2. (**Cycle Cover**)

Main Idea The main observation is that a cycle cover has exactly one incoming and one outgoing edge for each vertex. So, we can think of splitting each vertex into two: the left one and the right one, and think of edges as going from the left copies to the right copies of the vertices. Then a cycle cover corresponds to a matching between the left and the right vertices.

Algorithm On an input graph $G = (V, E)$,

1. Construct an undirected bipartite graph $G' = (L \cup R, E')$ such that $L = V$, $R = V$, and $(i, j) \in E$' if and only if $(i, j) \in E$.
2. Run the algorithm for Maximum Bipartite Matching from class on G'.
3. If it does not find a perfect matching, output "no cycle cover." If it finds a perfect matching M, generate a cycle cover C from it as follows:
   While M is not empty, repeat:
   3-1. Start at an arbitrary node i and set $v = i$.
   3-2. Remove the only edge of the form (v, j) from M, add it to the current cycle, and set $v = j$. Repeat this step until $v = i$.
   3-3. Add this cycle to C.
4. Output C.

Termination We show that the algorithm terminates. As the step 1 and 2 always halt, we focus on the two loops in step 3. Each iteration of the outer loop removes edges from M, so it can be performed at most $|M|$ times.

To show that the inner loop terminates, we have to prove that we will get back to the starting node i by following edges of the matching as described in the algorithm. Assume for contradiction that we cannot get back to i; that is, after removing (v, j) from M and adding it to the current cycle, we get stuck at $j \in L$ with $j \neq i$ since the edge of the form (j, w) is already removed. Suppose the edge (j, w) is removed at step t. Then, at step $t - 1$, an edge of the form (u, j) must have been removed and the current vertex v was set to j. It means M has two matching edges of the form (u, j) and (v, j), which contradicts the assumption that M is a perfect matching. Hence, the inner loop must terminate.

Claim 1. *If G has a cycle cover, then G' has a perfect matching.*

Proof. Let C be a cycle cover of G and let M be a set of edges of G', such that $(i, j) \in M$ iff (i, j) is an edge in a cycle of C. Since every node i is covered by exactly one cycle in C, all cycles in C contain exactly one incoming and one outgoing edge for i. That is, M contains exactly one edge of the form (v, i) for all $i \in R$ and exactly one edge of the form (i, v) for all $i \in L$. This defines a perfect matching. □

Claim 2. *If G' has a perfect matching M, then our algorithm returns a cycle cover C.*

By construction, our algorithm returns a set of cycles. We have to prove that these cycles are vertex-disjoint and cover all the vertices. The directed edges in C are from the perfect matching M, so none of these edges has the same starting vertex or the same ending vertex. This property is useful in proving that C is a cycle cover.

Proof. As none of the directed edges has the same starting vertex or ending vertex, each vertex has only one incoming edge and one outgoing edge. This proves that the cycles are vertex disjoint.

Assume that there are n vertices in G. As no two edges share the same starting vertex, n edges in C must have distinct starting vertices. The same holds for the ending vertices. This implies that the edges in C cover all vertices. □

Time and Space Complexity As the bipartite graph G' has $2n$ nodes and m edges, it takes $O(m+n)$ time and space to generate G'. Once G' is obtained, we invoke the algorithm for Maximum Bipartite Matching which runs in time $O(mn)$ and space $O(m+n)$. (For a bipartite graph with N nodes and M edges, we gave an algorithm for Maximum Bipartite Matching that runs in time $O(NM)$ and space $O(N+M)$ during the lecture.) Step 3 of the algorithm can be implemented to run in time $O(n)$, since M has n edges and each of them is processed exactly once. Hence, it takes $O(mn)$ time and $O(m+n)$ space.