COUNTING PERFECT MATCHINGS

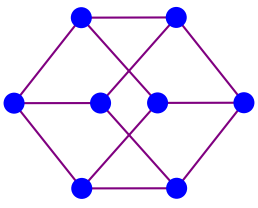
JOHN WILTSHIRE-GORDON

Abstract. Let G be a graph on n vertices. A perfect matching of the vertices of G is a collection of $n/2$ edges whose union is the entire graph. This definition only applies to graphs with an even number of vertices, however. We present a generalization of the notion of a perfect matching to include graphs with an odd number of vertices. Then we convince a skeptical reader of its relevance by producing a recurrence that counts perfect matchings, relying heavily on the new definition.

Contents

1. What is a Perfect Matching?

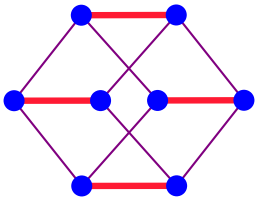
Here is a graph:



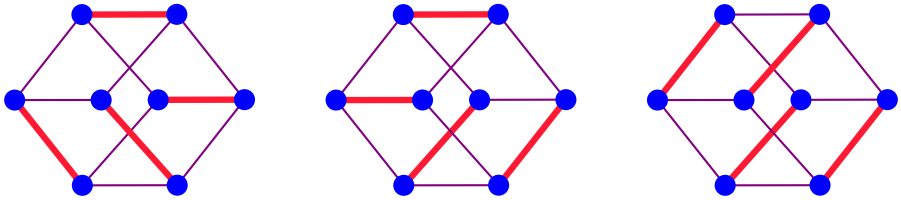
It is composed of a set of blue vertices connected by purple edges. An edge usually connects a single vertex to another, but we do allow an edge to loop back to its original vertex.

Here is a perfect matching:

Date: September 1, 2009.



We color four of the edges red so that every vertex is given a partner. We use four edges because there are eight vertices total, and each edge hits two. This graph has other perfect matchings, however:



One might ask how many perfect matchings a certain graph has. Unfortunately, this problem is quite computationally intensive in general (it belongs to #P, one of the prickliest complexity classes). Still, trial and error will certainly suffice for the example above: there are 9.

How about this graph?

This must be a trick question: there can be no way to pair up an odd number of vertices! More precisely, our current definition of "perfect matching" only applies when the number of vertices is even. Nevertheless, we will show:

$$\text{number of perfect matchings} \;=\; \frac{1}{3}\left(8-4\sqrt{2}\right)\sqrt{\frac{2}{\pi}} \;\approx\; 0.623187$$

The method we use is robust enough to handle odd graphs as well as run-of-the-mill even graphs.

2. The Key Observation

Let's define a function that uses the adjacency matrix of a graph to count its perfect matchings.

$$\mathcal{M}(G) := \#\{\text{perfect matchings of the graph specified by } G\}$$

Here's what we get when we put in a graph with $n = 4$ vertices:

$$\mathcal{M}\begin{pmatrix} 0 & a & b & d \\ a & 0 & c & e \\ b & c & 0 & f \\ d & e & f & 0 \end{pmatrix} = af + be + cd$$

Each term in this sum represents a potential perfect matching. If both a and f equal 1, for instance, then the edges $\{1, 2\}$ and $\{3, 4\}$ form a perfect matching. To check that this pair of edges indeed covers the graph, we need only verify that

every row and every column contains an a or an f, i.e. that the entries containing an a or an f form a permutation matrix. If fact, this permutation matrix must be symmetric in order to hit the right number of edges. Define

$$I_n := \{\sigma \in S_n \mid \sigma = \sigma^{-1} \wedge (\forall i)\sigma(i) \neq i\}$$

to be the set involutive derangements in the symmetric group on n elements. Since permutation matrices are orthogonal, these are exactly the traceless self-adjoint elements in the standard representation of the symmetric group. Now we may rewrite $\mathcal{M}(G)$ in the case where n is even:

$$\mathcal{M}(G) = \mathcal{M}((g_{i,j})) = \sum_{\sigma \in I_n} \prod_{i=1}^{n} \sqrt{g_{i,\sigma(i)}}$$

The square root compensates for the way σ hits each value twice, once on each side of the main diagonal.

