20191122

Reduction of k-Clique to Vertex Cover

As we have just seen, proving that a problem X is NP-Complete by reducing SAT (or CNF-SAT) to X can be arduous. Part of the difficulty is that Boolean expressions don't really look like graphs, or sets of integers, or points in a plane, or any of the thousands of other problem domains that we are interested in. So reducing SAT to a problem of some other type can require a lot of creativity.

Fortunately we usually don't have to do that. We saw in the previous notes that we can show a problem X is NP-Complete with a reduction from ***any*** NP-Complete problem. So now that we know k-Clique is NP-Complete, we can use it to show that other problems are NP-Complete … and then we can use them to show that still more problems are NP-Complete. Each new problem that is identified as NP-Complete gives us another tool that we can use to expand the NP-Complete set. In other words, proving that problems are NP-Complete gets easier and easier.

There are online lists of known NP-Complete problems. Here's one to get you started if you are interested in exploring this topic:

https://cgi.csc.liv.ac.uk/~ped/teachadmin/COMP202/annotated_np.html

Here is an illustration of how a reduction can be constructed between two problems that are in the same domain. This is one of the very earliest reductions found.

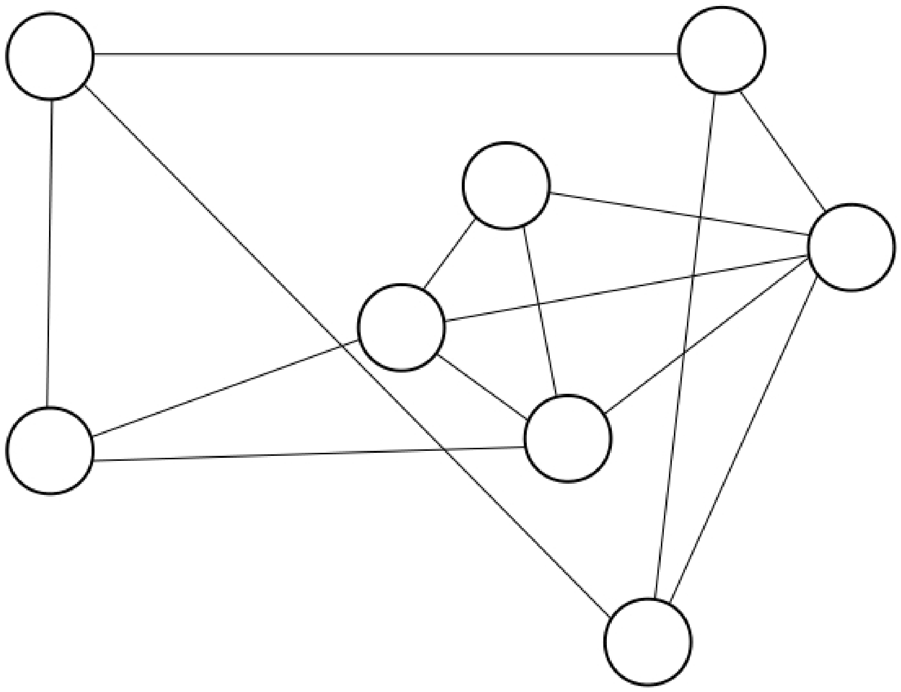
The k-Vertex Cover Problem: Let G be a graph on n vertices. A **vertex cover** of G is a subset S of G's vertex set with the property that every edge has at least one end in S. A **k-vertex cover** is a vertex cover that contains exactly k vertices.

Here's a simple (and almost realistic) application of this problem. Suppose the graph represents an art gallery – the edges represent hallways that are filled with valuable paintings, and the vertices represent rooms where the hallways meet. You need to place guards in the rooms so that every hallway can be watched by at least one of the guards. If the graph has a k-vertex cover then you can guard the entire gallery with k guards.

