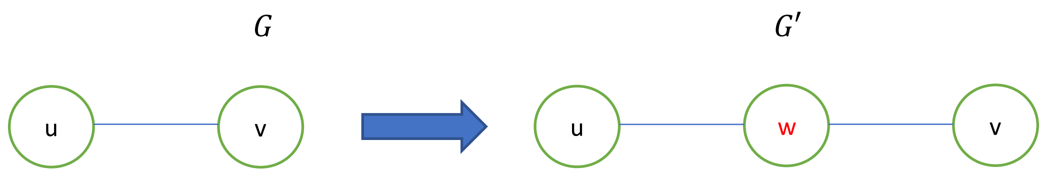
The Algorithm

We construct G' by taking G and adding a new node on every edge. For example, say we have edge (u, v) from G, and we add a new node w on it for G', it would become two edges (u, w) and (w, v).



So G' is constructed as such.
k' is k.
S' is V (all the original nodes/people from G).

Proof of Correctness

Claim: if $G = (V, E)$ has an independent set of size k, then G' has a very independent subset of the set S' of size k'.
Proof: We can rephrase this claim to *if $G = (V, E)$ has an independent set of size k, then G' has a very independent subset of the set S' of size at least k*, because if it has a subset of S' with more than $k' = k$ elements, say it has h elements, then just take away any $h - k$ nodes, as the remaining nodes would still be very independent. G' can't have less than k very independent people (nodes) because we've separated every old node from each other with a new node, so all the original independent people who shared a mutual friend no longer do, and will be classified as very independent. And of course, all the original independent people who didn't share mutual friends would still be correctly classified as very independent. In short, we don't "lose" independent people in my friend's code.

Claim: if $G = (V, E)$ has no independent set of size k, then G' also doesn't have a very independent subset of the set S' of size k'.
Proof: Say there are $h < k$ nodes in the original independent set from G. Using the previous claim we proved, this means G' has a very independent subset of S' of size h. If we were to assume that G' *does* have a very independent subset of S' of size $k' = k$, that means at least $k - h$ non-independent nodes from G are in it. But these nodes are not very independent! If they are not independent, that means some of them have edges connecting them, and those edges are still present in G', so they cannot be very independent either.

Runtime Analysis

The reduction algorithm (computing G', k', and S') is linear in the number of edges $|E|$ because we go to every edge in G and just add a new node, which effectively is deleting that old edge and adding two new edges. Getting k' and S' is constant because we're just equating them to old inputs. If we let $m = |E|$, this algorithm is $O(m)$.