Notice that this way of writing $\mathcal{M}$ is no help for odd graphs since there are no involutive derangements in S_n when n is odd. Here's the observation that will generalize:

$$\begin{aligned}\mathcal{M}(G) &= af + be + cd \\ &= \Big[(a+b+c+d+e+f)^2 - (a+b+c)^2 - (a+d+e)^2 - (b+d+f)^2 \\ &\qquad -(c+e+f)^2 + a^2 + b^2 + c^2 + d^2 + e^2 + f^2\Big]/2!\end{aligned}$$

Look at symmetric submatrices of G to see the pattern. Each term in the sum is the square of the sum of the values in a symmetric submatrix. The signs come from the parity of the number of rows/columns deleted.

Theorem 2.1. *Let G be the adjacency matrix of a graph on $\mathcal{V} = \{1, 2, 3, ..., n\}$. If $\mathcal{V}'$ is some subset of $\mathcal{V}$, define $G^{\mathcal{V}'}$ to be the adjacency matrix of the graph obtained by deleting the vertices in $\mathcal{V}'$. We have the following formula for the number of perfect matchings of G:*

$$\mathcal{M}(G) = \frac{1}{2^{n/2}(n/2)!} \sum_{\mathcal{V}' \subseteq \mathcal{V}} (-1)^{\#\mathcal{V}'} \left(\sum_{i,j} \left(G^{\mathcal{V}'}\right)_{i,j} \right)^{n/2}$$

Proof. We must show that the formula holds when n is even, since that's the only time $\mathcal{M}(G)$ is defined. First, some trivial manipulations:

$$(n/2)!\mathcal{M}(G) = \sum_{\mathcal{V}' \subseteq \mathcal{V}} (-1)^{\#\mathcal{V}'} \left(\sum_{i,j} \frac{1}{2} \left(G^{\mathcal{V}'}\right)_{i,j} \right)^{n/2}$$

Now the left side of the equation stands for the number of *ordered* perfect matchings. Also, the sum inside the parentheses corresponds to the sums we used before—things like $(a+b+c)^2$ instead of $(2a+2b+2c)^2$.

It will suffice to show:

$$(n/2)! \sum_{\sigma \in I_n} \prod_{i=1}^{n} \sqrt{g_{i,\sigma(i)}} = \sum_{\mathcal{V}' \subseteq \mathcal{V}} (-1)^{\#\mathcal{V}'} \left(\sum_{i,j} \frac{1}{2} \left(G^{\mathcal{V}'}\right)_{i,j} \right)^{n/2}$$

Define the set $U = \{g_{i,j} | i < j\}$ to be the entries in the upper triangle of the generic graph G. We will prove that the two sides of the above equation are identical as polynomials in the entries of G, that is, are identical as polynomials in the elements of U. Now

$$U^{n/2} = \{\text{words of length } n/2 \text{ in the entries of } G\}$$

Some of these words correspond to ordered perfect matchings, and some do not. The ones that cover an entire element of I_n are the ones we want, at least in the left side of the equation. Once again, we rewrite what we're trying to show:

$$\sum_{w \in U^{n/2}} \chi_{I_n} w = \sum_{\mathcal{V}' \subseteq \mathcal{V}} (-1)^{\#\mathcal{V}'} \left(\sum_{i,j} \frac{1}{2} \left(G^{\mathcal{V}'} \right)_{i,j} \right)^{n/2}$$

We use χ_{I_n} to be the characteristic function of I_n, that is, a function that gives 1 when given an ordered perfect matching, and 0 otherwise. Also note that the notation is a little sloppy—words in formal variables are the same as the ordered product of those same variables. In fact, the above identity should be considered to occur in $\mathbb{Z}[U^*]$ where U^* is the free monoid on U; an easy homomorphism will bring the result back to the land of numerical multiplication.

