24. We construct a polynomial time decider for $2SAT = \{\phi \mid \phi$ is a 2CNF boolean formula that is satisfiabl as follows.

The decider first checks that the given string is a valid encoding of a binomial formula in 2CNF form. If not, it rejects.

The decider then encodes the formula into a graph as follows. Firstly, it creates two nodes named T and F representing absolute truth and falsehood, and then adds an edge between them. This captures the fact that the boolean value of true and the boolean value of false have opposite boolean values.

Then the decider loops through the variables of the formula, and for each variable (call it x), constructs two nodes, one representing the x being true, which we'll refer to as $T(x)$, and another representing x being false, which we'll refer to as $F(x)$. The decider then adds an edge between $T(x)$ and $F(x)$. The edge is meant to capture the fact that a variable can't be set to both true and false in a given assignment. From here on, we'll use the shorthand $T(\overline{x}) = F(x)$ and $F(\overline{x}) = T(x)$.

Then the decider loops through the clauses and does the following. For clauses of the form $y \vee y$, since they reduce to y, the decider adds an edge between $T(y)$ and F. This captures the fact that y must be true in any satisfying assignment. For clauses of the form $y \vee w$ where $y \neq w$, the decider adds an edge between $F(y)$ and $F(w)$. This captures the fact that both y and w can't be false in a satisfying assignment.

The graph construction is now complete. Finally, the decider runs 2-coloring on the graph (with two colors true and false), coloring adjacent nodes with opposite colors, starting with the node T and coloring it with the color true. If it is able to obtain a 2-coloring without the need to color some node with both colors, then it declares the given formula as satisfiable. Else, not.

We'll now prove the correctness of this decision procedure. First, if the given formula is satisfiable, then we can obtain a 2-coloring of the corresponding graph as follows. If variable x is set to true, then color the node $T(x)$ true and $F(x)$ false. If variable x is set to false, then color the node $T(x)$ false and $F(x)$ true. Then color node T true, and node F false. It is clear that this produces a valid 2-coloring because all nodes are colored and no two adjacent nodes are colored the same (given how we have constructed the graph).

Now, we prove the converse. If the graph has a 2-coloring, then we can obtain an assignment as follows. If for variable x, $T(x)$ is colored true (and therefore $F(x)$ colored false), then set x to true. Else set x to false. Given how we have constructed the graph and the fact that we color the node T true always, it is clear that this produces a satisfying assignment.

Finally, we note that all steps in the above decision procedure took time that

is polynomial in the size of the formula's encoding. Therefore $2SAT \in P$. This concludes the proof.

27. Let's call an undirected graph with certain nodes labeled with integer values as a numbered graph. Such graphs can be encoded in a string much like undirected graphs, except now, there is an additional section in the string that contains the numberings. Further, let's refer to a numbered graph as minable if it is possible to place mines on the unnumbered nodes such that for each node that is numbered, the number of mined nodes adjacent to it is equal to the number on the node. We need to prove that the language $M = \{< G, N >| \ G$ is a minable graph with numbering scheme N $\}$ is NP-complete. It is easy to see that $M \in NP$ since we can build a verifier for M which operates using certificates that encode the mine placement itself. We'll prove that $M \in$ NP-complete by showing a polynomial-time reduction from 3-SAT to M.

Consider the problem of determining whether a given boolean formula ϕ belongs to 3-SAT. For each variable of ϕ (call it x), construct two unnumbered nodes $T(x)$ and $F(x)$ that respectively represent setting x to true and false. Add another numbered node $Var(x)$ with the number 1, and add two edges, one between $T(x)$ and $Var(x)$ and another between $F(x)$ and $Var(x)$. This construction captures the fact that x can't be set to both true and false in a given assignment.

Next, for each clause (call it c), add three nodes, firstly, a numbered node $Cl(c)$ with the number 3 and secondly, two other peripheral helper nodes $P(c)$. Now, add two edges, one for each node in $P(c)$ between that node and $Cl(c)$. Also, add edges between $Cl(c)$ and the nodes corresponding to the terms in the clause. Specifically, if x appears in clause c, then add an edge between $Cl(c)$ and $T(x)$. If $\overline{x}$ appears in clause c, add an edge between $Cl(c)$ and $F(x)$. This captures the fact that the clause must be satisfied by the assignment. The two peripheral nodes act as buffer in case fewer than three of the clause's terms is set to true in the assignment.

This completes the construction. Now, we'll prove that $\phi \in$ 3-SAT iff for the constructed numbered graph G with numbering N, $< G, N >\in M$.

Firstly, if we have a satisfying assignment for ϕ, then for each variable (call it x), put a mine on node $T(x)$ if x is true in the assignment, else put a mine on $F(x)$. This will satisfy the mine requirement for $Var(x)$. Then, for each clause (call it c), put as many mines in the $P(c)$ as required to meet the requirement of 3 mines surrounding the node $Cl(c)$. This is a valid mine assignment for graph G.

Secondly, if we have a valid mine assignment for graph G, then for each variable (call it x), there is a mine placed at either $T(x)$ or $F(x)$ but not both (in order to meet the mine requirement of $Var(x)$). If it is set at $T(x)$, then set x to true

in the assignment, else false. For each clause (call it c), there are at most two mines placed in $P(c)$, so one of the mines meeting the mine requirement of 3 for $Cl(c)$ must come from a node that corresponds to a variable being set to true. This implies that the generated assignment for ϕ satisfies all clauses.

Finally, the construction of G from ϕ can be done in polynomial time in the size of the ϕ's encoding. Therefore, $M \in$ NP-complete. This concludes the proof.