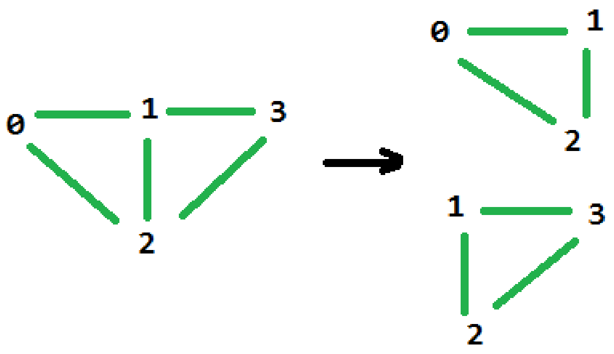
Number of Triangles in an Undirected Graph

Given an Undirected simple graph, We need to find how many triangles it can have. For example below graph have 2 triangles in it.



Graph with 2 triangles

Let A[][] be adjacency matrix representation of graph. If we calculate A^3, then the number of triangle in Undirected Graph is equal to trace(A^3) / 6. Where trace(A) is the sum of the elements on the main diagonal of matrix A.

```
Trace of a graph represented as adjacency matrix A[V][V] is,
trace(A[V][V]) = A[0][0] + A[1][1] + .... + A[V-1][V-1]

Count of triangles = trace(A^3) / 6
```

Below is C++ implementation of above formula.

```
// A C++ program for finding number of triangles in an
// Undirected Graph. The program is for adjacency matrix
// representation of the graph
#include <bits/stdc++.h>
using namespace std;

// Number of vertices in the graph
#define V 4

//  Utility function for matrix multiplication
void multiply(int A[][V], int B[][V], int C[][V])
{
   for (int i = 0; i < V; i++)
   {
      for (int j = 0; j < V; j++)
      {
         C[i][j] = 0;
         for (int k = 0; k < V; k++)
            C[i][j] += A[i][k]*B[k][j];
      }
   }
}

//  Utility function to calculate trace of a matrix (sum of
//  diagnonal elements)
int getTrace(int graph[][V])
{
   int trace = 0;
   for (int i = 0; i < V; i++)
      trace += graph[i][i];
   return trace;
}

//  Utility function for calculating number of triangles in graph
int triangleInGraph(int graph[][V])
{
   int aux2[V][V];  // To Store graph^2
   int aux3[V][V]; // To Store graph^3

   //  Initialising aux matrices with 0
   for (int i = 0; i < V; ++i)
      for (int j = 0; j < V; ++j)
         aux2[i][j] = aux3[i][j] = 0;

   // aux2 is graph^2 now  printMatrix(aux2);
   multiply(graph, graph, aux2);

   // after this multiplication aux3 is
   // graph^3 printMatrix(aux3);
   multiply(graph, aux2, aux3);

   int trace = getTrace(aux3);
   return trace / 6;
}

// driver program to test above function
int main()
{
  /* Let us create the example graph discussed above */
   int graph[V][V] = {{0, 1, 1, 0},
                      {1, 0, 1, 1},
                      {1, 1, 0, 1},
                      {0, 1, 1, 0}
                     };

   printf("Total number of Triangle in Graph : %d\n",
         triangleInGraph(graph));
   return 0;
}
```

Output:

```
Total number of Triangle in Graph : 2
```

How does this work?

If we compute A^n for an adjacency matrix representation of graph, then a value $A^n[i][j]$ represents number of distinct walks between vertex i to j in graph. In A^3, we get all distinct paths of length 3 between every pair of vertices.

A triangle is a cyclic path of length three, i.e. begins and ends at same vertex. So $A^3[i][i]$ represents a triangle beginning and ending with vertex i. Since a triangle has three vertices and it is counted for every vertex, we need to divide result by 3. Furthermore, since the graph is undirected, every triangle twice as i-p-q-j and i-q-p-j, so we divide by 2 also. Therefore, number of triangles is trace(A^3) / 6.

Time Complexity:

The time complexity of above algorithm is $O(V^3)$ where V is number of vertices in the graph, we can improve the performance to $O(V^{2.8074})$ using Strassen's matrix multiplication algorithm.

References:

http://www.d.umn.edu/math/Technical%20Reports/Technical%20Reports%202007-/TR%202012/yang.pdf
Number of Triangles in Directed and Undirected Graphs