Second way using kruskal :

void kruskal() {

int ans=0, u, v, w, a, b, diff = INT\_MAX;

sort( EdgeList.begin() , EdgeList.end() );

DjS ds;

Forp( i,1,V+1 ) ds.add\_comp( i );

for ( int i=0 ; i<EdgeList.size() ; i++ ) {

u = ds.find\_parent( EdgeList[i].start );

v = ds.find\_parent( EdgeList[i].end );

if ( u != v ) {

w = EdgeList[i].weight;

Forp( j, i+1, EdgeList.size() ) {

a = ds.find\_parent( EdgeList[j].start );

b = ds.find\_parent( EdgeList[j].end );

if ( ( a == u && b == v ) || ( a == v && b == u ) )

diff = min ( diff, EdgeList[ j ].weight - w ); }

ans+=w;

ds.merge( u,v ); } }

if ( ds.numOfComp != 1 ) cout<<"No way\n";

else if ( diff == INT\_MAX ) cout<<"No second way\n";

else cout<< ans + diff << endl;

//return ans; }

Floyd :

Forp( i, 0, n ) {

Forp( j, 0, n ) {

arr[ i ][ j ][ 0 ] = arr[ i ][ j ][ 1 ] = oo;

if( i==j ) arr[ i ][ j ][ 0 ] = 0; } }

Forp( i, 0, r ) {

scanf( "%d %d %d", &x, &y, &q );

arr[ x ][ y ][ 0 ] = arr[ y ][ x ][ 0 ] = q; }

Forp( i, 0, 2 ) {

Forp( j, 0, n ) {

Forp( k, 0, n ) {

Forp( l, 0, n ) {

if ( arr[ k ][ l ][ 0 ] > arr[ k ][ j ][ 0 ] + arr[ j ][ l ][ 0 ] )

{ arr[ k ][ l ][ 1 ] = arr[ k ][ l ][ 0 ];

arr[ k ][ l ][ 0 ] = arr[ k ][ j ][ 0 ] + arr[ j ][ l ][ 0 ]; }

else if ( arr[ k ][ l ][ 0 ] < arr[ k ][ j ][ 0 ] + arr[ j ][ l ][ 0 ] )

arr[ k ][ l ][ 1 ] = min ( arr[ k ][ l ][ 1 ], arr[ k ][ j ][ 0 ] + arr[ j ][ l ][ 0 ] );

if ( arr[ k ][ l ][ 0 ] < arr[ k ][ j ][ 1 ] + arr[ j ][ l ][ 0 ] ) arr[ k ][ l ][ 1 ] = min ( arr[ k ][ l ][ 1 ], arr[ k ][ j ][ 1 ] + arr[ j ][ l ][ 0 ] );

if ( arr[ k ][ l ][ 0 ] < arr[ k ][ j ][ 0 ] + arr[ j ][ l ][ 1 ] ) arr[ k ][ l ][ 1 ] = min ( arr[ k ][ l ][ 1 ], arr[ k ][ j ][ 0 ] + arr[ j ][ l ][ 1 ] ); } } } }

scanf( "%d", &q ); printf( "Set #%d\n", t++ );

while( q-- ) { scanf( "%d %d", &x, &y );

if ( arr[ x ][ y ][ 1 ] < oo ) printf( "%d\n", arr[ x ][ y ][ 1 ] );

else puts("?"); }