1. This problem asks you to develop polynomial-time algorithms for two (apparently) minor variants of 3SAT.

   (a) The input to **2SAT** is a boolean formula Φ in conjunctive normal form, with exactly **two** literals per clause, and the 2SAT problem asks whether there is an assignment to the variables of Φ such that every clause contains at least one TRUE literal.

   Describe a polynomial-time algorithm for 2SAT. *[Hint: This problem is strongly connected to topics covered earlier in the semester.]*

Solution: Let Φ be an arbitrary 2CNF formula, and suppose Φ has n variables $x_1, x_2, \ldots, x_n$ and m clauses. Trivially, $n \leq 2m$. We construct a directed graph $G = (V, E)$ as follows:

- $V = \{x_1, x_2, \ldots, x_n, \bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n\}$ is the set of all unique literals in Φ.
- E contains two directed edges for each clause in Φ, intuitively recording implications between the corresponding pair of variables. Specifically:
  - For each clause $(x_i \vee x_j)$, we include edges $\bar{x}_i \to x_j$ and $\bar{x}_j \to x_i$.
  - For each clause $(x_i \vee \bar{x}_j)$, we include edges $\bar{x}_i \to \bar{x}_j$ and $x_j \to x_i$.
  - For each clause $(\bar{x}_i \vee \bar{x}_j)$, we include edges $x_i \to \bar{x}_j$ and $x_j \to \bar{x}_i$.

Altogether, G has $2n \leq 4m$ vertices and $2m$ edges. We can construct G in ***O(m) time*** by brute force.

Now consider an arbitrary directed walk $v_1 \to v_2 \to \cdots \to v_k$ in G. The definition of the directed edges implies that in any satisfying assignment, if the literal corresponding to any vertex v_i is TRUE, then the literals corresponding to $v_{i+1}, v_{i+2}, \ldots, v_k$ must also be TRUE. Conversely, if the literal corresponding to any vertex v_i is FALSE, then the literals corresponding to $v_1, v_2, \ldots, v_{i-1}$ must also be FALSE.

It follows immediately that if two vertices u and v are strongly connected, then in every satisfying assignment for Φ, the literals corresponding to u and v must have the same truth value. In particular, if any variable x_i and its complement $\bar{x}_i$ lie in the same strong component of G, then the formula Φ is *not* satisfiable.

We can test this condition in $O(V + E) =$ ***O(m) time*** as follows. Compute the strong component graph H of G. Index the vertices of H (that is, the strong components of G) in topological order. For each vertex u in G, store the index of the strong component of u in a new field $comp(u)$. Finally, if $comp(x_i) = comp(\bar{x}_i)$ for any index i, return FALSE; otherwise, return TRUE.

We have already argued that the FALSE answers are always correct. It remains to prove that the TRUE answers are also correct. That is, we need to prove that if *no* variable is strongly connected to its negation, then Φ *is* satisfiable.

So suppose no variable vertex lies in the same strong component as its negation. We can assign consistent truth values to the variables of Φ as follows:

- If $comp(x_i) < comp(\bar{x}_i)$, set $x_i =$ FALSE.
- If $comp(x_i) > comp(\bar{x}_i)$, set $x_i =$ TRUE.

For the sake of argument, suppose that Φ contains a clause $(u \vee v)$ that is *not* satisfied by this assignment; both literals in this clause are FALSE. We derive a contradiction by considering the corresponding edges $\bar{u} \to v$ and $\bar{v} \to u$ in G as follows:

- The edge $\bar{u} \to v$ implies that $comp(\bar{u}) \leq comp(v)$.
- The assignment $u = \text{FALSE}$ implies that $comp(u) < comp(\bar{u})$.
- The edge $\bar{v} \to u$ implies that $comp(\bar{v}) \leq comp(u)$.
- The assignment $v = \text{FALSE}$ implies that $comp(v) < comp(\bar{v})$.

It is impossible to satisfy all four of these inequalities. We conclude that our assignment satisfies every clause of Φ; in other words, Φ is in fact satisfiable. ■

Rubric: 4 points: standard graph reduction rubric (scaled), *without* time analysis (covered in part (c)). This is not the only correct linear-time algorithm. This is more detail than necessary for full credit. In particular, a formal proof of correctness is not required.

However, any algorithm presented without justification *must* cite an external source to receive *any* credit. We already know you can use Google.

