Complexity Practice Problems

Note that your final covers ALL the material discussed in class. These practice problems are representative of the type of problems you might get on the final from the LAST SECTION ONLY but do NOT cover all of the material you are responsible for.

1. Prove that the following problem is NP-complete. Given n elements $E = \{e_1, \ldots, e_n\}$ and m subsets $S_i \subseteq E$, and integer k, find some subset of the sets $\{S_i\}$ with size $\leq k$ (in other words at most k of the sets) such that their union is E, if this exists.

Solution:

- First to show that this is in NP. Given such a set cover, we could verify that it has at most k elements, then iterate through every element in E and check that it's present in one of our sets S_i in polynomial time.
- There are a few ways to do this reduction, we will reduce from Vertex Cover.

  Let $SETCOVER$ be an algorithm that solves the above problem. Given an input to Vertex Cover, $(G = (V, E), k)$

  - Let E be the set of elements in the Set cover.
  - For each vertex $v \in V$, construct the set S_v which is the set of all edges incident to V. Let S be the collection of all these sets.
  - Run $SETCOVER(E, S, k)$, and output the vertices that correspond with the found sets.
- Again, we see that in this interpretation, $SETCOVER$ is a generalization of VC.

  Recall in the Set cover problem, we choose the fewest *sets* that cover all *elements*, and in Vertex cover, we are choosing the minimum number of *vertices* that cover all *edges*. An individual vertex covers all of its neighboring edges.

  So the second is a special case of the first, where each "element" is an edge, and each "set" is the set of neighboring edges for each vertex. Thus a solution to set cover (with elements as edges and sets as the neighborhoods of vertices) with at most k sets will exactly correspond to a vertex cover with at most k vertices.

  All steps done are polynomial time on polynomial inputs, thus a poly-time algorithm for $SETCOVER$ implies a poly time algorithm for Vertex Cover.

  Thus $SETCOVER$ is NP-complete.

2. Prove the following problem is NP-complete. Given a family of sets $\{S_1, S_2, \ldots, S_n\}$ and a budget b, find a set $H \subseteq \bigcup_i S_i$ of size $\leq b$ which intersects every S_i, if such an H exists. In other words, we want $H \cap S_i \neq \emptyset$ for all i.

Solution: Call this problem the Minimum Hitting Set(S, b) problem. First we show that this is in NP. Given a hitting set H, we could check that it has fewer than b elements, then

iterate through every set S_i and check that it has non-empty intersection with our chosen H, all in polynomial time.

We will again reduce from Vertex Cover . In the Hitting Set problem, we choose the fewest *elements* so that each *set* has at least one of them. In the Vertex Cover problem, we are choosing the minimum number of *vertices* so that each *edges* borders at least one of them.

Given an input $k, G = (V, E)$ to Vertex Cover , we do the following. Let V be the set of underlying elements, and create a set S_e for each edge: each set is a two-element set of vertices. A solution to minimum set cover with budget $b = k$ will be some set of vertices, so that every "set" or edge, contains at least one of the chosen vertices and uses at most k vertices total. This is exactly a minimum vertex cover.

Again, the proof of correctness of this reduction is trivial because it follows from the fact that this is again a generalization. A particular assignment of the inputs to Minimum Hitting Set turns the problem exactly into Vertex Cover - this makes Vertex Cover a special case of Minimum Hitting Set.

Thus Minimum Vertex Cover reduces to Minimum Hitting Set.

3. Prove that the following problem is NP-complete. Given a weighted graph G and bound B, find a cycle in G that visits every vertex exactly once and has the total sum of edge weights $\leq B$, if this exists.

Solution: This is the well known Traveling Salesman Problem problem. First to show that this is in NP. Given a cycle, we can verify that it is a cycle and visits every node once, and then verify that the summation of all weights is $\leq B$ all in polynomial time.

We reduce from Hamiltonian Cycle. Given a Hamiltonian Cycle input G, create a Traveling Salesman Problem instance with G and all edge weights set to 1. We then run Traveling Salesman Problem on this weighted graph setting $B = |V|$. As with the other generalizations, a Hamiltonian Cycle in G is exactly a Traveling Salesman Problem in the weighted G with total weight at most $|V|$. Thus Hamiltonian Path reduces to Traveling Salesman Problem .

4. Prove that the following problem is NP-complete: DegreeRestrictedMST: Given an undirected graph $G = (V, E)$ and an integer k, find a spanning tree T of G such that each vertex of the tree has maximum degree k, if such a tree exists. You may give a reduction from Hamiltonian Path, which you may assume is NP-complete for this problem only.

(Hamiltonian Path Problem: Given a graph G, find a **path** that visits each vertex exactly once, if one exists.)

Solution:

- First we show that this is in NP. Given such an MST T (we can convert it to an adjacency matrix representation in polytime if it isn't already), we can verify that it

is an MST in polytime by running DFS to ensure that all edges are connected, by checking that there are exactly $|V| - 1$ edges to make sure that it is a Tree, and then checking that each vertex is incident to at most k edges in an additional $O(V)$ time. This is all polynomial, thus this problem is in NP.

- We reduce from Hamiltonian Path as stated in the hint. The clever observation is that our problem is sneakily a generalization of Hamiltonian Path.

  Given an input G to Hamiltonian Path, run Degree-restricted MST with inputs G and 2, and output the MST returned.

- The observation is that an MST with max degree 2 MUST be a path! Thus finding an MST with max-degree 2 in a graph is equivalent to finding a path that connects all the vertices.