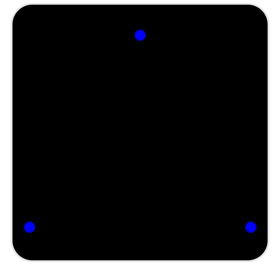
ChatGPT

Triangles in a Graph



In graph theory, a **triangle** (or 3-clique) is a set of three vertices all pairwise connected by edges, forming an undirected cycle of length 3. Equivalently, it is isomorphic to the complete graph K_3 (also denoted C_3) en.wikipedia.org . Detecting and counting triangles is fundamental: for example, in a social network a triangle represents three mutual friends, and in general triangles measure how tightly local neighborhoods are connected. In mathematical terms, an undirected graph contains a triangle if there exist vertices *u*, *v*, *w* such that edges (*u*,*v*), (*v*,*w*), and (*w*,*u*) all exist.

Counting Triangles: The total number of triangles in a simple graph can be computed from its adjacency matrix **A**. Each closed walk of length 3 contributes to the product $A_{(ij)}$ * $A_{(jk)}$ * $A_{(ki)}$. In fact, the sum $\sum_{i,j,k} A_{ij}A_{jk}A_{ki}$ counts each triangle 6 times (once for each cyclic ordering of its vertices). Hence the number of triangles is $\mathrm{trace}(A^3)/6$ en.wikipedia.org . This also appears in the global clustering coefficient formula: the numerator of the clustering coefficient is $3 \times (\text{number of triangles})$ en.wikipedia.org . Locally, the clustering coefficient of a vertex *i* can be defined as

$$C_i = \frac{2 \times (\text{number of edges among neighbors of } i)}{d_i(d_i - 1)} = \frac{2 \times T(i)}{d_i(d_i - 1)},$$

where d_i is the degree of i and $T(i)$ the triangles involving i. In other words, C_i is twice the count of triangles through i divided by $d_i(d_i - 1)$ en.wikipedia.org .

Algorithms for Finding Triangles: A naive algorithm checks every triple of vertices, taking $O(n^3)$ time (n = number of vertices). More efficient methods use graph structure:

- **Edge Intersection Method:** For each vertex *u*, iterate over each neighbor *v*. Then intersect the neighbor lists of *u* and *v* to find common neighbors *w*. Each common neighbor forms a triangle (*u*,*v*,*w*). If adjacency lists are sorted (or stored as hash sets), this finds each triangle once in time roughly proportional to the sum of degrees of endpoints.
- **Ordered-Vertex Method:** Sort vertices by degree (or label) and only search for triangles (*u*,*v*,*w*) with $u < v < w$. For each edge (*u*,*v*) with $u < v$, scan neighbors *w* of *v* with $w > v$ and check if *w* is a neighbor of *u*. This avoids duplicate counting.
- **Matrix Multiplication:** Use sparse matrix techniques to compute A^3 and extract triangles. This is practical for moderate-sized graphs using optimized libraries (cost roughly $O(n^{2.373})$ or less for dense multiplication, or faster with sparsity tricks).

Below is simple pseudocode illustrating the ordered-edge approach (counting each triangle once):

sql Copy

```sql
count = 0
for each vertex u in V:
    for each neighbor v of u where v > u:
        for each neighbor w of v where w > v:
            if edge (w,u) exists:
                count += 1
return count
```

Here we enforce an ordering $u < v < w$ to count each triangle exactly once. The time complexity is roughly $O(\sum_{(u,v)\in E} \min(d(u), d(v)))$, often much better than n^3.

```python
# Example Python: count triangles in an undirected adjacency list G
def count_triangles(G):
    count = 0
    for u in G:
        for v in G[u]:
            if v <= u:
                continue
            for w in G[v]:
                if w <= v:
                    continue
                if u in G[w]:  # check if edge (u,w) closes triangle
                    count += 1
    return count
```

This Python code assumes `G` is a dictionary of sets or lists of neighbors. It counts each triangle once by requiring `u<v<w` (assuming vertex labels can be ordered). Many graph libraries (e.g. NetworkX) provide built-in triangle or core-counting functions (`nx.triangles(G)` or `nx.clustering(G)`).