(b) The input to **MAJORITY3SAT** is a boolean formula Φ in conjunctive normal form, with exactly three literals per clause. MAJORITY3SAT asks whether there is an assignment to the variables of Φ such that every clause contains *at least two* TRUE literals.

Describe and analyze a polynomial-time reduction from MAJORITY3SAT to 2SAT. Don't forget to prove that your reduction is correct.

Solution: Let Φ be an arbitrary boolean formula Φ in conjunctive normal form, with exactly three literals per clause. We construct a new boolean formula Φ_2, also in conjunctive normal form, by replacing each three-literal clause in Φ with three two-literal clauses as follows:

$$(a \vee b \vee c) \longmapsto (a \vee b) \wedge (a \vee c) \wedge (b \vee c).$$

In particular, Φ and Φ_2 use the same variables.

Fix an arbitrary assignment of values to these variables. I claim that every clause in Φ_2 contains at least one TRUE literal if and only if a majority of the literals in each clause of Φ are TRUE.

$\Longleftarrow$ Suppose every clause in Φ_2 contains at least one TRUE literal. Consider an arbitrary clause $(a \vee b \vee c)$ of Φ. Each of the corresponding clauses $(a \vee b) \wedge (a \vee c) \wedge (b \vee c)$ in Φ_2 contains a TRUE literal. Thus, at most one of the literals a, b, and c is FALSE. We conclude that a majority of literals in every clause of Φ are TRUE.

$\Longrightarrow$ Suppose a majority of literals in every clause of Φ are TRUE. Consider an arbitrary clause $(a \vee b \vee c)$ of Φ. At most one of the literals a, b, and c is FALSE. Thus, each of the corresponding clauses $(a \vee b) \wedge (a \vee c) \wedge (b \vee c)$ in Φ_2 contains a TRUE literal. We conclude that every clause in Φ_2 contains a TRUE literal.

Suppose the input formula Φ has m clauses, and therefore total length $O(m)$. Then we can transform Φ into Φ_2 in ***$O(m)$ time*** by brute force. ■

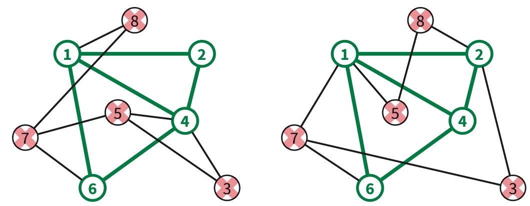
Rubric: 4 points: standard NP-hardness rubric. This is (slightly) more detail than necessary for full credit.

(c) Combining parts (a) and (b) gives us an algorithm for MAJORITY3SAT. What is the running time of this algorithm?

Solution: ***$O(m)$ time***, where m is the number of clauses (or the overall length) of the input 3CNF formula. ■

Rubric: 2 points = 1 point for analyzing part (a) + 1 point for analyzing part (b).

2. Suppose we are given two graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ with the same set of vertices $V = \{1, 2, \ldots, n\}$. Prove that is it NP-hard to find the smallest subset $S \subseteq V$ of vertices whose deletion leaves identical subgraphs $G_1 \setminus S = G_2 \setminus S$. For example, given the graphs below, the smallest subset has size 4.



Solution: We prove the problem NP-hard by reduction from MaxIndependentSet.

Let $G = (V, E)$ be an arbitrary input graph, and suppose G has n vertices. Let $G_2 = (V, \varnothing)$ be the graph obtained from G by deleting every edge. I claim that for any integer k, deleting k vertices leaves identical subgraphs of G and G_2 if and only if G contains an independent set of size $n - k$.

$\Longrightarrow$ Let S be any subset of V such that $G \setminus S = G_2 \setminus S$, and suppose $|S| = k$. The subgraph $G_2 \setminus S$ has no edges, which implies that the subgraph $G \setminus S$ has no edges. It follows that $V \setminus S$ is an independent set in G of size $n - k$.

$\Longleftarrow$ Suppose G has an independent set I of size $n - k$. Let $S = V \setminus I$. The definition of independent set implies that the subgraph $G \setminus S$ contains no edges. But $G_2 \setminus S$ also contains no edges, so $G \setminus S = G_2 \setminus S$.

Thus, if k is the minimum number of vertices k whose deletion makes G and G_2 identical, then $n - k$ is the size of the largest independent set in G.

