CIT 596 Online **Summer 2020**

Module 13–14

Name: Solutions

Questions requiring written answers.

1. We know that the subset sum problem is NP-Complete. Remember that the subset sum problem begins with an array of positive integers A and a targeted sum k. It then asks whether there exists a subset of A that sums up to this target value k. We would like to show that the following problem called Zero sum is also NP-Complete. Given a set of integers, is there a non-empty subset whose sum is zero. Show that the zero sum problem is in NP-Complete.

   Solution: Subset Sum $\le_p$ Zero Sum

   NP: The certificate for an instance of the Zero Sum problem is the non-empty subset of values. To verify if this is a solution, simply sum the values and check that it is zero.

   NP-Hard: We can show Zero Sum is NP-Hard by reducing Subset Sum to it. An instance of Subset Sum is an array of positive integers A and the target sum k. To reduce this to zero sum, simply append the value $-k$ to the array A and give this value as input into Zero Sum. The Subset Sum answer is YES iff the Zero Sum answer is YES.

   Correctness: We first show that if Zero Sum returns YES, then the answer to the Subset Sum problem is indeed YES. Note that if Zero Sum returns YES, then the solution subset of integers must include the value $-k$ as all other integers in the array are strictly positive, and we know that the subset is nonempty. Thus, in order for the sum of the solution to be zero, the remaining positive integers in the solution must add to k. Thus, this is the solution to the Subset Sum problem, and the answer should be YES.

   Now we show that if the answer to the Subset Sum problem is YES, then the Zero Sum problem it reduces to will return YES. Since Subset Sum should return YES, then there exists a set of the positive integers A that add to k. Then, in the Zero Sum problem it reduces to, we can add those integers together along with the added $-k$ value to yield zero, so the answer to Zero Sum must also be YES.

   Runtime: The reduction takes only constant time to add the value $-k$ to the array of integers.

2. The PARTITION problem is as follows:

   Instance: A (multi-)set of numbers $S = \{a_1, a_2, \dots, a_n\}$.

   Question: Can S be partitioned into two (multi-)sets A and B such that the sum of the numbers in A is equal to the sum of the numbers in B?

   Subset Sum $\le_p$ Partition

   Prove that PARTITION is NP-complete.

   Solution:

   PARTITION **is in NP**: Let the certificate represent the two sets A and B that form the partition of S. Then the certificate's validity can be verified by simply adding up the numbers in A and B and checking if their sums are equal, and then confirming that $S = A \cup B$ and $A \cap B = \emptyset$. All of this can be done in polynomial time, since it just involves iterating over the sets.

   PARTITION **is NP-hard**: We reduce from SUBSET-SUM. In the SUBSET-SUM problem, we are given a set S and a number t and we want to know whether there is a subset of S whose items sum to t. First, sum up the numbers in S; call this sum x. Add the number $2t - x$ to S to create S' and use S' as the input to PARTITION. Return YES iff PARTITION returns YES on S'.

   $\{S\}, t$

   $2t - x + \{S\}$

Correctness: Note that when the input is transformed into an input to the PARTITION problem, the sum of all values is now $x + 2t - x = 2t$, so PARTITION returns YES iff there are two disjoint subsets of S' such that each subset sums to t. Since the item added to S (which was $2t - x$) can only be in one of the two parts of the partition, the other part must sum to t and contain elements that were originally in S. Thus, PARTITION returns YES iff SUBSET-SUM returns YES.

Runtime: The reduction involves summing the numbers in S and adding a new value. This can clearly be done in polynomial time.

3. Exercise 5 of Chapter 8 on pages 506-507

Solution:

HITTING SET **is in NP**: The certificate is the hitting set H. To verify its correctness, iterate through all the sets B_i $(1 \le i \le m)$ and check that $|H \cap B_i| \ge 1$. In addition, check that $|H| \le k$. Accept the certificate iff all of these conditions are true.

HITTING SET **is NP-hard**: We reduce from VERTEX-COVER. In VERTEX-COVER, we are given a graph G and a parameter k, and we wish to know whether there is a set S of k vertices such that every edge has at least one endpoint in S. In the reduction, every vertex in G becomes an element in A, the set of all elements in the HITTING SET problem. Then for each edge $\{u, v\}$ in the graph, we create the set $\{u, v\}$ as one of the sets "to be hit". Return YES for VERTEX-COVER iff the instance of HITTING SET with our constructed sets, along with the parameter k, returns YES.

Correctness (sketch): Since the sets B_i correspond to edges in the original graph, all subsets in the HITTING SET problem are hit by H iff all edges in the VERTEX-COVER problem have at least one endpoint in the corresponding set of vertices of size k.

Runtime: There is a one-to-one correspondence between vertices and edges in the VERTEX-COVER problem and elements and subsets in the HITTING SET problem. Thus, the transformation can be done in linear (and hence, polynomial) time.

