
        depth.assign(n+1 , 0);
        first.assign(n+1 , 0);
        occur.reserve(n*2);
        dfs(root , 0);
        st.assign(log2(occur.size())+5 , vector<int> (occur.size()+100));
        build(occur.size());
    }

    void dfs(int u , int parent , int d = 0){
        depth[u] = d;
        first[u] = occur.size();
        occur.push_back(u);
        for(auto v : adj[u])
            if(v!=parent){
                dfs(v , u , d+1);
                occur.push_back(u);
            }
    }

    void build(int n){
        for(int i = 0 ; i < n ; i++) st[0][i] = occur[i];
        for(int j = 1 ; 1<<j <= n ; j++)
            for(int i = 0 ; i+(1<<j) <= n ; i++){
                int c1 = st[j-1][i] , c2 = st[j-1][i+(1<<j-1)];
                if(depth[c1] <= depth[c2]) st[j][i] = c1;
                else st[j][i] = c2;
            }
    }

    int inline get (int l , int r){
        l = first[l]; r = first[r];
        if(l > r) swap(l , r);
        int k = log2(r-l+1);
        int c1 = st[k][l] , c2 = st[k][r-(1<<k)+1];
        return (depth[c1] <= depth[c2]) ? c1 : c2;
    }
};
```