COL351: Analysis and Design of Algorithms **Fall 2024**

Practice Sheet 13

Announced on: Nov 15 (Thu)

1. Recall that in the VERTEX COVER problem, we are given an undirected graph $G = (V, E)$ and an integer k. The goal is to compute a subset of vertices $S \subseteq V$ such that $|S| \leqslant k$ and for every edge $e \in E$, there exists a vertex $v \in S$ such that e is incident to v. If no such subset exists, declare "no solution".

   Suppose you have algorithm A, which, given a graph G and a number k, outputs YES if and only if G has a vertex cover of size at most k. Assuming that A runs in polynomial time, show that you can find a vertex cover of minimum size in polynomial time.

2. The DIRECTED HAMILTONIAN CYCLE problem is as follows: Given a directed graph G, is there a cycle that contains all the vertices? Suppose you have a polynomial-time algorithm for this problem. Show that you can also find such a cycle (if it exists) in polynomial time.

3. The UNDIRECTED HAMILTONIAN CYCLE problem can be defined similarly as above for undirected graphs. The UNDIRECTED HAMILTONIAN PATH problem is as follows: Given an undirected graph G, is there a path that contains all the vertices? Show that UNDIRECTED HAMILTONIAN PATH reduces to UNDIRECTED HAMILTONIAN CYCLE.

4. Suppose you are consulting for a group that manages a high-performance real-time system in which asynchronous processes use shared resources. Thus, the system has a set of n processes and a set of m resources. At any given point in time, each process specifies a set of resources that it requests to use. Many processes might request each resource at once; but it can only be used by a single process at a time. Your job is to allocate resources to processes that request them. If a process is allocated all the resources it requests, then it is *active*; otherwise, it is *blocked*. You want to perform the allocation to maximize the number of active processes. Thus, we phrase the RESOURCE RESERVATION problem as follows: Given a set of processes and resources, the set of requested resources for each process, and a number k, is it possible to allocate resources to processes so that at least k processes will be active? Show that RESOURCE RESERVATION is NP-hard.

   Hint: Show a reduction from INDEPENDENT SET.

5. Given an undirected graph $G = (V, E)$, a *feedback vertex set* is a set $X \subseteq V$ with the property that $G - X$ has no cycles. The UNDIRECTED FEEDBACK VERTEX SET problem asks the following question: Given an undirected graph G and an integer k, does G contain a feedback

vertex set of size at most k? Prove that the UNDIRECTED FEEDBACK VERTEX SET is NP-hard.

Hint: Show a reduction from VERTEX COVER.

6. Consider a set $A = \{a_1, \ldots, a_n\}$ and a collection $B_1, B_2, \ldots B_m$ of subsets of A. (That is, $B_i \subseteq A$ for each i.) We say that a set $H \subseteq A$ is a *hitting set* for the collection $B_1, B_2, \ldots, B_m$ if H contains at least one element from each B_i, i.e., if $H \cap B_i$ is nonempty for each i. We now define the HITTING SET problem as follows: We are given a set $A = \{a_1, \ldots, a_n\}$, a collection $B_1, B_2, \ldots, B_m$ of subsets of A, and a number k. We are asked whether there exists a hitting set $H \subseteq A$ for $B_1, B_2, \ldots, B_m$ so that the size of H is at most k? Prove that HITTING SET is NP-hard.

   Hint: Show a reduction from VERTEX COVER.

7. In an undirected graph $G = (V, E)$, we say a subset $D \subseteq V$ is a *dominating set* if every $v \in V$ is either in D or adjacent to at least one member of D. In the DOMINATING SET problem, the input is a graph and a number k, and the aim is to find a dominating set in the graph of size at most k, if one exists. Prove that this problem is NP-hard.

   Hint: Show a reduction from VERTEX COVER.

8. Consider a greedy algorithm for VERTEX COVER that repeatedly picks a vertex that “covers” the most uncovered edges. Show that there exist bipartite graphs on n vertices for which the size of the vertex cover produced by the greedy algorithm can be at least $\Omega(\log_2 n)$ times the size of the optimal vertex cover.