We can obviously construct G_2 from G in polynomial time by brute force. ■

Solution: We prove the problem NP-hard by reduction from MaxClique.

Let $G = (V, E)$ be an arbitrary input graph, and suppose G has n vertices. Let $G_2 = (V, \binom{V}{2})$ be the graph obtained from G by inserting an edge between every pair of vertices that doesn't already have one. I claim that for any integer k, deleting k vertices leaves identical subgraphs of G and G_2 if and only if G contains an clique of size $n - k$.

$\Longrightarrow$ Let S be any subset of V such that $G \setminus S = G_2 \setminus S$, and suppose $|S| = k$. The subgraph $G_2 \setminus S$ is complete, because every induced subgraph of a complete graph is complete. Thus, $G \setminus S$ is also a complete graph, which implies that $V \setminus S$ is a clique of size $n - k$ in G.

$\Longleftarrow$ Suppose G contains a clique K of size $n - k$. Let $S = V \setminus K$; this set has size k. The definition of clique implies that the subgraph $G \setminus S$ is a complete graph. But $G_2 \setminus S$ is a complete graph with the same vertices as $G \setminus S$, so $G \setminus S = G_2 \setminus S$.

Thus, if k is the minimum number of vertices k whose deletion makes G and G_2 identical, then $n - k$ is the size of the largest clique in G.

We can obviously construct G_2 from G in polynomial time by brute force. ■

Solution: We prove the problem NP-hard by reduction from MinVertexCover.

Let $G = (V, E)$ be an arbitrary input graph, and suppose G has n vertices. Let $G_2 = (V, \varnothing)$ be the graph obtained from G by deleting every edge. Let S be an arbitrary subset of the vertices V. I claim that $G \setminus S = G_2 \setminus S$ if and only if S is a vertex cover in G.

$\Longrightarrow$ First, suppose $G \setminus S = G_2 \setminus S$. The subgraph $G_2 \setminus S$ has no edges, which implies that the subgraph $G \setminus S$ has no edges. It follows that every edge in G has at least one endpoint in S; in other words, S is a vertex cover.

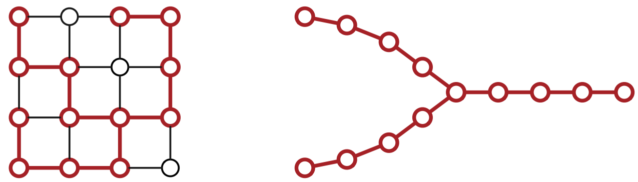
$\Longleftarrow$ Suppose S is a vertex cover of G. The definition of vertex implies that the subgraph $G \setminus S$ contains no edges. $G_2 \setminus S$ is an empty graph with the same vertices as $G \setminus S$. It follows that $G \setminus S = G_2 \setminus S$.

Thus, the minimum number of vertices k whose deletion makes G and G_2 identical is equal to the size of the smallest independent set in G.

We can obviously construct G_2 from G in polynomial time by brute force. ■

Rubric: 10 points: standard polynomial-time reduction rubric.

3. A ***wye*** is an undirected graph that looks like the capital letter Y. More formally, a wye consists of three paths of equal length with one common endpoint, called the *hub*.

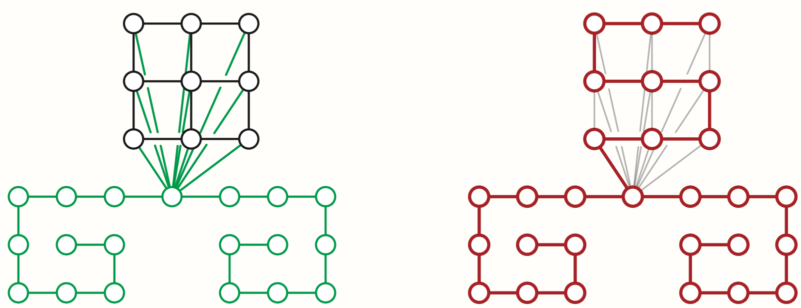


This grid graph contains a wye whose paths have length 4.

Prove that the following problem is NP-hard: Given an undirected graph G, what is the largest wye that is a subgraph of G? The three paths of the wye must not share any vertices except the hub, and they must have exactly the same length.

Solution: We can prove this problem is NP-hard by reduction from the Hamiltonian path problem in undirected graphs.