Consider a word on the left that corresponds to a perfect matching. It contains exactly one variable from each row and column. This same word must appear exactly once on the right too: the only term in the sum that has variables from every row and every column is the one where $\mathcal{V}' = \emptyset$, and this term contains exactly one of each word of length $n/2$:

$$\left(\sum_{i,j} \frac{1}{2} \left(G^{\mathcal{V}'} \right)_{i,j} \right)^{n/2} = \sum_{w \in U^{n/2}} w$$

Now consider a word on the left that does not correspond to a perfect matching. Such a word must miss some rows/columns. Let's say it misses k of them. Now the word might appear several times on the right. In fact, there will be 2^k subsets $\mathcal{V}' \subseteq \mathcal{V}$ that give rise to it. Luckily, these subsets are signed according to the number of elements they have, and we use the following identity:

$$\sum_{i=0}^{k} (-1)^i \binom{k}{i} = \chi_{\{0\}}(k)$$

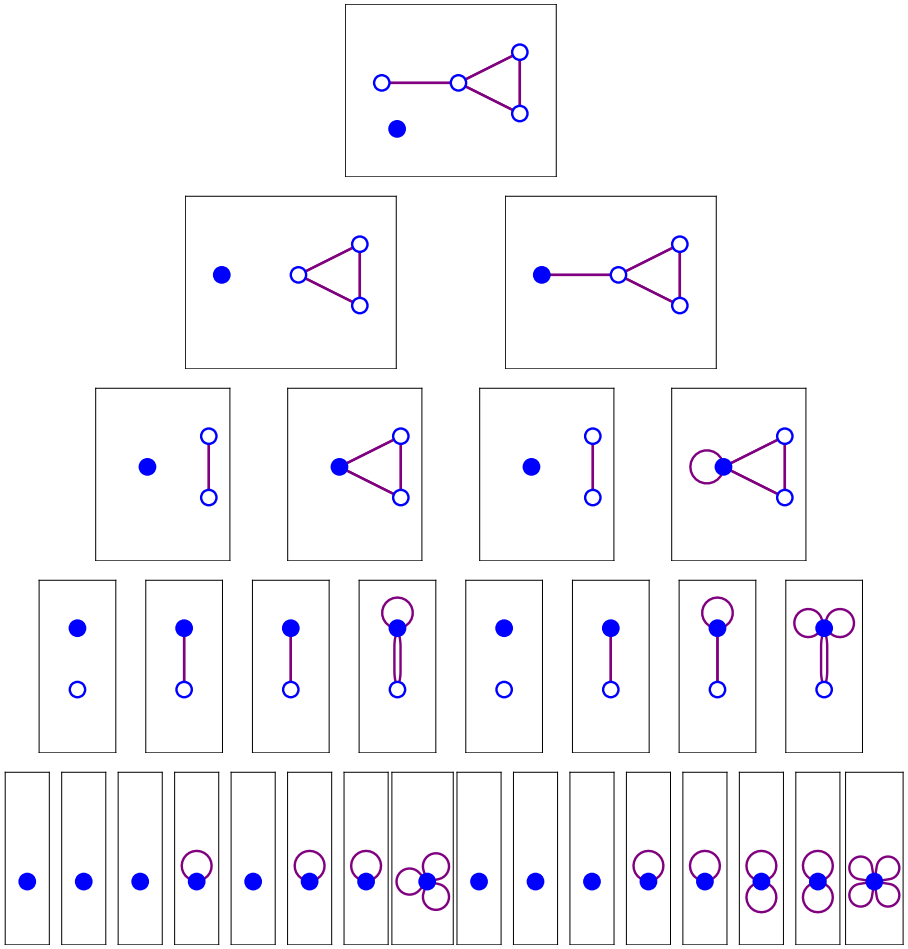
This identity is exactly the reason that a word missing 0 rows occurs exactly once, while all other words do not appear. The proof is complete.

□

3. A Graph Recursion for Counting Perfect Matchings

A graph recursion algorithm calculates a property of a graph based on slightly smaller graphs. In this way, calculating a property for a complex graph reduces to calculating the property for a lot of much smaller graphs. The algorithm we present re-writes the number of perfect matchings in a graph using two graphs that have one fewer vertex. This is unexpected since removing a single vertex changes the parity of the graph, seemingly rendering perfect matching information trivial.

If we know how to count perfect matchings for odd graphs too, however, the algorithm is quite natural. Here is an example of the procedure, which is called the "secret agent" recursion:



In the top frame, we see the secret agent—a solid blue vertex—eyeing the enemy agents he is about to neutralize. His plan of attack will proceed from left to right first, and top to bottom second. At each stage in the operation, the secret agent has a choice: simply kill the enemy agent, or kill him and impersonate him. If he chooses to kill, then he moves to the enemy's location, severing the enemy's contacts. If he decides to impersonate, then he moves to the enemy's location, but maintains the enemy's contacts. In the diagram, killing is on the left and impersonating is on the right.