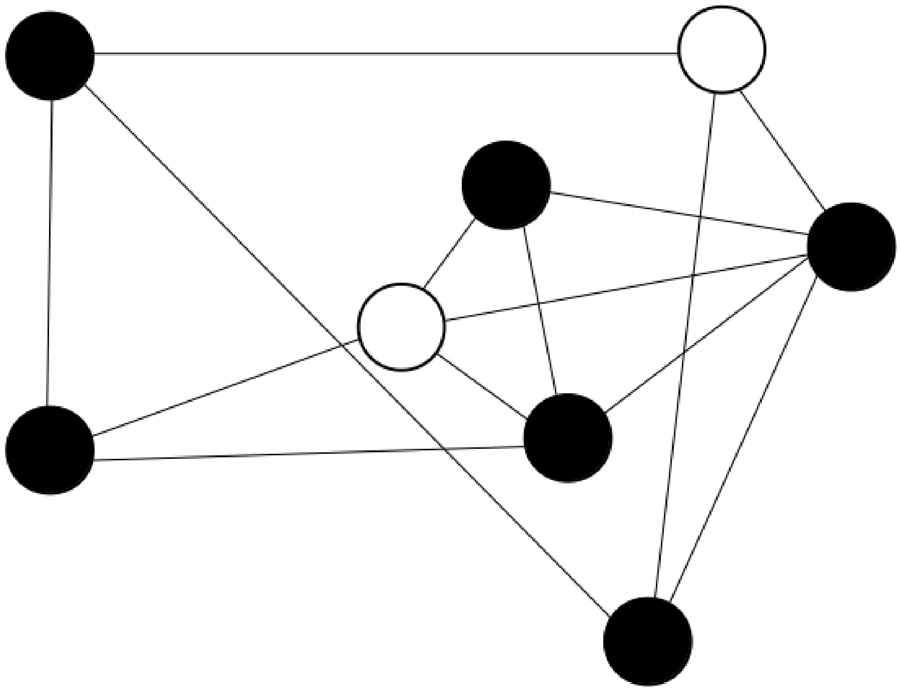
(This is a simplified version of a much more general problem in computational geometry. In the general problem the physical dimensions of the rooms and hallways are considered. This is a much-studied problem – with some very sophisticated results.)

Another application (more mundane but perhaps more practical) is to test for link-failures in a communication network. If each vertex represents a node in the network and the edges represent links that are potentially subject to failure, then a k-vertex cover gives us a set of k nodes that are capable of testing every link in the network by pinging the node at the other end.

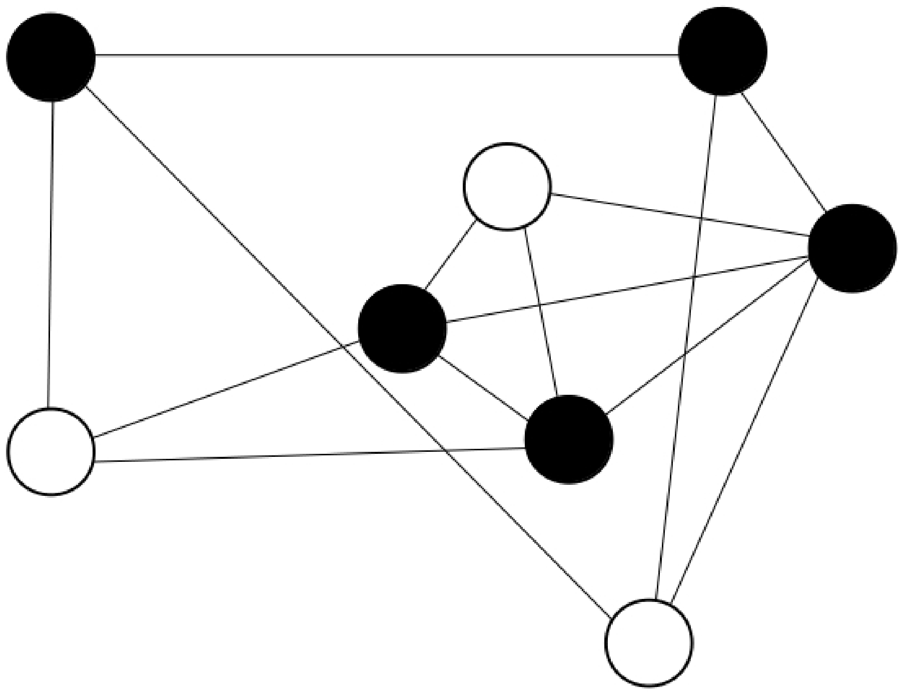
Here's a graph



and here's a vertex cover of size 6



and here's a vertex cover of size 5



You might want to convince yourself that even though the 6-vertex cover is not the smallest vertex cover of the graph, it has no redundant vertices – if we try to reduce it to a 5-vertex cover by taking out any of the vertices, we find that there is at least one edge of the graph that is not covered. (A quick way to see that this is true is to observe that each of the 6 vertices in the cover has at least one neighbour that is not in the cover.)

So this graph has a 5-vertex cover … but does it have a 4-vertex cover? Is there a better algorithm to answer this than the BFI algorithm that simply tries every combination of 4 vertices?

We're about to prove that the answer is (almost certainly) "No, there is no better algorithm for this problem."

Theorem: k-Vertex Cover is NP-Complete.

