

```

103 LCA by Binary Lifting:
104 #define log2(x) (31^__builtin_clz(x))
105
106 struct LCA{
107     int n, lg;
108     vector<int> lev; //depth
109     vector<vector<int>> par , adj;
110
111     LCA(vector<vector<int>> adj , int n , int root){
112         n++;
113         this→adj = adj;
114         this→n = n;
115         this→lg = log2(n);
116         lev.assign(n , 0);
117         par.assign(n , vector<int>(lg + 1, -1));
118         dfs(root, root, 0);
119     }
120
121     void dfs(int u, int p, int d){
122         lev[u] = d; par[u][0] = p;
123         for (int j = 1; 1 << j < n; j++)
124             if (par[u][j - 1] ≠ -1)
125                 par[u][j] = par[par[u][j - 1]][j - 1];
126         for (auto v : adj[u])
127             if (v ≠ p)
128                 dfs(v, u, d + 1);
129     }
130
131     int get(int u, int v)
132     {
133         if (lev[u] < lev[v])
134             swap(u, v);
135         int diff = lev[u] - lev[v];
136         for (int i = lg; i ≥ 0; i--)
137             if (diff & 1 << i)
138                 u = par[u][i];
139         if (u = v)
140             return u;
141         for (int i = lg; i ≥ 0; i--)
142             if (par[u][i] ≠ -1 && par[u][i] ≠ par[v][i])
143                 u = par[u][i], v = par[v][i];
144         return par[u][0];
145     }
146
147     //get jth ancestor of node u;
148     int getAncestorJ(int u , int distance){
149         for (int i = lg; i ≥ 0; i--)
150             if (distance & 1 << i)
151                 u = par[u][i];
152         return u;
153     }
154 };

```