Exercise 1. *Show that there is an algorithm to check whether a graph is bipartite that runs in $O(n^2)$ time, and show that no algorithm exists that has run time $o(n^2)$.*

Solution. In order to verify that a graph is bipartite we have to at least consider every possible edge, and since some graphs have $\Omega(n^2)$ edges, we cannot have an algorithm that runs in $o(n^2)$ time. In class, we discussed the following algorithm: for each connected component, fix a root vertex v and partition the set of vertices in that component by the parity of their distance to v (via a BFS which runs in $O(n^2)$ time). Since paths must alternate between partite sets in a bipartite graph, we now have that this graph (component) is bipartite if and only if this partition is correct. Therefore, we finish by checking that there are no "bad" edges in either of the partite sets, which is also done in $O(n^2)$ time.

Exercise 2. *Describe an algorithm to determine, given an undirected graph G as input, whether it is possible to direct each edge of G so that the resulting directed graph is strongly connected.*

Solution.

A graph G is *k-edge-connected* if it is connected and there is no set of $k-1$ edges whose removal disconnects G.

Let $H \leq G$. Then an *(open) ear* in G with respect to H is a path in G such that the endpoints of the path are in $V(H)$, while any internal vertices of the path are in $\overline{V(H)}$. A *closed ear* is a cycle such that exactly one of its vertices is in $V(H)$. We call an *ear decomposition* of G a chain $G_0 \leq G_1 \leq \cdots \leq G_k = G$, such that G_0 is a cycle, and each G_i is the union of G_{i-1} and an ear (wrt G_{i-1} in G).

Lemma 1. *A graph G is 2-edge-connected if and only if it has an ear decomposition (allowing open and closed ears).*

Proof. We will not write a full proof here, but one can follow the proof of Proposition 3.1.1 in Diestel. This is written to apply to vertex-connectivity, which allows only open ears, but it is easily adapted. □

An undirected graph G has a strongly connected orientation if and only if G is 2-edge-connected.

Proof. Suppose that G is not 2-edge connected. Then either G is not connected, in which case it clearly has no strongly connected orientation, or it has a bridge (a single edge whose removal disocnnects G). Let vw be a bridge edge of G and suppose G has some orientation with $v \to w$. Then there cannot be a directed path from w to v: such a path cannot use the vw edge, and there are no paths in G that avoid vw.

Now, suppose that G is 2-edge-connected. Then let $G_0 \leq G_1 \leq \cdots \leq G_k = G$ be an ear decomposition as in Lemma 1. We start by orienting G_0 with all edges in the same direction around the cycle. Then, orient each successive ear in one direction along its path or cycle. The choice of the two directions in each case is arbitrary. We claim that this leads to a strongly connetected orientation by induction by the subgraph chain. The base case, G_0, is clearly strongly connected. Now suppose that G_i is strongly connected. We direct the ear from one endpoint x to the other y (the endpoints might be identified). Now we have a path from any vertex in the ear to y. Since G_{i-1} is strongly connected, there is also a path from y to x, and therefore from y to any vertex in the ear. Furthermore, there is already a path

to and from y from any vertex in G_{i-1}, and combining all mentioned paths we obtain paths between any pair of vertices in G_i. □

Note that we did not phrase this as an explicit algorithm yet. Informally: ear decompositions can be found greedily. We can use a BFS to find any cycle in G to be G_0 (how?). Now, while $G_i \neq G$, take any edge vw that is not in G_i, but has at least one endpoint v in G_i (why must such an edge exist?). From w, use a BFS to search for a path to a vertex in G_i such that any internal vertices avoid G_i (why does such a path exist?). Now we have our next ear. Repeat until all edges are included. Convince yourself that this algorithm fails exactly if G is not 2-edge-connected.

Exercise 3. *Describe the complexity of finding the connected components in an undirected graph. Then design an algorithm to find the strongly connected components in a directed graph and describe its complexity.*

Solution. For the undirected case, we use the following algorithm. While there is an unclassified vertex v, run a BFS search from v to find all vertices in its component, then store this component and mark its members as classified. As we have seen, this algorithm runs in time $O(m)$, where m is the number of edges in G.

For directed graphs and strongly connected components, the above approach cannot be applied directly. A BFS search from a vertex v finds the vertices w such that there is a vw-path, but not necessarily a wv-path. One solution is to run BFS searches from every vertex and combine the information to find the partition in some way. There are various ways to implement this, and they will likely run in polynomial, but not linear, time in the number of edges. However, there are more efficient algorithms that do run $O(m)$ time, such as Kosaraju's algorithm.

Exercise 4. *Complete the proof of Claim 9.*

Solution. We claim that Ψ is satisfied if and only if for every literal x, x and $\overline{x}$ lie in different strongly connected components. By the previous exercise we have seen that strongly connected components can be determined in $O(m)$ time, where m is the number of edges, and we obtain 2 edges for each clause, which implies that $m = O(n)$.

An edge $x \to y$ in the graph means that if x is True then y must be True in order to satisfy the formula. Therefore, literals forming strongly connected components must be be either all True or all False. So, if x and $\overline{x}$ for some lieral x lie in the same strongly connected component then Ψ is not satisfiable. Conversely, suppose that this does not happen for any literal. First, we observe that if x and y are together in a strongly connected component, then $\overline{x}$ and $\overline{y}$ must also be together in a (different) strongly connected component. Therefore, the strongly connected components come in "opposite" pairs, and we must choose one from each pair. If any "opposite" pair of strongly connected components, let's call them C_1 and C_2 have a path $C_1 \to C_2$ (wlog), then we must choose C_2 to avoid a contradiction. Since they are separate strongly connected components, such a path can only exist in one direction. If there are no paths, we can choose either. Now, our choice corresponds to a satisfying assignment of all variables.

Exercise 5. *As in the example above, design a gadget for an AND-gate.*

Solution. The following gadget works: the final vertex on the right can only be colored T if both a and b are colored T (since the middle unlabeled vertices must be colored F and B respectively).