Proof:

Claim: k-Vertex Cover is in $\mathcal{NP}$

If the answer to an instance of k-Vertex Cover is **Yes** and we are told which k vertices form the cover, we can check all edges to confirm that each edge has at least one end in the cover in polynomial time. Thus k-Vertex Cover is in $\mathcal{NP}$.

Claim: k-Clique $\propto$ k-Vertex Cover

This proof has the potential to be confusing because the "target value" will change. That is, we will start with an instance of k-Clique where the size of the required set of vertices has a particular value and we will construct an instance of k-Vertex Cover where the size of the desired set has a different value. But we will see that it all works out in the end (ie the transformation is answer-preserving).

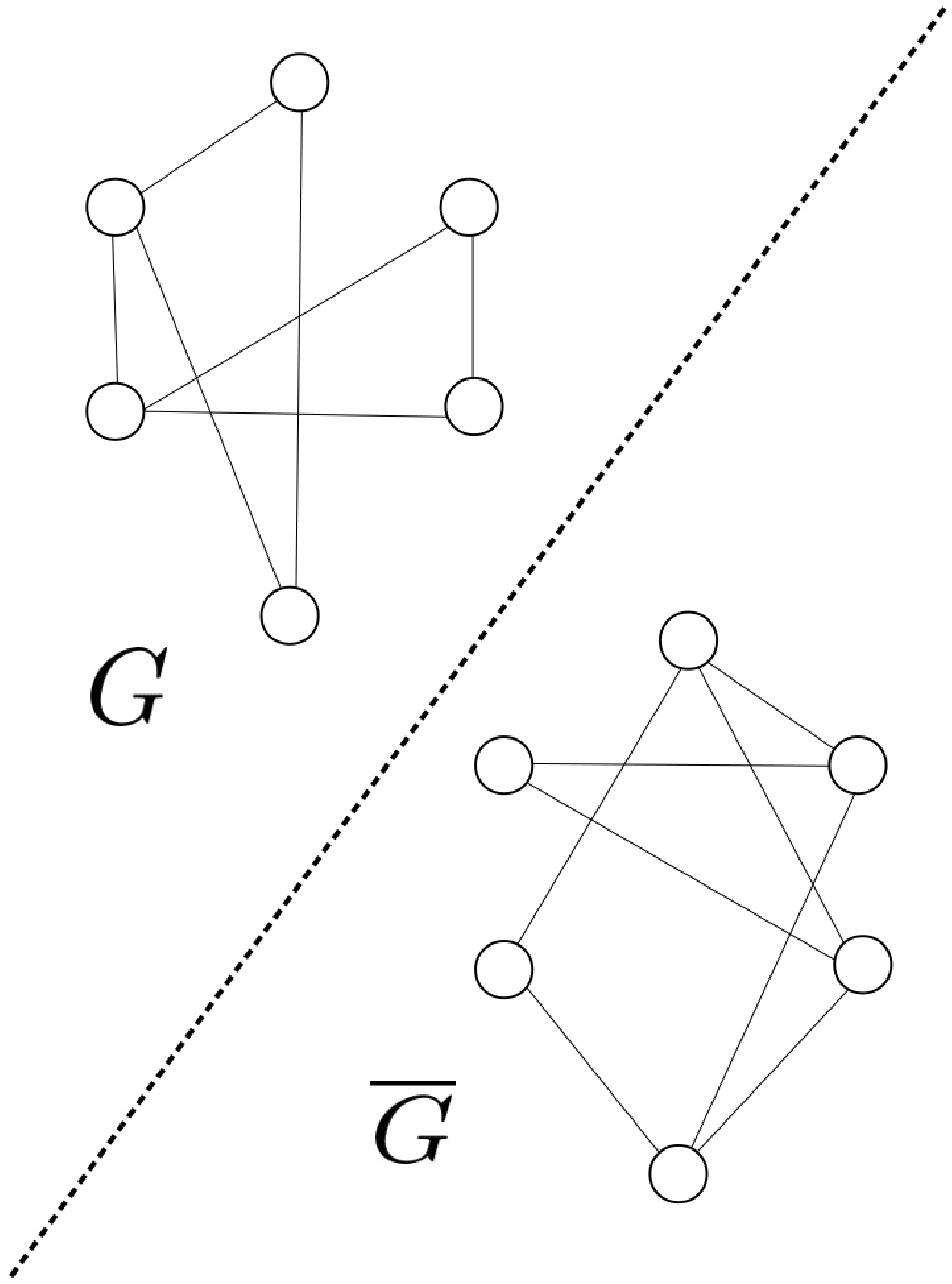
Let (G, k) be an instance of k-Clique, where G is a graph on n vertices and k is an integer. That is, we are asking "Does G contain a set of k vertices that are all adjacent?"