As the operation unfolds, the secret agent may accumulate self-loops. At the end, in fact, all the enemy agents are dead and we count the number of self-loops on the agent:

$$(0, 0, 0, 1, 0, 1, 1, 3, 0, 0, 0, 1, 1, 2, 2, 4)$$

We raise each number to the power q,

$$(0^q, 0^q, 0^q, 1^q, 0^q, 1^q, 1^q, 3^q, 0^q, 0^q, 0^q, 1^q, 1^q, 2^q, 2^q, 4^q)$$

And we give each one a sign according to it's location in the tree: each kill multiplies by (-1). The pattern of signs is called the Thue-Morse sequence (OEIS A010060).

$$\begin{aligned}\mathcal{C}(q) &= 0^q - 0^q - 0^q + 1^q - 0^q + 1^q + 1^q - 3^q - 0^q + 0^q + 0^q - 1^q + 1^q - 2^q - 2^q + 4^q \\ &= 3 \cdot 1^q - 2 \cdot 2^q - 3^q + 4^q\end{aligned}$$

This function has a combinatorial meaning: $\mathcal{C}(q)$ gives the number of ordered edge-coverings of the enemy graph by q edges (this is an easy generalization; simply replace $n/2$ with q in the proof of the theorem) . In particular, if we calculate $\mathcal{C}(2)/2!$ we get the number of perfect matchings of the enemy network:

$$\mathcal{C}(2)/2! = \frac{1}{2}\left(3 \cdot 1^2 - 2 \cdot 2^2 - 3^2 + 4^2\right) = \frac{1}{2}\left(3 - 8 - 9 + 16\right) = \frac{1}{2}\left(2\right) = 1$$

which is the correct answer. Before we state the recursion as a theorem, let's have some definitions:

Definition 3.1. A *subverted graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a graph with one vertex, $x_{\mathcal{G}} \in \mathcal{V}$, designated as the secret agent. We may *subvert* a graph $\mathcal{H}$ by adjoining a secret agent vertex and no edges; the new subverted graph is called $\tilde{\mathcal{H}}$.

Definition 3.2. We define a function recursively on subverted graphs $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. If $v \in \mathcal{V}$, $v \neq x_{\mathcal{G}}$, let $\mathcal{G}_{\text{kill } v} = \mathcal{G} \setminus \{v\}$ be the graph $\mathcal{G}$ without the vertex v, and let $\mathcal{G}_{\text{keep } v} = \mathcal{G}/(x_{\mathcal{G}} \sim v)$ be the graph that identifies v and the secret agent. The function is called $\mathcal{C}_{\mathcal{G}}(q)$, and is defined by the following recursion:

$$\mathcal{C}_{\mathcal{G}}(q) := \mathcal{C}_{\mathcal{G}_{\text{keep } v}}(q) - \mathcal{C}_{\mathcal{G}_{\text{kill } v}}(q)$$

For the base case, we use graphs containing only the secret agent and self-loops. Say $\mathcal{X}_n$ is a graph with a single vertex—$x_{\mathcal{X}_n}$—and n self-loops. We define $\mathcal{C}$ in this case:

$$\mathcal{C}_{\mathcal{X}_n}(q) := n^q$$

Theorem 3.3. *We prove the following facts about $\mathcal{C}_{\mathcal{G}}(q)$:*
(a) $\mathcal{C}$ is well-defined; that is, the recursion doesn't depend on the choice of v.
(b) If $\mathcal{G}$ is a graph, then $\mathcal{C}_{\tilde{\mathcal{G}}}(q) = \#\{$ ordered coverings of $\mathcal{G}$ by q edges $\}$.