4. We say that graph $G_1 = (V_1, E_1)$ is isomorphic to graph $G_2 = (V_2, E_2)$ if there is a bijective mapping f from V_1 to the vertices of G_2 such that V_2 such that $(u, v) \in E_1$ if and only if $(f(u), f(v)) \in E_2$. The SUBGRAPH ISOMORPHISM problem is as follows:

Instance: Two graphs G and H.

Question: Does H have a subgraph H' such that G is isomorphic to H'?

Prove that SUBGRAPH ISOMORPHISM is NP-complete.

Solution:

SUBGRAPH ISOMORPHISM **is in NP**: The certificate is the subgraph H' that should be isomorphic to G and the bijective function between the vertex sets. To verify the certificate's validity, we simply check that H' is isomorphic to G (by using the given bijective function). Then we check that H' is indeed a subgraph of H by looking through the adjacency lists (or matrices) and checking that all of the vertices and edges in H' are also in H.

SUBGRAPH ISOMORPHISM **is NP-hard**: We reduce from INDEPENDENT SET. In the independent set problem, we are given a graph G and a number k, and wish to answer the question "is there an independent set of size k in G?" To transform this input to an input to SUBGRAPH ISOMORPHISM simply use the graph G as the input H in the SUBGRAPH ISOMOPRHISM problem; for the input G in the SUBGRAPH ISOMORPHISM problem, simply use a graph with k vertices and no edges. Return YES iff the SUBGRAPH ISOMORPHISM problem returns YES.

Correctness: If the SUBGRAPH ISOMORPHISM problem returns YES, then clearly there is an independent set of size k in the original graph, since the subgraph is itself the independent set. If the

SUBGRAPH ISOMORPHISM problem returns NO, then there must not be an independent set of size k in the original graph, otherwise the SUBGRAPH ISOMORPHISM problem would have returned YES because that independent set is the isomorphic subgraph H'.

Runtime: Note that k is less than the number of nodes in the G, the input graph to the independent set problem (otherwise the problem is trivial). Thus, constructing the graph of k vertices and no edges can clearly be done in polynomial time.

5. A Hamilton path in a graph is a simple path that visits all vertices and a Hamilton cycle is a simple cycle that visits all vertices. The HAMILTON PATH (respectively, HAMILTON CYCLE) problem is the following:

   Instance: An undirected graph G.

   Question: Does G contain a Hamilton path (respectively, Hamilton cycle)?

   (a) Assume without proof that HAMILTON PATH is NP-complete. Prove that HAMILTON CYCLE is NP-complete.

   (b) Now do the reverse: In other words, assume that HAMILTON CYCLE is NP-complete and prove that HAMILTON PATH is NP-complete. (This is the hard direction. It is easy to find a mapping that maps YES-instances of HAMILTON CYCLE to YES-instances of HAMILTON PATH. The challenge is to find a reduction that does this, but also maps NO-instances to NO-instances.

Solution: Ham Path $\leq_p$ Ham Cycle

(a) **NP**: The certificate is the ordering of vertices (or edges) that constitute the cycle. Simply check that such a cycle indeed exists in the given graph.

NP-Hard: We reduce from HAMILTON PATH. Given an instance of such problem, which is a graph G, add a single vertex to the graph v. Then, add edges from v to every other vertex in the graph and feed this new graph G' into HAMILTON CYCLE. Return YES for HAMILTON PATH iff HAMILTON CYCLE returns YES.

Correctness: If HAMILTON CYCLE returns YES, then there is a cycle that traverses every vertex of G'. Thus, it also traverses the added vertex v. Let the edges involving v be (a, v) and (v, b). Consider the path from a to b that does not include v (such a path must exist since by definition of cycle, there are two paths between the two vertices). Since this path is part of the Hamilton Cycle, it traverses every vertex in the original graph G. This is the Hamilton Path.

Now we prove that if there is a Hamilton Path in G, then there is a Hamilton Cycle in G'. Consider the path in G. Assume that it starts at vertex a and ends at vertex b. Then, in G', we know that a and b have edges to the new vertex v by construction. Thus, adding the edges (v, a) to the beginning of the path and (b, v) to the end of the path forms a Hamilton Cycle in G'.

Running Time: The reduction takes polynomial time ($O(n)$) to add one vertex and add n edges.

(b) **NP**: The certificate is the edges that comprise the Hamilton Path. Simply check that the edges given actually exist in the graph to form a path that touches every vertex.

NP-Hard: We are given an instance of HAMILTON CYCLE which is a graph G. To reduce this to HAMILTON PATH, choose any vertex $u \in G$. Add a new vertex u' that is only connected to u. Add another new vertex v' that is connected to all the neighbors of u. Finally add a third new vertex o, which is connected only to v'. This will be the graph G' that is the instance of Hamilton Path.

Correctness: We first prove that if G has a Hamilton Cycle, then G' will have a Hamilton path. This is straightforward: The cycle must use two edges incident on u, say (u, a) and (u, b). Break the cycle at (u, a) this gives a Hamilton path in the original graph G. We can easily extend this path by including the edges (u', u), (a, v') and (v', o) to form a Hamilton Path in G'.