Let G be an arbitrary undirected graph, and suppose G has n vertices. We construct a new graph H from G by adding a path of $2n+1$ new vertices $x_n, x_{n-1}, \dots, x_1, y, z_1, z_2, \dots, z_n$, along with edges from the middle vertex y to every original vertex of G. The resulting graph H has $3n+1$ vertices. I claim that H contains a wye whose arms have length n if and only if G contains a Hamiltonian path.



$\Longrightarrow$ Suppose G contains a Hamiltonian path $v_1, v_2, \dots, v_n$. Adding the edge yv_1 and the new path $x_n, x_{n-1}, \dots, x_1, y, z_1, z_2, \dots, z_n$ to this Hamiltonian path gives us a wye in H with arm-length n.

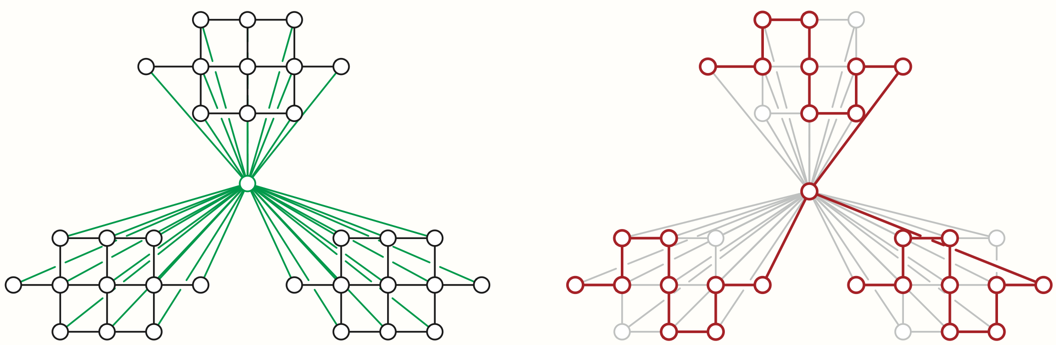
$\Longleftarrow$ Suppose H contains a wye Y whose arms have length n. Then Y must contain all $3n+1$ vertices of H, and in particular must contain the entire path $x_n, \dots, x_1, y, z_1, \dots, z_n$. The hub of Y must be y (the only vertex on this path with degree more than 2), and two of the arms must be $x_n, \dots, x_1$ and $z_1, \dots, z_n$. Let $y, v_1, v_2, \dots, v_n$ be the third arm of Y. Every vertex v_i is a vertex of G; thus, the subpath $v_1, v_2, \dots, v_n$ is a Hamiltonian path in G.

We can construct H from G in polynomial time by brute force. ■

Solution: We can prove this problem NP-hard by reduction from the longest path problem in undirected graphs.

Let G be an arbitrary undirected graph, and suppose G has n vertices. We construct a new graph H from three copies G_1, G_2, G_3 of G by adding a new vertex y and new edges from y to every vertex of every copy of G. The resulting graph H has $3n+1$ vertices.

I claim that the length of the longest path in G is one less than the arm-length of the largest wye in H. As usual, we prove this claim to two steps.



$\Longrightarrow$ Suppose the longest path in G has length k. Connecting the three copies of this path to y gives us a wye in H with arm-length $k+1$.

$\Longleftarrow$ Let Y be the largest wye in H, and suppose the arms of Y have length ℓ. Clearly $\ell \geq 1$. There are two cases to consider:

- Suppose the hub of Y is the central vertex y. Then each arm of Y contains a path of length $\ell - 1$ in one of the copies of G.
- On the other hand, suppose the hub of Y is *not* the central vertex y. At most one path of Y can pass through the central vertex y. Thus, by removing one arm of Y, we obtain a path of length $2\ell > \ell - 1$ in one of the copies of G.[a]

In both cases, we find a path of length at least $\ell - 1$ in G.

We can construct H from G in polynomial time by brute force. ■

[a]The forward argument now implies that H contains a wye with arm-length $2\ell + 1$, contradicting our assumption that Y is the largest wye in H, but the proof is already done, so I don't care.

Rubric: 10 points, standard polynomial-time reduction rubric. These are not the only correct solutions. (In particular, there *is* a correct reduction from 3SAT, because wyes have three arms, but it's pretty ugly.)

—1 for assuming *without proof* that the hub of the largest wye in H must be vertex y.