To build a related instance of k-Vertex Cover (remember, the k is going to be different!) we first construct the graph $\overline{G}$ which is the **complement** of G.

Definition: Let G be a graph on n vertices. Then $\overline{G}$, the complement of G, is defined on the same set of vertices, with the edge set of $\overline{G}$ containing exactly the edges that are not present in G. Formally, we have

$$\forall \text{ distinct vertices } x \text{ and } y,\ \text{edge}\,(x, y) \in \overline{G} \text{ iff } (x, y) \notin G$$

Here’s an example:



Note that given G, we can easily construct $\overline{G}$ in polynomial time. Now we need to determine how we can phrase a vertex cover problem on $\overline{G}$ that is answer-preserving with the original k-clique problem on G.

We're going prove this result:

$\forall k \geq 1, G$ has a clique of size k iff G has a vertex cover of size n-k

Part 1: Suppose G has a k-clique. Without loss of generality we can assume that the vertices in the k-clique are $\{v_1, v_2, \ldots v_k\}$ and the rest of the vertices are $\{v_{k+1}, \ldots v_n\}$

Note that all the edges of the form (v_i, v_j) where $1 \leq i < j \leq k$ are in G (this is the definition of a k-clique) … which means that none of these edges are in $\overline{G}$. Therefore all edges of $\overline{G}$ have at least one end in $\{v_{k+1}, \ldots v_n\}$. Therefore $\{v_{k+1}, \ldots v_n\}$ is a vertex cover of $\overline{G}$, with size $n - k$

Thus "G has a k-Clique" ⇒ "$\overline{G}$ has an (n-k)-Vertex Cover"

Part 2: Suppose $\overline{G}$ has an (n-k)-Vertex Cover. Again without loss of generality, assume the vertices that form this vertex cover are numbered $\{v_{k+1}, \ldots v_n\}$. This means every edge in $\overline{G}$ has at least one end in this set. Therefore no edge of $\overline{G}$ has both ends in the set $\{v_1, \ldots v_k\}$. Therefore every possible edge with both ends in $\{v_1, \ldots v_k\}$ is in G. Therefore the vertex set $\{v_1, \ldots v_k\}$ forms a k-clique in G.

Thus "$\overline{G}$ has an (n-k)-Vertex Cover" ⇒ "G has a k-Clique"

Putting Parts 1 and 2 together we are able to conclude that

G has a k-clique iff $\overline{G}$ has an (n-k)-vertex cover

And that's our reduction! We transform the question "Does G have a k-clique?" into the question "Does $\overline{G}$ have an (n-k)-vertex cover?"

We've shown that the construction of $\overline{G}$ takes polynomial time, and we have shown that the transformation is answer-preserving.

Thus we conclude that k-Clique $\propto$ k-Vertex Cover

Therefore k-Vertex Cover is NP-Complete.

Note 1: I can't stress strongly enough that when we write something like "k-Clique $\propto$ k-Vertex Cover" , the "k" is really just a signal that there is an integer parameter that is an essential part of each instance of the problem. It's not a variable that is bound to have the same value on both sides of the $\propto$ relation. When we write "k-Clique $\propto$ k-Vertex Cover" what we mean is "each instance of the clique problem with a specified parameter value can be transformed into an instance of the vertex cover problem with a (possibly different) specified parameter value, in an answer-preserving way".

Note 2: When we first introduced the concept of an answer-preserving transformation between two problems X and Y, we said "Instances of X with Yes answers must map onto instances of Y with Yes answers, and instances of X with No answers must map onto instances of Y with No answers".

But in the two proofs of NP-Completeness we have done, we have done something that looks a bit different: we have proved that when an instance of X is transformed into an instance of Y, the answer to the original X instance is Yes iff the answer to the instance of Y is Yes.   It may look like we are not requiring that No answers are preserved.

In fact, we **are** preserving No answers as well as Yes answers. To see that this is true, suppose we have proved that the X answer is Yes iff the Y answer is Yes. Now suppose at least one No answer is not preserved. This would mean that there is some instance of X where the answer is No, which gets transformed into an instance of Y where the answer is Yes. But our original supposition was that we have proved that if the Y answer is Yes, then the X answer must also be Yes. So we are in a state of contradiction: the X answer is No but it must be Yes. The most recent supposition was that some No answer to X is not preserved. This supposition led to a contradiction so it must be false. Therefore if we have proved that the X answer is Yes iff the Y answer is Yes, then all No answers **are** preserved as well.