Pastebin: http://paste.ubuntu.com/24556603/

/\*

Call lca\_init(root) to do precal stuffs

Call lca\_query(u, v) to get the lca

\*/

#define NODES 30105

#define LOG\_NODES 20

using namespace std;

int n; //Number of nodes

int par[NODES]; //Parent of each node

int depth[NODES]; //Calculates the depth of each node

int p[NODES][LOG\_NODES]; //Processing array for LCA

vector <int> adj[NODES]; //Adjacency List

void lca\_dfs(int u){ //Determines the parent of each node

for(int i=0; i<adj[u].size(); i++){

int v=adj[u][i];

if(v!=par[u]){

depth[v]=1+depth[u];

par[v]=u;

lca\_dfs(v);

}

}

}

void lca\_precal(){ //Calculates array p

for(int i=0; i<n; i++){

p[i][0]=par[i];

}

for(int j=1; (1<<j)<n; j++){

for(int i=0; i<n; i++){

if(p[i][j-1]!=-1){

p[i][j]=p[p[i][j-1]][j-1];

}

}

}

}

void lca\_init(int root){

memset(par, -1, sizeof(par));

memset(p, -1, sizeof(p));

depth[root]=0;

lca\_dfs(root);

lca\_precal();

}

int lca\_query(int u, int v){ //Finds the lca of u and v

int lg;

if(depth[u]<depth[v]){

swap(u, v);

}

for(lg=1; (1<<lg)<=depth[u]; lg++);

lg--;

for(int i=lg; i>=0; i--){

if(depth[u]-(1<<i)>=depth[v]){

u=p[u][i];

}

}

if(u==v){

return u;

}

for(int i=lg; i>=0; i--){

if(p[u][i]!=-1 && p[u][i]!=p[v][i]){

u=p[u][i];

v=p[v][i];

}

}

return par[u];

}