Theoretical Insights: Graphs with no triangles (triangle-free) are well-studied: by Mantel's theorem, a triangle-free graph on *n* vertices has at most $\lfloor n^2/4 \rfloor$ edges (achieved by a complete bipartite graph). On the other hand, dense regions of a graph often yield many triangles; the *global clustering coefficient* (transitivity) measures the ratio of triangles to connected triples as $C = 3 \times (\text{number of triangles})/(\text{number of triplets})$ en.wikipedia.org .

Applications: Triangles are key in network analysis. In social networks, the *triadic closure* principle says two individuals with a common friend are likely to become friends, so social graphs tend to have many triangles. The clustering coefficient of a network is based on triangle counts en.wikipedia.org . In computational biology, triangles appear as simple network motifs indicating feedback loops. Triangles are also used in graph mining (finding frequent subgraphs) and in graph embedding/clustering: communities often have high triangle density. Fast triangle-counting algorithms are used in performance-sensitive areas like large-scale social network analysis and subgraph mining.

In graph theory, a clustering coefficient is a measure of the degree to which nodes in a graph tend to cluster together. Evidence suggests that in most real-world networks, and in particular social networks, nodes tend to create tightly knit groups characterised by a relatively high density of ties; this likelihood tends to be greater than the average probability of a tie randomly established between two nodes (Holland and Leinhardt, 1971;[1] Watts and Strogatz, 1998[2]).

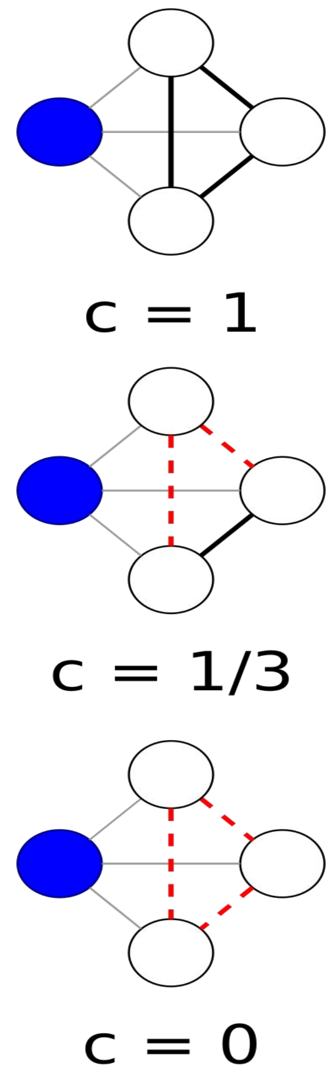
Two versions of this measure exist: the global and the local. The global version was designed to give an overall indication of the clustering in the network, whereas the local gives an indication of the extent of "clustering" of a single node.

Local Clustering Coefficient: For a given node, it measures the proportion of possible connections between its neighbors that are actually realized (i.e., present in the graph). If a node has k neighbors, the maximum number of edges that could exist between these neighbors is .
The local clustering coefficient C for a node is calculated as: C=

$C = 2E/(k(k-1))$

where E is the number of edges that actually exist between the neighbors of the node.

Global Clustering Coefficient: This is a broader measure that considers the overall tendency of nodes in the graph to cluster together. It can be calculated in various ways, one of which involves the ratio of the number of closed triplets (triangles) to the number of open triplets.



Example local clustering coefficient on an undirected graph. The local clustering coefficient of the blue node is computed as the proportion of connections among its neighbours which are actually realised compared with the number of all possible connections. In the figure, the blue node has three neighbours, which can have a maximum of 3 connections among them. In the top part of the figure all three possible connections are realised (thick black segments), giving a local clustering coefficient of 1. In the middle part of the figure only one connection is realised (thick black line) and 2 connections are missing (dotted red lines), giving a local cluster coefficient of 1/3. Finally, none of the possible connections among the neighbours of the blue node are realised, producing a local clustering coefficient value of 0.