Proof. Both of these facts follow easily from previous considerations. Let's prove *(b)* first. Here's what we need to show:

$$\mathcal{C}_{\tilde{\mathcal{G}}}(q) = \sum_{\mathcal{V}' \subseteq \mathcal{V}} (-1)^{\#\mathcal{V}'} \left(\sum_{i,j} \frac{1}{2} \left(G^{\mathcal{V}'} \right)_{i,j} \right)^q$$

In fact, since the secret agent has no edges, it's just as good to show

$$\mathcal{C}_{\tilde{\mathcal{G}}}(q) = \sum_{\mathcal{V}' \subseteq \mathcal{V}, x_{\tilde{\mathcal{G}}} \in \mathcal{V}'} (-1)^{\#\mathcal{V}'} \left(\sum_{i,j} \frac{1}{2} \left(G^{\mathcal{V}'} \right)_{i,j} \right)^q$$

Naturally, we break it up into two cases: one for $\mathcal{G}_{\text{keep } v}$ and one for $\mathcal{G}_{\text{kill } v}$:

$$\begin{aligned}\mathcal{C}_{\tilde{\mathcal{G}}}(q) &= \sum_{\mathcal{V}'\subseteq\mathcal{V}\wedge v, x_{\tilde{\mathcal{G}}}\in\mathcal{V}'} (-1)^{\#\mathcal{V}'}\left(\sum_{i,j}\frac{1}{2}\left(G^{\mathcal{V}'}\right)_{i,j}\right)^q \\ &\quad + \sum_{\mathcal{V}'\subseteq\mathcal{V}, v\notin\mathcal{V}', x_{\tilde{\mathcal{G}}}\in\mathcal{V}'} (-1)^{\#\mathcal{V}'}\left(\sum_{i,j}\frac{1}{2}\left(G^{\mathcal{V}'}\right)_{i,j}\right)^q\end{aligned}$$

The second sum is the same as $\mathcal{C}_{\tilde{\mathcal{G}}\setminus\{v\}}(q)$, and the first sum will be the same if we identify $x_{\tilde{\mathcal{G}}}$ and v. We have proved that this satisfies the recursion:

$$\mathcal{C}_{\tilde{\mathcal{G}}}(q) = \mathcal{C}_{\tilde{\mathcal{G}}_{\text{kill } v}}(q) - \mathcal{C}_{\tilde{\mathcal{G}}_{\text{keep } v}}(q)$$

In fact, the same proof shows that for subverted graphs in general we have

$$\mathcal{C}_{\mathcal{G}}(q) = \sum_{\mathcal{V}'\subseteq\mathcal{V}\wedge x_{\mathcal{G}}\in\mathcal{V}'} (-1)^{\#\mathcal{V}'}\left(\sum_{i,j}\frac{1}{2}\left(G^{\mathcal{V}'}\right)_{i,j}\right)^q$$

which also takes care of part *(a)* since this formula doesn't reference v at all. All that remains is to verify the base case. Indeed:

$$\mathcal{C}_{\mathcal{X}_n}(q) = \sum_{\mathcal{V}'\subseteq\mathcal{V}\wedge x_{\mathcal{G}}\in\mathcal{V}'} (-1)^{\#\mathcal{V}'}\left(\sum_{i,j}\frac{1}{2}\left(G^{\mathcal{V}'}\right)_{i,j}\right)^q = n^q$$

□

4. An Example Calculation

We conclude by calculating the number of perfect matchings in the odd graph we mentioned earlier:

Here's the graph's adjacency matrix:

$$\begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

Now we just use the signed sum over the number of edges in subgraphs, as seen before:

$$\begin{aligned}\mathcal{C}(q) &= (1+1+0)^q - 1^q - 1^q - 0^q \\ &= 2^q - 2\end{aligned}$$

As before, $\mathcal{C}(q)$ gives the number of ordered covers of the graph using q edges. For a perfect matching, we should take $q = n/2 = 3/2$. Despite the seeming absurdity, we now have that the number of ordered perfect matchings is

$$\mathcal{C}(3/2) = 2\sqrt{2} - 2$$

To find the number of unordered perfect matchings, we divide by $q!$. Since q is not an integer, we use the gamma function to calculate

$$\text{number of perfect matchings} \;=\; \mathcal{C}(3/2)/(3/2)! \;=\; \frac{1}{3}\left(8-4\sqrt{2}\right)\sqrt{\frac{2}{\pi}} \;\approx\; 0.623187$$

5. Acknowledgements

Thanks to my mentors, Ilya and Spencer; they had to listen to me talk about a lot of things, none of which had much to do with this paper. Thanks to Jim Fowler, who kept the whole REU running smoothly. Finally, thanks to Peter May, whose class I'm going to try to take next year, in spite of what Paul Sally will say.

References

[1] J. P. May. A Concise Course in Algebraic Topology. University of Chicago Press. 1999.