Now we prove that if G' has a Hamilton Path, then G has a Hamilton cycle. A Hamilton Path in G' must start at u' and end at o since these are both vertices of degree 1 and cannot be in the middle of the Hamilton Path. So the path must go from u' to u, traverse all the vertices in G, ending at one of the neighbors a of u, and from there go to v' and o. The key is that the Hamilton Path portion in G must end at a neighbor of u in order to be able to continue to v' and o. Because it ends at a neighbor a, we can take this Hamilton Path in G and add the edge (u, a) to get a Hamilton Cycle in G.

Running Time: The reduction involves just adding three vertices and at most $n + 1$ edges to the original graph m times, which is clearly polynomial.

6. Suppose we are given a polynomial-time algorithm for solving the Hamilton Cycle decision problem on directed graphs. Describe how you could make just polynomially many calls to this algorithm to actually find a Hamilton cycle in a directed graph that has one.

   Solution:

   Algorithm: Iterate through the vertices in the graph. For each vertex, consider all of its outgoing edges. For each outgoing edge e, select it and discard all other outgoing edges; then use the blackbox to determine whether the graph still has a Hamiltonian cycle. If it does, then discard all edges outgoing from the current vertex except e, and move onto the next vertex. After iterating through all of the vertices, the remaining graph is a Hamiltonian cycle in G.

   Correctness: At each stage of the algorithm, we maintain the invariant that there is a Hamiltonian cycle in the remaining graph. This is clearly true at the beginning of the algorithm (this is given to us). After deleting some edges at each vertex in the graph, this is also true by construction. Since the Hamiltonian cycle must go through every vertex, we know that at each vertex, at least one of its outgoing edges must be in the Hamiltonian cycle, so the algorithm will always be able to find an edge to keep for each vertex.

   Runtime: We consider each edge in the graph once, so this algorithm uses m calls to the black box, which is polynomial with respect to the size of the input.

7. The Set Cover problem is the following:

   Instance: A set $U = \{1, 2, \ldots, n\}$ of n elements, a collection of subsets $S_1, S_2, \ldots, S_m$ of U, and an integer K.

   Question: Are there (at most) K sets among the S_i's whose union is equal to U? In other words, are there K or fewer sets which together cover all the elements of U?

   Starting with a problem that we have shown to be NP-complete, prove that Set Cover is NP-complete.

   Solution:

   NP: The certificate is the collection of $\leq K$ sets. It is easily verifiable whether or not the union of them is equal to U.

   NP-Hard: We can reduce Vertex Cover to Set Cover. The input Vertex Cover is a graph G and a number k and we wish to transform this into a universe, a collection of subsets, and an integer. We let the edges of G form the universe U. Then, each subset corresponds to a vertex from G such that the subset for vertex v is the collection of edges incident on v. Finally, the integer parameter is just k. Then, the answer to Vertex Cover is YES iff the answer to Set Cover is YES.

   Correctness: We first show that if there exists a vertex cover with size $\leq k$, then there is a set cover with size $\leq k$. Consider the vertex cover i.e. the collection of vertices. Now after the transformation, choose the subsets corresponding to the vertices of the vertex cover. Since the sets correspond to the edges incident on the vertex, and since we know that the vertices of the vertex cover are collectively incident to every edge in the graph, then clearly the chosen subsets collectively cover the universe, which are just the edges of the graph.

Now we show that if there exists a set cover of size $\leq k$, then there exists a vertex cover of size $\leq k$. Since we established a bijection between subsets of SET COVER and the vertices of VERTEX COVER, select the vertices corresponding to the SET COVER solution. We claim that every edge of the graph is covered. Consider any edge e. It corresponds to an element of the universe U. Then we know there is some subset S_i that covers e. By construction, S_i corresponds to a vertex on which e is incident, so e is covered by that vertex. Thus, all edges are covered.

Running Time: Converting the edges to the universe takes either constant or $O(m)$ time, which is polynomial in either case. Forming the subset then takes at most $O(nm)$ time in the case where every vertex has $n-1$ edges. Thus, the reduction is polynomial.

8. We have seen how the independent set (decision) problem is NP complete. Suppose we are given a subroutine that will solve this problem. Show how you can use the subroutine to find a maximum independent set in a graph.

   Solution:

   We can first find the true size K of the independent set by asking if the graph contains an independent set of size k, iterating from $k = 1$ to n. Once this value is found, iterate through the vertices of the graph. For each, temporarily remove the vertex from the graph and use the subroutine to check if there still exists an independent set of size K. If so, then the vertex chosen is not necessary for our solution and can be permanently deleted. If not, then we know the vertex must be part of our maximum independent set. As a result, we should place it back into the graph and instead remove all of its neighbors from the graph. This process takes at most n iterations. Once complete, we are left with the maximum independent set. The total number of calls to the subroutine is at most n to find the true value K and at most n to iterate through the vertices.