MATH 61DM
FALL 2018
COUNTING CLIQUES IN GRAPHS [M10]

Suppose we are given a particular graph G on n vertices, and want to know whether it contains a k-clique. More generally, we might want to *count* how many k-cliques G has.

Question 1. *Given an arbitrary graph G on n vertices and a (fixed) positive integer k, compute (as efficiently as possible) the number of k-cliques in G, and in particular determine whether it is zero.*

Here we care about k some small fixed integer and n large. In the first instance we consider $k = 3$, i.e. we want to count triangles.

One way is to simply enumerate all triples of vertices $\{x, y, z\}$ and check whether each one is a triangle. This uses $\binom{n}{3}$, or crudely about $O(n^3)$ operations. We can do slightly better by considering every edge xy and counting counting how many common neighbors they have, i.e. the number of vertices z with xz, $yz \in E$, then summing up. However, if G has at least (say) $n^2/10$ edges (which it might) this doesn't change the asymptotic.

It turns out, surprisingly, that it is possible to do significantly better than this.

Theorem 2. *We can count the triangles in a graph G on n vertices in time $O(n^{2.373})$.*

Proof. Identify the vertices of G with $\{1, \dots, n\}$. Given our graph G, we'll write down its *adjacency matrix* A. This is the $n \times n$ matrix such that

$$A_{ij} = \begin{cases} 1 & : ij \in E \\ 0 & : ij \notin E \end{cases} .$$

Note A is symmetric (i.e., $A_{ij} = A_{ji}$) and has zeroes on the diagonal.

Now we're going to consider the matrix $B = A^2 = AA$. Its entries are

$$B_{ik} = \sum_{j=1}^{n} A_{ij} A_{jk} = \#\{j \in \{1, \dots, n\} \colon ij \in E \text{ and } jk \in E\},$$

i.e. B_{ik} counts the number of common neighbours of i and j.

So, to find a triangle $\{i, j, k\}$, it suffices to look for i and k such that (i) $ik \in E$ and (ii) $B_{ik} > 0$, i.e. i and k have a common neighbour. In fact, the number of triangles is given by

$$\begin{aligned} 6\#\text{triangles} &= \sum_{i,j,k=1}^{n} A_{ij} A_{jk} A_{ik} \\ &= \sum_{i,k=1}^{n} A_{ik} \sum_{j=1}^{k} A_{ij} A_{jk} \\ &= \sum_{i,k=1}^{n} A_{ik} B_{ik} . \end{aligned}$$

Note the 6 is because each triangle $\{i, j, k\}$ will appear 6 times in the sum, once for each ordering of i, j, k.

So, if we can calculate the $n \times n$ matrix B somehow in $O(n^{2.373})$ time, we can count triangles in an extra $O(n^2)$ operations, which is much smaller. Hence we're done if we know the following.

Theorem 3. *We can multiply two $n \times n$ matrices in time $O(n^{2.373})$.*

This is the state of the art as of 2018 (due to Le Gall, 2014). It's a slight improvement on the Coppersmith–Winograd algorithm (1990), which achieved $O(n^{2.376})$. It's common to write ω for "whatever the exponent in matrix multiplication is"; then we can also count triangles in time $O(n^{\omega})$. It's widely believed that any $\omega > 2$ is achievable, but this is a big open problem.

These modern results are beyond our scope. However, we can give a sketch of the first result along these lines, due to Strassen: he showed you can multiply matrices in $O(n^{\log_2 7})$ time (so, about $O(n^{2.8074})$).

Sketch proof of 2.8074. Instead of counting "operations" we'll count how many multiplications we need to do to multiply $n \times n$ matrices. It turns out counting additions etc. as well does not change the overall answer.

The first step is to show that you can multiply two 2×2 matrices using only 7 multiplications. This is suprising because naive matrix multiplication uses 8. This step is totally unenlightening: to compute

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}$$

we compute the 7 products

$$\begin{aligned} m_1 &= (a_{11} + a_{22})(b_{11} + b_{22}) \\ m_2 &= (a_{21} + a_{22})b_{11} \\ m_3 &= a_{11}(b_{12} - b_{22}) \\ m_4 &= a_{22}(b_{21} - b_{11}) \\ m_5 &= (a_{11} + a_{12})b_{22} \\ m_6 &= (a_{21} - a_{11})(b_{11} + b_{12}) \\ m_7 &= (a_{12} - a_{22})(b_{21} + b_{22}) \end{aligned}$$

and observe that every entry of the product matrix can be formed by a sum of $m_1, \dots, m_7$:

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} = \begin{pmatrix} m_1 + m_4 - m_5 + m_7 & m_3 + m_5 \\ m_2 + m_4 & m_1 - m_2 + m_3 + m_6 \end{pmatrix}$$

i.e. using no further multiplications. [If these equations differ from Wikipedia, believe Wikipedia.]

In the second step, suppose we have 4×4 matrices A and B. We can think of them as 2×2 matrices whose entries are 2×2 matrices:

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} A_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}.$$

We can evaluate this product using 7 2×2 matrix multiplications in the same way as step 1 (note we didn't use in an important way that a_{ij}, b_{ij} or that multiplication commutes). Each of these matrix multiplications requires 7 multiplications; so we need $7^2 = 49$ multiplications overall.

In the third step, suppose more generally have $2^k \times 2^k$ matrices. Again by writing this as a 2×2 matrix of $2^{k-1} \times 2^{k-1}$ matrices, we can do this recursively using 7^k multiplications. Since $7^k = (2^k)^{\log_2 7}$, rounding n up to the nearest power of 2 (filling with zeros) we can multiply an $n \times n$ matrix in $O(n^{\log_2 7})$ operations. □

This finishes the triangle-counting proof. □

That's good for counting 3-cliques. What about k-cliques, for general (fixed) k? If k is a multiple of 3, we can also prove:

Theorem 4. *We can count $3k$-cliques in G in time $O_\varepsilon(n^{\omega k})$.*

Proof. The trick is to use the triangle case. Given G, we'll build a new graph G' with $O(n^k)$ vertices, such that triangles in G' correspond exactly to $3k$-cliques in G. Running the triangle-counting algorithm on G' then proves the result.

We choose $G' = (V', E')$ as follows: the vertices V' are exactly the k-cliques in G, and two k-cliques S, T form an edge whenever (i) $S \cup T$ is a $2k$-clique in G (so in particular, S, T are disjoint), and (ii) to avoid over-counting, we insist that every vertex $i \in S$ has a smaller label than every vertex $j \in T$ (or the same exchanging S and T).

Then every triangle $\{S_1, S_2, S_3\}$ in G' corresponds to a $3k$-clique $S_1 \cup S_2 \cup S_3$ in G, and conversely for every $3k$-clique A in G there is a unique way to split it into a triangle $\{S_1, S_2, S_3\}$ in G' with $A = S_1 \cup S_2 \cup S_3$.

Note we can just list k-cliques and $2k$-cliques in G in time $O(n^{2k})$, which is smaller than $O(n^{\omega k})$. $\square$

Remark 5. We can do something similar for k-cliques where k is not a multiple of 3, by splitting $k = k_1 + k_2 + k_3$ as evenly as possible.

Remark 6. These algorithms are the best known way of counting k-cliques in a graph G for fixed k.